Uniwersytet Wrocławski
Wydział Matematyki i Informatyki

Rekonstrukcja poliomin wypukłych

Maciej Gębala

Praca doktorska

Promotor: prof. Leszek Pacholski

Instytut Informatyki
ul. Przesmyckiego 20
51-151 Wrocław
email: mgc@ii.uni.wroc.pl

Wrocław 2001

Spis treści

Wstęp

Tomografia komputerowa jest dziedziną zajmującą się rekonstrukcją obiektów, które są opisane tylko przez pewien zbiór swoich rzutów. W tomografii dyskretnej obiektem, który chcemy zrekonstruować jest skończony zbiór pól w dwu- lub trójwymiarowej kracie. Wszystkie wiadomości o danym zbiorze pól są dostępne tylko w postaci dyskretnych rzutów. W najprostszym przypadku, rzut skończonego zbioru S w kierunku u jest funkcją, której wartości opisują liczbę punktów zbioru na każdej linii równoległej do u lub liczbę punktów zbioru S należących do podprzestrzeni prostopadłej do u. Problemy dyskretnej tomografii komputerowej są również studiowane w kontekście przetwarzania obrazów i kompresji danych.

Ważnym podproblemem jest tutaj rekonstrukcja dyskretnych zbiorów dwuwymiarowych z ich rzutów ortogonalnych, to jest poziomego i pionowego. Dla większości klas takich zbiorów problem rekonstrukcji jest NP-zupełny. W tej pracy zajmujemy się opisem i analizą pewnej klasy zbiorów dyskretnych, dla których rekonstrukcja jest możliwa w czasie wielomianowym. Klasa ta to poliomina wypukłe, czyli spójne zbiory o ciągłych przekrojach ortogonalnych. Zajmujemy się rekonstrukcją takich zbiorów z rzutów dokładnych oraz rekonstrukcją z rzutów przybliżonych, gdzie rzuty dopuszczalnego rozwiązania mogą się różnić o określony błąd od podanych rzutów wejściowych. Podajemy też wielomianowy algorytm rekonstrukcji z rzutów dokładnych dla pewnej klasy poliomin trójwymiarowych. Jest to pierwszy algorytm wielomianowy dla zbiorów trójwymiarowych.

Rozprawa jest podzielona na sześć rozdziałów. Rozdział pierwszy zawiera przedstawienie problemu oraz przegląd dotychczas uzyskanych wyników. W rozdziale drugim ujęte zostały pewne właściwości poliomin wypukłych, przydatne do konstruowania algorytmów rekonstrukcji. Rozdział trzeci zawiera opis i analizę wielomianowego algorytmu rekonstrucji dwuwymiarowych poliomin wypukłych z rzutów ortogonalnych. W rozdziale

czwartym dostosowujemy i uogólniamy ten algorytm do pewnej klasy trójwymiarowych poliomin wypukłych. Przedostatni, piąty rozdział dotyczy rekonstrukcji dwuwymiarowych poliomin wypukłych z rzutów przybliżonych. Pokazujemy w nim, że problem rekonstrukcji dwuwymiarowych zbiorów z przybliżonych rzutów jest co najmniej tak trudny, jak rekonstrukcja z dokładnych rzutów, oraz podajemy wielomianowy algorytm rekonstrukcji dla poliomin wypukłych. Na koniec, w rozdziale szóstym, podajemy niektóre problemy otwarte, które wiążą się bezpośrednio z problemami rozważanymi w tej pracy.

Rozdział 1

Wprowadzenie

1.1 Definicja Problemu

Niech $\mathcal{R}$ oznacza prostokątną kratę, o wymiarach m na n, której jednostkowe kwadratowe pola są lub nie są zaciemnione. Kratę $\mathcal{R}$ możemy identyfikować z czarno-białym rysunkiem rastrowym. Matematycznym modelem takiej kraty, którym będziemy posługiwać się w tej pracy, będzie macierz zerojedynkowa R o m wierszach i n kolumnach, gdzie elementy odpowiadające zaciemnionym polom mają wartość 1 a polom jasnym 0. Niech $R[i,j]$ oznacza element macierzy R znajdujący się w wierszu i i kolumnie j. Niech $Q(R)$ oznacza zbiór wszystkich elementów macierzy R, czyli

$$Q(R) = \{\ [i,j]\ :\ 1 \leqslant i \leqslant m\ \wedge\ 1 \leqslant j \leqslant m\ \}.$$

Niech $S(R)$ oznacza zbiór elementów macierzy R równych 1, tj.

$$S(R) = \{\ [i,j]\ :\ R[i,j] = 1\ \}.$$

Ponieważ R będzie na ogół ustalone, zamiast $Q(R)$ i $S(R)$ będziemy często pisali Q i S.

Definicja 1.1 *Dla zbioru $S = S(R)$ definiujemy $h_i(S)$ – rzut i-tego wiersza S, jako liczbę tych elementów i-tego wiersza R, które są równe 1, czyli*

$$h_i(S) = \sum_{j=1}^{n} R[i,j],$$

dla $i \in \{1,\ldots,m\}$. Analogicznie definiujemy $v_j(S)$ – rzut j-tej kolumny S, jako liczbę elementów j-tej kolumny R, które są równe 1, czyli

$$v_j(S) = \sum_{i=1}^{m} R[i,j],$$

dla $j \in \{1, \ldots, n\}$. *Rzuty wierszy będziemy nazywać rzutami poziomymi a rzuty kolumn pionowymi. (Jeśli zbiór* S *jest ustalony, to piszemy krócej* h_i *i* v_j*.)*

Dwa pola w kracie $\mathcal{R}$ nazywamy sąsiednimi, jeśli mają wspólną krawędź. Stąd dwa elementy $[i, j]$ i $[i', j']$ w macierzy R są sąsiednimi, jeśli jedne z ich współrzędnych, pionowe albo poziome, różnią się dokładnie o jeden, czyli

$$|i - i'| = 1 \quad \text{i} \quad j = j' \qquad \text{lub} \qquad i = i' \quad \text{i} \quad |j - j'| = 1.$$

Zbiór S nazywamy spójnym, jeśli dla każdych dwóch elementów należących do S są one sąsiednie lub istnieje ścieżka w S, przechodząca kolejno przez sąsiednie elementy należące do S, która łączy te dwa elementy.

W tej pracy będziemy badali następujące własności zbioru S:

Definicja 1.2

(I) *Zbiór* S *ma własność* p*, jeśli* S *jest spójnym zbiorem skończonym.*

(II) *Zbiór* S *ma własność* v*, jeśli każda kolumna zbioru* S *jest spójna, to znaczy nie istnieje w macierzy* R *kolumna zawierająca 0 między dwoma elementami równymi 1.*

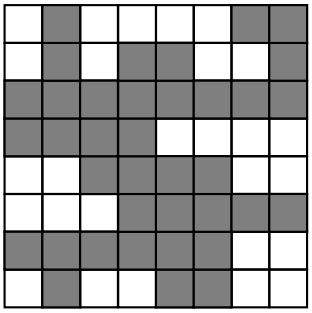
(III) *Zbiór* S *ma własność* h*, jeśli każdy wiersz zbioru* S *jest spójny, to znaczy nie istnieje w macierzy* R *wiersz zawierający 0 między dwoma elementami równymi 1.*

Mówimy, że zbiór S *należy do klasy* (x) *(*$S \in (x)$*) wtedy i tylko wtedy, gdy* S *ma własność* x*. Jeśli* $S \in (x)$ *i* $S \in (y)$*, to piszemy* $S \in (x, y)$*.*

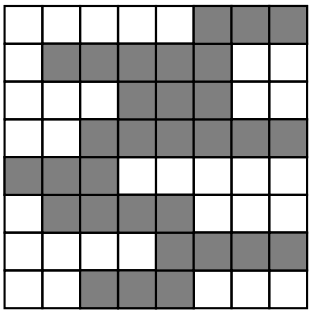
Teraz wprowadzamy następujące pojęcia

- *zbiór* S *jest poliominem, jeśli* $S \in (p)$ *(zobacz Rysunek 1.1),*
- *zbiór* S *jest poliominem ze spójnymi wierszami, jeśli* $S \in (p, h)$ *(zobacz Rysunek 1.2),*
- *zbiór* S *jest poliominem ze spójnymi kolumnami, jeśli* $S \in (p, v)$*, oraz*
- *zbiór* S *jest wypukłym poliominem (to jest poliominem ze spójnymi wierszami i kolumnami), jeśli* $S \in (p, h, v)$ *(zobacz Rysunek 1.3).*

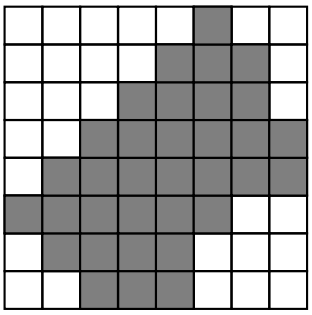
Dla każdej z wyżej zdefiniowanych klas można sformułować problem rekonstrukcji zbioru S tej klasy z jego prostopadłych rzutów:



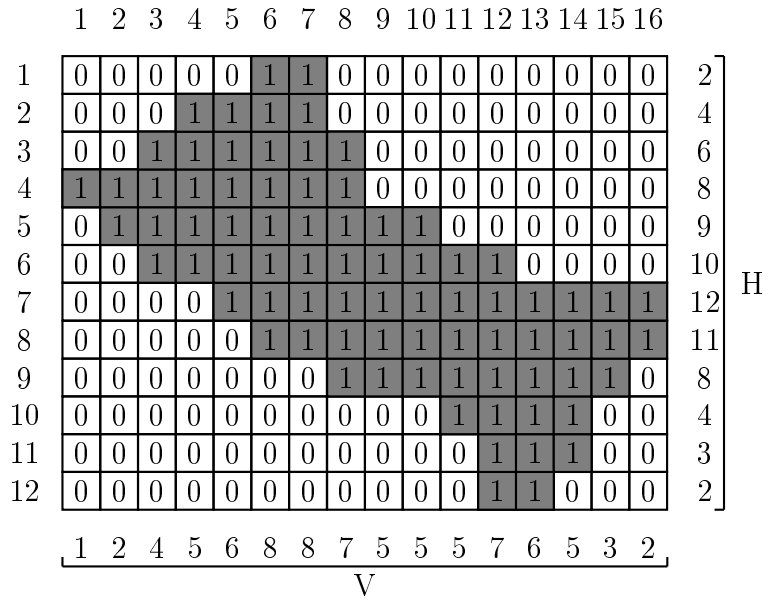
Rysunek 1.1: Przykład poliomina bez dodatkowych warunków na spójność wierszy i kolumn



Rysunek 1.2: Przykład poliomina ze spójnymi wierszami



Rysunek 1.3: Przykład wypukłego poliomina (to jest mającego spójne wiersze i kolumny)



Rysunek 1.4: Przykład wypukłego poliomina, które realizuje ustaloną parę rzutów prostopadłych (H, V)

Definicja 1.3 *Niech będą dane dwa wektory*

$$H = (h_1, \ldots, h_m) \in \{1, \ldots, n\}^m$$

oraz

$$V = (v_1, \ldots, v_n) \in \{1, \ldots, m\}^n.$$

Mówimy, że S realizuje (H, V) w klasie (x), jeśli $S \in (x)$ oraz $h_i(S) = h_i$ dla $i \in \{1, \ldots, m\}$ i $v_j(S) = v_j$ dla $j \in \{1, \ldots, n\}$.

Para (H, V) ma realizację w klasie (x), jeśli istnieje co najmniej jedna macierz R o m wierszach i n kolumnach taka, że $S = S(R)$ realizuje (H, V) w klasie (x).

W tej pracy skupimy się na rekonstrukcji poliomin wypukłych, czyli znalezieniu zbioru S realizującego parę prostopadłych rzutów (H, V) w klasie (p, h, v) lub stwierdzeniu, że taki zbiór nie istnieje.

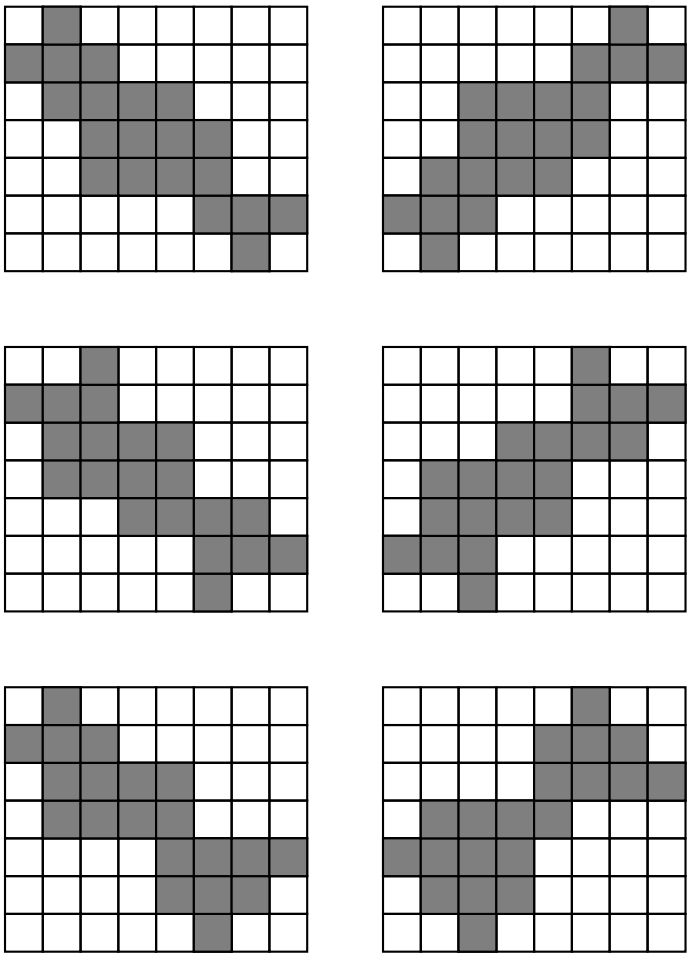
Głównym problemem, z którym spotkamy się przy rekonstrukcji poliomin, jest niejednoznaczność rozwiązania. W niektórych przypadkach istnieje bardzo dużo zbiorów mających takie same wektory rzutów ortogonalnych (H, V). W [dLNP96] pokazano, że może istnieć wykładniczo wiele takich wypukłych poliomin. Na Rysunku 1.5 pokazujemy przykład sześciu poliomin o takich samych rzutach.

1.2 Przegląd wyników

Pierwszy H. Ryser [R63], a następnie S.K. Chang [Ch71] i X.G. Wang [W75] studiowali istnienie zbioru S realizującego parę rzutów prostopadłych (H, V) bez żadnych dodatkowych warunków. Pokazali oni, że problem istnienia takiego zbioru może być rozwiązany w czasie $O(mn)$, gdzie m jest rozmiarem wektora rzutów poziomych a n – pionowych. Wymienieni autorzy wprowadzili również pierwsze algorytmy rekonstruujące zbiór S na podstawie jego rzutów prostopadłych (H, V).

Kolejny etap prac nad problemem rekonstrukcji nastąpił w latach dziewięćdziesiątych. G.J. Woeginger [W96] udowodnił, że problemy rekonstrukcji zbiorów, które mają spójne wiersze i kolumny (czyli należą do klasy (h, v)) oraz poliomin bez warunków na spójność wierszy i kolumn (klasa (p)) są NP-zupełne.

Z kolei w [BdLNP96a] E. Barcucci, A. Del Lungo, M. Nivat i R. Pinzani pokazali, że problem rekonstrukcji jest NP-zupełny również jeśli rekonstruowane zbiory są poliominami ze spójnymi kolumnami lub poliominami ze



Rysunek 1.5: Przykład sześciu różnych wypukłych poliomin o takich samych rzutach prostopadłych, poziomym $H = (1, 3, 4, 4, 4, 3, 1)$ i pionowym $V = (1, 3, 4, 3, 3, 3, 2, 1)$

spójnymi wierszami (odpowiednio klasy (p, v) i (p, h)), a również kiedy są tylko zbiorami ze spójnymi kolumnami lub, symetrycznie, spójnymi wierszami (klasy (v) i (h)). Stąd jedynie gdy rozważamy zbiory z wszystkimi trzema cechami (p, h, v), czyli wypukłe poliomina, problem rekonstrukcji może być rozwiązany w czasie wielomianowym.

Algorytm, który ustala istnienie wypukłego poliomina realizującego parę ustalonych wektorów prostopadłych (H, V) w czasie wielomianowym, został po raz pierwszy opisany w [BdLNP96a]. Główną ideą tego algorytmu było ustalenie pozycji początkowych dla pewnych zer i jedynek a następnie uruchamianie procedury wypełniania w oparciu o te pozycje. Liczba różnych początkowych ustawień wynosiła $O(m^2n^2)$, a procedura wypełniania miała złożoność $O(m^2n^2)$. Stąd cały algorytm wymagał czasu rzędu $O(m^4n^4)$. W rozdziale 3 pokazujemy wariant tego algorytmu, opisany w [G98], którego złożoność wynosi $O(\min(m, n)^2 \cdot mn \log mn)$. Najnowszy algorytm opisany w pracy [ChD99], którego autorami są M. Chrobak i Ch. Dürr, rekonstruuje wypukłe poliomina w czasie $O(\min(m, n)^2 \cdot mn)$. Główna jego idea polega na konstruowaniu formuł 2SAT, które są spełnialne wtedy i tylko wtedy, gdy istnieje wypukłe poliomino z pewnymi ustalonymi, początkowymi pozycjami jedynek.

W rozdziale 4 przedstawiamy algorytm rekonstrukcji pewnej klasy wypukłych poliomin trójwymiarowych, wykorzystujący opisane w tej pracy własności dwuwymiarowych poliomin wypukłych.

Wymienione powyżej wyniki dotyczą rekonstrukcji wypukłych poliomin w przypadku, gdy mamy dane dokładne rzuty. Czasami jednak rzuty są podane tylko z pewną dokładnością. W rozdziale 5 rozważamy więc rekonstrukcję poliomin z takimi właśnie przybliżonymi rzutami, opisaną częściowo w pracy [G01]. Pokazujemy, że jest to problem co najmniej tak trudny jak rekonstrukcja z dokładnych rzutów a w związku z tym dla poliomin bez żadnych warunków na spójność kolumn i wierszy, poliomin ze spójnymi wierszami i poliomin ze spójnymi kolumnami jest on NP-zupełny. Natomiast dla wypukłych poliomin podajemy algorytm ich rekonstrukcji z przybliżonych rzutów pracujący w czasie $O(m^3n^3)$. Wyniki te rozwiązują problem postawiony przez G.J. Woeginger w [GN97]. Dodatkowo przedstawiamy równoległą wersję tego algorytmu, z której wynika, że problem należy do klasy NC_1. Ponieważ algorytm ten działa także dla dokładnych danych, więc problem rekonstrukcji poliomin wypukłych z dokładnych rzutów prostopadłych też należy do klasy NC_1.

Rozdział 2

Pewne własności poliomin wypukłych

W tym rozdziale udowodnimy pewne własności poliomin wypukłych, które będą pomocne w algorytmach rekonstrukcji zawartych w dalszej części tej pracy.

2.1 Ogólne własności

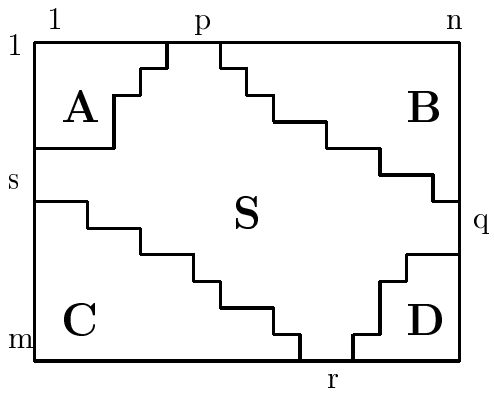
Na początek zdefiniujemy rozłączne obszary macierzy R z wpisanym do niej wypukłym poliominem S.

