Proiectarea Algoritmilor, Tema 1

Deadline: **23.04.2025 23:55**

Cuprins

1 Descriere Temă

Considerăm că, pentru a ne dezvolta abilitatea de a rezolva eficient probleme de algoritmică, este esențial:

- Să abordăm cât mai multe probleme, pentru a fi expuși la cât mai multe situații diferite în care putem aplica tehnicile studiate.
- Să ne dedicăm timpul necesar pentru a rezolva singuri problemele întâmpinate. Este cel mai bun mod de a ne dezvolta atât abilitățile de rezolvare cât și încrederea în noi.

Din acest motiv, prima tema conține trei seturi de probleme, a căror dificultate crește progresiv. Punctajul total se va calcula luând în considerare cele mai mari două punctaje obținute din primul set de probleme, precum și cel mai mare punctaj din fiecare dintre celelalte două seturi. Astfel, pentru punctaj maxim, trebuie să rezolvați cel puțin două probleme din primul set, una din al doilea și una din al treilea.

Pentru fiecare problemă puteți acumula punctaj parțial în funcție de numărul de teste rezolvate. Rezolvările vor fi verificate atât automat, precum și prin prezentarea lor la laborator.

1.1 Tips & Tricks

Primul set conține probleme relativ ușoare.

- Citiți cu atenție enunțul problemei și constrângerile impuse variabilelor de intrare.
- Puteți începe prin a reduce problema la una mai simplă.
- Cum ați testa problema? Gândiți-vă la cazurile particulare care pot să apară.
- Puteți porni de la o soluție care garantează rezultatul corect printr-o căutare exhaustivă. Puteți descoperi o regulă pentru a evita unele calcule?
- Nu abandonați dacă nu vă merge implementarea din prima. Dacă ați petrecut deja mult timp pe o problemă și primiți același mesaj de eroare **uitați-vă și pe altă problemă și reveniți ulterior**. E important să nu rămâneți blocați pe o anumită idee - uneori ajută să faceți o mică pauză pentru a putea căuta o soluție și din alt unghi.

Dificultatea crește progresiv la următoarele seturi de probleme. Pentru punctaj maxim este posibil să fie nevoie:

- Să combinați mai multe tehnici de programare (de exemplu, Divide et Impera cu PD).
- Să aplicați optimizările specifice discutate la curs/laborator.

1.1.1 Links

- Debugging și Structuri de Date
- Coding Tips pentru teme la PA (OCW)

1.2 Punctare

- Punctajul total al temei este de 100 de puncte.
- La corectarea manuală se pot aplica depunctări pentru calitatea comentariilor din sursă/README/coding style sau alte aspecte. Pentru exemple de depunctări puteți să vă uitați și peste regulile generale de trimitere a temelor.
- O rezolvare care nu compilează sau nu trece niciun test va fi punctată cu 0.
- Fiecare problemă are o limită de timp pe test. Dacă execuția programului pe un test al acelei probleme va dura mai mult decât limita de timp, veți primi automat 0 puncte pe testul respectiv și execuția va fi întreruptă.
- În fișierul `README.md` va trebui **să descrieți soluția** pe care ați ales-o pentru fiecare problemă, **să precizați complexitatea** pentru fiecare și alte lucruri pe care le considerați utile de menționat.

Setul 1

2 Feribot

2.1 Enunț

Un convoi de **N** masini, trebuie să traverseze un râu (păstrându-se ordinea din convoi) folosind feribotul, iar în total sunt disponibile **K** feriboturi. Stim despre a i-a mașină că are greutatea G_i. Un feribot poate să susțină oricâte mașini, însă costul de traversare al acestuia este este egal cu suma greutăților tuturor mașinilor pe care le transportă. Notăm cu C_i costul pentru feribotul i.

Ținând cont că cele K feriboturi trebuie să treacă toate mașinile pe partea opusă a râului, fiecare făcând un singur drum, care este cea mai mică valoare posibilă a lui **C**, unde **C** este maximul dintre $C_1, C_2, \ldots, C_K$? (Cu alte cuvinte, niciun feribot să nu aibă costul de traversare mai mare decat **C**)

2.2 Date de intrare

Fișierul de intrare **feribot.in** conține pe prima linie numerele **N** si **K**.
A doua linie conține **N** numere intregi, al i-lea număr reprezentând greutatea celei de-a i-a mașini.

2.3 Date de ieșire

Fișierul de ieșire **feribot.out** va conține un singur număr, reprezentând greutatea minimă calculată.

