Tema 1 – Proiectarea algoritmilor (Seria CD)
Deadline: 16.04.2020

Mihai Nan – mihai.nan.cti@gmail.com, David Iancu – davidiancudvd@gmail.com

Anul universitar 2019 – 2020

1 Temele buclucașe – 50 de puncte

1.1 Enunț

Gigel are de realizat în acest semestru o serie de teme, primind enunțurile pentru toate din prima săptămână și cunoscând pentru fiecare punctajul pe care îl primește dacă o realizează și în ce săptămână are deadline-ul. Deoarece Gigel este un student silitor și exact, atunci când citește enunțul unei teme își da seama de câte săptămâni are nevoie pentru a o realiza. Gigel este foarte perfecționist și ajunge să fie foarte concentrat pe tema respectivă. De aceea, nu mai reușește să lucreze și la o altă temă în acest interval. Cu alte cuvinte, în orice săptămână, Gigel poate lucra la maximum o temă. Toate temele lui Gigel au deadline-ul hard, ceea ce înseamnă că trebuie să le realizeze până la deadline. Sarcina voastră este să-l ajutați pe Gigel să își aranjeze temele astfel încât să reușească să acumuleze un punctaj cât mai mare în acest semestru.

1.2 Precizări

Fișierul de intrare `teme.in` conține pe prima linie numărul de teme din acest semestru. Pe următoarele linii se vor afla câte trei numere naturale reprezentând timpul de care are nevoie Gigel pentru a realiza tema, punctajul pe care îl primește pentru tema respectivă și săptămâna în care aceasta are deadline-ul.

Fișierul de ieșire `teme.out` va conține pe prima linie punctajul maxim pe care îl poate obține Gigel și numărul de teme pe care trebuie să le realizeze pentru a obține acest punctaj, iar pe următoarea linie indicii temelor pe care le poate realiza Gigel pentru a obține acest punctaj maxim, în ordinea în care trebuie să le realizeze. Dacă vor exista mai multe variante prin care poate obține același punctaj, atunci Gigel o să aleagă varianta cea mai mică din perspectiva ordinii lexicografice.

1.3 Restricții

- $1 \leq numar\_teme \leq 1.000$
- $1 \leq deadline\_max \leq 1.000$
- $1 \leq punctaj \leq 100$
- Timp maxim de execuție: **1 secundă / test**
- Limbajul acceptat: **C / C++**

1.4 Exemple

Input

```
5
8 100 13
1 100 1
1 100 3
1 100 2
4 100 6
```

Output

```
400 4
2 4 3 1
```

În prima săptămână face tema 2, pentru ea are nevoie de 1 săptămână și are deadline în săptămâna 2 (deci o termină la finalul săptămânii 1 înainte de deadline).

În a doua săptămână începe tema 4, pentru ea are nevoie de 1 săptămână și are deadline în săptămâna 2 (deci o termină la finalul săptămânii 2 înainte de deadline).

În a treia săptămână începe tema 3, pentru ea are nevoie de 1 săptămână și are deadline-ul în săptămâna 3 (deci o termină la finalul săptămânii 3 înainte de deadline).

În a patra săptămână începe tema 1, pentru ea are nevoie de 8 săptămâni și are deadline-ul în săptămâna 13 (deci o termină la sfârșitul săptămânii 11).

Input

```
10
8 50 13
3 100 14
2 50 3
1 50 3
4 65 6
1 100 2
1 100 3
2 70 3
5 100 10
1 100 1
```

Output

```
500 5
10 6 7 9 2
```

Input

```
5
1 100 5
2 100 5
3 100 5
1 90 6
2 80 7
```

Output

```
370 4
1 2 4 5
```

2 Numărul magic – 25 de puncte

2.1 Enunț

Gigel este foarte pasionat de matematică și magie. Fiind blocat în casă în această perioadă, el inventează un joc pe care să îl joace împreună cu sora lui, Gigica. Prin acest joc, Gigel dorește să o determine pe sora sa să recapituleze operațiile de adunare, scădere și înmulțire. El realizează N bilețele cu numere întregi pe care le așează pe o masă. Fiecare jucător trebuie să extragă, la fiecare rundă a jocului, două bilețele și să realizeze următoarele calcule (fie *stg* indicele bilețelului din stânga selectat și *drp* indicele celui din dreapta):

