

Budowa algorytmów - przegląd z nawrotami

Wstęp do Informatyki i Programowania

Maciek Gębala

18 grudnia 2025

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Sformułowanie problemu

Mamy dany zbiór wartości D nazywany dziedziną.

Niech $\Omega = D^n$. Na zbiorze Ω określona jest funkcja $C : \Omega \rightarrow \{true, false\}$ wyznaczająca zbiór rozwiązań dopuszczalnych $S \subseteq \Omega$.

Warunek logiczny C , wyznaczający zbiór rozwiązań dopuszczalnych, nazywać będziemy ograniczeniami.

Problem spełnienia ograniczeń

Interesuje nas znalezienie elementów zbioru S (wszystkich albo co najmniej jednego).

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Przeszukiwanie z nawrotami

W przypadku dowolnej liczby zmiennych n i dziedziny k -elementowej $D = \{1, \dots, k\}$ potrzebna jest procedura dokonująca rekurencyjnie przeglądu z nawrotami.

```
1: procedure PEŁENPRZEGLĄD( $n, i$ )  
2:                                     ▷ ustalenie wartości  $X_1, \dots, X_n$   
3:   if  $i = n + 1$  then  
4:     if  $C(X_1, \dots, X_n)$  then  
5:       drukuj rozwiązanie  $X_1, \dots, X_n$   
6:     end if  
7:   else  
8:     for  $X_i \in \{1, \dots, k\}$  do  
9:       PEŁENPRZEGLĄD( $n, i + 1$ )  
10:    end for  
11:  end if  
12: end procedure  
13: PEŁENPRZEGLĄD( $n, 1$ )           ▷ początkowe wywołanie
```

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Przeszukiwanie z nawrotami

Procedura PEŁENPRZEGLĄD pracuje bardzo nieefektywnie, bo najpierw generuje cały układ wartości $X_1, \dots, X_n$ a dopiero na koniec sprawdza, czy spełnia on ograniczenia C (takie przeszukiwanie nazywa się generowaniem i testowaniem).

W takim przeszukiwaniu można zmienić sposób generowania np. na nierekurencyjny.

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Notatki

Notatki

Notatki

Ulepszenia

Niech warunek $\text{MOŻLIWE}(X_1, \dots, X_i)$, dla $i \leq n$, jest spełniony, gdy istnieją takie wartości $X_{i+1}, \dots, X_n$, że zachodzi $C(X_1, \dots, X_i, \dots, X_n)$.

Warunek MOŻLIWE można zinterpretować w ten sposób, że ocenia on węzeł drzewa poszukiwań i przewiduje, czy w poddrzewie zakorzenionym w ocenianym węźle znajduje się liść odpowiadający rozwiązaniu dopuszczalnemu (elementowi ze zbioru S).

Ponieważ najczęściej aby być pewnym odpowiedzi z warunku MOŻLIWE musiałby on przejrzeć całe rozpatrywane poddrzewo, więc będziemy zakładać, że warunek MOŻLIWE udziela bez pełnego przeglądu jedynie niepewnej odpowiedzi tj.

- jeśli warunek nie jest spełniony, to w analizowanym poddrzewie **na pewno** nie ma rozwiązania dopuszczalnego;
- jeśli warunek jest spełniony, to w analizowanym poddrzewie **może** jest rozwiązanie dopuszczalne.

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Ulepszenia

Poniższa procedura dokonuje częściowego przeglądu, co może w wielu sytuacjach bardzo przyspieszyć poszukiwania.

```
1: procedure CZĘŚCIOWYPRZEGLĄD( $n, i$ )
2:   if  $i = n + 1$  then
3:     if  $C(X_1, \dots, X_n)$  then
4:       drukuj rozwiązanie  $X_1, \dots, X_n$ 
5:     end if
6:   else
7:     for  $X_i \in \{1, \dots, k\}$  do
8:       if  $\text{MOŻLIWE}(X_1, \dots, X_i)$  then
9:         CZĘŚCIOWYPRZEGLĄD( $n, i + 1$ )
10:      end if
11:    end for
12:  end if
13: end procedure
14: CZĘŚCIOWYPRZEGLĄD( $n, 1$ )
```

▷ początkowe wywołanie

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Ulepszenia

Często łatwiej rozstrzygnąć warunek $\text{NIEMOŻLIWE}(X_1, \dots, X_i)$, który zachodzi gdy nie można rozszerzyć wartości pierwszych i wartości do n wartości spełniających ograniczenia C . Wtedy w linii 8 użyjemy warunku $\neg \text{NIEMOŻLIWE}$.

Gdy warunek MOŻLIWE zawsze prowadzi do rozwiązania dopuszczalnego, lub warunek NIEMOŻLIWE zawsze uniemożliwia doprowadzenie do rozwiązania dopuszczalnego, to możemy zrezygnować ze sprawdzania warunku C .

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Przykład: Problem skoczka szachowego

Ścieżka

Na szachownicy $m \times n$ chcemy znaleźć taką sekwencję ruchów skoczka, aby odwiedził każde pole szachownicy dokładnie raz.

Cykl

Dodajemy warunek, że z ostatniego pola które odwiedził może przeskoczyć na pierwsze.

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Istnienie rozwiązania

Twierdzenie

Dla każdej planszy $m \times nm$, gdzie $\min(m, n) \geq 5$ istnieje ścieżka konika szachowego.

