Zadanie

1. Zaimplementuj poniższe problemy na co najmniej dwa sposoby (Umieść oba algorytmy w jednym programie).
2. Jeden z algorytmów powinien mieć optymalny czasowy koszt pesymistyczny.
3. Za operację elementarną przyjmij najbardziej charakterystyczną operację algorytmu.
4. W ciele algorytmów umieść licznik operacji elementarnych.
5. Wyznacz teoretycznie czasowe koszty pesymistyczne obu algorytmów i określ rzędy funkcji kosztów.
6. Komunikacja programu z użytkownikiem powinna się odbywać za pomocą plików o podanych niżej formatach.

Zadanie 1 (symbol Newtona)

Zaimplementuj dwa spośród pięciu algorytmów wyznaczających wartość symbolu Newtona (ozn. SN).

1. Algorytm I wyznacza wartość SN1(n, k) z definicji.
2. Algorytm II wyznacza wartość SN2(n, k) z definicji po uproszczeniu przez większy czynnik mianownika.
3. Algorytm III wyznacza wartość SN3(n, k) rekurencyjnie ze wzoru

$$\binom{n}{k} = \begin{cases} 1 & \text{dla } k = 0 \text{ lub } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{w pozostałych przypadkach} \end{cases}$$

   Nie trzeba wyznaczać kosztu powyższego algorytmu.
   Należy umieć narysować drzewo wywołań rekurencyjnych dla podanych danych wejściowych.
4. Algorytm IV wyznacza wartość SN4(n, k) iteracyjnie z trójkąta Pascala z wykorzystaniem wektora.
5. Algorytm V wyznacza wartość SN5(n, k) iteracyjnie z trójkąta Pascala z wykorzystaniem tablicy 2-wymiarowej.

Zestawy: a(I, V), b(II, IV), c(I, III), d(II, V).

Dane:

- W pierwszej linii pliku In0101.txt umieszczone są dwie wartości n i k (k≤n) oddzielone pojedynczą spacją.

Wyjście:

- W pierwszej i drugiej linii pliku Out0101.txt (o formacie zgodnym z poniższym przykładem) wstaw wyniki realizacji programu.

Przykład

In0101.txt
8 2 // n k (k≤n)
Out0101.txt
n=8 k=2
SN1 = 28; licz = 14
SN2 = 28; licz = 2

Zadanie 2 (zbiory 2-elementowe)

Na wejściu danych jest n ($1 \le n \le 100$) różnych liczb N_+ z przedziału [1,k] ($1 \le k \le 150$). Zaimplementuj algorytm (o możliwie najniższym koszcie czasowym), który wykorzystując zadane liczby, generuje zbiory 2-elementowe {x, y} spełniające własność x+y<=k. Każdą liczbę należy użyć dokładnie jeden raz. W przypadku, gdy dla zadanej wartości x nie można już znaleźć takiej liczby y, by spełniona była powyższą własność, należy utworzyć zbiór 1-elementowy {x}. Zadanie polega na znalezieniu możliwie najmniejszej ilości tego typu zbiorów.

Dane:

- W pierwszej linii pliku In0102.txt znajdują się dwie liczby n i k (oddzielone pojedynczą spacją).
- W następnych n liniach podane są liczby naturalne, z których należy utworzyć zbiory.

Wyjście:

- W pierwszej linii pliku Out0102.txt wpisz minimalną liczbę zbiorów.
- W kolejnych liniach podaj elementy wygenerowanych zbiorów. (Ponieważ możliwości jest wiele, wystarczy dowolny przykład)

Przykład

In0102.txt
8 140 // n k
60
70
80
56
67
78
81
68
Out0102.txt
5
56 81
60 80
78
67 70
68

Zadanie 3 (największy spójny fragment)

Na wejściu dany jest n-elementowy ciąg liczb całkowitych z przedziału [-50000,50000].
Zaimplementuj algorytm (o możliwie najmniejszym koszcie czasowym) znajdujący dla danego ciągu spójny fragment o największej sumie.

Dane:

- W pierwszym wierszu pliku In0103.txt znajduje się liczba n reprezentująca rozmiar tablicy o elementach typu int.
- W kolejnych n liniach umieszczone są wartości z przedziału [-50000, 50000].

Wyjście:

- W pierwszej linii pliku Out0103.txt zapisz sumę elementów oraz pierwszy i ostatni indeks wyznaczonego fragmentu.

Przykład

In0103.txt:
10 // n
5
1
-10
6
8
2
-1
6
8
-10
Out0103.txt:
29 4 9

Zadanie 4 (liczby pierwsze)

Należy wyznaczyć wszystkie liczby pierwszych w przedziale domkniętym [a, b] (2≤a≤b≤1000).
Program powinien:

- Czytać z pliku tekstowego początek i koniec przedziału.