2.4 Restricții și precizări

- $1 \leq K \leq N \leq 10^5$
- $1 \leq G_i \leq 10^{12}$

2.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **feribot.c**, **feribot.cpp** sau **Feribot.java**.

2.6 Exemple

feribot.in	feribot.out	Explicație
10 5 3 4 2 1 6 7 1 2 2 3	8	Pe primul feribot intra primele 2 masini (3+4=7), pe al doilea urmatoarele doua (2+1=3), pe al treilea urmatoarea masina (6), pe al patrulea urmatoarele doua (7+1=8), iar pe ultimul ultimele trei masini (2+2+3=7).

3 Stocks

3.1 Enunț

Gigel a economisit B dolari pe care acum dorește să îi investească la bursă. El a găsit online o listă cu N acțiuni studiate de cei mai buni specialiști ai lumii. Ei au determinat pentru fiecare acțiune prețul minim și prețul maxim pe care aceasta îl va putea avea la sfârșitul anului.

Convins că specialiștii nu pot să greșească, Gigel s-a hotărât să cumpere acțiuni din această listă și să vândă tot la final de an. El dorește să aleagă niște acțiuni astfel încât în cel mai rău scenariu posibil să nu piarda mai mult de L dolari și să obțină profitul maxim în cel mai bun caz posibil.

Pentru că a auzit că este bine să îți diversifici portofoliul de acțiuni, el a decis că nu va cumpăra mai mult de o acțiune de un anumit tip. De asemenea, pe platforma online pe care Gigel tranzacționează, se pot cumpăra doar unități întregi (nu poate cumpăra fracții precum 0.3 dintr-o acțiune). Care este profitul maxim pe care îl poate obține?

3.2 Date de intrare

Pe prima linie a fișierului **stocks.in** se află N, B și L.
Pe fiecare din următoarele N linii se află 3 numere, $currentValue$, $minValue$ și $maxValue$.

3.3 Date de ieșire

În fișierul **stocks.out** se va afla profitul maxim care poate fi obținut.

3.4 Restricții și precizări

- $1 \leq N \leq 100$
- $1 \leq B \leq 500$
- $1 \leq L \leq 100$
- $1 \leq minValue < currentValue < maxValue \leq 100$

3.5 Testare și punctare

Sursa care conține funcția **main** trebuie obligatoriu denumită: **stocks.c**, **stocks.cpp** sau **Stocks.java**.

3.6 Exemple

stocks.in	stocks.out	Explicație
5 30 6 9 7 17 12 5 16 7 6 8 3 2 6 5 2 6	12	Gigel va alege acțiunile 1,3 și 4. Va investi 9 + 7 + 3 = 19 dolari. În cel mai rău caz, acțiunile vor valora 7 + 6 + 2 = 15 dolari. Pierderea va fi de 19 - 15 = 4 dolari. În cel mai bun caz, acțiunile vor valora 17 + 8 + 6 = 31 de dolari. Profitul va fi prin urmare 31 - 19 = 12 dolari.

4 Nostory

4.1 Enunț

Se dau $2 * N$ numere **distincte**, împărțite în două liste de lungime **N**: **A** și **B**. Avem voie să facem mutări de tipul: alegem două numere din cele două liste, și le interschimbăm. Putem alege ambele numere din aceeași listă.

După ce facem mutările, calculăm un scor astfel: pentru fiecare pereche de numere care se găsesc pe aceeași poziție în **A** și **B**, păstrăm maximul dintre ele; suma maximelor este scorul. Ne dorim să **maximizăm** acest scor.
Această problemă are două subpuncte:

1. putem face un număr nelimitat de mutări.
2. putem face cel mult **K** mutări.

4.2 Date de intrare

Pe prima linie a fișierului **nostory.in** se află un număr **T**, care reprezintă subpunctul pe care îl rezolvăm.
Pe a doua linie se află numerele **N**, sau **N** și **K**, în funcție de subpunct.
Pe liniile 3 și 4 se află două liste de câte **N** numere: **A** și **B**.

4.3 Date de ieșire

În fișierul **nostory.out** se va afișa scorul maxim pe care îl putem obține.

4.4 Restricții și precizări

- $\mathbf{T} \in \{1, 2\}$
- $1 \leq \mathbf{K} \leq \mathbf{N} \leq 10^5$
- $1 \leq A_i, B_i \leq 10^9$

4.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **nostory.c**, **nostory.cpp** sau **Nostory.java**.