Definicja 2.1 *Dla wypukłego poliomina $S(R)$ zbiór elementów równych 0 w macierzy R rozpada się na spójne, rozłączne obszary zawierające rogi macierzy. Przez A oznaczamy spójny obszar wypełniony zerami w lewym górnym rogu R, to znaczy, że $[1,1] \in A$, jeśli $R[1,1] = 0$ lub $A = \emptyset$.*

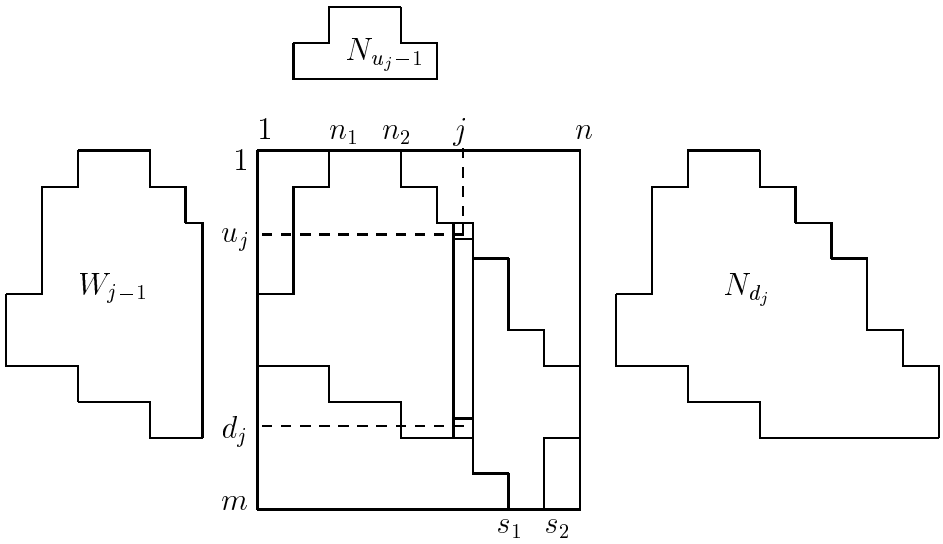
Analogicznie oznaczmy jako B obszar zer w prawym górnym rogu macierzy R (jeśli $R[1,n] = 0$, to $[1,n] \in B$, a w p.p. $B = \emptyset$), jako C obszar zer w lewym dolnym rogu R (jeśli $R[m,1] = 0$, to $[m,1] \in C$, a w p.p. $C = \emptyset$) a jako D obszar zer w prawym dolnym rogu R (jeśli $R[m,n] = 0$, to $[m,n] \in D$, a w p.p. $D = \emptyset$).

Teraz dla tak zdefiniowanych obszarów narożnych można sformułować następujący lemat (dowód jego jest oczywisty więc go pomijamy):

Lemat 2.2 *Dla narożnych obszarów zer wypukłego poliomina S w macierzy R spełnione są następujące implikacje*



Rysunek 2.1: Wypukłe poliomino S zakotwiczone w (p, q, r, s), z narożnymi obszarami zer A, B, C i D



Rysunek 2.2: Ilustracja własności wypukłego poliomina z Lematów 2.5 i 2.6. Mamy następujące zawierania: $N_{u_{j-1}} \subset W_{j-1}$ i $W_j \subset N_{d_j}$, gdzie W_j jest równy sumie W_{j-1} i zbioru jedynek w kolumnie j

1. *Jeśli* $[i+1,j] \in A$ *lub* $[i,j+1] \in A$, *to* $[i,j] \in A$.
2. *Jeśli* $[i+1,j] \in B$ *lub* $[i,j-1] \in B$, *to* $[i,j] \in B$.
3. *Jeśli* $[i-1,j] \in C$ *lub* $[i,j+1] \in C$, *to* $[i,j] \in C$.
4. *Jeśli* $[i-1,j] \in D$ *lub* $[i,j-1] \in D$, *to* $[i,j] \in D$.

□

Z definicji wypukłego poliomina i powyższego lematu otrzymujemy

Lemat 2.3 *[ChD99] Zbiór S jest wypukłym poliominem wtedy i tylko wtedy, gdy*

$$S = Q \setminus (A \cup B \cup C \cup D),$$

gdzie A, B, C i D są rozłącznymi obszarami narożnymi (odpowiednio lewym górnym, prawym górnym, lewym dolnym i prawym dolnym) spełniającymi własności z Lematu 2.2 oraz takimi, że

(i) jeśli $[i-1,j-1] \in A$, *to* $[i,j] \notin D$,

(ii) jeśli $[i-1,j+1] \in B$, *to* $[i,j] \notin C$.

□

Teraz zajmiemy się własnościami poliomin wypukłych przydatnymi do ustalania położenia pewnych należących do nich elementów. Własności te będą przydatne w algorytmach rekonstrukcji, opisanych w następnych rozdziałach, do wyznaczania pozycji niektórych jedynek w macierzy R. Wszystkie opisane poniżej własności zachodzą również, gdy słowo wiersz zastąpimy kolumną a kolumna – wierszem. Dla większej czytelności pomijamy zapis tej drugiej wersji.

Definicja 2.4 *Dla wypukłego poliomina S niech* $\langle n_1, n_2 \rangle$ *oznacza pozycje jedynek w pierwszym wierszu macierzy R, to znaczy* $R[1,j] = 1$ *wtedy i tylko wtedy, gdy* $n_1 \leqslant j \leqslant n_2$.

Analogicznie, niech $\langle s_1, s_2 \rangle$ *oznacza pozycje jedynek w ostatnim wierszu macierzy R, to znaczy* $R[m,j] = 1$ *wtedy i tylko wtedy, gdy* $s_1 \leqslant j \leqslant s_2$.

Pozycje tych elementów nazywamy w skrócie odpowiednio górnymi i dolnymi pozycjami brzegowymi.

W poniższym lemacie niech W_j oznacza zbiór jedynek w pierwszych j kolumnach macierzy R, czyli

$$W_j = \{\ [i,k]\ :\ R[i,k] = 1\ \wedge\ k \leqslant j\ \},$$

oraz niech N_i oznacza zbiór jedynek w pierwszych i wierszach R, czyli

$$N_i = \{\ [k,j]\ :\ R[k,j] = 1\ \wedge\ k \leqslant i\ \}$$

(zobacz rysunek 2.2). Ponadto, niech u_j i d_j będą odpowiednio najmniejszym i największym indeksem takim, że $R[u_j, j] = 1$ oraz $R[d_j, j] = 1$. Wtedy następujący lemat jest prawdziwy

Lemat 2.5 *Niech S będzie wypukłym poliominem z górnymi pozycjami brzegowymi $\langle n_1, n_2\rangle$. Wtedy dla wszystkich $j \in \{n_1, \ldots, n\}$, to jest numerów kolumn za początkiem górnych pozycji brzegowych w wypukłym poliominie, prawdziwe jest następujące zawieranie*

$$N_{u_j - 1} \subset W_{j-1}.$$

Dowód: Załóżmy, że $[i,k] \in N_{u_j-1}$ dla pewnego j takiego, że $n_1 \leqslant j \leqslant n$. Chcemy pokazać, że $k < j$. Jak łatwo zauważyć, idąc z lewej do prawej, począwszy od kolumny n_1, współrzędne pierwszego elementu równego 1 w kolumnach tworzą ciąg niemalejący (z definicji obszaru narożnego B i Lematu 2.2). Stąd, dla $k \geqslant j$ zachodzi $u_k \geqslant u_j$. Lecz jest to niemożliwe, ponieważ u_k jest z definicji najmniejszym indeksem takim, że $R[u_k, k] = 1$, więc $u_k \leqslant i$. Z założenia $[i,k] \in N_{u_j-1}$ mamy jednak, że $i < u_j$. Dlatego $u_k \leqslant i < u_j$ a więc $k < j$. □

Lemat 2.6 *Niech S będzie wypukłym poliominem z dolnymi pozycjami brzegowymi $\langle s_1, s_2\rangle$. Wtedy dla wszystkich $j \in \{1, \ldots, s_2\}$, to jest numerów kolumn mniejszych od dolnych pozycji brzegowych w wypukłym poliominie, prawdziwe jest następujące zawieranie*

$$W_j \subset N_{d_j}.$$

Dowód: Załóżmy, że $[i,k] \in W_j$ dla j takiego, że $1 \leqslant j \leqslant s_2$. Chcemy pokazać, że $i \leqslant d_j$. Jak łatwo zauważyć, idąc od pierwszej kolumny do kolumny s_2, współrzędne ostatniego elementu równego 1 w kolumnach tworzą ciąg niemalejący (z definicji obszaru narożnego C i Lematu 2.2). Dla $i > d_j$ zachodzi $d_k \geqslant i$. Stąd, jeśli $i > d_j$, to $d_k > d_j$. Ale to zachodzi tylko wtedy, gdy $k > j$. Sprzeczność z założeniem, że $[i,k] \in W_j$, a więc $k \leqslant j$. Dlatego $i \leqslant d_j$. □

Teraz zdefiniujemy pomocnicze wartości będące sumami częściowymi dla wektorów rzutów.

Definicja 2.7 *Wprowadzamy następujące oznaczenia:*

$$H_k = \sum_{j=1}^{k} h_j,$$

dla $k \in \{1, \ldots, m\}$ *i* $H_0 = 0$, *oraz*

$$V_k = \sum_{i=1}^{k} v_i,$$

dla $k \in \{1, \ldots, n\}$ *i* $V_0 = 0$.

Fakt 2.8 Równość

$$\sum_{j=1}^{m} h_j = \sum_{i=1}^{n} v_i \qquad \text{(to jest } H_m = V_n\text{)}$$

jest warunkiem koniecznym istnienia poliomina.

Z powyższych lematów i ich wariantów otrzymujemy następujące wnioski:

Wniosek 2.9 *Niech* S *będzie wypukłym poliominem z górnymi pozycjami brzegowymi* $\langle n_1, n_2 \rangle$. *Wtedy dla wszystkich* $j \in \{1, \ldots, n_2\}$ *spełnione są nierówności*

$$H_{u_j - 1} \leqslant V_n - V_j \qquad i \qquad H_m - H_{u_j - 1} \geqslant V_j.$$

□

Wniosek 2.10 *Niech* S *będzie wypukłym poliominem z górnymi pozycjami brzegowymi* $\langle n_1, n_2 \rangle$. *Wtedy dla wszystkich* $j \in \{n_1 + 1, \ldots, n\}$ *spełnione są nierówności*

$$H_{u_j - 1} \leqslant V_{j-1} \qquad i \qquad H_m - H_{u_j - 1} \geqslant V_n - V_{j-1}.$$

□

Wniosek 2.11 *Niech* S *będzie wypukłym poliominem z dolnymi pozycjami brzegowymi* $\langle s_1, s_2 \rangle$. *Wtedy dla wszystkich* $j \in \{1, \ldots, s_2\}$ *spełnione są nierówności*

$$H_{d_j} \leqslant V_j \qquad i \qquad H_m - H_{d_j} \geqslant V_n - V_j.$$

□

Wniosek 2.12 *Niech S będzie wypukłym poliominem z dolnymi pozycjami brzegowymi $\langle s_1, s_2\rangle$. Wtedy dla wszystkich $j \in \{s_1, \ldots, n\}$ spełnione są nierówności*

$$H_{d_j} \leqslant V_n - V_{j-1} \qquad i \qquad H_m - H_{d_j} \geqslant V_{j-1}.$$

□

Na koniec sformułujemy lemat wyznaczający elementy $[i, j]$ zbioru Q, które na pewno będą należały do wypukłego poliomina S, czyli takie, że $R[i, j] = 1$.

Lemat 2.13 *[dLNPS98] Niech S będzie poliominem wypukłym. Wtedy, jeśli i, j spełniają nierówności*

$$V_j \geqslant H_{i-1}, \qquad H_i \geqslant V_{j-1}, \qquad H_i \geqslant V_n - V_j \quad i \quad V_n - V_{j-1} \geqslant H_{i-1},$$

to $R[i, j] = 1$ (taki element $[i, j]$ nazywamy medianą poliomina). □

Liczba elementów będących medianą poliomina wypukłego jest ograniczona i w zależności od własności rzutów może wynosić odpowiednio 1, 2 lub 4.

2.2 Oszacowania

Teraz wykorzystamy niektóre z lematów i wniosków z poprzedniego podrozdziału do stworzenia narzędzia pomocnego w oszacowaniu niektórych pozycji elementów równych 1 w macierzy R opisującej poliomino wypukłe.

Definicja 2.14 *Dla $j \in \{1, \ldots, n\}$ zdefiniujmy następujące wielkości:*

$$\begin{aligned} D_j^{\searrow} &= \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_m - H_i < H_m - V_j \\ &\qquad\qquad \vee \quad i = m\} \end{aligned}$$

oraz

$$U_j^{\searrow} = \max\{\, i \in \{2, \ldots, m\} \;:\; H_{i-1} < V_{j-1} \quad \vee \quad i = 1\}.$$

Obszar zawarty między pozycjami $U_j^{\searrow}$ a $D_j^{\searrow}$ (elementy $[j, U_j^{\searrow}], \ldots, [j, D_j^{\searrow}]$) dla $j \in \{1, \ldots, n\}$ nazywamy obszarem oszacowań wzdłuż przekątnej.

Analogicznie dla $j \in \{1, \ldots, n\}$ zdefiniujmy następujące wielkości:

$$D_j^{\swarrow} = \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_m - H_i < V_{j-1} \quad \vee \quad i = m\}$$

oraz

$$U_j^{\swarrow} = \max\{\, i \in \{2, \ldots, m\} \;:\; H_{i-1} < H_m - V_j \quad \vee \quad i = 1\}.$$

Obszar zawarty między pozycjami $U_j^{\swarrow}$ *a* $D_j^{\swarrow}$ *(elementy* $[j, U_j^{\swarrow}], \ldots, [j, D_j^{\swarrow}]$*) dla* $j \in \{1, \ldots, n\}$ *nazywamy obszarem oszacowań wzdłuż wstecznej przekątnej.*

Powyżej zdefiniowane wartości są dla poszczególnych kolumn oszacowaniami dolnej i górnej granicy występowania jedynek w wypukłych poliominach zgodnie z Wnioskami 2.9–2.12. Jednak są to tylko przybliżenia i nie na całym obszarze wyznaczonym przez te oszacowania występują jedynki. Poniżej przedstawiamy pewne własności tych oszacowań, które czynią z nich pomocne narzędzie w rekonstrukcji poliomin wypukłych.

Lemat 2.15 *Dla wszystkich* $j \in \{1, \ldots, n\}$ *mamy*

$$U_j^{\searrow} \leqslant D_j^{\searrow} \qquad i \qquad U_j^{\swarrow} \leqslant D_j^{\swarrow},$$

czyli oszacowanie elementu równego 1 o najniższym numerze w danej kolumnie jest niewiększe niż oszacowanie jedynki o najwyższym numerze w tej kolumnie.

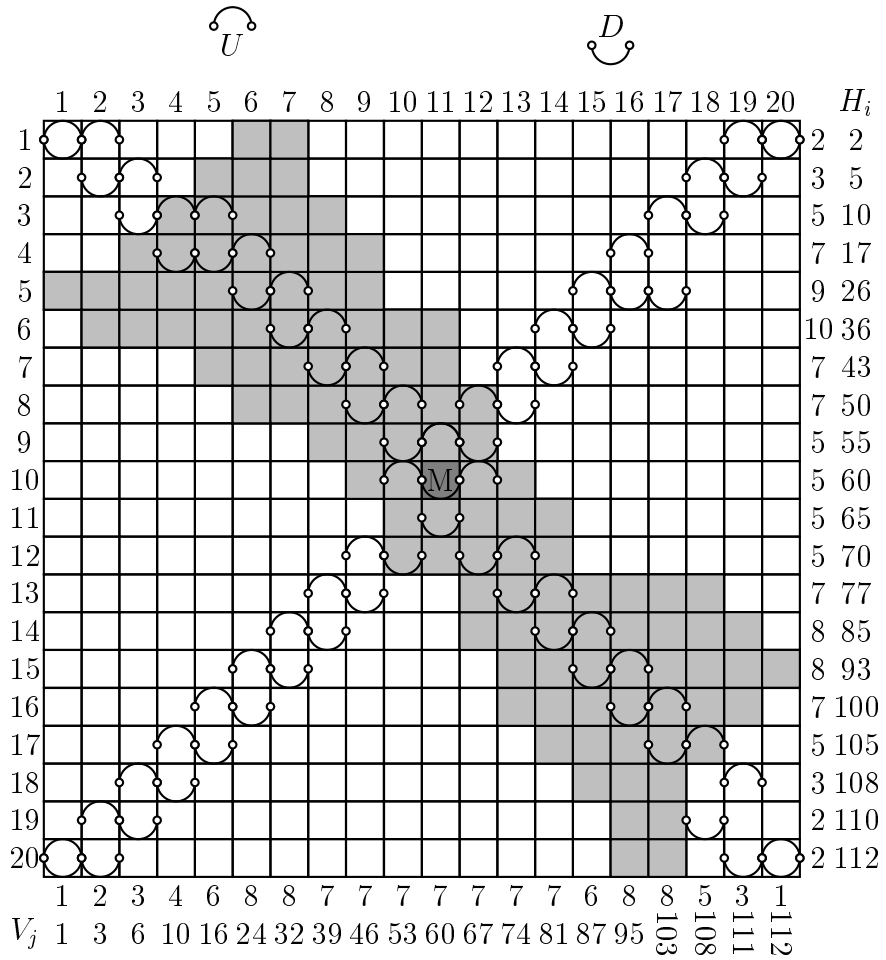
Dowód: Zgodnie z Definicją 2.14 dla przekątnej mamy

$$\begin{aligned}
U_j^{\searrow} &= \max\{\, i \in \{2, \ldots, m\} \;:\; H_{i-1} < V_{j-1} \,\} \\
&= \max\{\, i \in \{1, \ldots, m-1\} \;:\; H_i < V_{j-1} \,\} + 1 \\
&\leqslant \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_i > V_{j-1} \,\} \\
&\leqslant \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_i > V_{j-1} + v_j \,\} \\
&= \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_i > V_j \,\} \\
&= \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_m - H_i < V_n - V_j \,\} \\
&= D_j^{\searrow}.
\end{aligned}$$

Analogicznie, dla wstecznej przekątnej mamy odpowiednio

$$\begin{aligned}
U_j^{\swarrow} &= \max\{\, i \in \{2, \ldots, m\} \;:\; H_{i-1} < V_n - V_j \,\} \\
&= \max\{\, i \in \{1, \ldots, m-1\} \;:\; H_i < V_n - V_j \,\} + 1 \\
&\leqslant \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_i > V_n - V_j \,\} \\
&\leqslant \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_i > V_n - V_j + v_j \,\} \\
&= \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_i > V_n - V_{j-1} \,\} \\
&= \min\{\, i \in \{1, \ldots, m-1\} \;:\; H_m - H_i < V_{j-1} \,\} \\
&= D_j^{\swarrow}.
\end{aligned}$$

□



Rysunek 2.3: Poliomino, odpowiadające mu oszacowania przekątniowe i mediana (M)

Ponadto obszary oszacowań w sąsiednich kolumnach nie są od siebie odległe. Każdy wiersz zawiera komórkę należącą do oszacowań.

Lemat 2.16 *Dla wszystkich $j \in \{1, \ldots, n-1\}$ i dla oszacowań wzdłuż przekątnej mamy*

$$D_j^{\searrow} + 1 \geqslant U_{j+1}^{\searrow}.$$

A dla wszystkich $j \in \{1, \ldots, n-1\}$ i dla oszacowań wzdłuż wstecznej przekątnej mamy

$$D_{j+1}^{\swarrow} + 1 \geqslant U_j^{\swarrow}.$$

Dowód: Pierwsza nierówność:

$$\begin{aligned}
U_{j+1}^{\searrow} &= \max\{\ i \in \{2, \ldots, m\}\ :\ H_{i-1} < V_j\ \} \\
&= \max\{\ i \in \{1, \ldots, m-1\}\ :\ H_i < V_j\ \} + 1 \\
&= \max\{\ i \in \{1, \ldots, m-1\}\ :\ H_m - H_i > V_n - V_j\ \} + 1 \\
&\leqslant \min\{\ i \in \{1, \ldots, m-1\}\ :\ H_m - H_i < V_n - V_j\ \} + 1 \\
&= D_j^{\searrow} + 1.
\end{aligned}$$

Oraz druga nierówność:

$$\begin{aligned}
U_j^{\swarrow} &= \max\{\ i \in \{2, \ldots, m\}\ :\ H_{i-1} < V_m - V_j\ \} \\
&= \max\{\ i \in \{1, \ldots, m-1\}\ :\ H_i < V_m - V_j\ \} + 1 \\
&= \max\{\ i \in \{1, \ldots, m-1\}\ :\ H_m - H_i > V_j\ \} + 1 \\
&\leqslant \min\{\ i \in \{1, \ldots, m-1\}\ :\ H_m - H_i < V_j\ \} + 1 \\
&= D_{j+1}^{\swarrow} + 1.
\end{aligned}$$

□

Teraz dodatkowo podamy lemat o poprawności oszacowań wzdłuż przekątnych.