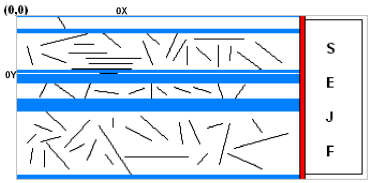
- Zapisać do pliku tekstowego ilość liczb pierwszych, kolejne liczby pierwsze z zadanego przedziału oraz liczbę wykonanych operacji elementarnych (dla każdej z metod).

Dane:
- W pierwszej linii pliku In0104.txt znajdują się dwie liczby naturalne a, b (takie że $2 \le a \le b \le 1000$) oddzielone pojedynczą spacją.

Wyjście:
- Plik wyjściowy Out0104.txt powinien mieć następujący format
  przedział: [a, b]
  I metoda:
  ... ... // liczby pierwsze z przedziału [a, b] oddzielone pojedynczą spacją, wyznaczone metodą pierwszą
  ilość liczb pierwszych = ... ; ilość wykonanych operacji elementarnych = ...
  II metoda:
  ... ... // liczby pierwsze z przedziału [a, b] oddzielone pojedynczą spacją, wyznaczone metodą drugą
  ilość liczb pierwszych = ... ; ilość wykonanych operacji elementarnych = ...

Przykład
In0104.txt:
2 60
Out0104.txt:
przedział: [2, 60]
I metoda:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
ilość liczb pierwszych = 16 ; ilość wykonanych operacji elementarnych = ...
II metoda:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
ilość liczb pierwszych = 16 ; ilość wykonanych operacji elementarnych = ...

Zadanie 5 (sejf króla Baitdocji)



Droga do sejfu króla Bajtdocji prowadzi przez korytarz (o szerokości n metrów ($n \in N$, $1 \le n \le 10000$)) zabezpieczony sprzętem laserowym. Urządzenie stanowi m ($m \in N$, $3 \le m \le 50000$)) prętów zamocowanych do sufitu, w które wbudowano emitery laserowe (liczba emiterów na jednym pręcie zależy od jego długości). Działanie urządzenia polega na aktywowaniu i dezaktywowaniu kolejnych prętów. Z każdego uruchomionego pręta emitowana jest przez moment wiązka światła (z losowo wybranego emitera), po czym uruchamiany jest kolejny pręt. Zmiana pręta następuje co 1 µs. Kolejność aktywacji prętów jest określona i powtarza się cyklicznie.
Aby wyłączyć zabezpieczenie (przy braku znajomości hasła) należy naprowadzić pocisk na dowolne miejsce poziomej (czerwonej) listwy opasającej ścianę sejfu (szerokość sejfu jest równa szerokości korytarza). Twoje zadanie polega na znalezieniu wszystkich możliwych bezpiecznych pasm (na całej szerokości korytarza) dla toru pocisku.
Algorytm powinien:
- Czytać z pliku tekstowego In0105.txt opis korytarza i lokalizację prętów.
- Wyznaczać wszystkie możliwe bezpieczne pasma dla toru pocisku.
- Zapisać wyznaczone pasma w rosnącej (lub malejącej) kolejności do pliku tekstowego

Dane:
- W pierwszym wierszu pliku In0105.txt znajdują się dwie liczby całkowite n (szerokość korytarza) i m (liczba prętów) oddzielone pojedynczą spacją.
- W kolejnych m liniach umieszczone są po cztery liczby całkowite, oddzielone pojedynczą spacją, reprezentujące współrzędne prętów: $x1_i$ $y1_i$ $x2_i$ $y2_i$. Zakładamy, że $(0 \le x1_i \le x2_i \le n)$ $(0 \le y1_i \le y2_i \le n)$.
- Oś OX układu prowadzi od początku korytarza do ściany sejfu. Oś OY - od strony lewej korytarza do prawej.
- Zakładamy, że dowolny emiter z i-tego pręta o współrzędnych ($x1_i$, $y1_i$, $x2_i$, $y2_i$) nie może wysyłać wiązki poza pasmem ($y1_i$, $y2_i$).

Wyjście:
- W pierwszym wierszu pliku Out0105.txt należy wpisać ilość bezpiecznych pasm oraz liczbę operacji elementarnych.
- W kolejnych liniach – współrzędne $y1_i$, $y2_i$ ($y1_i \ne y2_i$) oddzielone pojedynczą spacją, reprezentujące bezpieczne pasma.

Przykład
In0105.txt:
11 5 // n m
2 5 2 6
4 1 4 4
4 10 10 10 (uwaga: pręt może być zamocowany wzdłuż korytarza)
1 6 5 9
3 8 7 9
Out0105.txt:
liczba przedziałów=4; licz=...
0 1
4 5
9 10
10 11

Zestawy

Zestaw	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Zad1	1a	1b	1c	1d	1a	1b	1c	1d	1a	1b	1c	1d	1a	1b	1c	1d
Zad2	2	3	4	5	3	4	5	2	4	5	2	3	5	2	3	4