4.6 Exemple

nostory.in	nostory.out	Explicație
1 3 1 5 10 6 3 9	25	Avem la dispoziție un număr nelimitat de mutări. Putem interschimba, de exemplu, pe 3 cu 9, apoi pe 3 cu 10. Listele finale vor fi: 1 5 3 6 9 10 Scorul final va fi 6 + 9 + 10 = 25.
2 5 2 3 6 10 9 5 1 8 4 7 2	40	Avem la dispoziție două mutări. Putem interschimba pe 3 cu 8, și pe 7 cu 2. Listele finale vor fi: 8 6 10 9 5 1 3 4 2 7 Scorul final va fi 8 + 6 + 10 + 9 + 7 = 40.

5 Walsh

5.1 Enunț

Gigel vrea să își implementeze un canal secret de comunicație. După ce a făcut niște cercetări, el a descoperit că o modalitate bună de criptare este folosirea Tabelelor Walsh. Acestea au forma unei matrice pătratice, de dimensiuni **N**×**N**, unde **N** este o putere a lui 2. Notăm W_1 tabelul de dimensiune 1×1, W_2 tabelul de dimensiune 2×2, W_4 tabelul de dimensiune 4×4 și așa mai departe. Tabelele Walsh se generează după următoarea regulă:

$$W_1 = \begin{bmatrix} 0 \end{bmatrix} \quad W_{2n} = \begin{bmatrix} W_n & W_n \\ W_n & \overline{W_n} \end{bmatrix}$$

unde $\overline{W_n}$ va conține elementele din W_n negate (1 devine 0, iar 0 devine 1). Observăm că așa vor arăta W_2, W_4 si W_8, generate pe baza relației de mai sus.

$$W_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad W_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad W_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Având acest tabel, Gigel se întreabă: Pentru **K** perechi $(\mathbf{x}, \mathbf{y})$: ce valoare se află în tabelul Walsh la linia **x**, coloana **y**? (de exemplu, pentru: $\mathbf{x} = 3$ și $\mathbf{y} = 2 \implies 0$)

5.2 Date de intrare

Fișierul de intrare **walsh.in** conține pe prima linie numărul natural **N**, ce reprezintă dimensiunea tabelei Walsh.

Pe a doua linie se află numărul natural **K**, iar pe următoarele **K** linii se află perechi de numere naturale $(\mathbf{x}, \mathbf{y})$ $(1 \leq \mathbf{x}, \mathbf{y} \leq \mathbf{N})$, câte o pereche pe o linie. Valorile scrise pe aceeași linie sunt separate prin câte un spațiu.

5.3 Date de ieșire

Fișierul de ieșire **walsh.out** va conține **K** linii. Pe cea de-a **i**-a linie va fi scrisă valoarea aflată pe linia **x** și coloana **y** pentru cea de-a **i**-a pereche din fișierul de intrare.

5.4 Restricții și precizări

- $1 \leq \mathbf{N} \leq 2^{30}$, **N** este o putere a lui 2.
- $1 \leq \mathbf{x}, \mathbf{y} \leq \mathbf{N}$
- $1 \leq \mathbf{K} \leq 10^5$

5.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **walsh.c**, **walsh.cpp** sau **Walsh.java**.

5.6 Exemple

walsh.in	walsh.out	Explicație
4 5 1 3 4 2 3 3 1 2 2 1	0 1 1 0 0	Pe linia 1, coloana 3 valoarea este 0 Pe linia 4, coloana 2 valoarea este 1 Pe linia 3, coloana 3 valoarea este 1 Pe linia 1, coloana 2 valoarea este 0 Pe linia 2, coloana 1 valoarea este 0

Setul 2

6 BadGPT

6.1 Enunț

Gigel a decis să folosească noul tool EnunțGPT pentru a genera enunțuri la temele de la Politehnică. Acest tool nu reușește să genereze neaparat enunțuri interesate, dar folosind o nouă extensie (creată chiar de el) GPT2PDF preia enunțul și îl transformă în format PDF, numai bun pentru a putea fi publicat.

Problema lui Gigel este că extensia lui transformă enunțul în format PDF nu prin copierea textului, ci prin crearea unei imagini pe baza textului original. Acest tool însa nu reușește să proceseze corect literele **m** și **w**. Litera **m** va apărea în enunț ca **nn**, iar litera **w** va apărea ca **uu** (EnunțGPT preferă engleza). De exemplu, dacă în enunțul din PDF va apărea secvența de litere **anna**, atunci secvența originală putea să fie **anna** sau **ama**.

Gigel este curios să afle dacă extensia lui este bună sau nu. El vrea să știe, pornind de la șirul de caractere din PDF, câte șiruri distincte puteau sta la baza enuntului. Speriat de eventualul număr mare de posibilități, rezultatul se va afișa modulo $10^9 + 7$ (așa poate rezultatul devine 1 și Gigel va fi satisfăcut).