Lemat 2.17 *Obszar oszacowań wzdłuż przekątnej jest rozłączny z obszarem narożnym B i z obszarem narożnym C.*

Analogicznie obszar oszacowań wzdłuż wstecznej przekątnej jest rozłączny z obszarem narożnym A i z obszarem narożnym D.

Dowód: Pierwszej część lematu wynika bezpośrednio z Wniosków 2.10 i 2.11 oraz z Definicji 2.14.

Analogicznie druga część wynika z Wniosków 2.9 i 2.12 oraz Definicji 2.14. □

Poniżej sformułowany lemat ma kluczowe znaczenie dla rekonstrukcji wypukłych poliomin umożliwiając łączenie różnych wyznaczonych obszarów jedynek. Niech podkratą rozpiętą między elementami $[i,j]$ i $[i',j']$, będzie zbiór tych elementów $[k,l]$, dla których $\min\{i,i'\} \leqslant k \leqslant \max\{i,i'\}$ i $\min\{j,j'\} \leqslant l \leqslant \max\{j,j'\}$.

Lemat 2.18 *Niech S będzie wypukłym poliominem i niech $[i,j]$ będzie jego medianą. Wtedy, jeśli $[i',j']$ należy do S, to oszacowania wzdłuż przekątnej (jeśli $i' < i \ \wedge \ j' < j$ lub $i' > i \ \wedge \ j' > j$) lub wzdłuż wstecznej przekątnej (jeśli $i' > i \ \wedge \ j' < j$ lub $i' < i \ \wedge \ j' > j$) zawarte w podkracie Q', rozpiętej między elementami $[i',j']$ i $[i,j]$, należą do poliomina S.*

Dowód: Po pierwsze zauważmy, że mediana należy do obszaru obu oszacowań przekątniowych i do poliomina S.

Na początek załóżmy, że $i' < i$ i $j' < j$. Wtedy w podkracie Q', rozpiętej między elementami $[i',j']$ i $[i,j]$, zawarty będzie fragment oszacowania wzdłuż przekątnej. Z Lematu 2.17 oszacowania te są rozłączne z obszarami narożnymi B i C. Ponadto obszar podkraty Q' jest również rozłączny z obszarami narożnymi A i D. Gdyby bowiem do podkraty Q' należał element z obszaru narożnego A, to zgodnie z Lematem 2.2 $[i',j']$ również musiałby należeć do obszaru A. A to jest sprzeczne z założeniem, że należy do poliomina S. Analogicznie, gdyby w podkracie Q' istniał element należący do obszaru narożnego D, to zgodnie z Lematem 2.2 $[i,j]$ również musiałby należeć do obszaru D, a wiemy, że należy do poliomina S. Stąd sprzeczność.

W pozostałych trzech przypadkach, tj. gdy $i < i'$ i $j' < j$ albo $i' < i$ i $j < j'$ albo $i < i'$ i $j < j'$, dowód przebiega podobnie. □

Zauważmy jeszcze, że jeśli $i' = i$ lub $j' = j$ to wszystkie elementy między $[i',j']$ a $[i,j]$ należą do S, ponieważ S z definicji ma spójne obszary jedynek w kolumnach i wierszach.

Z powyższego lematu oraz z Lematów 2.15 i 2.16, a także ze spójności obszarów jedynek w wierszach i kolumnach wypukłego poliomina, otrzymujemy następujący wniosek

Wniosek 2.19 *Niech S będzie wypukłym poliominem i niech $[i,j]$ będzie jego medianą. Wtedy, jeśli $[i',j']$ należy do S, to w każdym wierszu i każdej kolumnie podkraty rozpiętej między elementami $[i,j]$ i $[i',j']$ możemy wyznaczyć pozycję co najmniej jednego elementu należącego do S (są to elementy odpowiednich obszarów oszacowań).*

Dowód: Bezpośrednio z Lematów 2.15 i 2.16 oraz z własności spójności obszaru jedynek dla kolumn i wierszy. □

Lemat 2.20 *Niech S będzie wypukłym poliominem i niech $[i, j]$ będzie jego medianą. Wtedy, jeśli element $[i', j']$ nie należy do S, a należy do obszaru oszacowań wzdłuż przekątnej, to*

1. *jeśli $i' \leqslant i$ oraz $j' \leqslant j$, to podkrata rozpięta między $[1, 1]$ a $[i', j']$ jest rozłączna z S;*

2. *jeśli $i' \geqslant i$ oraz $j' \geqslant j$, to podkrata rozpięta między $[i', j']$ a $[m, n]$ jest rozłączna z S.*

Analogicznie, jeśli element $[i', j']$ nie należy do S, a należy do obszaru oszacowań wzdłuż wstecznej przekątnej, to

1. *jeśli $i' \leqslant i$ oraz $j' \geqslant j$, to podkrata rozpięta między $[1, n]$ a $[i', j']$ jest rozłączna z S;*

2. *jeśli $i' \geqslant i$ oraz $j' \leqslant j$, to podkrata rozpięta między $[i', j']$ a $[m, 1]$ jest rozłączna z S.*

Dowód: Na początek załóżmy, że $i' < i$ i $j' < j$. Z Lematu 2.17 wiemy, że element $[i', j']$ nie należy do obszarów narożnych B i C. Gdyby należał do obszaru narożnego D, to zgodnie z Lematem 2.2 element $[i, j]$ też należałoby do obszaru D, ale to jest sprzeczne z założeniem, że $[i, j]$ jest medianą. Tak więc $[i', j']$ musi należeć do obszaru narożnego A, a zgodnie z Lematem 2.2 wszystkie komórki z podkraty rozpiętej między $[1, 1]$ a $[i', j']$ należą do obszaru A, a więc nie należą do poliomina S.

Pozostałe trzy przypadki dowodzimy podobnie. □

2.3 Pozycje brzegowe

Teraz udowodnimy kilka ciekawych własności pozycji brzegowych w poliominach wypukłych.

Definicja 2.21 *Dla wypukłego poliomina S z wektorem rzutów poziomych $H = \{h_1, \ldots, h_m\} \in \{1, \ldots, n\}^m$ definiujemy*

$$c \;=\; \max\{\; h_i \;:\; i \in \{1, \ldots, m\} \;\}.$$

Liczba c jest maksymalną liczbą jedynek w wierszu macierzy R.

Lemat 2.22 *Niech S będzie wypukłym poliominem realizującym daną parę prostopadłych wektorów (H, V) i posiadającym odpowiednio pozycje brzegowe $\langle n_1, n_2 \rangle$ i $\langle s_1, s_2 \rangle$. Wtedy następujące implikacje są prawdziwe*

1. *Jeśli* $n_2 < s_1$, *to* $n_2 \leqslant c$ *i* $s_1 \geqslant n - c + 1$.

2. *Jeśli* $s_2 < n_1$, *to* $s_2 \leqslant c$ *i* $n_1 \geqslant n - c + 1$.

Dowód: Udowodnimy tylko pierwszą część pierwszej implikacji. Pozostałe implikacje dowodzi się podobnie. Przypuśćmy, że $n_2 = c + 1$. Wtedy w kolumnie n_2 mamy najpierw v_{n_2} jedynek a następnie pomiędzy $v_{n_2} + 1$ a m mamy zera. Z Lematu 2.3 wynika, że w pierwszej kolumnie na pozycjach od $v_{n_2} + 1$ do m również muszą być zera. Stąd blok jedynek w pierwszej kolumnie musi znajdować się powyżej wiersza $v_{n_2}+1$. Lecz odległość między pierwszą a n_2-ą kolumną wynosi $c+1$, a każdy wiersz zawiera co najwyżej c jedynek w spójnym bloku. Daje to sprzeczność z warunkiem, że to wypukłe poliomino. Dlatego $n_2 \leqslant c$. □

Definicja 2.23 *Dla wypukłego poliomina* S *z wektorem rzutów pionowych* $V = \{v_1, \ldots, v_n\} \in \{1, \ldots, m\}^n$ *definiujemy*

$$l = \min\{\ j\ :\ v_j > v_{j+1}\ \textit{dla}\ j \in \{1, \ldots, n\}\ \},$$

$$r = \max\{\ j\ :\ v_{j-1} < v_j\ \textit{dla}\ j \in \{1, \ldots, n\}\ \}.$$

Ciąg $v_1, v_2, \ldots, v_l$ *jest niemalejący i nie ma dłuższego takiego ciągu zaczynającego się od* v_1. *A ciąg* $v_r, v_{r+1}, \ldots, v_n$ *jest nierosnący i nie ma dłuższego takiego ciągu kończącego się na* v_n.

Lemat 2.24 *Dla wypukłego poliomina* S, *realizującego parę prostopadłych wektorów* (H, V) *i posiadającego pozycje brzegowe* $\langle n_1, n_2\rangle$ *i* $\langle s_1, s_2\rangle$, *następujące implikacje są prawdziwe*

1. *Jeśli* $n_2 < s_1$, *to* $n_2 \leqslant l$ *i* $s_1 \geqslant r$.

2. *Jeśli* $s_2 < n_1$, *to* $s_2 \leqslant l$ *i* $n_1 \geqslant r$.

Dowód: Podobnie jak w poprzednim lemacie udowodnimy tylko pierwszą część pierwszej implikacji. Przypuśćmy, że istnieje wypukłe poliomino takie, że $n_2 = l+1$. Wtedy w kolumnie n_2 mamy najpierw v_{n_2} jedynek a następnie pomiędzy pozycją $v_{n_2} + 1$ a m mamy zera. Z Lematu 2.3 wynika, że w kolumnie $l = n_2 - 1$ na pozycjach od $v_{n_2} + 1$ do m również są zera. Jednak z definicji l mamy $v_l > v_{l+1} = v_{n_2}$, więc wolny obszar, w którym mogą być jedynki, jest mniejszy niż liczba jedynek. Czyli takie wypukłe poliomino nie istnieje.

Pozostałe implikacje dowodzimy podobnie. □

Definicja 2.25 *Dla wypukłego poliomina S z wektorem rzutów poziomych $H = \{h_1, \ldots, h_m\} \in \{1, \ldots, n\}^m$ dla wszystkich $j \in \{1, \ldots, m\}$ definiujemy*

$$\begin{aligned} \hat{c}_j &= \max\{\ h_i\ :\ i \in \{1, \ldots, j\}\ \} \qquad i \\ \check{c}_j &= \max\{\ h_i\ :\ i \in \{n-j+1, \ldots, m\}\ \}. \end{aligned}$$

$\hat{c}_j$ jest maksymalną liczbą jedynek w wierszu dla pierwszych j wierszy a $\check{c}_j$ jest maksymalną liczbą jedynek w wierszu dla ostatnich j wierszy (to jest od $n-j+1$-go do ostatniego wiersza).

Lemat 2.26 *Dla wypukłego poliomina S realizującego parę wektorów (H, V) i posiadającego pozycje brzegowe $\langle n_1, n_2\rangle$ i $\langle s_1, s_2\rangle$, następujące implikacje są prawdziwe*

1. *Jeśli $n_2 < s_1$, to $n_2 \leqslant \hat{c}_{v_{n_2}}$ i $s_1 \geqslant n - \check{c}_{v_{s_1}} + 1$.*
2. *Jeśli $s_2 < n_1$, to $s_2 \leqslant \check{c}_{v_{s_2}}$ i $n_1 \geqslant n - \hat{c}_{v_{n_1}} + 1$.*

Dowód: To jest oczywisty wniosek z Lematu 2.22 i Lematu 2.24. □

Definicja 2.27 *Dla wypukłego poliomina S definiujemy*

$$\begin{aligned} \hat{l} &= \max\{\ j\ :\ j \leqslant \hat{c}_{v_j} \text{ dla } j \in \{1, \ldots, l\}\ \}, \\ \check{r} &= \min\{\ j\ :\ j \geqslant n + 1 - \check{c}_{v_j} \text{ dla } j \in \{r, \ldots, n\}\ \}, \\ \check{l} &= \min\{\ j\ :\ j \leqslant \check{c}_{v_j} \text{ dla } j \in \{1, \ldots, l\}\ \} \quad i \\ \hat{r} &= \max\{\ j\ :\ j \leqslant n + 1 - \check{c}_{v_j} \text{ dla } j \in \{r, \ldots, n\}\ \}. \end{aligned}$$

Wniosek 2.28 *Dla wypukłego poliomina S realizującego parę wektorów (H, V) i posiadającego pozycje brzegowe $\langle n_1, n_2\rangle$ i $\langle s_1, s_2\rangle$, następujące implikacje są prawdziwe*

1. *Jeśli $n_2 < s_1$, to $n_2 \leqslant \hat{l}$ i $s_1 \geqslant \check{r}$.*
2. *Jeśli $s_2 < n_1$, to $s_2 \leqslant \check{l}$ i $n_1 \geqslant \hat{r}$.*

Dowód: Bezpośrednio z Lematu 2.24 i Lematu 2.26. □

Lemat 2.29 *Załóżmy, że w wypukłym poliominie S dla każdego wiersza i, $i \in \{1, \ldots, m\}$, jego rzut poziomy $h_i(S) > \lfloor \frac{n}{2} \rfloor$. Wtedy dla co najmniej jednego j, $j \in \{1, \ldots, n\}$, $h_j(S) = m$, to znaczy istnieje kolumna zawierająca tylko jedynki.*

Dowód: W każdym wierszu mamy spójny zbiór jedynek, którego rozmiar wynosi co najmniej $\lfloor \frac{n}{2} \rfloor + 1$. Możemy umieścić ten blok (początek bloku) na co najwyżej $n - \lfloor \frac{n}{2} \rfloor - 1$ początkowych pozycjach w wierszu. Lecz wtedy na pozycji $\lfloor \frac{n}{2} \rfloor + 1$ zawsze mamy jedynkę. Stąd przynajmniej dla $j = \lfloor \frac{n}{2} \rfloor + 1$ mamy $v_j = m$. □

Rozdział 3

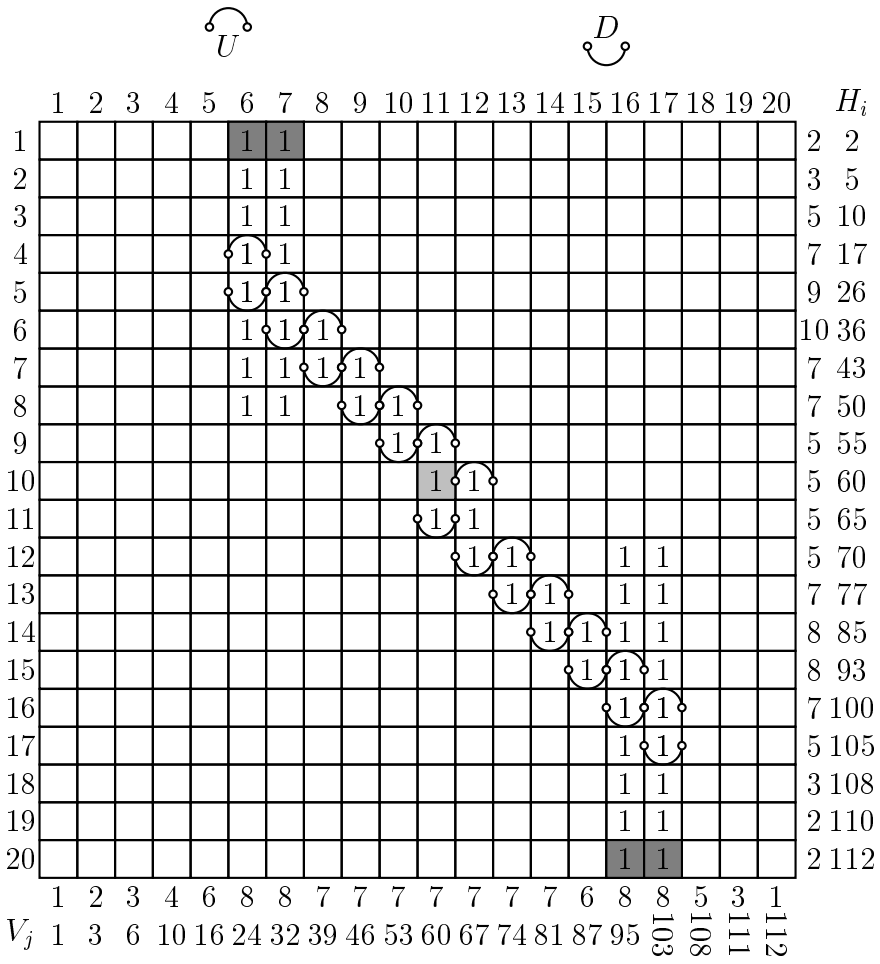
Rekonstrukcja wypukłych poliomin z dokładnych prostopadłych rzutów

W tym rozdziale zdefiniujemy algorytm, który sprawdza istnienie wypukłego poliomina realizującego parę danych wektorów (H, V) – prostopadłych rzutów, odpowiednio poziomego i pionowego. Nasz algorytm ma dwa etapy. W pierwszym etapie wybieramy początkowe pozycje dla pewnego zbioru jedynek. W drugim etapie uruchamiamy procedurę wypełniania, która próbuje odtworzyć wypukłe poliomino zawierające ustalone wcześniej jedynki i realizujące wejściową parę wektorów (H, V). Oba etapy powtarzamy dla wszystkich możliwych pozycji startowych, których liczba jest ograniczona. Złożoność algorytmu jest więc równa iloczynowi liczby testów i złożoności procedury wypełniania.

3.1 Pozycje startowe

W naszym algorytmie będziemy testować wszystkie możliwe pozycje jedynek w pierwszym i ostatnim wierszu macierzy R, to jest pozycje brzegowe. Jeśli pozycje brzegowe będą już ustalone, to użyjemy Wniosku 2.19 z poprzedniego rozdziału do obliczenia pozycji niektórych jedynek w kolumnach pomiędzy pozycjami brzegowymi. Naszym celem jest wstawienie co najmniej jednej jedynki w każdym wierszu macierzy R przed uruchomieniem procedury wypełniania. Ma to zapewnić prawidłową pracę tej procedury.

Lemat 3.1 *Dla każdych pozycji brzegowych wypukłego poliomina można*



Rysunek 3.1: Przykład oszacowania pozycji początkowych: stopki, mediana i odpowiadające im oszacowania wzdłuż przekątnej

wstawić jedynki do macierzy R w taki sposób, aby w każdym wierszu macierzy była co najmniej jedna jedynka.

Dowód: Ustalmy n_1 i s_1. Wtedy $\langle n_1, n_2\rangle$, gdzie $n_2 = n_1 + h_1 - 1$, i $\langle s_1, s_2\rangle$, gdzie $s_2 = s_1 + h_m - 1$, są pozycjami brzegowymi. Do wszystkich kolumn wyznaczonych przez te pozycje wstawiamy wszystkie zera i jedynki zgodnie z wartościami odpowiednich rzutów pionowych. Dodatkowo, zgodnie z Lematem 2.13, mamy wyznaczoną medianę. Teraz, korzystając z Wniosku 2.19, uzupełniamy jedynkami obszar macierzy R między brzegami a medianą. Stąd mamy co najmniej jedną jedynkę w każdym wierszu macierzy. □

Pozycje wszystkich obliczonych zgodnie z powyższym lematem jedynek będziemy nazywać *pozycjami startowymi.*

Lemat 3.2 *Jeśli istnieje wypukłe poliomino S, które realizuje parę wektorów (H, V) i posiada pozycje brzegowe $\langle n_1, n_2\rangle$ i $\langle s_1, s_2\rangle$, wtedy wszystkie wstawiane do macierzy R jedynki na pozycjach startowych należą do S.*

Dowód: Z Wniosku 2.19. □

Lemat 3.3 *Obliczenie pozycji wstawianych jedynek wymaga co najwyżej $0(m + n)$ kroków.*

Dowód: Wszystkie obliczenia wielkości oszacowań z Definicji 2.14 oraz wyznaczenie mediany wymagają tylko $O(m+n)$ kroków, ponieważ wyznaczane ciągi są słabo monotoniczne i można je policzyć w czasie liniowym. □

Koszt wyznaczania pozycji startowych jest mniejszy niż złożoność opisanej niżej procedury wypełniania. Stąd to procedura zdeterminuje złożoność całego algorytmu.