- Va realiza suma numerelor care sunt scrise pe toate bilețelele aflate înaintea celui selectat (fie *suma_stg* suma în cazul bilețelului cel mai din stânga și *suma_drp* pentru cel mai din dreapta). Dacă va extrage primul bilețel, atunci nu va avea nimic de calcula, considerând $suma\_stg = 0$.
- Va calcula diferența dintre *suma_stg* și *suma_drp* (fie *suma* această diferență).
- Va înmulți *suma* și *suma* (fie *suma_patrat* rezultatul acestei înmulțiri).
- Va calcula diferența dintre *stg* și *drp* (fie $diferenta$ rezultatul acestei operații).
- Va înmulți $diferenta$ și $diferanta$ (fie $diferenta\_patrat$ rezultatul acestei înmulțiri).
- În final, pentru a determina numărul norocos pe care trebuie să îl comunice în această rundă, jucătorul trebuie să adune *suma_patrat* și $diferenta\_patrat$.

Acest joc continuă până când unul dintre jucători ajunge să extragă bilețelele care vor genera numărul magic al acestui joc. Gigel a stabilit că numărul magic este reprezentat de cel mai mic număr care se poate obține prin realizarea acestor calcule.

Gigica vă roagă să o ajutați să îi demonstreze fratelui ei că este foarte bună la matematică prin determinarea numărului magic și a bilețelelor care vor genera acest număr magic.

2.2 Precizări

Dacă vor exista mai multe variante de selectare a bilețelelor prin care se poate ajunge la numărul magic, o să fie considerată soluție prima variantă.

Fișierul de intrare `magic.in` conține pe prima linie numărul N de bilețele pe care urmează să le așeze Gigel pe masă. Pe următoarea linie vor exista N numere întregi reprezentând numerele pe care le va scrie Gigel pe bilețele.

Fișierul de ieșire `magic.out` va conține numărul magic și indicii pentru cele două bilețele pe care trebuie să le aleagă Gigica pentru a obține acest număr magic.

2.3 Restricții

- $1 \leq N \leq 100.000$
- $-100 \leq numar \leq 100$
- Timp maxim de execuție: **1 secundă / test**
- Limbajul acceptat: **C / C++**

2.4 Exemple

Input	*Output*
3 2 -1 3	2 2 3

Cazul 1: $stg = 1, drp = 2 \Rightarrow suma\_stg = 0, suma\_drp = 2 \Rightarrow suma = suma\_stg - suma\_drp = 0 - 2 = -2 \Rightarrow suma\_patrat = (-2) \cdot (-2) = 4, diferenta = stg - drp = 1 - 2 = -1 \Rightarrow diferenta\_patrat = (-1) \cdot (-1) = 1 \Rightarrow numar\_norocos = 4 + 1 = 5$

Cazul 2: $stg = 1, drp = 3 \Rightarrow suma\_stg = 0, suma\_drp = 2 + (-1) = 1 \Rightarrow suma = suma\_stg - suma\_drp = 0 - 1 = -1 \Rightarrow suma\_patrat = (-1) \cdot (-1) = 1, diferenta = stg - drp = 1 - 3 = -2 \Rightarrow diferenta\_patrat = (-2) \cdot (-2) = 4 \Rightarrow numar\_norocos = 1 + 4 = 5$

Cazul 3: $stg = 2, drp = 3 \Rightarrow suma\_stg = 2, suma\_drp = 2 + (-1) = 1 \Rightarrow suma = suma\_stg - suma\_drp = 2 - 1 = 1 \Rightarrow suma\_patrat = 1 \cdot 1 = 1, diferenta = stg - drp = 2 - 3 = -1 \Rightarrow diferenta\_patrat = (-1) \cdot (-1) = 1 \Rightarrow numar\_norocos = 1 + 1 = 2$

$numar\_magic = minimum(\{5, 5, 2\}) = 2$

Input	*Output*
5 2 7 3 5 -8	5 1 2

Input	*Output*
5 20 7 3 5 -8	10 3 4

3 Rățuștele – 25 de puncte

3.1 Enunț

În mijlocul unui ocean există o insulă de forma unui grid cu M linii și N coloane. În cadrul acestui grid există plasate diverse rățișoare, fiecare într-o celulă distinctă, iar pentru fiecare știm orientarea (S - stânga sau D - dreapta). Fiecare rățișoară se va deplasa doar pe orizontală în funcție de orientarea pe care o are. Dacă două rățișoare se intersectează, atunci ele își schimbă orientarea și își continuă traseul. Pentru fiecare rățișoară cunoaștem poziția celulei în care se află ea inițial, un identificator unic (un număr natural cuprins între 1 și P), orientarea sa și viteza cu care se deplasează (constantă și egală cu o celulă pe unitatea de timp pentru fiecare). Atunci când o rățișoară iese din grid, o să sară în apă. Ciocnirea a două rățișoare și operația rezultată în urma acestei acțiuni (schimbarea orientărilor) nu sunt operații consumatoare de timp.