6.2 Date de intrare

Fișierul de intrare **badgpt.in** va conține pe prima linie un singur șir de caractere, comprimat sub forma $l_1 n_1 l_2 n_2 \ldots$, unde secvența $l_i n_i$ sugerează ca litera l_i va apărea de n_i ori la rând.

Se garantează faptul că nu va exista aceeași literă pe două poziții consecutive. De exemplu, nu putem avea șirul codificat $u3u4$. Acesta va fi $u7$.

6.3 Date de ieșire

Fișierul de ieșire **badgpt.out** va conține pe prima linie numărul de șiruri distincte care puteau sta la baza enunțului, modulo $10^9 + 7$.

6.4 Restricții și precizări

- $1 \leq \mathbf{L}, n_i \leq 10^{18}$, unde **L** este lungimea șirului necomprimat.
- $1 \leq \mathbf{G} \leq 10^5$, unde **G** este numărul de grupuri comprimate. Un grup comprimat reprezintă o pereche $l_i n_i$ ce semnifică faptul că litera l_i va apărea de n_i ori la rând.
- Pentru teste in valoare de 10 puncte, $1 \leq \mathbf{L} \leq 2 * 10^6$, unde L este lungimea șirului necomprimat.

6.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **badgpt.c**, **badgpt.cpp** sau **Badgpt.java**.

6.6 Exemple

badgpt.in	badgpt.out	Explicație
a1c3n2	2	Codificarea corespunde șirului **acccnn**. Există 2 șiruri inițiale posibile: **acccnn** și **acccm**.

7 Prinel

7.1 Enunț

Gigel trebuie sa aibă grijă de fratele său mai mic, Prinel. Pentru a îl ține ocupat, în timp ce el se uită la meciul naționalei de fotbal a României, îi dă o lista **a** de **N** numere, inițial toate fiind **1**, pe care poate aplica următoarea operație: alege un număr **a[i]** și un număr **x**, divizor al lui **a[i]**, apoi înlocuiește **a[i]** cu **a[i]** + **x**.

Scopul este să transforme numerele din **a** în numerele dintr-o altă listă **target** dată, tot de lungime **N**, până se termină meciul de fotbal, folosind operația anterioară de maxim **K** ori.

Gigel îi mai dă și o listă **p**, **p[i]** = numărul de puncte obținute dacă **a[i]** = **target[i]** la finalul operațiilor. Prinel se intreabă care este numărul maxim de puncte pe care le poate obține?

7.2 Date de intrare

Pe prima linie a fișierului **prinel.in** se află **N** și **K**, separate prin spațiu.
Pe a doua linie se află vectorul **target** ce conține **N** elemente, separate prin spațiu.
Pe a treia linie se află vectorul **p** ce conține **N** elemente, separate prin spațiu.

7.3 Date de ieșire

În fișierul **prinel.out** se va afla numărul maxim de puncte ce pot fi obținute.

7.4 Restricții și precizări

- $1 \leq \mathbf{N} \leq 10^3$
- $1 \leq \mathbf{K} \leq 10^6$
- $1 \leq \mathbf{target[i]} \leq 10^5$
- $1 \leq \mathbf{p[i]} \leq 10^6$

7.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **prinel.c**, **prinel.cpp** sau **Prinel.java**.

7.6 Exemple

prinel.in	prinel.out	Explicație
4 4 1 7 5 2 2 6 5 2	9	Pentru **target[0]** nicio operație. Pentru **target[1]** 4 operații : $1 \to 2 \to 4 \to 6 \to 7$ Pentru **target[2]** 3 operații: $1 \to 2 \to 4 \to 5$ Pentru **target[3]** o operație: $1 \to 2$ Alegem **target[0]**, **target[2]** și **target[3]**, obținând $2 + 5 + 2 = 9$

Setul 3

8 Crypto

8.1 Enunț

După ani de minat criptomonede, Gigel s-a decis că în sfârșit e timpul să le folosească. Când a vrut să acceseze portofelul, a realizat că are nevoie de o cheie **K** formată din litere mici ale alfabetului englez, pe care și-o notase cu mulți ani în urmă pe o bucată de hârtie. După lungi căutări, a găsit hârtia salvatoare, însă o parte din caractere se șterseseră din cauza trecerii ireversibile a timpului.

Pe spatele foii mai era scris un șir **S**, despre care Gigel știe că are următoarele proprietăți:

- **S** este un subșir al lui **K**
- **K** conține doar litere din **S**

Gigel a scris cheia **K** pe o altă bucată de hârtie, punand caracterul **?** în fiecare din pozițiile unde lipsea un caracter.