3.2 Procedura wypełniania

W procedurze opisanej poniżej będziemy używać zrównoważonych drzew binarnych (jak na przykład drzewa AVL) z opisanymi niżej operacjami, o podanej złożoności obliczeniowej:

empty(*tree*) — funkcja zwracająca *true*, czyli prawdę, jeśli *tree* jest pustym drzewem lub $false$, czyli fałsz, w przeciwnym przypadku. Jej wywołanie zawsze kosztuje $O(1)$.

delete(k, $tree$) — procedura usuwająca element k z drzewa $tree$. Złożoność tej procedury wynosi $O(\log |tree|)$, gdzie $|tree|$ oznacza rozmiar drzewa $tree$ (to jest liczbę elementów w $tree$).

insert(k, $tree$) — procedura wstawiająca element k do drzewa $tree$, przy założeniu, że $k \notin tree$, lub nie wykonująca żadnych zmian w drzewie w przeciwnym przypadku. Koszt tej operacji jest równy $O(\log |tree|)$.

min($tree$) — funkcja zwracająca minimalny element drzewa $tree$. Jej koszt to $O(\log |tree|)$.

max($tree$) — funkcja zwracająca maksymalny element drzewa $tree$. Jej koszt to również $O(\log |tree|)$.

Będziemy posługiwać się również dwoma zmiennymi globalnymi – tree_{col} i tree_{row}, które będą zbalansowanymi drzewami binarnymi. W drzewach tych będziemy przechowywać numery kolumn i wierszy macierzy R, które były modyfikowane w bieżącym kroku procedury i będą musiały być przejrzane w następnym.

Dla każdego wiersza i w macierzy R, gdzie $i \in \{1, \ldots, m\}$, definiujemy następujące zmienne pomocnicze (zobacz również Rysunek 3.2):

l^i – minimalna pozycja komórki zawierającej jedynkę w wierszu i,

r^i – maksymalna pozycja komórki zawierającej jedynkę,

p^i – minimalna pozycja komórki nie zawierającej zera,

q^i – maksymalna pozycja komórki nie zawierającej zera,

$\tilde{l}^i$ – tymczasowa wartość l_i którą zmieniamy w trakcie modyfikacji kolumn,

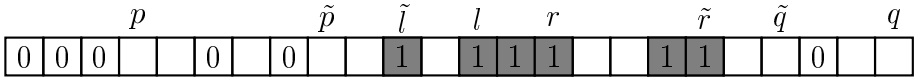
$\tilde{r}^i$ – tymczasowa wartość r_i,

$\tilde{p}^i$ – tymczasowa wartość p_i,

$\tilde{q}^i$ – tymczasowa wartość q_i,

free0^i – zbalansowane drzewo binarne zawierające pozycje wszystkich zer w komórkach między pozycjami $\tilde{p}$ i $\tilde{q}$, czyli nie należących do któregoś z krańcowych bloków zer.

Dla każdej kolumny j w macierzy R, gdzie $j \in \{1, \ldots, n\}$, definiujemy analogicznie zmienne l_j, r_j, p_j, q_j, $\tilde{l}_j$, $\tilde{r}_j$, $\tilde{p}_j$, $\tilde{q}_j$ i free0_j.



Rysunek 3.2: Pozycje zmiennych pomocniczych dla częściowo wypełnionego wiersza

Zmiennym tym nadajemy w wierszach następujące wartości początkowe:

$$\begin{aligned} &l = \tilde{l} = n + 1, \\ &r = \tilde{r} = 0, \\ &p = \tilde{p} = 1, \\ &q = \tilde{q} = n \quad \text{i} \\ &\text{free0} = nil, \end{aligned}$$

a w kolumnach

$$\begin{aligned} &l = \tilde{l} = m + 1, \\ &r = \tilde{r} = 0, \\ &p = \tilde{p} = 1, \\ &q = \tilde{q} = m \quad \text{i} \\ &\text{free0} = nil, \end{aligned}$$

gdzie *nil* oznacza puste drzewo.

Teraz wprowadzimy dwie pomocnicze operacje wstawiania symboli:

wstaw 0 w i-tym wierszu na pozycji j:

```
if R[i,j] = 1 then exit( fail )
        {w miejscu wstawiania zera jest już jedynka więc mamy }
        {sprzeczność i przerywamy procedurę}
if R[i,j] ≠ 0 then
        {wstawiamy nowe zero – nie było go tam wcześniej}
    R[i,j] ← 0
    insert( j, tree_col )
            {zapamiętujemy numer kolumny, która jest modyfikowana}
    if r_j ⩾ l_j then
            {kolumna j zawiera już jedynki}
        if i < l_j and i ⩾ p̃_j then p̃_j ← i + 1
        if i > r_j and i ⩽ q̃_j then q̃_j ← i − 1
    else
            {kolumna j nie zawiera żadnej jedynki}
        if i < p̃_j + v_j and i ⩾ p̃_j then
```

```
                {próbujemy scalić początkowy obszar zer}
            p̃_j ← i + 1
            while not empty( free0_j ) and
                    (k ←min( free0_j ))< p̃_j + v_j do
                delete( k, free0_j )
                p̃_j ← k + 1
        if i > q̃_j − v_j and i ⩽ q̃_j then
                {próbujemy scalić końcowy obszar zer}
            q̃_j ← i − 1
            while not empty( free0_j ) and
                    (k ←max( free0_j ))> q̃_j − v_j do
                delete( k, free0_j )
                q̃_j ← k − 1
        if p̃_j + v_j ⩽ i ⩽ q̃_j − v_j then insert( i, free0_j )
```

wstaw 1 w i-tym wierszu na pozycji j:

```
if R[i,j] = 0 then exit( fail )
        {w miejscu wstawiania jedynki jest już zero więc mamy }
        {sprzeczność i przerywamy procedurę}
if R[i,j] ≠ 1 then
        {wstawimy nową jedynkę – nie było jej tam wcześniej}
    R[i,j] ← 1
    insert( j, tree_col )
            {zapamiętujemy numer kolumny, która jest modyfikowana}
    if r_j < l_j then
            {kolumna j nie zawiera jedynek – to jest pierwsza jedynka}
        l_j ← r_j ← l̃_j ← r̃_j ← i
        if p̃_j < i − v_j + 1 then p̃_j ← i − v_j + 1
        if q̃_j < i + v_j − 1 then q̃_j ← i + v_j − 1
        while not empty( free0_j ) do
                {robimy spójnymi oba krańcowe zbiory zer}
            k ←min( free0_j )
            delete( k, free0_j )
            if k < i and k + 1 > p̃_j then p̃_j ← k + 1
            if k > i and k − 1 < q̃_j then q̃_j ← k − 1
    else
            {kolumna j zawiera już jedynki}
        if i < l̃_j then l̃_j ← i
```

`if` $i > \tilde{r}_j$ `then` $\tilde{r}_j \leftarrow i$

Operacje opisywane powyżej zapamiętują w zmiennej globalnej tree_{col} numery kolumn, które są modyfikowane, kiedy wstawiamy jakiś nowy element w wierszu. Ponadto modyfikujemy równocześnie odpowiednie wartości zmiennych tymczasowych w tych kolumnach. Modyfikacje te są następnie pomocne w wykonywaniu operacji na kolumnach. Operacje wstawiania nie dokonują żadnych modyfikacji, jeśli wstawiany symbol jest już na danej pozycji. Natomiast jeśli przy wstawianiu symbolu na danej pozycji znajdują symbol przeciwny, to przerywają działanie procedury. Oznacza to, że wypukłe poliomino o zadanych parametrach nie istnieje.

Analogicznie definiujemy operacje wstawiania dla kolumn.

Teraz zdefiniujemy operacje $\oplus, \ominus, \otimes, \odot$ modyfikujące wiersz (zobacz Rysunek 3.3):

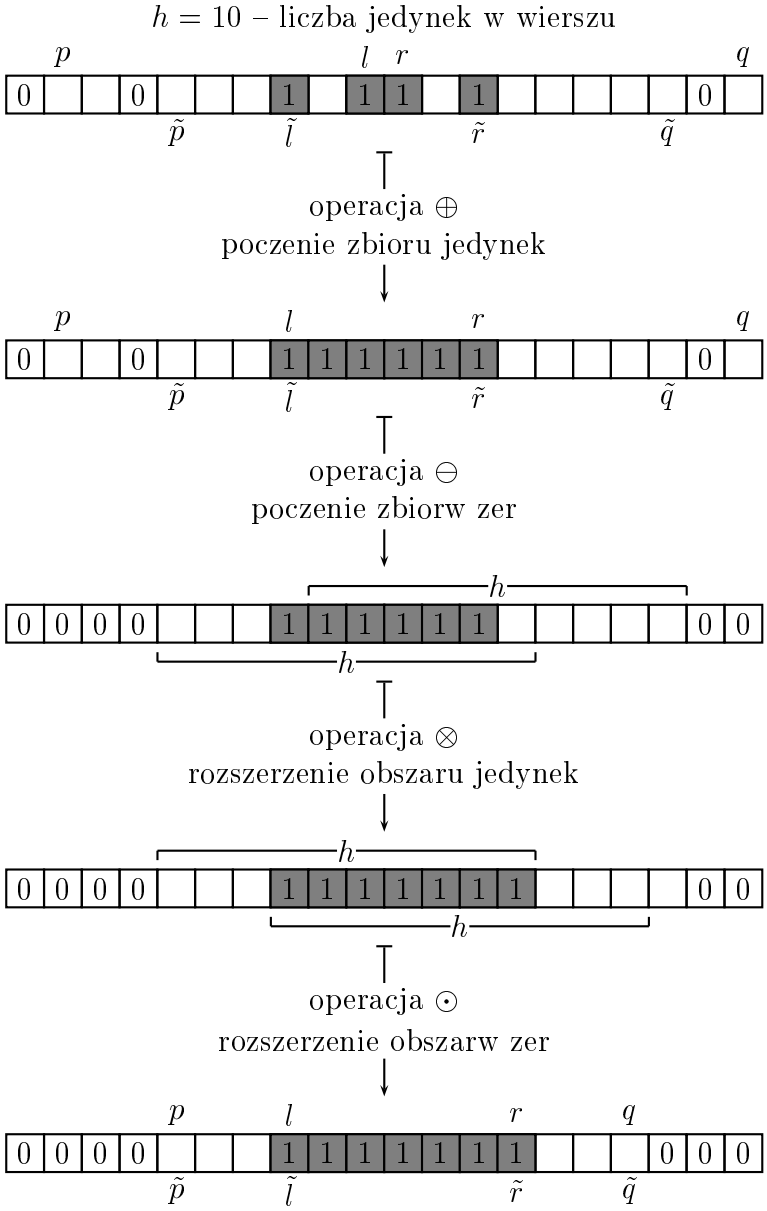
operacja $\oplus$ w i-tym wierszu:

```
if l̃^i < l^i then
    for j ← l̃^i to l^i − 1 do
        wstaw 1 w i-tym wierszu na pozycji j
    l^i ← l̃^i
if r̃^i > r^i then
    for j ← r^i + 1 to r̃^i do
        wstaw 1 w i-tym wierszu na pozycji j
    r^i ← r̃^i
```

Operacja $\oplus$ wypełnia wolne miejsca między blokami jedynek w wierszu tak, aby istniał tylko jeden spójny blok jedynek. Ponadto modyfikuje zmienne wyznaczające ten blok, czyli l i r.

operacja $\ominus$ na i-tym wierszu:

```
if p^i < p̃^i then
    for j ← p^i to p̃^i − 1 do
        wstaw 0 w i-tym wierszu na pozycji j
    p^i ← p̃^i
if q^i > q̃^i then
    for j ← q̃^i + 1 to q^i do
        wstaw 0 w i-tym wierszu na pozycji j
    q^i ← q̃^i
```



Rysunek 3.3: Operacje $\oplus$, $\ominus$, $\otimes$ i $\odot$ w wierszu

Operacja $\ominus$ wypełnia puste miejsca między blokami zer dla obu końców wiersza i modyfikuje wyznaczające je zmienne, czyli p i q.

Po wykonaniu obu powyższych operacji zmienne l, r, p i q są równe swoim wartościom tymczasowym.

operacja $\otimes$ w i-tym wierszu:

```
if l^i > r^i and p^i + h_i − 1 ⩾ q^i − h_i + 1 then
    l^i ← l̃^i ← q^i − h_i + 1
    r^i ← r̃^i ← p^i + h_i − 1
    for j ← l^i to r^i do
        wstaw 1 w i-tym wierszu na pozycji j
if l^i ⩽ r^i and q^i − h_i + 1 < l^i then
    for j ← q^i − h_i + 1 to l^i − 1 do
        wstaw 1 w i-tym wierszu na pozycji j
    l^i ← l̃^i ← q^i − h_i + 1
if l^i ⩽ r^i and p^i + h_i − 1 > r^i then
    for j ← r^i + 1 to p^i + h_i − 1 do
        wstaw 1 w i-tym wierszu na pozycji j
    r^i ← r̃^i ← p^i + h_i − 1
```

Operacja $\otimes$ rozszerza obszar jedynek w wierszu, jeśli przestrzeń bez zer jest odpowiednio wąska (zobacz Rysunek 3.3).

operacja $\odot$ na i-tym wierszu:

```
if l^i ⩽ r^i and p^i ⩽ r^i − h_i then
    for j ← p^i to r^i − h_i do
        wstaw 0 w i-tym wierszu na pozycji j
    p^i ← p̃^i ← r^i − h_i + 1
if l^i ⩽ r^i and q^i ⩾ l^i + h_i then
    for j ← l^i + h_i to q^i do
        wstaw 0 w i-tym wierszu na pozycji j
    q^i ← q̃^i ← l^i + h_i − 1
```

Operacja $\odot$ rozszerza obszary zer na końcach wiersza, jeśli blok jedynek jest odpowiednio szeroki (zobacz Rysunek 3.3).

Operacje $\oplus$, $\otimes$ wstawiają nowe jedynki do macierzy R a operacje $\ominus$, $\odot$ wstawiają nowe zera. Dla kolumn definiujemy powyższe operacje analogicznie.

Główna pętla procedury wypełniania ma teraz następującą postać

```
repeat
    while not empty( tree_row ) do
        k ←min( tree_row )
        delete( k, tree_row )
        wykonaj operacje ⊕, ⊖, ⊗, ⊙ na k-tym wierszu
    while not empty( tree_col ) do
        k ←min( tree_col )
        delete( k, tree_col )
        wykonaj operacje ⊕, ⊖, ⊗, ⊙ na k-tej kolumnie
until empty( tree_row ) and empty( tree_col )

if istnieją puste komórki w macierzy R then
    zbuduj i rozwiąż odpowiednią formułę 2SAT
```

Kiedy wyznaczamy pozycje startowe opisane w poprzednim podrozdziale, nie wstawiamy żadnych zer czy jedynek do macierzy R. Modyfikujemy tylko zmienne $\tilde{p}$ i $\tilde{q}$ dla odpowiednich kolumn, oraz wstawiamy numery tych kolumn do drzewa tree_{col}. Wstawimy te zera i jedynki, kiedy będziemy wykonywać główną pętlę procedury (zobacz operacje $\ominus$ i $\otimes$).

Lemat 3.4 *Załóżmy, że pozycje startowe są takie, że jest wstawiona co najmniej jedna jedynka w każdym wierszu macierzy R. Wtedy procedura wypełniania pracuje prawidłowo.*

Dowód: Jeśli procedura wypełniania zwraca *fail*, to wiemy, że wypukłe poliomino, które ma prostopadłe rzuty H i V oraz ustalone pozycje startowe, nie istnieje.

Natomiast jeśli drzewa tree_{row} i tree_{col} są puste a procedura nie odpowiedziała *fail*, to mamy dwa przypadki:
przypadek 1: Każda komórka macierzy R zawiera zero lub jedynkę. Mamy rozwiązanie. Zbiór jedynek tworzy zbiór spójny, ze spójnymi kolumnami i wierszami, więc jest wypukłym poliominem i realizuje wektory rzutów (H, V).
przypadek 2: Każdy wiersz zawiera co najmniej jedną jedynkę i są komórki w macierzy R, które nie zawierają ani jedynki ani zera (zobacz Rysunek 3.4, komórki w kolorze jasnoszarym). Jeśli jakiś wiersz lub kolumna zawiera takie puste komórki i co najmniej jedną jedynkę, wtedy zmienne pomocnicze w tym wierszu lub kolumnie muszą spełniać równość:

$$l - p = q - r \neq 0.$$

0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	c_1	1	1	c_{18}	0	0	0	0	0	0	0	0	0	0	0
0	0	0	c_{13}	c_7	1	1	1	1	c_{12}	c_6	0	0	0	0	0	0	0	0	0
0	a_1	b_1	1	1	1	1	1	1	1	a_6	b_6	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	a_2	b_2	c_{14}	1	1	1	1	1	1	1	1	1	1	1	a_3	b_3	c_{15}	0	0
0	0	0	0	c_8	c_2	1	1	1	1	1	1	1	1	1	1	1	1	c_9	c_3
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	c_{17}	c_{11}	c_5	1	1	1	1	1	1	c_{16}	c_{10}	c_4
0	0	0	0	0	0	0	0	0	0	a_5	b_5	1	1	1	a_4	b_4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Rysunek 3.4: Trzy różne cykle do wypełnienia: $(a_1, \ldots, a_6)$, $(b_1, \ldots, b_6)$ i $(c_1, \ldots, c_{18})$

Wynika to bezpośrednio z własności operacji $\otimes$. Jeśli natomiast pewna kolumna nie zawiera jedynek i posiada puste komórki, to muszą one tworzyć dwa bloki, każdy o długości równej liczbie jedynek w tej kolumnie. Oba bloki mogą ze sobą sąsiadować. W każdym wierszu lub kolumnie liczba wolnych komórek jest równa dwukrotności liczby brakujących jedynek. Te puste komórki, niezawierające ani jedynki ani zera, tworzą cykle parzystej długości zawierające co najmniej 4 komórki. Cykle te tworzymy znajdując dla danej pustej komórki $R[i,j]$ komórkę $R[i',j']$ taką, że $i = i'$ i $|j - j'| = h_i$ albo $j = j'$ i jeśli kolumna j zawiera co najmniej jedną jedynkę, to $|i - i'| = v_j$, a w przeciwnym wypadku $|i - i'|$ jest równe sumie odległości między blokami wolnymi od zer i liczbą jedynek w kolumnie j, czyli v_j. Cykle te możemy na przemian etykietować jedynkami i zerami. Etykietowania pewnych cykli mogą zależeć od siebie. Budujemy więc odpowiednią formułę 2SAT, której rozwiązanie daje prawidłowe etykietowanie pustych komórek. W tym celu każdemu cyklowi nadajemy nazwę w postaci zmiennej boolowskiej. Każdy cykl etykietujemy na przemian odpowiadającą mu zmienną i jej negacją. Teraz jeśli w jakimś wierszu lub kolumnie mamy obok siebie dwie komórki zaetykietowane tymi zmiennymi, to do formuły 2SAT dodajemy implikację, że ze zmiennej zewnętrznej w stosunku do bloku jedynek

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	x	1	1	$\bar{x}$	0	0	0	0	0	0	0	0	0	0	0	0	3
3	0	0	0	y	1	1	1	1	$\bar{y}$	0	0	0	0	0	0	0	0	0	0	0	5
4	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	7
5	z	1	1	1	1	1	1	1	1	$\bar{z}$	0	0	0	0	0	0	0	0	0	0	9
6	$\bar{z}$	1	1	1	1	1	1	1	1	1	z	0	0	0	0	0	0	0	0	0	10
7	0	0	0	$\bar{y}$	1	1	1	1	1	1	y	0	0	0	0	0	0	0	0	0	7
8	0	0	0	0	$\bar{x}$	1	1	1	1	1	1	x	0	0	0	0	0	0	0	0	7
9	0	0	0	0	0	0	0	x	1	1	1	1	$\bar{x}$	0	0	0	0	0	0	0	5
10	0	0	0	0	0	0	0	0	y	1	1	1	1	$\bar{y}$	0	0	0	0	0	0	5
11	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	5
12	0	0	0	0	0	0	0	0	0	z	1	1	1	1	$\bar{z}$	0	0	0	0	0	5
13	0	0	0	0	0	0	0	0	0	0	$\bar{z}$	1	1	1	1	1	1	z	0	0	7
14	0	0	0	0	0	0	0	0	0	0	$\bar{y}$	1	1	1	1	1	1	1	y	0	8
15	0	0	0	0	0	0	0	0	0	0	0	$\bar{x}$	1	1	1	1	1	1	1	x	8
16	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	1	1	1	1	$\bar{x}$	7
17	0	0	0	0	0	0	0	0	0	0	0	0	0	y	1	1	1	1	$\bar{y}$	0	5
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	z	1	1	$\bar{z}$	0	0	3
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	2
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	2
	1	2	3	4	6	8	8	7	7	7	7	7	7	7	6	8	8	5	3	1	

