

Wstęp do Informatyki i Programowania

Laboratorium nr 6 8, 13, 14, 15, 21 stycznia

Przeczytaj opis problemu **8 hetmanów** (ang. **8 queens**). Zauważ, że rozwiązanie problemu w postaci listy pozycji w kolejnych kolumnach szachownicy musi być permutacją wierszy (z warunku niebicia w poziomie i pionie).

Zadanie 1C (6 pkt)

Napisz w języku C program, który rozwiązuje problem n hetmanów następującym algorytmem:

Generujemy kolejne n elementowe permutacje (funkcja z listy na ćwiczeniach) i sprawdzamy następną funkcją, czy permutacja jest rozwiązaniem (czy nie ma bić po skosie).

Program po podaniu n w linii poleceń powinien wypisać wszystkie znalezione rozwiązania oraz na końcu ich liczbę.

Przykładowa sesja powinna wyglądać następująco:

```
1 szmaragd:~/lab6/queens 3
2 Number of solutions: 0
3 szmaragd:~/lab6/queens 4
4   2 4 1 3
5   3 1 4 2
6 Number of solutions: 2
7 szmaragd:~/lab6/queens 6
8   2 4 6 1 3 5
9   3 6 2 5 1 4
10  4 1 5 2 6 3
11  5 3 1 6 4 2
12 Number of solutions: 4
```

Sprawdź liczbę rozwiązań dla n od 1 do 12.

Zadanie 1A (6 pkt)

Napisz w języku Ada program, który rozwiązuje problem n hetmanów następującym algorytmem:

Generujemy kolejne n elementowe permutacje (funkcja z listy na ćwiczeniach) i sprawdzamy następną funkcją, czy permutacja jest rozwiązaniem (czy nie ma bić po skosie).

Program po podaniu n w linii poleceń powinien wypisać wszystkie znalezione rozwiązania oraz na końcu ich liczbę.

Przykładowa sesja powinna wyglądać następująco:

```
1 szmaragd:=~/lab6/queens 3
2 Number of solutions: 0
3 szmaragd:=~/lab6/queens 4
4   2 4 1 3
5   3 1 4 2
6 Number of solutions: 2
7 szmaragd:=~/lab6/queens 6
8   2 4 6 1 3 5
9   3 6 2 5 1 4
10  4 1 5 2 6 3
11  5 3 1 6 4 2
12 Number of solutions: 4
```

Sprawdź liczbę rozwiązań dla n od 1 do 12.

Algorytm 1 Pseudokod algorytmu z nawrotami dla problemu hetmanów.

```
1: procedure HETMANI( $n$ )
2:   position( $1 : n$ )  $\leftarrow$  all 0
3:   bije_wiersz( $1 : n$ )  $\leftarrow$  all false
4:   bije_przek1( $2 : 2n$ )  $\leftarrow$  all false
5:   bije_przek2( $-n + 1 : n - 1$ )  $\leftarrow$  all false
6:   procedure USTAW( $i$ )
7:     for  $j$  from 1 to  $n$  do
8:       if  $\neg(\text{bije\_wiersz}[j] \vee \text{bije\_przek1}[i + j] \vee \text{bije\_przek2}[i - j])$  then
9:         position[ $i$ ]  $\leftarrow$   $j$ 
10:        bije_wiersz[ $j$ ]  $\leftarrow$  bije_przek1[ $i + j$ ]  $\leftarrow$  bije_przek2[ $i - j$ ]  $\leftarrow$  true
11:        if  $i < n$  then
12:          USTAW( $i + 1$ )
13:        else
14:          DRUKUJ ROZWIĄZANIE
15:        end if
16:        position[ $i$ ]  $\leftarrow$  0
17:        bije_wiersz[ $j$ ]  $\leftarrow$  bije_przek1[ $i + j$ ]  $\leftarrow$  bije_przek2[ $i - j$ ]  $\leftarrow$  false
18:      end if
19:    end for
20:  end procedure
21:  USTAW(1)
22: end procedure
```

Zadanie 1P (6 pkt)

Napisz w języku Python program, który rozwiązuje problem n hetmanów następującym algorytmem:

Generujemy kolejne n elementowe permutacje (funkcja z listy na ćwiczeniach) i sprawdzamy następną funkcją, czy permutacja jest rozwiązaniem (czy nie ma bić po skosie).

Program po podaniu n w linii poleceń powinien wypisać wszystkie znalezione rozwiązania oraz na końcu ich liczbę.

Przykładowa sesja powinna wyglądać następująco:

```
1 szmaragd:~/lab6/python queens.py 3
2 Number of solutions: 0
3 szmaragd:~/lab6/python queens.py 4
4   2 4 1 3
5   3 1 4 2
6 Number of solutions: 2
7 szmaragd:~/lab6/python queens.py 6
8   2 4 6 1 3 5
9   3 6 2 5 1 4
10  4 1 5 2 6 3
11  5 3 1 6 4 2
12 Number of solutions: 4
```

Sprawdź liczbę rozwiązań dla n od 1 do 12.