Twierdzenie Schwenk'a

Dla każdej planszy $m \times n$ (gdzie $m \leq n$) istnieje cykl konika szachowego chyba, że jeden z następujących warunków jest prawdziwy:

- 1 m i n są nieparzyste,
- 2 $m = 1, 2$ lub 4 , oraz $n > 1$,
- 3 $m = 3$ i $n = 4, 6$ lub 8 .

Istnieje liniowy algorytm wyznaczania ścieżki lub cyklu skoczka, ale przedstawimy metodę częściowego przeglądu z nawrotami.

Maciek Gębala

Budowa algorytmów - przegląd z nawrotami

Implementacja ścieżki w Adzie

```
1 with Ada.Text_IO; use Ada.Text_IO;
2 with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4 procedure Konik2 is
5   procedure Sciezka (n, m : Integer) is
6     plansza : array (1 .. m, 1 .. n) of Integer :=
7       (others => (others => 0));
8     dx : array (1 .. 8) of Integer :=
9       (1, 2, 2, 1, -1, -2, -2, -1);
10    dy : array (1 .. 8) of Integer :=
11      (2, 1, -1, -2, -2, -1, 1, 2);
12    licznik : Integer := 0;
13
14    procedure Wypisz is
15    begin
16      for i in 1 .. m loop
17        for j in 1 .. n loop
18          Put (plansza (i, j));
19        end loop;
20        New_Line;
21      end loop;
22      New_Line;
23    end Wypisz;
```

Maciek Gębala

Budowa algorytmów - przegląd z nawrotami

Notatki

Notatki

Implementacja ścieżki w Adzie

```
25 procedure NastRuch (x, y, l : Integer) is
26   nx, ny : Integer;
27 begin
28   plansza (x, y) := 1;
29   if l = m * n then
30     Wypisz;
31     licznik := licznik + 1;
32   else
33     for i in 1 .. 8 loop
34       nx := x + dx (i);
35       ny := y + dy (i);
36       if (1 <= nx and nx <= m and 1 <= ny and ny <= n)
37         and then plansza (nx, ny) = 0
38       then
39         NastRuch (nx, ny, l + 1);
40       end if;
41     end loop;
42   end if;
43   plansza (x, y) := 0;
44 end NastRuch;
```

Maciek Gębala

Budowa algorytmów - przegląd z nawrotami

Implementacja ścieżki w Adzie

```
45 begin
46   for i in 1 .. m loop
47     for j in 1 .. n loop
48       NastRuch (i, j, 1);
49     end loop;
50   end loop;
51   Put (licznik);
52   New_Line;
53 end Sciezka;
54 begin
55   Sciezka (6, 6);
56 end Konik2;
```

Maciek Gębala

Budowa algorytmów - przegląd z nawrotami

Notatki

Implementacja cyklu w Adzie

```
1 with Ada.Text_IO; use Ada.Text_IO;
2 with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4 procedure Konik3 is
5   procedure Cykl (n, m : Integer) is
6     plansza : array (1 .. m, 1 .. n) of Integer :=
7       (others => (others => 0));
8     dx : array (1 .. 8) of Integer :=
9       (1, 2, 2, 1, -1, -2, -2, -1);
10    dy : array (1 .. 8) of Integer :=
11      (2, 1, -1, -2, -2, -1, 1, 2);
12    licznik : Integer := 0;
13
14    procedure Wypisz is
15    begin
16      for i in 1 .. m loop
17        for j in 1 .. n loop
18          Put (plansza (i, j));
19        end loop;
20        New_Line;
21      end loop;
22      New_Line;
23    end Wypisz;
```

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Implementacja cyklu w Adzie

```
25 procedure NastRuch (x, y, l : Integer) is
26   nx, ny : Integer;
27 begin
28   plansza (x, y) := 1;
29   if l = m * n then
30     for i in 1 .. 8 loop
31       nx := x + dx (i);
32       ny := y + dy (i);
33       if (1 <= nx and nx <= m and 1 <= ny and ny <= n)
34         and then plansza (nx, ny) = 1
35       then
36         Wypisz;
37         licznik := licznik + 1;
38       end if;
39     end loop;
```

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Implementacja cyklu w Adzie

```
40 else
41   for i in 1 .. 8 loop
42     nx := x + dx (i);
43     ny := y + dy (i);
44     if (1 <= nx and nx <= m and 1 <= ny and ny <= n)
45       and then plansza (nx, ny) = 0
46     then
47       NastRuch (nx, ny, l + 1);
48     end if;
49   end loop;
50 end if;
51 plansza (x, y) := 0;
52 end NastRuch;
53 begin
54   NastRuch (1, 1, 1);
55   Put (licznik);
56   New_Line;
57 end Cykl;
58 begin
59   Cykl (6, 6);
60 end Konik3;
```

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki

Dla efektywnego rozwiązywania problemów sformułowanych w postaci problemu spełnienia ograniczeń opracowano specjalną metodologię programowania nazywaną *Constraint programming*.

Constraint programming opracowano początkowo dla języka programowania logicznego Prolog, który w naturalny sposób wykonuje algorytm z nawrotami, ale dostępna jest ona także dla innych języków programowania jak np. C++.

Maciek Gębala Budowa algorytmów - przegląd z nawrotami

Notatki