Rysunek 3.5: Przykład trzech cykli etykietowanych odpowiednio przez zmienne x, y i z oraz formuły 2SAT utworzonej dla tych cykli (z wyeliminowanymi powtórzeniami):

$$(z \to y) \wedge (\bar{y} \to \bar{z})$$

Istnieje sześć różnych wartościowań zmiennych $\{x, y, z\}$, które spełniają formułę: $\{0,0,0\}$, $\{0,1,0\}$, $\{0,1,1\}$, $\{1,0,0\}$, $\{1,1,0\}$ i $\{1,1,1\}$.

w tym wierszu lub kolumnie wynika zmienna wewnętrzna. W kolumnie, która nie zawiera bloku jedynek, tworzymy cykl implikacji dla zmiennych każdego bloku osobno, tak aby wymusić zgodne etykietowanie całego bloku. Tak zbudowana formuła może być rozwiązana w czasie liniowym. Jej długość jest zaś ograniczona przez ilość niewypełnionych komórek, czyli wynosi co najwyżej $O(mn)$ (każda niewypełniona komórka uczestniczy w nie więcej niż czterech implikacjach). Wartościowanie tak otrzymanej formuły 2SAT zadaje sposób wypełnienia poliomina. □

Lemat 3.5 *Procedura wypełniania kosztuje co najwyżej* $O(mn \log mn)$ *kroków.*

Dowód: Oszacujmy koszt głównej pętli procedury wypełniania. Na każdej pozycji $[i, j]$ macierzy R wykonujemy operację **wstaw** tylko dwa razy, raz w wierszu i i raz w kolumnie j. Co więcej, jeśli wykonujemy ciąg operacji $\oplus$, $\ominus$, $\otimes$, $\odot$ w wierszu lub w kolumnie, to wykonujemy co najmniej jedną operację **wstaw** (odwiedzamy kolumnę lub wiersz tylko wtedy gdy coś wymaga modyfikacji). Stąd w sumie odwiedzamy co najwyżej $O(mn)$ wierszy i kolumn. Odwiedzenie każdego wiersza kosztuje sumę kosztu operacji ściągnięcia numeru wiersza z drzewa oraz kosztu wykonanych operacji **wstaw**, czyli

$$O(\log m) + [\text{koszt operacji } \mathbf{wstaw}].$$

Analogicznie koszt odwiedzenia każdej kolumny wynosi

$$O(\log n) + [\text{koszt operacji } \mathbf{wstaw}]$$

Stąd ogólny koszt głównej pętli procedury wypełniania wynosi

$$O(mn(\log m + \log n)) + [\text{koszt wszystkich operacji } \mathbf{wstaw}].$$

Teraz wyznaczmy koszt wszystkich operacji **wstaw**. W wierszu i, w trakcie wykonywania operacji **wstaw** możemy wywołać co najwyżej n operacji insert w drzewie tree_{col}. To kosztuje $O(n \log n)$. Dla wszystkich wierszy kosztuje to więc co najwyżej $O(mn \log n)$. Analogicznie, dla wszystkich kolumn koszt operacji insert w drzewie tree_{row} wynosi $O(mn \log m)$.

Ponieważ operacje insert w drzewie free0^i dla każdego wiersza są wykonywane co najwyżej raz dla każdej pozycji w tym wierszu, więc ogólnie kosztują co najwyżej $O(n \log n)$. Operacji delete wykonujemy tyle samo co insert. Funkcje min i max wywołujemy tylko wtedy, kiedy modyfikujemy zmienne $\tilde{p}^i$ i $\tilde{q}^i$. Stąd liczba tych operacji również nie przekracza n. Wszystkie operacje na drzewie free0^i kosztują więc co najwyżej $O(n \log n)$. Dla wszystkich m

wierszy koszt operacji na drzewach free0 wynosi $O(mn \log n)$. Analogicznie, dla kolumn koszt operacji na drzewach free0 wynosi $O(mn \log m)$.

Złożoność dodatkowych operacji, w tym budowy i rozwiązania formuły 2SAT, nie przekracza $O(mn)$.

Sumując powyższe koszty otrzymujemy, że całkowity koszt procedury wypełniania wynosi $O(mn(\log m + \log n))$. □

3.3 Złożoność algorytmu

Twierdzenie 1 *Problem rekonstrukcji wypukłego poliomina z rzutów prostopadłych kosztuje* $O(\min(m,n)^2 \cdot mn \log mn)$.

Dowód: Załóżmy, że istnieje wypukłe poliomino S realizujące wektory rzutów (H,V). Jeśli prawidłowo zgadniemy pozycje brzegowe S w macierzy R, a ponieważ testujemy wszystkie możliwe pozycje brzegowe, musimy w końcu trafić na prawidłowe, to zgodnie z Lematem 3.2 wyznaczone pozycje startowe są prawidłowe i należą do S. Stąd mamy w każdym wierszu co najmniej jedną jedynkę, procedura wypełniania działa prawidłowo (zgodnie z Lematem 3.4) i musi zwrócić poprawne wypukłe poliomino.

Jeśli dla pary wektorów (H,V) nie istnieje wypukłe poliomino S realizujące (H,V) wtedy procedura wypełniania odpowie *fail* dla wszystkich próbowanych pozycji brzegowych.

Liczba wszystkich testowanych pozycji brzegowych jest równa co najwyżej $\min(m,n)^2$. Z Lematu 3.3 i Lematu 3.5 wyznaczanie pozycji startowych i wypełnianie kosztują co najwyżej $O(mn \log mn)$. Stąd ogólny koszt rekonstrukcji wypukłego poliomina z rzutów poziomego i pionowego jest iloczynem powyższych wielkości i jest równy $O(\min(m,n)^2 \cdot mn \log mn)$. □

Implementacja algorytmu może być jeszcze bardziej efektywna dzięki redukcji liczby wybieranych pozycji brzegowych. Możemy tutaj wykorzystać na przykład Wniosek 2.28.

Możemy rozpatrzyć również pewne szczególne przypadki.

Wniosek 3.6 *Jeśli istnieje kolumna j taka, że jej pionowy rzut v_j jest równy m (to jest istnieje kolumna zawierająca tylko jedynki), wtedy koszt rekonstrukcji wypukłego poliomina wynosi* $O(mn \log mn)$.

Dowód: Jest to konsekwencja Lematu 3.4, ponieważ kolumna j zawiera jedynki w każdym wierszu a jej końce należą do pozycji brzegowych, których w tym wypadku nie zgadujemy, od razu przystępujemy do procedury

wypełniania, startując z wypełnionej kolumny j. Stąd koszt tego przypadku jest równy tylko kosztowi procedury wypełniania. □

Wniosek 3.7 *Jeśli* $m = \Theta(n)$ *i* $c = \max\{\ h_i\ :\ i \in \{1, \ldots, m\}\ \}$ *to rekonstrukcja wypukłego poliomina kosztuje* $O(c^2 \cdot cm \log m)$.

Dowód: Z Lematu 2.22 musimy testować co najwyżej $O(c^2)$ pozycji brzegowych.

Ponadto, ponieważ mamy co najwyżej c jedynek w każdym wierszu możemy w prosty sposób zmodyfikować procedurę wypełniania tak, aby wypełniała co najwyżej obszar $O(cm)$ komórek. Dla reszty po prostu od razu założymy, że są wypełnione zerami. W każdym wierszu, po wyznaczeniu pozycji początkowych, ograniczamy obszar pracy do $2c$ komórek, korzystając z operacji $\otimes$. W ten sposób łatwo utworzymy wypełniony zerami brzeg obszaru roboczego, korzystając ze spójności zbioru jedynek i Lematu 2.2. Dodatkowo przy bokach macierzy możemy użyć Lematu 2.24. Stąd, ponieważ obszar na którym będziemy pracować nie zawiera więcej niż $2cm$ komórek, możemy go wypełnić w czasie $O(cm \log m)$ (wynika to bezpośrednio z odpowiedniej modyfikacji dowodu Lematu 3.5). □

Rozdział 4

Trójwymiarowe poliomina

W tym rozdziale wprowadzimy pewną klasę trójwymiarowych poliomin wypukłych, dla której podamy wielomianowy algorytm rekonstrukcji. Będzie on w dużej mierze korzystał z właściwości dwuwymiarowych poliomin wypukłych.

4.1 Definicja trójwymiarowego poliomina

Na początek zdefiniujmy obiekty, którymi będziemy się zajmować w tym rozdziale. Niech $\mathcal{R}$ oznacza trójwymiarową kratę o wymiarach $n \times n \times n$, której jednostkowe sześcienne pola są lub nie są wypełnione. Matematycznym modelem tej kraty będzie zerojedynkowa, trójwymiarowa macierz R, gdzie elementy odpowiadające wypełnionym polom kraty mają wartość 1 a niewypełnionym – 0. Niech Q(R) oznacza zbiór wszystkich elementów macierzy R, czyli

$$Q(R) = \{\ [i,j,k]\ :\ 1 \leqslant i,j,k \leqslant n\ \}.$$

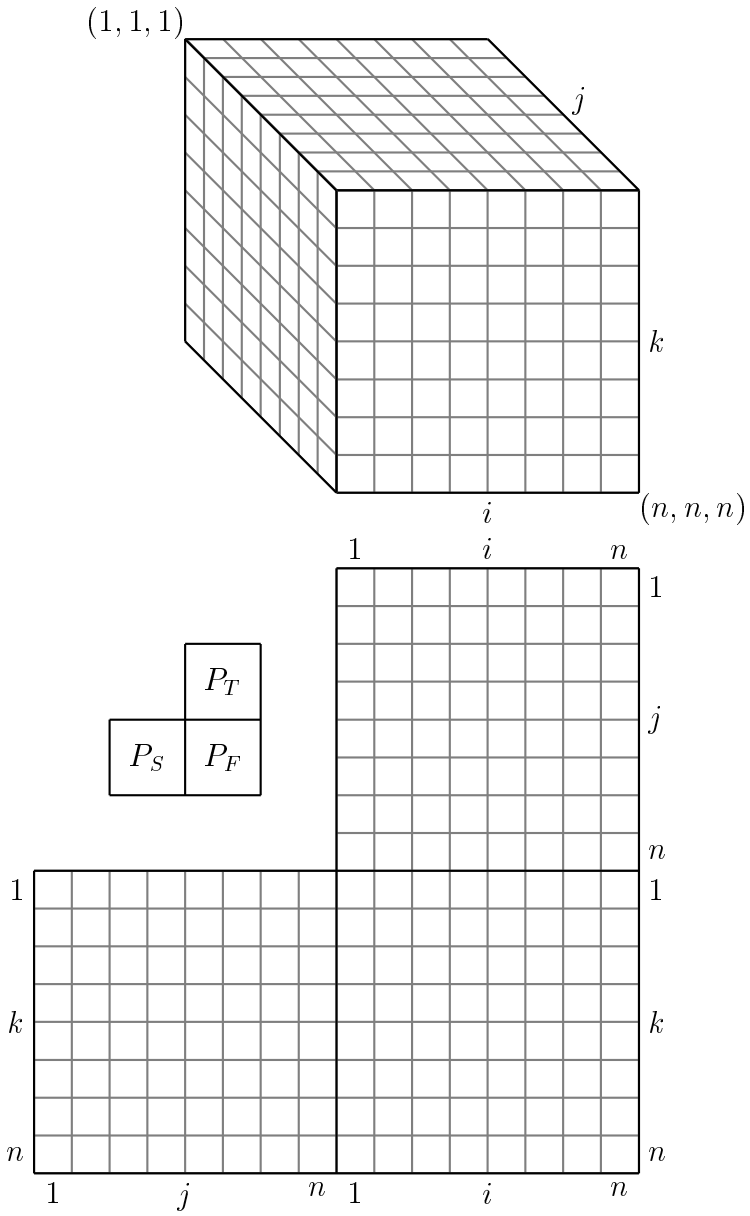
Niech $S(R)$ oznacza zbiór elementów macierzy R równych 1, tj.

$$S(R) = \{\ [i,j,k]\ :\ R[i,j,k] = 1\ \}.$$

Ponieważ R będzie na ogół ustalone zamiast $Q(R)$ i $S(R)$ będziemy często pisali Q i S.

Na początek zdefiniujemy podział macierzy R na odpowiednie fragmenty, którymi będziemy posługiwać się w następnych podrozdziałach

Definicja 4.1 *Warstwą nazywamy dwuwymiarową macierz będącą prostopadłym wycinkiem macierzy R. I tak dla każdego $i \in \{1,\ldots,n\}$ definiujemy macierz $L^i_X \in \{0,1\}^{n\times n}$ taką, że $L^i_X[j,k] = R[i,j,k]$.*



Rysunek 4.1: Trójwymiarowa macierz R i odpowiednie macierze rzutów prostopadłych P_T, P_F i P_S, kolejno z góry, z przodu i z boku

Analogicznie, dla $j \in \{1, \ldots, n\}$ *definiujemy warstwy* $L_Y^j \in \{0,1\}^{n \times n}$ *takie, że* $L_Y^j[i,k] = R[i,j,k]$*, oraz dla* $k \in \{1, \ldots, n\}$ *warstwy* $L_Z^k \in \{0,1\}^{n \times n}$ *takie, że* $L_Z^k[i,j] = R[i,j,k]$.

Definicja 4.2 *Kolumną nazywamy wektor w macierzy R wyznaczony zgodnie z jednym z prostopadłych kierunków. I tak dla* $j,k \in \{1, \ldots, n\}$ *definiujemy kolumny* $C_X^{j,k} \in \{0,1\}^n$ *takie, że* $C_X^{j,k}[i] = R[i,j,k]$.

Analogicznie, dla $i,k \in \{1, \ldots, n\}$ *definiujemy kolumny* $C_Y^{i,k} \in \{0,1\}^n$ *takie, że* $C_Y^{i,k}[j] = R[i,j,k]$*, oraz dla* $i,j \in \{1, \ldots, n\}$ *kolumny* $C_Z^{i,j} \in \{0,1\}^n$ *takie, że* $C_Z^{i,j}[k] = R[i,j,k]$.

Kolumnami pionowymi będziemy nazywać kolumny C_Z.

Dodatkowo dla macierzy R i pozycji $[i,j,k]$ kolumnami zawierającymi $[i,j,k]$ będziemy nazywać kolumny $C_X^{j,k}$, $C_Y^{i,k}$ i $C_Z^{i,j}$.

Teraz zdefiniujmy pojęcie rzutów prostopadłych dla trójwymiarowej macierzy

Definicja 4.3 *Dla danego zbioru* $S = S(R)$ *definiujemy* $P_T(S)$ *jako macierz o rozmiarze* $n \times n$ *taką że*

$$P_T(S)[i,j] = \sum_{k=1}^{n} R[i,j,k].$$

Macierz tą nazywamy macierzą rzutów prostopadłych z góry zbioru S. Elementy $P_T(S)[i,j]$ *opisują liczbę jedynek w kolumnie* $C_Z^{i,j}$.

Analogicznie definiujemy macierz rzutów prostopadłych z przodu – $P_F(S)$

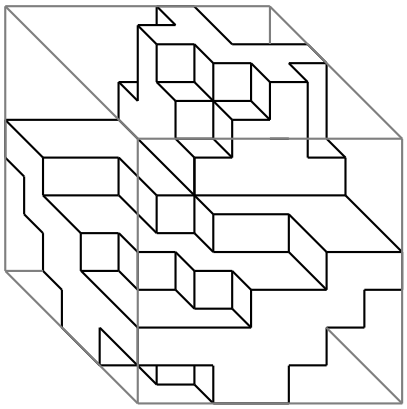
$$P_F(S)[i,k] = \sum_{j=1}^{n} R[i,j,k]$$

oraz macierz rzutów prostopadłych z boku – $P_S(S)$

$$P_S(S)[j,k] = \sum_{i=1}^{n} R[i,j,k].$$

Jeśli zbiór S jest ustalony, to piszemy krócej P_T, P_F *i* P_S.

Definicja 4.4 *Trójwymiarowy zbiór* $S = S(R)$ *nazywamy pełnym poliominem wypukłym, jeśli jest spójny, każda warstwa macierzy R zawiera dwuwymiarowe wypukłe poliomino oraz przynajmniej jedna macierz rzutów nie zawiera zer (jest pełna).*



P_T

1						n
1	1	1	2	3	1	1
	2	2	4	3	1	1
	2	1	2	3	3	3
	3	2	2	3	4	2
	2	3	2	2	2	1
	1	2	3	2	1	2	2
n	1	1	2	3	2	2	1

P_S

1						n
1	1	2	3	1	0	0	0
	3	4	4	2	0	0	0
	2	2	4	4	0	0	0
	4	4	2	5	4	2	2
	0	3	3	4	5	4	3
	0	0	1	2	3	4	5
n	0	0	0	1	2	2	2

P_F

1						n
1	0	0	0	1	3	2	1
	0	0	0	2	4	4	3
	0	0	0	4	4	2	2
	2	2	4	5	2	4	4
	3	4	5	4	3	3	0
	5	4	3	2	1	0	0
n	2	2	2	1	0	0	0

Rysunek 4.2: Przykład trójwymiarowego pełnego poliomina wypukłego wraz z jego macierzami rzutów

Dla ułatwienia, w dalszej części tego rozdziału przyjmiemy, że pełną macierzą jest zawsze macierz rzutów prostopadłych z góry – P_T.

Definicja 4.5 *Niech będą dane trzy macierze*

$$P_S, P_F \in \{0, \ldots, n\}^{n \times n} \qquad i \qquad P_T \in \{1, \ldots, n\}^{n \times n}.$$

Mówimy, że trójwymiarowy zbiór S realizuje (P_T, P_S, P_F) w klasie pełnych poliomin wypukłych, jeśli S jest trójwymiarowym pełnym poliominem wypukłym oraz $P_T(S) = P_T$, $P_S(S) = P_S$ i $P_F(S) = P_F$.

Trójka (P_T, P_S, P_F) ma realizację w klasie trójwymiarowych pełnych poliomin wypukłych, jeśli istnieje co najmniej jedna macierz $R \in \{0, 1\}^{n \times n \times n}$ taka, że $S = S(R)$ realizuje (P_T, P_S, P_F) w klasie pełnych poliomin wypukłych.

4.2 Pozycje startowe

Algorytm, który opisujemy w tym rozdziale, ma dwa etapy. W pierwszym etapie wyznaczymy pozycje niektórych jedynek i zer. Natomiast w drugim etapie, opisanym w następnym podrozdziale, wykonamy procedurę wypełniania macierzy w oparciu o wyznaczone pozycje startowe i macierze rzutów.

Korzystając z tego, że zgodnie z założeniem macierz rzutów pionowych P_T nie zawiera zer, czyli każda pionowa kolumna $C_Z^{i,j}$ macierzy R, dla $i, j \in \{1, \ldots, n\}$, zawiera co najmniej jedną jedynkę, wyznaczmy arbitralnie pozycje pierwszych jedynek w czterech narożnych kolumnach $C_Z^{1,1}$, $C_Z^{1,n}$, $C_Z^{n,1}$ i $C_Z^{n,n}$. Niech będą to odpowiednio n_1, n_2, n_3 i n_4 (czyli $R[1, 1, n_1] = R[1, n, n_2] = R[n, 1, n_3] = R[n, n, n_4] = 1$). Pozycje te będziemy nazywać *początkowymi*.

Teraz rozpatrujemy dwie warstwy: L_X^1 i L_X^n. Każda z nich musi zawierać, zgodnie z założeniem, dwuwymiarowe poliomino wypukłe. W skrajnych kolumnach (brzegowych) tych warstw zostały już wstawione jedynki. Stąd zgodnie z Lematami 3.1 i 3.2 możemy oszacować pozycje co najmniej jednej jedynki w każdej pionowej kolumnie tych warstw, czyli $C_Z^{1,j}$ i $C_Z^{n,j}$, dla $j \in \{1, \ldots, n\}$, odpowiednio dla warstw L_X^1 i L_X^n.