Zadanie 2C (6 pkt)

Przeanalizuj pseudokod opisany w Algorytmie 1 i zaimplementuj go w języku C.

Napisz program, który po podaniu n w linii poleceń wypisuje wszystkie znalezione rozwiązania oraz na końcu ich liczbę.

Sprawdź liczbę rozwiązań dla n od 1 do 12. Który algorytm jest szybszy.

Zadanie 2A (6 pkt)

Przeanalizuj pseudokod opisany w Algorytmie 1 i zaimplementuj go w języku Ada.

Napisz program, który po podaniu n w linii poleceń wypisuje wszystkie znalezione rozwiązania oraz na końcu ich liczbę.

Sprawdź liczbę rozwiązań dla n od 1 do 12. Który algorytm jest szybszy.

Zadanie 2P (6 pkt)

Przeanalizuj pseudokod opisany w Algorytmie 1 i zaimplementuj go w języku Python.

Napisz program, który po podaniu n w linii poleceń wypisuje wszystkie znalezione rozwiązania oraz na końcu ich liczbę.

Sprawdź liczbę rozwiązań dla n od 1 do 12. Który algorytm jest szybszy.

Gra Mastermind W grze łamie się ukryty kod złożony z sekwencji czterech cyfr wybranych spośród sześciu (od 1 do 6, cyfry mogą powtarzać się w sekwencji).

Gra dwóch graczy. Jeden układa kod z czterech cyfr (nazywać będziemy go koderem) a drugi stara się go odgadnąć (nazywać będziemy go dekerem).

Dekoder podaje sekwencję czterech cyfr i dostaje od koderem w odpowiedzi informację ile cyfr jest poprawnych i na swoich miejscach, a ile poprawnych ale na złych miejscach.

Celem jest napisanie programu, który będzie łamał ukryty kod (grał jako deker). Zakładamy, że osoba uruchamiająca program jest koderem i zapisała sobie na kartce kod złożony z czterech cyfr. Program cyklicznie drukuje swoją propozycję kodu (cztery cyfry z zakresu od 1 do 6) i czeka na wprowadzenie przez koderem liczby cyfr na swoich miejscach i liczby cyfr dobrych ale nie na swoich miejscach.

Program nie musi łamać kodu minimalną liczbą pytań. Wystarczy, że będzie to robił w kilku pytaniach (maksymalnie ośmiu).

W poniższym przykładzie program znalazł poprawny kod po zadaniu pięciu pytań:

```
1 $ ./mastermind
2 1: 1 1 1 1 ?
3 Na swoim miejscu: 1
4 Nie na swoim miejscu: 0
5 2: 1 2 2 2 ?
6 Na swoim miejscu: 0
7 Nie na swoim miejscu: 1
8 3: 3 1 3 3 ?
9 Na swoim miejscu: 1
10 Nie na swoim miejscu: 1
11 4: 3 4 1 4 ?
12 Na swoim miejscu: 2
13 Nie na swoim miejscu: 2
14 5: 3 4 4 1 ?
15 Na swoim miejscu: 4
16 Nie na swoim miejscu: 0
17 Wygrałem.
```

W tym przykładzie program wykrył po pięciu pytaniach, że koder oszukał go:

```
1 $ ./mastermind
2 1: 1 1 1 1 ?
3 Na swoim miejscu: 1
```

```
4 Nie na swoim miejscu: 0
5 2: 1 2 2 2 ?
6 Na swoim miejscu: 1
7 Nie na swoim miejscu: 0
8 3: 1 3 3 3 ?
9 Na swoim miejscu: 1
10 Nie na swoim miejscu: 0
11 4: 1 4 4 4 ?
12 Na swoim miejscu: 1
13 Nie na swoim miejscu: 0
14 5: 1 5 5 5?
15 Na swoim miejscu: 0
16 Nie na swoim miejscu: 1
17 Oszukujesz!
```

Wskazówka: Rozważ następujący algorytm: Wygeneruj wszystkie możliwe kody (wariacje z powtórzeniami), a następnie wypisuj pierwszy dostępny i na podstawie odpowiedzi kodera eliminuj te które nie pasują.

Zadanie 3C (6 pkt)

Napisz program mastermind w języku C. Czy może on być uogólniony do wszystkich 10 cyfr?

Zadanie 3A (6 pkt)

Napisz program mastermind w języku Ada. Czy może on być uogólniony do wszystkich 10 cyfr?

Zadanie 3P (6 pkt)

Napisz program mastermind w języku Python. Czy może on być uogólniony do wszystkich 10 cyfr?