Ne dorim să știm care este rățișoara care ajunge a k-a în apă dintre cele P rățișoare care se aflau inițial pe insulă. Dacă două rățișoare ar trebui să ajungă în apă în același timp, vom considera că prima care va sări în apă este cea cu identificatorul mai mic.

3.2 Precizări

În configurația inițială fiecare rățușcă se va afla într-o celulă distinctă.

Fișierul de intrare `ratisoare.in` conține pe prima linie patru numere reprezentând: numărul de linii, numărul de coloane, valoarea k, numărul P de rățișoare. Pe următoarele P linii se vor afla câte 4 elemente: identificatorul, indicele liniei, indicele coloanei, orientarea.

Fișierul de ieșire `ratisoare.out` va conține o singură valoare reprezentând identificatorul pentru cea de-a k-a rață care ajunge în apă.

3.3 Restricții

- $1 \leq P \leq 100.000$
- $1 \leq M \leq 10.000$
- $1 \leq N \leq 10.000$
- Timp maxim de execuție: **1 secundă / test**
- Limbajul acceptat: **C / C++**

3.4 Exemple

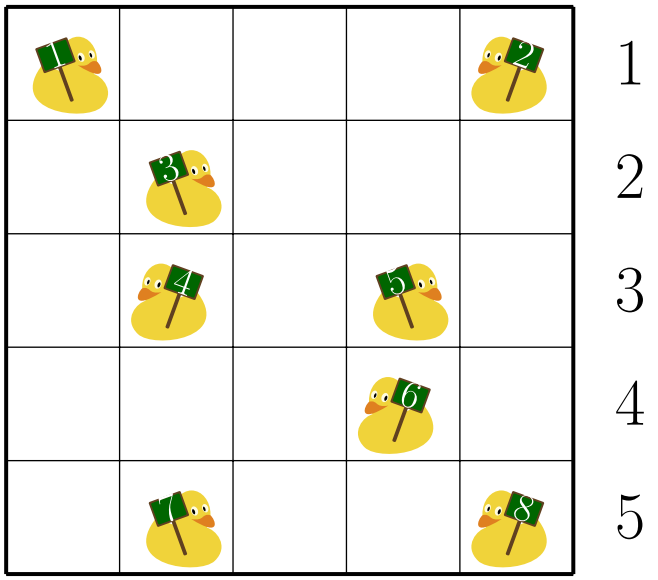
Input

```
5 5 3 8
1 1 1 D
2 1 5 S
3 2 2 D
4 3 2 S
5 3 4 D
6 4 4 S
7 5 2 D
8 5 5 S
```

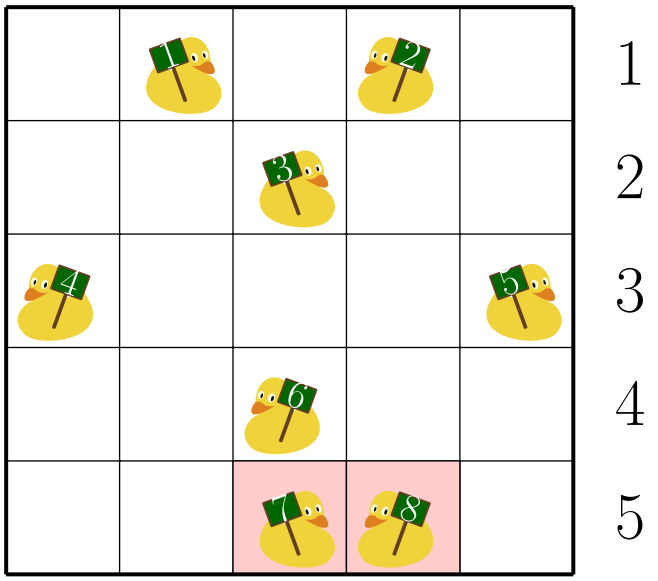
Output

```
3
```

Timpul $t = 0$ (starea inițială)