Ponieważ mamy teraz co najmniej jedną jedynkę w każdej kolumnie $C_Z^{1,j}$ i $C_Z^{n,j}$, dla $j \in \{1, \ldots, n\}$, rozpatrzymy wszystkie warstwy L_Y^j. Każda z tych warstw jest również dwuwymiarowym poliominem wypukłym i zawiera jedynki w obu skrajnych (brzegowych) kolumnach. Możemy więc powtórzyć rozumowanie i oszacować zgodnie z Lematami 3.1 i 3.2 pozycje co najmniej jednej jedynki dla każdej kolumny położonej między brzegowymi. Stąd dla

każdego $i, j \in \{1, \ldots, n\}$ każda pionowa kolumna $C_Z^{i,j}$ zawiera co najmniej jedną jedynkę.

Powyżej opisane argumenty uzasadniają więc następujący lemat:

Lemat 4.6 *Dla każdych pozycji początkowych w czterech kolumnach narożnych macierzy R można wstawić jedynki do macierzy R w taki sposób, aby w każdej kolumnie pionowej była co najmniej jedna jedynka.* □

Pozycje wszystkich obliczonych powyżej jedynek będziemy nazywać *pozycjami startowymi*. Teraz musimy jeszcze pokazać, że wstawione jedynki są na poprawnych pozycjach.

Lemat 4.7 *Jeśli istnieje pełne wypukłe poliomino S, które realizuje macierze rzutów P_T, P_F i P_S oraz zawiera pozycje początkowe $[1, 1, n_1]$, $[1, n, n_2]$, $[n, 1, n_3]$ i $[n, n, n_4]$, to wszystkie wstawione do macierzy R jedynki na pozycjach startowych należą do S.*

Dowód: Z Lematu 3.2 bezpośrednio wynika, że wyznaczone pozycje jedynek są prawidłowe dla warstw L_X^1 i L_X^n. Stosując Lemat 3.2 do warstw L_Y otrzymujemy, że pozostałe jedynki również są postawione prawidłowo. □

Pozostaje nam jeszcze oszacować koszt obliczenia pozycji startowych

Lemat 4.8 *Obliczenie pozycji startowych wymaga co najwyżej $O(n^2)$ kroków.*

Dowód: Wykonujemy $n + 2$ obliczeń opisanych w poprzednim rozdziale. Koszt każdego z nich zgodnie z Lematem 3.3 wynosi $O(n)$. Stąd całkowity koszt równy jest $O(n^2)$. □

Koszt wyznaczania pozycji startowych jest także w tym przypadku mniejszy niż koszt procedury wypełniania. Stąd to procedura zdeterminuje złożoność algorytmu.

4.3 Procedura wypełniania

Procedura opisana tutaj będzie trójwymiarową modyfikacją procedury wypełniania opisanej w poprzednim rozdziale.

Będziemy w niej posługiwać się trzema zbalansowanymi drzewami binarnymi, pamiętanymi w zmiennych globalnych tree_X, tree_Y i tree_Z. Elementami pamiętanymi przez te drzewa będą pary liczb określające kolumnę, która będzie wymagała modyfikacji.

Każdej kolumnie $C_X^{j,k}$, $C_Y^{i,k}$, $C_Z^{i,k}$, dla $i,j,k \in \{1,\ldots,n\}$, przypiszemy zbiór zmiennych pomocniczych $l, r, p, q, \tilde{l}, \tilde{r}, \tilde{p}, \tilde{q}$ oraz free0, które będą miały takie samo znaczenie jak w poprzednim rozdziale. Wartości początkowe tych zmiennych będą następujące

$$\begin{aligned} &l = \tilde{l} = n+1, \\ &r = \tilde{r} = 0, \\ &p = \tilde{p} = 1, \\ &q = \tilde{q} = n \quad \text{i} \\ &\text{free0} = nil, \end{aligned}$$

gdzie *nil* oznacza puste drzewo.

Teraz można zdefiniować odpowiednie operacje wstawiania symboli.

wstaw 0 w kolumnie $C_X^{j,k}$ na pozycji i (na pozycji $[i,j,k]$):

`if` $R[i,j,k] = 1$ `then exit(` fail `)`
{w miejscu wstawiania zera jest już jedynka więc mamy }
{sprzeczność i przerywamy procedurę}
`if` $R[i,j,k] \neq 0$ `then`
{wstawiamy nowe zero – nie było go tam wcześniej}
$R[i,j,k] \leftarrow 0$
insert`(` (i,k)`,` tree_Y `)`
insert`(` (i,j)`,` tree_Z `)`
{zapamiętujemy numery kolumn, które są modyfikowane}
modyfikuj0 kolumnę $C_Y^{i,k}$ na pozycji j z macierzą rzutów P_F
modyfikuj0 kolumnę $C_Z^{i,j}$ na pozycji k z macierzą rzutów P_T

wstaw 1 w kolumnie $C_X^{j,k}$ na pozycji i (na pozycji $[i,j,k]$):

`if` $R[i,j,k] = 0$ `then exit(` fail `)`
{w miejscu wstawiania jedynki jest już zero więc mamy }
{sprzeczność i przerywamy procedurę}
`if` $R[i,j,k] \neq 1$ `then`
{wstawimy nową jedynkę – nie było jej tam wcześniej}
$R[i,j,k] \leftarrow 1$
insert`(` (i,k)`,` tree_Y `)`
insert`(` (i,j)`,` tree_Z `)`
{ zapamiętujemy numery kolumn, które są modyfikowane}
modyfikuj1 kolumnę $C_Y^{i,k}$ na pozycji j z macierzą rzutów P_F
modyfikuj1 kolumnę $C_Z^{i,j}$ na pozycji k z macierzą rzutów P_T

gdzie podoperacje **modyfikuj** mają następującą postać

modyfikuj0 kolumnę $C^{j,k}$ na pozycji i z macierzą rzutów P:

```
with C^{j,k} do
    if r ⩾ l then
            {kolumna zawiera już jedynki}
        if i < l and i ⩾ p̃ then p̃ ← i + 1
        if i > r and i ⩽ q̃ then q̃ ← i − 1
    else
            {kolumna nie zawiera żadnej jedynki}
        if i < p̃ + P[j,k] and i ⩾ p̃ then
                {próbujemy scalić początkowy obszar zer}
            p̃ ← i + 1
            while not empty( free0 ) and
                    (m ←min( free0 ))< p̃ + P[j,k] do
                delete( m, free0 )
                p̃ ← m + 1
        if i > q̃ − P[j,k] and i ⩽ q̃ then
                {próbujemy scalić końcowy obszar zer}
            q̃ ← i − 1
            while not empty( free0 ) and
                    (m ←max( free0 ))> q̃ − P[j,k] do
                delete( m, free0 )
                q̃ ← m − 1
        if p̃ + P[j,k] ⩽ i ⩽ q̃ − P[j,k] then insert( i, free0 )
```

modyfikuj1 kolumnę $C^{j,k}$ na pozycji i z macierzą rzutów P:

```
with C^{j,k} do
    if r < l then
            {kolumna nie zawiera jedynek – to jest pierwsza jedynka}
        l ← r ← l̃ ← r̃ ← i
        if p̃ < i − P[j,k] + 1 then p̃ ← i − P[j,k] + 1
        if q̃ < i + P[j,k] − 1 then q̃ ← i + P[j,k] − 1
        while not empty( free0 ) do
                {scalamy oba krańcowe zbiory zer}
            m ←min( free0 )
            delete( m, free0 )
            if m < i and m + 1 > p̃ then p̃ ← n + 1
```

```
            if m > i and m − 1 < q̃ then q̃ ← n − 1
    else
            {kolumna zawiera już jedynki}
        if i < l̃ then l̃ ← i
        if i > r̃ then r̃ ← i
```

Operacje opisywane powyżej zapamiętują w zmiennych globalnych tree_X, tree_Y i tree_Z numery kolumn, które są modyfikowane, kiedy wstawiamy nowy symbol. Ponadto modyfikujemy równocześnie odpowiednie wartości zmiennych tymczasowych w tych kolumnach. Modyfikacje te są następnie pomocne w wykonywaniu modyfikacji na tych kolumnach. Operacje wstawiania nie dokonują żadnych modyfikacji, jeśli wstawiany symbol jest już na danej pozycji. Natomiast jeśli przy wstawianiu symbolu na danej pozycji znajdują symbol przeciwny, to przerywają działanie procedury. Oznacza to, że wypukłe poliomino o zadanych parametrach nie istnieje.

Operacje wstawiania zer i jedynek dla kolumn C_Y i C_Z definiujemy podobnie (zmieniając odpowiednio rodzaje kolumn i macierze rzutów, przy czym kolumny C_X wiążemy z macierzą rzutów P_S).

Teraz ponownie definiujemy operacje $\oplus, \ominus, \otimes, \odot$ modyfikujące daną kolumnę

operacja $\oplus$ w kolumnie $C_X^{j,k}$:

```
with C_X^{j,k} do
    if l̃ < l then
        for i ← l̃ to l − 1 do
            wstaw 1 w kolumnie C_X^{j,k} na pozycji i
        l ← l̃
    if r̃ > r then
        for i ← r + 1 to r̃ do
            wstaw 1 w kolumnie C_X^{j,k} na pozycji i
        r ← r̃
```

operacja $\ominus$ w kolumnie $C_X^{j,k}$:

```
with C_X^{j,k} do
    if p < p̃ then
        for i ← p to p̃ − 1 do
            wstaw 0 w kolumnie C_X^{j,k} na pozycji i
```

```
        p ← p̃
    if q > q̃ then
        for i ← q̃ + 1 to q do
            wstaw 0 w kolumnie C_X^{j,k} na pozycji i
        q ← q̃
```

operacja $\otimes$ w kolumnie $C_X^{j,k}$:

```
with C_X^{j,k} do
    if l > r and p + P_S[j,k] − 1 ⩾ q − P_S[j,k] + 1 then
        l ← l̃ ← q − P_S[j,k] + 1
        r ← r̃ ← p + P_S[j,k] − 1
        for i ← l to r do
            wstaw 1 w kolumnie C_X^{j,k} na pozycji i
    if l ⩽ r and q − P_S[j,k] + 1 < l then
        for i ← q − P_S[j,k] + 1 to l − 1 do
            wstaw 1 w kolumnie C_X^{j,k} na pozycji i
        l ← l̃ ← q − P_S[j,k] + 1
    if l ⩽ r and p + P_S[j,k] − 1 > r then
        for i ← r + 1 to p + P_S[j,k] − 1 do
            wstaw 1 w kolumnie C_X^{j,k} na pozycji i
        r ← r̃ ← p + P_S[j,k] − 1
```

operacja $\odot$ w kolumnie $C_X^{j,k}$:

```
with C_X^{j,k} do
    if l ⩽ r and p ⩽ r − P_S[j,k] then
        for i ← p to r − P_S[j,k] do
            wstaw 0 w kolumnie C_X^{j,k} na pozycji i
        p ← p̃ ← r − P_S[j,k] + 1
    if l ⩽ r and q ⩾ l + P_S[j,k] then
        for i ← l + P_S[j,k] to q do
            wstaw 0 w kolumnie C_X^{j,k} na pozycji i
        q ← q̃ ← l + P_S[j,k] − 1
```

Oczywiście dla kolumn C_Y i C_Z definiujemy powyższe operacje analogicznie, wykorzystując odpowiednio macierze rzutów P_F i P_T.

Główna pętla procedury ma teraz następującą postać

```
repeat
    while not empty( tree_X ) do
        (j,k) ←min( tree_X )
        delete( (j,k), tree_X )
        wykonaj operacje ⊕,⊖,⊗,⊙ w kolumnie C_X^{j,k}
    while not empty( tree_Y ) do
        (i,k) ←min( tree_Y )
        delete( (i,k), tree_Y )
        wykonaj operacje ⊕,⊖,⊗,⊙ w kolumnie C_Y^{i,k}
    while not empty( tree_Z ) do
        (i,j) ←min( tree_Z )
        delete( (i,j), tree_Z )
        wykonaj operacje ⊕,⊖,⊗,⊙ w kolumnie C_Z^{i,j}
until empty( tree_X ) and empty( tree_Y ) and empty( tree_Z )

if istnieją puste komórki w macierzy R then
    zbuduj i rozwiąż odpowiednią formułę 2SAT
```

Oczywiście kiedy wyznaczamy pozycje startowe jedynek w kolumnach C_Z (opisane w poprzednim podrozdziale), to nie wstawiamy ich od razu do macierzy R a tylko odpowiednio modyfikujemy zmienne $\tilde{p}$ i $\tilde{q}$ dla każdej z tych kolumn. Wpisujemy również numery tych kolumn do drzewa tree_Z. Stąd pętla będzie miała co robić już w pierwszym przejściu i wstawi wszystkie jedynki startowe.

Lemat 4.9 *Załóżmy, że wyznaczone pozycje startowe są takie, że w każdej pionowej kolumnie macierzy R jest co najmniej jedna jedynka. Wtedy procedura wypełniania działa prawidłowo.*

Dowód: Jeśli pętla procedury zwróci *fail*, to oznacza, że przy danych pozycjach startowych i macierzach rzutów prostopadłych, warunki jakie musi spełniać wypukłe trójwymiarowe poliomino są sprzeczne i takie poliomino nie istnieje.

Natomiast jeśli pętla procedury nie przerwała działania ze względu na sprzeczność, to po jej zakończeniu mamy dwa przypadki:
przypadek 1: Każda komórka macierzy R zawiera zero lub jedynkę. Mamy rozwiązanie. Z użytych w procedurze warunków wynika, że zbiór jedynek tworzy poliomino, realizujące macierze rzutów P_T, P_F i P_S, oraz każdy warstwa macierzy R zawiera dwuwymiarowe poliomino wypukłe.

przypadek 2: Pozostały w macierzy R komórki niewypełnione, nie zawierające ani zera ani jedynki. Z Lematu 4.6 wiemy jednak, że w każdej pionowej kolumnie mieliśmy wyznaczoną co najmniej jedną jedynkę. Z własności operacji $\otimes$ wynika, że w takiej kolumnie liczba pustych komórek jest równa dokładnie dwukrotności liczby brakujących jedynek. Stąd liczba niewypełnionych komórek w całej macierzy R jest równa dokładnie dwukrotności liczby brakujących jedynek.

W kolumnach które nie zawierają jedynek a mają niewypełnione miejsca, puste komórki tworzą dwa bloki, każdy o długości równej liczbie brakujących jedynek w tych kolumnach. Bloki te mogą sąsiadować ze sobą.

Na początek pogrupujmy puste komórki macierzy R w cykle, odpowiednio je przy tym etykietując. Jeśli weźmiemy pustą komórkę, to etykietujemy ją wolną zmienną np. x. Komórka ta będzie miała za sąsiadów w cyklu dokładnie trzy niewypełnione komórki, po jednej w każdej z trzech kolumn zawierających tą komórkę, które albo będą odległe od niej o wartość rzutu dla tej kolumny albo o wartość tego rzutu plus odległość między pustymi blokami komórek, jeśli w kolumnie nie ma żadnej jedynki. Te trzy komórki etykietujemy zaprzeczeniem x, tj. $\bar{x}$, i następnie dla każdej z nich wyznaczamy trzech następnych sąsiadów w cyklu (jednym z nich będzie na pewno komórka z której wyszliśmy) i nadajemy im etykietę x. Powtarzamy ten krok dla każdej nowozaetykietowanej komórki cyklu, aż nie znajdziemy, zgodnie z tymi warunkami, żadnej nowej komórki do zaetykietowania. W ten sposób tworzymy trójwymiarowy cykl w macierzy R, którego komórki są na przemian etykietowane przez x i $\bar{x}$, oraz każda komórka posiada w tym cyklu dokładnie trzech sąsiadów.

Takie cykle tworzymy dopóki nie zapełnimy wszystkich pustych komórek etykietami. Oczywiście każdy cykl etykietujemy inną zmienną.

Teraz tworzymy formułę 2SAT. Jeśli w jakiejś kolumnie mamy obok siebie dwie puste komórki zaetykietowane tymi zmiennymi, to do formuły 2SAT dodajemy implikację, że ze zmiennej zewnętrznej w stosunku do bloku jedynek w tej kolumnie wynika zmienna wewnętrzna. W kolumnie, która nie zawiera bloku jedynek, tworzymy cykl implikacji dla zmiennych każdego bloku osobno, tak aby wymusić zgodne etykietowanie całego bloku. Tak zbudowana formuła może być rozwiązana w czasie liniowym. Jej długość jest ograniczona przez ilość niewypełnionych komórek, czyli wynosi co najwyżej $O(n^3)$ (każda niewypełniona komórka uczestniczy w nie więcej niż ośmiu implikacjach). Stąd wartościowanie formuły 2SAT wyznacza trójwymiarowe pełne poliomino wypukłe.

Jeśli dla zadanych macierzy rzutów istnieje wypukłe poliomino zawierające wyznaczone pozycje startowe, to procedura wypełniania wygeneruje

poliomino realizujące wejściowe macierze rzutów (P_T, P_F, P_S). □

Teraz oszacujemy maksymalny koszt powyższej procedury.

Lemat 4.10 *Procedura wypełniania kosztuje co najwyżej* $O(n^3 \log n)$.

Dowód: Oszacujmy koszt głównej pętli procedury wypełniania. Dla każdej pozycji $[i, j, k]$ wykonujemy operację **wstaw** tylko trzy razy, po jednym w każdym z trzech kierunków. Co więcej, ilekroć odwiedzamy jakąś kolumnę wykonując operacje $\oplus, \ominus, \otimes, \odot$, wykonujemy co najmniej jedną operację **wstaw**. Stąd w sumie odwiedzamy co najwyżej $O(n^3)$ kolumn. Odwiedzenie każdej kolumny kosztuje sumę kosztu ściągnięcia numeru kolumny z drzewa oraz kosztu wykonanych operacji **wstaw**, czyli

$$O(\log n) + [\text{koszt operacji } \textbf{wstaw}].$$

Stąd ogólny koszt głównej pętli procedury wynosi

$$O(n^3 \log n) + [\text{koszt wszystkich operacji } \textbf{wstaw}].$$

Wyznaczmy więc ogólny koszt operacji **wstaw**. W danej kolumnie w trakcie wykonywania operacji **wstaw** możemy wywołać co najwyżej $2n$ operacji insert na drzewach tree. To kosztuje $O(n \log n)$. Dla wszystkich $3n^2$ kolumn kosztuje to więc co najwyżej $O(n^3 \log n)$.

Pozostają jeszcze operacje na drzewie free0. Dla każdej kolumny możemy wykonać co najwyżej n operacji insert i taką samą liczbę operacji delete. Funkcje min i max wywołujemy tylko wtedy, kiedy modyfikujemy zmienne $\tilde{p}$ i $\tilde{q}$, stąd ich liczba nie przekracza $2n$. Koszt tych operacji dla wszystkich kolumn nie przekracza więc $O(n^3 \log n)$.

Złożoność pozostałych operacji, w tym budowy i rozwiązania formuły 2SAT, nie przekracza $O(n^3)$.

Sumując powyższe koszty otrzymujemy, że całkowita złożoność procedury wypełniania wynosi $O(n^3 \log n)$. □

4.4 Złożoność rekonstrukcji

Twierdzenie 2 *Problem rekonstrukcji trójwymiarowych pełnych poliomin wypukłych z danych trzech macierzy rzutów prostopadłych może być rozwiązany w czasie* $O(n^7 \log n)$.

Dowód: Załóżmy, że istnieje pełne wypukłe poliomino S realizujące zadane macierze rzutów prostopadłych. Jeśli prawidłowo zgadniemy pozycje jedynek w kolumnach narożnych, to zgodnie z Lematem 4.7 wszystkie wyznaczone pozycje startowe są prawidłowe i należą do S. Stąd procedura wypełniania zgodnie z Lematem 4.9 musi nam zwrócić poprawne poliomino.

Aby trafić na właściwe pozycje w kolumnach narożnych możemy wypróbować wszystkie możliwości a jest ich co najwyżej n^4.

Jeśli poliomino o zadanych właściwościach i macierzach rzutów nie istnieje, to procedura wypełniania odpowie *fail* w każdym przypadku.

Koszt całego algorytmu jest więc równy iloczynowi możliwych punktów startowych, czyli $O(n^4)$, i kosztów algorytmu, czyli wyznaczenia pozycji startowych i procedury wypełniania, co wynosi $O(n^3 \log n)$. Stąd całkowity koszt rekonstrukcji wynosi $O(n^7 \log n)$. □

Tak więc problem rekonstrukcji zdefiniowanej w tym rozdziale klasy trójwymiarowych poliomin należy do P.