Înainte de a se apuca de bruteforce pentru a găsi cheia punând litere în loc de toate caracterele **?**, Gigel vrea să știe de câte ori apare subșirul **S** în toate cheile **K** posibile. Deoarece rezultatul poate fi mare, se dorește afișarea lui modulo $10^9 + 7$.

Notă: Un subșir al unui șir este format din elemente (nu neapărat consecutive) ale șirului respectiv, în ordinea în care acestea apar în șir.

8.2 Date de intrare

Pe prima linie a fișierului **crypto.in** vor fi **N** și **L**, separate prin spațiu.
Pe a doua linie va fi cheia **K**, cu **?** în locul caracterelor lipsă.
Pe a treia linie va fi subșirul **S**.

8.3 Date de ieșire

În fișierul **crypto.out** se va afla un număr, reprezentând numărul de apariții ale subșirului **S** în cheile **K**.

8.4 Restricții și precizări

- $1 \leq \mathbf{N} \leq 10^5$, lungimea cheii **K**
- $1 \leq \mathbf{L} \leq 10$, lungimea subșirului **S**

8.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **crypto.c**, **crypto.cpp** sau **Crypto.java**.

8.6 Exemple

8.6.1 Exemplul 1

Exemplul 1		
crypto.in	crypto.out	Explicație
4 2 ?x?y xy	11	Șirurile posibile sunt: xxxy → 3 subșiruri xy xxyy → 4 subșiruri xy yxxy → 2 subșiruri xy yxyy → 2 subșiruri xy $3 + 4 + 2 + 2 = 11$

9 Regele

9.1 Enunț

După toate încercările prin care a trecut de-a lungul temelor de la PA, Gigel a devenit cel mai înțelept om și a fost ales regele regatului Gigeland. În acest moment, regatul se confruntă cu o problemă ce necesită o rezolvare imediată. Regatul are o rețea comercială ce constă în **N** orașe aflate pe o dreaptă. Orașul **i** se află la coordonata **coord[i]**. Un oraș poate face comerț doar cu orașele vecine cu el (cel mai din stânga și cel mai din drepta au un singur vecin).

O rută comercială dintre două orașe este **activă** dacă îi putem aloca un număr de negustori egal cu distanța dintre cele două orașe. Dacă o rută este activă, este activă pentru ambele orașe pe care le leagă.

Un oraș este considerat **activ** dacă ambele sale rute sunt active. Din păcate, în regat nu există suficienți negustori pentru a se ocupa de toate rutele comerciale.

Cum în fiecare zi numărul de negustori din regat poate să crească sau să scadă, Givel vrea să răspundă la **Q** intrebari de genul: Având **M** negustori disponibili, care este numărul maxim **X**, astfel încât oricum am alege **X** orașe, negustorii disponibili pot fi distribuiți pe rutele comerciale astfel încât cele **X** orașe să fie active comercial?

9.2 Date de intrare

Pe prima linie a fișierului **regele.in** se află **N**, numărul de orașe.
Pe a doua linie se află **N** numere, reprezentând coordonatele orașelor, separate prin spațiu, **sortate crescător**.
Pe a treia linie se află **Q**, numărul de întrebări.
Pe următoarele **Q** linii se află un număr **M** de negustori.

9.3 Date de ieșire

În fișierul **regele.out** vor fi scrise **Q** linii a câte un număr, reprezentând numărul maxim **X**, cu proprietatea din enunț.

9.4 Restricții și precizări

- $1 \leq \mathbf{Q} \leq 5 \cdot 10^5$
- $1 \leq \mathbf{N} \leq 2000$
- $1 \leq \mathbf{coord[i]} \leq 10^9$

9.5 Testare și punctare

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **regele.c**, **regele.cpp** sau **Regele.java**.

9.6 Exemple

9.6.1 Exemplul 1

Exemplul 1		
regele.in	regele.out	Explicație
6 2 5 10 12 17 19 3 15 10 20	2 1 6	**2** → daca **X** ar fi 3, pentru activarea orașelor de la pozițiile 5, 12 și 17 este nevoie de 17 negustori. Orice combinație de 2 orașe poate fi activată folosind 15 negustori. **1** → orice oraș poate fi activat daca avem 10 negustori. Activarea simultană a orașelor de la coordonatele 5 și 17 nu este posibilă cu 10 negustori (este nevoie de 15). **6** → pentru activarea tuturor orașelor este nevoie de 17 negustori.