Timpul $t = 1$



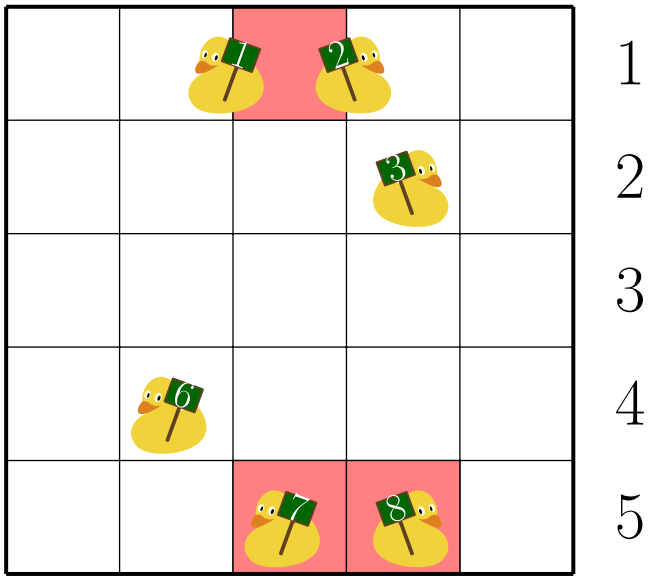
Timpul $t = 2$

După cum putem observa, acum ar urma ca rățuștele cu identificatorii 1 și 2 să se intersecteze, deoarece ajung amândouă în celula 3. Deci o să fie nevoie să le schimbăm orientarea celor două.

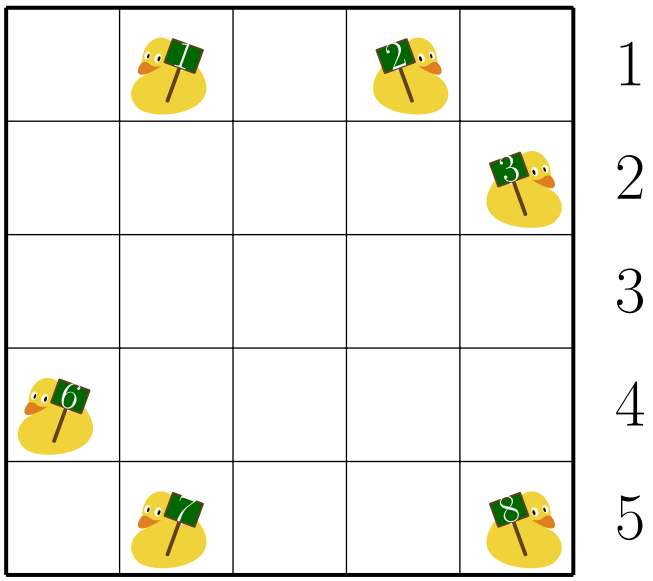
Pentru a ajunge în următoarea coloană rățusca 7 ea ar trebui să se intersecteze cu rățușca 8, deci ele își vor schimba orientările, dar vor rămâne pe aceeași poziție, deoarece vom presupune că ele s-au deplasat jumătate de celulă, s-au intersectat, și-au schimbat orientarea și s-au deplasat la loc altă jumătate de celulă.

La acest pas, rățușca 4 va sări în apă, ea fiind prima care ajunge în apă, după care și rățușca 5 poate sări în apă, fiind a doua care ajunge în apă.

Restul vor fi dispuse în grid după cum este prezentat în figura următoare:



Timpul $t = 3$



Timpul $t = 4$

La acest pas va ajunge în apă și a 3-a rățușca, ea fiind cea cu identificatorul 3. Deci răspunsul problemei noastre pentru acest exemplu o să fie 3.

Input

```
6 4 3 10
1 1 2 D
2 1 3 S
4 2 2 S
3 2 1 D
6 2 4 S
5 2 3 D
8 4 4 D
7 3 2 D
10 5 4 D
9 5 1 S
```

Output

```
10
```

4 Jocul numerelor impare – 50 de puncte

4.1 Enunț

Gigel s-a plictisit de statul în casă și concepe un alt joculeț. Pornește de la un șir cu N numere naturale. El caută în acest șir K subsecvențe distincte și disjuncte, conținând în total un număr L de elemente, astfel încât să existe un număr impar de numere impare printre cele L elemente. Având în vedere că se pot forma mai multe subsecvențe de acest tip, Gigel dorește să calculeze cea mai mare sumă care poate fi obținută cu aceste L numere. După ce a realizat aceste reguli, Gigel își dă seama că este prea complicat să determine care este suma maximă pe care o poate obține, respectând regulile jocului, și a decis să vă ceară ajutorul.