Rozdział 5

Rekonstrukcja wypukłych poliomin z przybliżonych prostopadłych rzutów.

W tym rozdziale rozwiążemy problem sformułowany przez G.J. Woeginger w [GN97]. Pokażemy istnienie wielomianowego algorytmu, który mając jako wejście wektor rzutów poziomych $\tilde{H} \in \mathbb{R}_+^m$ i wektor rzutów pionowych $\tilde{V} \in \mathbb{R}_+^n$ odpowiada na pytanie, czy istnieje poliomino S z przybliżonymi rzutami $H^* \in \{1,\ldots,n\}^m$ i $V^* \in \{1,\ldots,n\}^m$ takimi, że H^* i V^* są przybliżeniem odpowiednio $\tilde{H}$ i $\tilde{V}$.

Rozważymy dwie wersje pojęcia „przybliżenie"

(1) przybliżenie z błędem absolutnym – każdy współczynnik pary wektorów $(\tilde{H}, \tilde{V})$ różni się o co najwyżej jeden od odpowiadającego mu współczynnika wektorów (H^*, V^*), lub

(2) przybliżenie z błędem logarytmicznym – dla każdego współczynnika $\tilde{h}_i$ i $\tilde{v}_j$ mamy spełnione nierówności

$$|\tilde{h}_i - h_i^*| \leqslant \log(\tilde{h}_i + 1) \qquad \text{i} \qquad |\tilde{v}_j - v_j^*| \leqslant \log(\tilde{v}_j + 1),$$

dla $i = 1,\ldots,m$ i $j = 1,\ldots,n$.

Ponadto, dla rekonstrukcji wypukłego poliomina uogólnimy funkcję błędu. Będziemy rekonstruować wypukłe poliomino w ten sposób, że dla każdego wejściowego rzutu p będziemy wymagać, aby wyjściowy rzut p' należał do ustalonego zbioru (przedziału) $[\check{p}, \hat{p}]$, gdzie $\check{p} \leqslant p \leqslant \hat{p}$.

5.1 Trudność problemu

W tym podrozdziale przedstawimy redukcję problemu rekonstrukcji poliomina z dokładnych rzutów prostopadłych (dokł-RP) do problemu rekonstrukcji poliomina z przybliżonych rzutów prostopadłych (przybl-RP). Niech

$$\begin{aligned} H &= (h_1, \ldots, h_m) \in \{1, \ldots, n\}^m \qquad \text{i} \\ V &= (v_1, \ldots, v_n) \in \{1, \ldots, m\}^n \end{aligned}$$

będzie ustalonym przykładem problemu dokł-RP. Ponadto zakładamy, że

$$\sum_{i=1}^{m} h_i = \sum_{j=1}^{n} v_j,$$

gdyż w przeciwnym przypadku poliomino z rzutami (H, V) nie istnieje. Z powyższego przykładu chcemy skonstruować wektor rzutów poziomych $\tilde{H} = (\tilde{h}_1, \ldots, \tilde{h}_m) \in \mathbb{R}_+^m$ i wektor rzutów pionowych $\tilde{V} = (\tilde{v}_1, \ldots, \tilde{v}_n) \in \mathbb{R}_+^n$ dla problemu przybl-RP, odpowiednio dla rodzaju rozpatrywanego błędu.

Definicja 5.1 *Dla przybliżenia z błędem absolutnym konstruujemy*

$$\begin{aligned} \tilde{h}_i &= h_i - \frac{1}{2}, \qquad \textit{dla } i \in \{1, \ldots, m\} \textit{ i} \\ \tilde{v}_j &= v_j + \frac{1}{2}, \qquad \textit{dla } j \in \{1, \ldots, n\}. \end{aligned}$$

Oraz dla przybliżenia z błędem logarytmicznym definiujemy $\tilde{h}_i$ takie, że

$$h_i \leqslant \tilde{h}_i + \log(\tilde{h}_i + 1) < h_i + 1$$

oraz $\tilde{v}_j$ takie, że

$$v_j - 1 < \tilde{v}_j - \log(\tilde{v}_j + 1) \leqslant v_j.$$

(Wybór jest zawsze możliwy, ponieważ funkcje $x - \log(x+1)$ oraz $x + \log(x+1)$ są ciągłe i ściśle monotoniczne na $\mathbb{R}_+$.)

Lemat 5.2 *Jeśli istnieje poliomino S takie, że $H^* = H(S)$ i $V^* = V(S)$ i wektory (H^*, V^*) są przybliżeniem z błędem absolutnym (logarytmicznym) wektorów $(\tilde{H}, \tilde{V})$, wtedy istnieje poliomino z rzutami (H, V).*

Dowód: Dla poliomina S następujące własności są prawdziwe:

(i) $\sum_i h_i^* = \sum_j v_j^*$ (sumy są równe liczbie jedynek w poliominie S), i

(ii) $|h_i^* - \tilde{h}_i| \leqslant 1$ i $|v_j^* - \tilde{v}_j| \leqslant 1$, w przypadku rozważania błędu absolutnego, lub $|h_i^* - \tilde{h}_i| \leqslant \log(\tilde{h}_i + 1)$ i $|v_j^* - \tilde{v}_j| \leqslant \log(\tilde{v}_j + 1)$ dla błędu logarytmicznego.

Lecz z Definicji 5.1 powyższe własności zachodzą wtedy i tylko wtedy, gdy dla wszystkich i h_i^* jest równe maksymalnej dopuszczalnej wartości, to jest w pierwszym przypadku

$$h_i^* = \lfloor \tilde{h}_i + 1 \rfloor = h_i$$

lub w drugim przypadku

$$h_i^* = \lfloor \tilde{h}_i + \log(\tilde{h}_i + 1) \rfloor = h_i.$$

Oraz gdy dla wszystkich j v_j^* jest równe minimalnej dopuszczalnej wartości czyli odpowiednio

$$v_j^* = \lceil \tilde{v}_j - 1 \rceil = v_j$$

lub

$$v_j^* = \lceil \tilde{v}_j - \log(\tilde{v}_j + 1) \rceil = v_j.$$

Stąd S realizuje również (H, V). □

Lemat 5.3 *Jeśli istnieje poliomino S z rzutami prostopadłymi (H, V), wtedy istnieje poliomino z rzutami poziomymi $H^* \in \{1, \ldots, n\}^m$ i rzutami pionowymi $V^* \in \{1, \ldots, m\}^n$ takimi, że (H^*, V^*) są przybliżeniem z błędem absolutnym (logarytmicznym) wektorów $(\tilde{H}, \tilde{V})$.*

Dowód: Z definicji wektorów $(\tilde{H}, \tilde{V})$ (dla obu wersji błędów) mamy, że każdy współczynnik $(\tilde{H}, \tilde{V})$ może być zaokrąglony do odpowiedniego współczynnika (H, V). Stąd S jest również realizacją poliomina spełniającego $(\tilde{H}, \tilde{V})$ w odpowiednim błędem. □

Ponieważ wiemy, że problem rekonstrukcji poliomin z dokładnych rzutów prostopadłych dla poliomin bez warunku spójności kolumn i wierszy, poliomin ze spójnymi wierszami i poliomin ze spójnymi kolumnami są NP-zupełne (zobacz [BdLNP96a, W96]) otrzymujemy z Lematu 5.2 i Lematu 5.3 następujące twierdzenie

Twierdzenie 3 *Problemy rekonstrukcji poliomin, poliomin ze spójnymi wierszami i poliomin ze spójnymi kolumnami z przybliżonych prostopadłych rzutów są NP-zupełne.* □

5.2 Rekonstrukcja wypukłych poliomin

W algorytmie opisanym poniżej uogólnimy pojęcie błędu przybliżenia i założymy, że ma on formę funkcji f, która jest dodatnia na $\mathbb{R}_+$. Dla przykładu, rozważany błąd absolutny jest zadany stałą funkcją równą 1, a błąd logarytmiczny funkcją $f(x) = \log(x+1)$.

Definicja 5.4 *Definiujemy ciąg pomocniczych wyrażeń*

$$\begin{aligned} \check{v}_j &= \max\{1, \lceil \tilde{v}_j - f(\tilde{v}_j) \rceil\}, \\ \hat{v}_j &= \min\{m, \lfloor \tilde{v}_j + f(\tilde{v}_j) \rfloor\}, \end{aligned}$$

dla $j \in \{1, \ldots, n\}$, *oraz*

$$\begin{aligned} \check{h}_i &= \max\{1, \lceil \tilde{h}_i - f(\tilde{h}_i) \rceil\}, \\ \hat{h}_i &= \min\{n, \lfloor \tilde{h}_i + f(\tilde{h}_i) \rfloor\}, \end{aligned}$$

dla $i \in \{1, \ldots, m\}$.

Powyższe wyrażenia mają następujące własności: $\check{h}_i$ jest minimalną dodatnią liczbą naturalną, która dla rzutu poziomego h_i różni się o co najwyżej o $f(h_i)$. $\hat{h}_i$ jest maksymalną taką dodatnią liczbą naturalną. Analogicznie, $\check{v}_j$ i $\hat{v}_j$ są odpowiednio minimalną i maksymalną dopuszczalną wartością rzutu pionowego ze względu na funkcję błędu f. Stąd możemy rozważać ogólnie rekonstrukcję z rzutami należącymi do odpowiedniego ustalonego przedziału dla każdej kolumny i wiersza.

Definicja 5.5 *Wypukłe poliomino S jest zakotwiczone w (p, q, r, s), jeśli*

$$[1, p], [q, n], [m, r], [s, 1] \in S$$

(to jest powyższe komórki nie należą do żadnego obszaru narożnego).

Główną ideą naszego algorytmu jest, mając cztery wektory rzutów

$\check{H} \in \{1, .., n\}^m$ - wektor dolnych ograniczeń dla rzutów poziomych,

$\check{V} \in \{1, .., m\}^n$ - wektor dolnych ograniczeń dla rzutów pionowych,

$\hat{H} \in \{1, .., n\}^m$ - wektor górnych ograniczeń dla rzutów poziomych i

$\hat{V} \in \{1, .., m\}^n$ - wektor górnych ograniczeń dla rzutów pionowych,

skonstruować formułę 2SAT

$$F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$$

taką, że $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$ jest spełnialna wtedy i tylko wtedy, gdy istnieje wypukłe poliomino S zakotwiczone w (p,q,r,s) i mające prostopadłe rzuty (H,V) takie, że każdy współczynnik (H,V) należy do przedziału zdefiniowanego przez dolne i górne ograniczenie wyznaczone przez odpowiednie współczynniki wektorów $(\check{H},\hat{H},\check{V},\hat{V})$. Użyjemy ponadto Definicji 2.1, Lematu 2.2 i Lematu 2.3.

Definicja 5.6 *Definiujemy następujące zbiory klauzul 2SAT dla ustalonych wektorów $(\check{H},\hat{H},\check{V},\hat{V})$ i zakotwiczenia (p,q,r,s)*

„Rogi" – *obszary narożne z Definicji 2.1*

$$Cor \equiv \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow A_{i-1,j} & A_{i,j} \Rightarrow A_{i,j-1} \\ B_{i,j} \Rightarrow B_{i-1,j} & B_{i,j} \Rightarrow B_{i,j+1} \\ C_{i,j} \Rightarrow C_{i+1,j} & C_{i,j} \Rightarrow C_{i,j-1} \\ D_{i,j} \Rightarrow D_{i+1,j} & D_{i,j} \Rightarrow D_{i,j+1} \end{array} \right\}$$

„Spójność" – *spójność wypukłego poliomina S z Lematu 2.3*

$$Con \equiv \bigwedge_{i,j} \{A_{i,j} \Rightarrow \overline{D}_{i+1,j+1} \quad B_{i,j} \Rightarrow \overline{C}_{i+1,j-1}\}$$

„Zakotwiczenie" – *zakotwiczenie w (p,q,r,s) z Definicji 5.5*

$$Anc_{p,q,r,s} \equiv \bigwedge \left\{ \begin{array}{llll} \overline{A}_{1,p} & \overline{B}_{1,p} & \overline{C}_{1,p} & \overline{D}_{1,p} \\ \overline{A}_{q,n} & \overline{B}_{q,n} & \overline{C}_{q,n} & \overline{D}_{q,n} \\ \overline{A}_{m,r} & \overline{B}_{m,r} & \overline{C}_{m,r} & \overline{D}_{m,r} \\ \overline{A}_{s,1} & \overline{B}_{s,1} & \overline{C}_{s,1} & \overline{D}_{s,1} \end{array} \right\}$$

„Dolne ograniczenie na sumy w kolumnach" *(LBC) – dla wszystkich kolumn wyznaczamy minimalną odległość między rejonami narożnymi (dla kolumny j jest ona równa $\check{v}_j$)*

$$LBC \equiv \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow \overline{C}_{i+\check{v}_j,j} & A_{i,j} \Rightarrow \overline{D}_{i+\check{v}_j,j} \\ B_{i,j} \Rightarrow \overline{C}_{i+\check{v}_j,j} & B_{i,j} \Rightarrow \overline{D}_{i+\check{v}_j,j} \end{array} \right\}$$

$$\wedge \bigwedge_{j} \left\{ \begin{array}{l} \overline{C}_{\check{v}_j,j} \\ \overline{D}_{\check{v}_j,j} \end{array} \right\}$$

„Górne ograniczenie na sumy w kolumnach" *(UBC) – dla wszystkich kolumn wyznaczamy maksymalną odległość między rejonami narożnymi (dla kolumny j jest ona równa $\hat{v}_j$)*

$$UBC_{p,r} \equiv \bigwedge_i \left\{ \begin{array}{ll} \bigwedge\limits_{j \leqslant \min\{p,r\}} & \overline{A}_{i,j} \Rightarrow C_{i+\hat{v}_j,j} \\ \bigwedge\limits_{p \leqslant j \leqslant r} & \overline{B}_{i,j} \Rightarrow C_{i+\hat{v}_j,j} \\ \bigwedge\limits_{r \leqslant j \leqslant p} & \overline{A}_{i,j} \Rightarrow D_{i+\hat{v}_j,j} \\ \bigwedge\limits_{\max\{p,r\} \leqslant j} & \overline{B}_{i,j} \Rightarrow D_{i+\hat{v}_j,j} \end{array} \right\}$$

„Dolne ograniczenie na sumy w wierszach" *(LBR) – dla wszystkich wierszy wyznaczamy minimalną odległość między rejonami narożnymi (dla wiersza i jest ona równa $\check{h}_i$)*

$$LBR \equiv \bigwedge_{i,j} \left\{ \begin{array}{ll} A_{i,j} \Rightarrow \overline{B}_{i,j+\check{h}_i} & A_{i,j} \Rightarrow \overline{D}_{i,j+\check{h}_i} \\ C_{i,j} \Rightarrow \overline{B}_{i,j+\check{h}_i} & C_{i,j} \Rightarrow \overline{D}_{i,j+\check{h}_i} \end{array} \right\}$$

$$\wedge \bigwedge_i \left\{ \begin{array}{l} \overline{B}_{i,\check{h}_i} \\ \overline{D}_{\check{h}_i} \end{array} \right\}$$

„Górne ograniczenie na sumy w wierszach" *(UBR) – dla wszystkich wierszy wyznaczamy maksymalną odległość między rejonami narożnymi (dla wiersza i jest ona równa $\hat{h}_i$)*

$$UBR_{s,q} \equiv \bigwedge_j \left\{ \begin{array}{ll} \bigwedge\limits_{i \leqslant \min\{s,q\}} & \overline{A}_{i,j} \Rightarrow B_{i,j+\hat{h}_i} \\ \bigwedge\limits_{s \leqslant i \leqslant q} & \overline{C}_{i,j} \Rightarrow B_{i,j+\hat{h}_i} \\ \bigwedge\limits_{q \leqslant j \leqslant s} & \overline{A}_{i,j} \Rightarrow D_{i,j+\hat{h}_i} \\ \bigwedge\limits_{\max\{s,q\} \leqslant j} & \overline{C}_{i,j} \Rightarrow D_{i,j+\hat{h}_i} \end{array} \right\}$$

Wszystkie zmienne z indeksami z poza zbioru $\{1,\ldots,m\} \times \{1,\ldots,n\}$ są wartościowane jako 1 (true).

Zwartościowanie zmiennej $X_{i,j}$ jako true oznacza, że pozycja $[i,j]$ należy do zbioru X, a zwartościowanie jako false oznacza, że $[i,j] \notin X$.

Definicja 5.7

$$F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V}) = Cor \wedge Con \wedge Anc_{p,q,r,s} \wedge LBC \wedge UBC_{p,r} \wedge LBR \wedge UBR_{q,s}.$$

Teraz możemy napisać algorytm rekonstrukcji.

Input: $\check{H} \in \{1,\ldots,n\}^m$, $\hat{H} \in \{1,\ldots,n\}^m$,
$\check{V} \in \{1,\ldots,m\}^n$, $\hat{V} \in \{1,\ldots,m\}^n$

`for all` $p,r \in \{1,\ldots,n\}$ `and` $q,s \in \{1,\ldots,m\}$ `do`
  `if` $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$ jest spełnialna `then`
    `return` $S = Q \setminus (A \cup B \cup C \cup D)$ `and halt`
`return` „NO”

Lemat 5.8 *Formuła $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$ jest spełnialna wtedy i tylko wtedy, gdy S jest wypukłym poliominem zakotwiczonym w (p,q,r,s), z prostopadłymi rzutami (H,V) takimi, że $\check{h}_i \leqslant h_i \leqslant \hat{h}_i$, dla wszystkich $i \in \{1,\ldots,m\}$, i $\check{v}_j \leqslant v_j \leqslant \hat{v}_j$, dla wszystkich $j \in \{1,\ldots,n\}$.*

Dowód: ($\Leftarrow$) Jeśli S jest wypukłym poliominem z własnościami jak w lemacie, wtedy niech A, B, C i D będą obszarami narożnymi z Definicji 2.1. Z Lematu 2.3 A, B, C i D spełniają warunki „Rogi” i „Spójność”. Warunek „Zakotwiczenie” jest prawdziwy, ponieważ S jest zakotwiczone w (p,q,r,s). Ponadto dla wszystkich $i \in \{1,\ldots,m\}$ mamy $\check{h}_i \leqslant h_i \leqslant \hat{h}_i$, a stąd warunki LBR i UBR są spełnione. Analogicznie, dla wszystkich $j \in \{1,\ldots,n\}$ mamy $\check{v}_j \leqslant v_j \leqslant \hat{v}_j$, a stąd warunki LBC i UBC są spełnione. Stąd cała formuła $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$ jest spełniona.

($\Rightarrow$) Jeśli formuła $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$ jest spełniona, to weźmy

$$S = Q \setminus (A \cup B \cup C \cup D).$$

Z warunków „Rogi”, „Spójność”, LBC i LBR mamy, że zbiory A, B, C i D spełniają Lemat 2.2 (warunki LBC i LBR gwarantują tutaj rozłączność obszarów narożnych). Stąd zbiór S jest wypukłym poliominem. Ponadto z warunku „Zakotwiczenie” mamy, że S jest zakotwiczone w (p,q,r,s). Z warunków LBR i UBR mamy, że dla każdego wiersza i, gdzie $i \in \{1,\ldots,m\}$, liczba jedynek w tym wierszu, czyli h_i spełnia warunek $\check{h}_i \leqslant h_i \leqslant \hat{h}_i$. Analogicznie z warunków LBC i UBC mamy, że $\check{v}_j \leqslant v_j \leqslant \hat{v}_j$ dla każdej kolumny j, gdzie $j \in \{1,\ldots,n\}$. Stąd S musi być wypukłym poliominem zakotwiczonym w (p,q,r,s), z prostopadłymi rzutami (H,V) takimi, że $\check{h}_i \leqslant h_i \leqslant \hat{h}_i$,

dla wszystkich $i \in \{1, \ldots, m\}$, i $\check{v}_j \leqslant v_j \leqslant \hat{v}_j$, dla wszystkich $j \in \{1, \ldots, n\}$. □

Teraz możemy udowodnić następujące twierdzenie

Twierdzenie 4 *Jeśli mamy cztery wektory:*

$\check{H} \in \{1, .., n\}^m$ - *wektor dolnych ograniczeń na rzuty poziome,*

$\check{V} \in \{1, .., m\}^n$ - *wektor dolnych ograniczeń na rzuty pionowe,*

$\hat{H} \in \{1, .., n\}^m$ - *wektor górnych ograniczeń na rzuty poziome i*

$\hat{V} \in \{1, .., m\}^n$ - *wektor górnych ograniczeń na rzuty pionowe,*

to rekonstrukcja wypukłego poliomina S z prostopadłymi rzutami (H, V) takimi, że $h_i(S) \in \{\check{h}_i, \ldots, \hat{h}_i\}$, dla $i \in \{1, \ldots, m\}$, oraz $v_j(S) \in \{\check{v}_j, \ldots, \hat{v}_j\}$, dla $j \in \{1, .., n\}$, kosztuje co najwyżej $O(m^3 n^3)$.

Dowód: Algorytm testuje co najwyżej $m^2 n^2$ formuł $F_{p,q,r,s}(\check{H}, \hat{H}, \check{V}, \hat{V})$. Każda taka formuła ma rozmiar $O(mn)$ i jest konstruowana w czasie liniowym. Ponieważ formuły 2SAT mogą być rozwiązane w czasie liniowym względem swojej długości, otrzymujemy złożoność algorytmu.

Lemat 5.8 gwarantuje poprawność algorytmu. □

Twierdzenie 5 *Problem rekonstrukcji wypukłych poliomin z przybliżonych prostopadłych rzutów może być rozwiązany w czasie $O(m^3 n^3)$.* □

5.3 Algorytm równoległy

W tym podrozdziale opiszemy krótko równoległy algorytm oparty o wyniki z poprzedniego podrozdziału, czyli możliwość zakodowania problemu za pomocą zbioru formuł 2SAT. Mając już formuły 2SAT możemy posłużyć się algorytmem równoległym opisanym w [CL88, Ch92], który wyznacza wartościowanie zmiennych spełniające ustaloną formułę 2SAT, o ile takie wartościowanie istnieje. Następujące twierdzenie mówi o złożoności tego algorytmu

Twierdzenie 6 *[Ch92] Problem znalezienia wartościowania spełniającego ustaloną formułę 2SAT o długości n może być rozwiązany w czasie $O(\log n)$ przy pomocy $O(n^3)$ procesorów na maszynie CRCW PRAM albo w czasie $O(\log^2 n)$ za pomocą $O(n^{2.376})$ procesorów na maszynie EREW PRAM.* □

Użyta w twierdzeniu liczba procesorów dla EREW PRAM jest złożonością najszybszego sekwencyjnego mnożenia dwóch macierzy o rozmiarze n^2.

Sam algorytm będzie miał następującą postać

Input: $\check{H} \in \{1,\ldots,n\}^m$, $\hat{H} \in \{1,\ldots,n\}^m$,
$\check{V} \in \{1,\ldots,m\}^n$, $\hat{V} \in \{1,\ldots,m\}^n$

`for all` $p,r \in \{1,\ldots,n\}$ `and` $q,s \in \{1,\ldots,m\}$ `do parallel`
`(1)` skonstruuj formułę $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$
`(2)` znajdź wartościowanie zmiennych spełniające formułę
$F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$
`if` istnieje spełnialna formuła `then`
wybierz jedno rozwiązanie `and return` $S = Q \setminus (A \cup B \cup C \cup D)$
`else`
`return` „NO”

Najpierw rozważmy złożoność naszego algorytmu na maszynie CRCW PRAM. Łatwo zauważyć, że każdą formułę $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$, z Definicji 5.7, budujemy w czasie stałym przy użyciu $O(mn)$ procesorów. A więc uzyskanie wszystkich m^2n^2 formuł jest możliwe w tym samym czasie za pomocą $O(m^3n^3)$ procesorów.

Teraz stosując algorytm z Twierdzenia 6 dla każdej formuły znajdujemy spełniające ją wartościowanie zmiennych, jeśli takie istnieje, w czasie $O(\log mn)$ za pomocą $O(m^3n^3)$ procesorów, czyli dla wszystkich formuł za pomocą $O(m^5n^5)$ procesorów.

Następnie w czasie stałym wybieramy jedno z rozwiązań, na przykład to o najniższym numerze. Można to zrobić w czasie stałym używając co najwyżej $O(m^4n^4)$ procesorów.

Na koniec, dla wybranego rozwiązania formuły 2SAT, generujemy poliomino w czasie stałym za pomocą $O(mn)$ procesorów.

Cały algorytm wykonuje więc na CRCW PRAM $O(\log nm)$ kroków wykorzystując $O(m^5n^5)$ procesorów.

W przypadku EREW PRAM mamy trochę trudniejsze zadanie. W pierwszym kroku musimy stworzyć m^2n^2 kopii wektorów $(\check{H},\hat{H},\check{V},\hat{V})$, co uzyskujemy w czasie $O(\log mn)$ za pomocą $O(m^3n^3)$ procesorów. Następnie dla każdej kopii tworzymy odpowiednią formułę 2SAT $F_{p,q,r,s}(\check{H},\hat{H},\check{V},\hat{V})$. Każdą taką formułę możemy utworzyć w czasie $O(\log mn)$ za pomocą $O(mn)$ procesorów. W sumie więc cały ten krok wymaga $O(m^3n^3)$ procesorów i czasu $O(\log mn)$.

Teraz każdą formułę rozwiązujemy stosując algorytm z Twierdzenia 6 w czasie $O(\log^2 mn)$ używając ogółem $O(m^{4.376}n^{4.376})$ procesorów.

Następnie wyznaczamy jedno z rozwiązań, jeśli takie istnieje, lub stwierdzamy, że rozwiązanie nie istnieje, w czasie $(\log mn)$ za pomocą co najwyżej m^2n^2 procesorów.

Dla wybranego rozwiązania formuły 2SAT możemy wygenerować poliomino w czasie stałym za pomocą mn procesorów.

Tak więc ogólny czas pracy algorytmu na maszynie EREW PRAM wynosi $O(\log^2 mn)$ przy wykorzystaniu $O(m^{4.376}n^{4.376})$ procesorów.

Z powyższych rozważań otrzymujemy następujące twierdzenie

Twierdzenie 7 *Problem rekonstrukcji wypukłego poliomina z rzutów prostopadłych spełniających wektory górnych i dolnych ograniczeń* $(\check{H}, \check{V}, \hat{H}, \hat{V})$, *może być rozwiązany w czasie* $O(\log mn)$ *przy pomocy* $O(m^5n^5)$ *procesorów na CRCW PRAM albo w czasie* $O(\log^2 mn)$ *za pomocą* $O(m^{4.376}n^{4.276})$ *procesorów na EREW PRAM.* □

Wniosek 5.9 *Problem rekonstrukcji poliomin wypukłych należy do klasy* NC_1. □

Rozdział 6

Otwarte problemy

6.1 Liczba dwuwymiarowych poliomin wypukłych

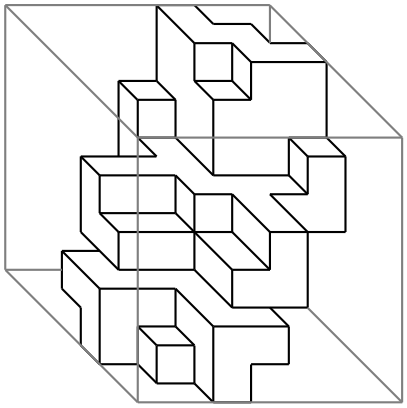
Jak już wspomnieliśmy w rozdziale 1, para rzutów prostopadłych nie wyznacza nam jednoznacznie dwuwymiarowego poliomina wypukłego. Jak pokazali Del Lungo, Nivat i Pinzani w [dLNP96] liczba możliwych wypukłych poliomin o identycznych rzutach poziomym i pionowym może być nawet wykładnicza.

Czasami jednak chcielibyśmy znać, jeśli już nie wszystkie rozwiązania szczegółowo, to przynajmniej ich liczbę. Opisane w rozdziałach 3 i 5 algorytmy rekonstrukcji poliomin wypukłych mogłyby być podstawą do próby wyznaczenia tej liczby. Łatwo zauważyć, że byłaby ona wówczas równa, przy odpowiednich modyfikacjach, sumie ilości rozwiązań zbioru pewnych formuł 2SAT uzyskanych w tych algorytmach. Tak więc mamy tu do czynienia z redukcją do problemu obliczenia liczby rozwiązań pewnej szczególnej formuły 2SAT.

Problem obliczenia liczby rozwiązań dla ogólnych formuł 2SAT jest #P-zupełny (zobacz [V79a]). Również jeśli ograniczymy się do pozytywnego 2SAT ([V79b]) czy hornowskiego pozytywnego 2SAT ([L86]) to są one #P-zupełne. Niestety formuły uzyskiwane w obu algorytmach wydają się prostsze niż wymienione podklasy, w związku z czym wydaje się, że redukcja tych podklas do problemu rekonstrukcji poliomin wypukłych nie jest możliwa.

Z kolei najbliższym nie #P-zupełnym problemem, który moglibyśmy tutaj rozważyć jest pozytywny hornowski 2SAT, dla którego istnieje planarny graf skierowany opisujący formułę. Tutaj jednak również nie udało nam się uzyskać redukcji problemu rekonstrukcji do tego problemu.

Tak więc problem czy ilość rozwiązań problemu rekonstrukcji dwuwy-



P_T

	1					n
1	0	0	0	2	3	1	0
	0	0	0	1	4	2	1
	0	2	2	2	3	4	3
	1	2	3	3	2	1	2
	2	2	2	3	2	0	0
	0	1	2	2	2	0	0
n	0	0	2	1	0	0	0

P_S

	1					n
1	1	2	2	0	0	0	0
	2	3	3	0	0	0	0
	2	1	3	1	0	0	0
	1	2	5	4	1	1	0
	0	0	3	4	2	2	0
	0	0	0	4	4	2	2
n	0	0	0	1	4	2	1

P_F

	1					n
1	0	0	0	0	2	2	1
	0	0	0	0	3	3	2
	0	0	0	1	3	1	2
	0	1	1	4	5	2	1
	0	2	2	4	3	0	0
	2	2	4	4	0	0	0
n	1	2	4	1	0	0	0

Rysunek 6.1: Przykład trójwymiarowego wypukłego poliomina wraz z macierzami rzutów

miarowych poliomin wypukłych należy do klasy #P-zupełnych jest nadal otwarty.

Oczywiście, gdyby udało się znaleźć redukcję ogólnego problemu 2SAT do rekonstrukcji wypukłego poliomina, to oprócz tego, że problem liczby rozwiązań byłby #P-zupełny, moglibyśmy uzyskać NL-zupełność problemu rekonstrukcji, o ile algorytm redukcji należałby do klasy L, czyli byłby wykonalny w pamięci logarytmicznej. Wynika to z NL-zupełności problemu 2SAT.

6.2 Trójwymiarowe poliomina wypukłe

W rozdziale 4 omówiliśmy algorytm rekonstrukcji pewnej szczególnej klasy trójwymiarowych poliomin wypukłych. Otwartym problemem pozostaje jednak istnienie wielomianowego algorytmu rekonstrukcji w przypadku, gdy zrezygnujemy z warunku istnienia pełnej macierzy rzutów, zostawiając tylko warunki na kształt przekrojów, czyli dla zdefiniowany poniżej trójwymiarowych poliomin wypukłych

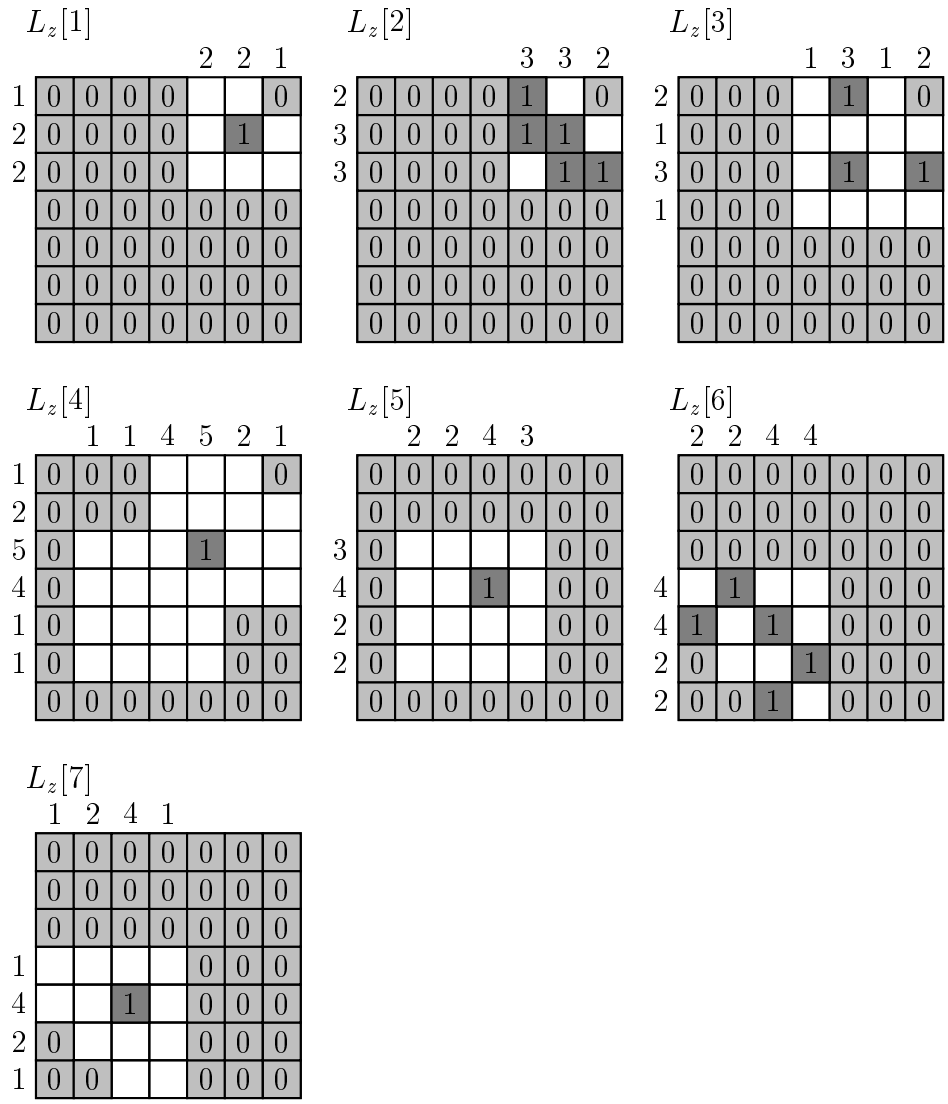
Definicja 6.1 *Trójwymiarowy zbiór $S = S(R)$ nazywamy poliominem wypukłym, jeśli jest spójny i każda warstwa macierzy R zawiera dwuwymiarowe poliomino wypukłe.*

Przykład takiego poliomina mamy na rysunku 6.1. Innym, prostym przykładem mógłby być obiekt zbliżony do kuli wpisany w sześcienną macierz, której wszystkie trzy macierze rzutów miałyby zera w obszarach narożnych.

W algorytmie opisanym w rozdziale 4 punktami startowymi rekonstrukcji były narożne kolumny prostopadłe do pełnej macierzy rzutów. Tutaj niestety te narożne kolumny mogą być puste, ponieważ żadna z macierzy rzutów nie musi być pełna.

Na początek rozważmy więc, jakie komórki możemy dla wypukłego trójwymiarowego poliomina zawsze wypełnić na podstawie znajomości macierzy rzutów. Po pierwsze kolumny, których rzut jest równy zero, możemy od razu w całości wypełnić zerami, ograniczając w ten sposób obszar rekonstrukcji. Ponadto, ponieważ każda warstwa macierzy R ma zawierać dwuwymiarowe poliomino wypukłe, więc zgodnie z Lematem 2.13 dla każdej takiej warstwy możemy wyznaczyć co najmniej jedną jedynkę, czyli medianę. Na rysunku 6.2 mamy zaznaczone wszystkie powyższe punkty dla przykładowego poliomina z rysunku 6.1.

Oczywiście, korzystając z Wniosku 2.19, jeśli w którejś warstwie oprócz mediany pojawi się jakaś inna jedynka, możemy je połączyć ciągiem jedynek.



Rysunek 6.2: Poziome warstwy poliomina z rysunku 6.1 z wstawionymi zerami w pustych wierszach i medianami wszystkich warstw, tj. jedynkami

Również dla median warstw brzegowych możemy wypełnić całą prostopadłą do tej warstwy kolumnę zawierającą tą jedynkę (obszar jedynek jest spójny i zaczyna się od skraju kolumny, więc jest wyznaczony na podstawie wartości rzutu jednoznacznie).

Niestety zbiór powyżej opisanych wypełnionych komórek często okazuje się być niewystarczający do prawidłowego funkcjonowania procedury wypełniania, czyli do pełnej rekonstrukcji. Powstaje więc pytanie, czy istnieje łatwy do wyznaczenia, dodatkowy, skończony zbiór komórek macierzy, który odpowiednio wypełniony zapewni nam prawidłowe działanie procedury wypełniania.

Co do samej procedury wypełniania, to można poszerzyć jej możliwości korzystając z Wniosku 2.19 (łączenie obszarów jedynek występujących w jednej warstwie poliomina) i Lematu 2.20 (wstawianie zer w obszarach narożnych warstw).

Bibliografia

[BdLNP96a] Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Reconstructing Convex Polyominoes from Horizontal and Vertical Projections. Theoretical Computer Science **155** (1996) 321–347 7, 9, 55

[BdLNP96b] Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Reconstructing Convex Polyominoes from Horizontal and Vertical Projections II. (1996) Preprint

[BdLPS96] Barcucci, E., Del Lungo, A., Pinzani, R., Sprugnoli R.: Polyominoes defined by their vertical and horizontal projections. Theoretical Computer Science **159** (1996) 129–136

[Ch71] Chang, S.K.: The Reconstruction of Binary Patterns from their Projections. Communications of the ACM **14** (1971) 21–25 7

[Ch92] Chen, Z.: A fast and efficient parallel algorithm for finding a satisfying truth assigment to a 2-CNF formula. Information Processing Letters **43** (1992) 191–193 60

[ChD99] Chrobak, M., Dürr, Ch.: Reconstructing hv-Convex Polyominoes from Orthogonal Projections. Information Processing Letters **69** (1999) 283–289 9, 12

[CL88] Cook, S., Luby, M.: A simple parallel algorithm for finding a satisfying truth assigment to a 2-CNF formula. Information Processing Letters **27** (1988) 141–145 60

[G98] Gębala, M.: The Reconstruction of Convex Polyominoes from Horizontal and Vertical Projections. Proceedings of the 25th Annual Conference on Current Trends in Theory and Practice of Informatics, SOFSEM'98, Lecture Notes of Computer Science **1521** (1998) 350–359 9

[G01] Gębala, M.: The Reconstruction of Polyominoes from Approximately Orthogonal Projections. Proceedings of the 25th Annual Conference on Current Trends in Theory and Practice of Informatics, SOFSEM'2001, Lecture Notes of Computer Science 9

[GN97] Gritzmann, P., Nivat, M. (editors): Discrete Tomography: Algorithms and Complexity. Dagstuhl-Seminar-Report; 165 (1997) 9, 53

[L86] Linial, N.: Hard enumeration problems in geometry and combinatorics. SIAM Journal on Algebraic and Discrete Methods **7** (1986) 331–335 63

[dL94] Del Lungo, A.: Polyominoes defined by two vectors. Theoretical Computer Science **127** (1994) 187–198

[dLNP96] Del Lungo, A., Nivat, M., Pinzani, R.: The number of convex polyominoes reconstructible from their orthogonal projections. Discrete Mathematics **157** (1996) 65–78 7, 63

[dLNPS98] Del Lungo, A., Nivat, M., Pinzani, R., Sorri, L.: The medians of discrete sets. Information Processing Letters **65** (1998) 293–299 15

[MS96] Métivier, Y., Saheb, N.: Medians and centres of polyominoes. Information Processing Letters **57** (1996) 175–181

[R63] Ryser, H.: Combinatorial Mathematics. The Carus Mathematical Monographs vol. 14 (The Mathematical Association of America, Rahway, 1963) 7

[V79a] Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science **8** (1979) 189–201 63

[V79b] Valiant, L.G.: The complexity of enumeration and reliability problem. SIAM Journal on Computing **8** (1979) 410–421 63

[W75] Wang, X.G.: Characterisation of Binary Patterns and their Projections. IEEE Transactions on Computers **C-24** (1975) 1032–1035 7

[W96] Woeginger, G.J.: The Reconstruction of Polyominoes from Their Orthogonal Projections. Technical report **SFB-65**, TU Graz, Austria, (1996) 7, 55