Pornind de la un șir $S = (s_1, s_2, \ldots, s_N)$, o subsecvență a șirului este de forma $(s_i, s_{i+1}, \ldots, s_j)$ cu $1 \leq i \leq j \leq N$, iar suma subsecvenței este $s_i + s_{i+1} \cdots + s_j$.

Două subsecvențe S_1 și S_2 vor fi considerate distincte dacă ele nu compun o altă subsecvență validă. De exemplu $S_1 = \{s_1, s_2\}$ și $S_2 = \{s_4, s_5\}$ sunt subsecvențe distincte, dar $S_3 = \{s_1, s_2\}$ și $S_4 = \{s_3, s_4\}$ nu sunt subsecvențe distincte, deoarece ele compun $S_5 = \{s_1, s_2, s_3, s_4\}$ care este o subsecvența validă.

Două subsecvențe S_1 și S_2 sunt disjuncte dacă ele nu conțin elemente comune $S_1 \cap S_2 = \varnothing$.

4.2 Precizări

Fișierul de intrare `joc.in` conține pe prima linie numerele naturale N, K și L, având semnificația din enunț. Pe a doua linie se vor găsi N numere naturale despărțite printr-un spațiu, reprezentând numerele din șirul pe care îl folosește Gigel.

Fișierul de ieșire `joc.out` va conține un singur număr reprezentând suma maximă pe care o poate obține Gigel prin aplicarea regulilor jocului.

Dacă nu va exista o soluție, se va afișa -1.

4.3 Restricții

- $1 \leq N \leq 200$
- $1 \leq K \leq 100$
- $1 \leq L \leq 200$
- Timp maxim de execuție: **2 secunde / test**
- Limbajul acceptat: **C / C++**

4.4 Exemple

Input	*Output*
`8 3 5` `1 2 3 4 5 6 7 8`	`27`

Subsecvențele $\{2\}$, $\{4\}$, $\{6, 7, 8\}$ au un număr impar de elemente impare (un singur element impar, 7), suma lor este 27 și este maximul care se poate obține cu 5 numere care respectă proprietatea cerută.

Input

```
10 4 7
1 2 3 4 5 6 7 8 9 10
```

Output

```
43
```

Input

```
10 5 10
2 4 6 7 8 9 10 11 12 14
```

Output

```
-1
```

5 Informații importante

- Temele vor fi testate automat folosind platforma vmchecker.
- Singurele limbaje acceptate pentru realizarea temei sunt C/C++.
- Va trebui să încărcați o arhivă având formatul **zip** care să conțină următoarele fișiere:
  - Fișierul / fișierele sursă cu implementările problemelor
  - Fișierul `Makefile` (vezi indicațiile)
  - Fișierul `README` (vezi indicațiile)
- Fișierul `Makefile` trebuie să conțină următoarele reguli:
  - `build` – care va compila sursele și va genera executabilele;
  - `run-p1` – care va rula executabilul pentru problema 1;
  - `run-p2` – care va rula executabilul pentru problema 2;
  - `run-p3` – care va rula executabilul pentru problema 3;
  - `run-p4` – care va rula executabilul pentru problema 4;
  - `clean` – care va șterge executabilele generate;
- Temele o să fie verificate doar folosind sistemul de operare Linux!
- Numele regulilor, din fișierul `Makefile`, trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea unui punctaj de **0 puncte** pe testele asociate problemei rezolvate de regula greșită.
- Fișierul `README` trebuie să conțină pentru fiecare problemă rezolvată următoarele:
  - explicația implementării realizate, prin evidențierea tehnicii de programare folosită;
  - estimarea complexității pentru soluția propusă.
- Absența fișierului `README` se penalizează cu o depunctare de **-10 puncte**.

Important

- **Nu este permisă compilarea cu opțiuni de optimizare a codului (`-O1`, `O2`, etc.).**
- **Tema este individuală! Toate soluţiile trimise vor fi verificate, folosind o unealtă pentru detectarea plagiatului.**
- **Temele vor fi punctate doar pentru testele care sunt trecute pe `vmchecker`.**