

PA - Tema 1

- Tehnici de Programare (și Gigel) -

Responsabili:

Andrei Preda, Cristian Pătrașcu, Ioan Pop,
Ioana Dabelea, Ioan Popescu, Alexandru Dima, Cristian Tudorache

Deadline soft: **18.04.2024 23:55**

Deadline hard: **25.04.2024 23:55**

CUPRINS

1	Problema 1: Alimentare Servere	3
1.1	Enunț	3
1.2	Date de intrare	3
1.3	Date de ieșire	3
1.3.1	Afișare	3
1.4	Restricții și precizări	4
1.5	Testare și punctare	4
1.6	Exemple	4
2	Problema 2: Colorare	5
2.1	Enunț	5
2.2	Date de intrare	5
2.3	Date de ieșire	5
2.4	Restricții și precizări	5
2.5	Testare și punctare	6
2.6	Exemple	6
3	Problema 3: Compresie	7
3.1	Enunț	7
3.2	Date de intrare	7
3.3	Date de ieșire	7
3.4	Restricții și precizări	7
3.5	Testare și punctare	7
3.6	Exemple	8
4	Problema 4: Criptat	9

4.1	Enunț	9
4.2	Date de intrare	9
4.3	Date de ieșire	9
4.4	Restricții și precizări	9
4.5	Testare și punctare	9
4.6	Exemple	10
5	Problema 5: Oferta	11
5.1	Enunț	11
5.2	Date de intrare	11
5.3	Date de ieșire	11
5.4	Restricții și precizări	12
5.5	Testare și punctare	12
5.6	Exemple	12
6	Punctare	13
6.1	Checker	13
7	Folosire ChatGPT	14
8	Format arhivă	14
9	Links	15

1 PROBLEMA 1: ALIMENTARE SERVERE

1.1 Enunț

Avem la dispoziție un grup de N servere, fiecare cu o putere de calcul proprie P_i . Serverele funcționează optim când sunt alimentate cu o cantitate de curent C_i . Dacă un server este subalimentat sau supraalimentat, puterea sa de calcul scade cu o unitate pentru fiecare unitate de curent *mismatched* (puterea finală poate să fie **inclusiv negativă**).

Suntem nevoiți să alimentăm toate serverele cu aceeași cantitate de curent, dar, din fericire, controlăm această cantitate. Care este puterea de calcul maximă a sistemului, știind că clusterul va fi **limitat de cea mai mică putere individuală**?

1.2 Date de intrare

Pe prima linie din fișierul **servere.in** va fi numărul de servere, N .

Pe a doua linie vor fi N numere naturale reprezentând puterile de calcul ale serverelor (vectorul P).

Pe a treia linie vor fi N numere naturale reprezentând limitele de alimentare ale serverelor (vectorul C).

1.3 Date de ieșire

În fișierul **servere.out** veți scrie puterea de calcul maximă care poate fi atinsă, cea mai mică dintre puterile individuale. Afișați puterea cu **exact o zecimală**.

1.3.1 Afișare

În C++ puteți folosi biblioteca `<iomanip>`, și funcțiile `fixed` și `setprecision` pentru a afișa un număr cu o zecimală, astfel:

```
#include <iomanip>

// În interiorul unei funcții
fout << fixed << setprecision(1) << rezultat << "\n";
```

În Java puteți folosi metoda `String.format`, astfel:

```
fout.println(String.format("%.1f", rezultat));
```

1.4 Restricții și precizări

- $1 \leq N \leq 10^5$, numărul de servere
- $1 \leq P_i \leq 10^9$, puterile de calcul
- $1 \leq C_i \leq 10^9$, pragurile de alimentare optime
- Toate numerele din datele de intrare sunt întregi.
- Puterea de alimentare aleasă de voi **poate să nu fie număr întreg**.

1.5 Testare și punctare

- Punctajul maxim este de 25 puncte.
- Timpul de execuție:
 - C/C++: 1 s
 - Java: 2 s
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **servere.c**, **servere.cpp** sau **Servere.java**.

1.6 Exemple

servere.in	servere.out	Explicație
4 6 9 7 5 2 4 1 8	2.5	<p>Sunt 4 servere, având puterile 6, 9, 7 și 5. Pragurile lor de alimentare sunt 2, 4, 1, și 8. Alimentând cu 5.5 unități vom obține puterile individuale:</p> <ul style="list-style-type: none"> • $6 - 5.5 - 2 = 2.5$ • $9 - 5.5 - 4 = 7.5$ • $7 - 5.5 - 1 = 2.5$ • $5 - 5.5 - 8 = 2.5$ <p>Puterea sistemului va fi cea mai mică dintre ele, adică 2.5. Nu putem obține o putere mai mare de atât dacă alimentăm la alt prag (nici mai mare, nici mai mic).</p>

2 PROBLEMA 2: COLORARE

2.1 Enunț

Gigel a început să fie din ce în ce mai pasionat de pictură. Problema este, însă, că nu prea are talent, așa că a decis să își cumpere picturi pe numere. Uitându-se pe AlgoExpress, a găsit o ofertă de nerefuzat pentru un tablou și a început să citească mai multe despre el.

Tabloul are o lungime de N centimetri și o lățime de 2 centimetri, fiecare pătrățel în care poți colora având o suprafață de 1 cm^2 . Regulile de colorare sunt următoarele:

- nu poți folosi decât **3 culori** (galben, roz și mov)
- zonele de colorare sunt împărțite în dreptunghiuri de dimensiune 1×2 , așezate fie pe orizontală, fie pe verticală
- două zone de colorare care au o **latură comună** trebuie să fie **colorate diferit**.

Fascinat de acest tablou, Gigel a început să se gândească în câte moduri diferite ar putea arăta la final tabloul. Ajutați-l pe Gigel să afle **numărul de modele distincte**, modulo $10^9 + 7$.

2.2 Date de intrare

Pe prima linie din fișierul **colorare.in** va fi un număr natural, K .

Pe a doua linie vor fi K perechi de forma " $X \ T$ ", cu semnificația: "următoarele X dreptunghiuri din tablou sunt de tipul T (verticale sau orizontale)".

2.3 Date de ieșire

În fișierul **colorare.out** veți scrie numărul de tablouri distincte care se pot realiza, modulo $10^9 + 7$.

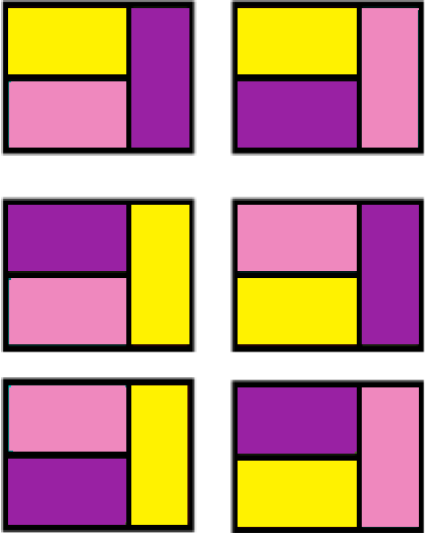
2.4 Restricții și precizări

- $1 \leq N \leq 2 * 10^6$, pentru 15 puncte.
- $2 * 10^6 \leq N \leq 7 * 10^9$, pentru alte **10 puncte bonus**.
- $1 \leq X \leq N$, număr de dreptunghiuri consecutive cu aceeași orientare.
- T va fi caracterul " H " pentru dreptunghiuri orizontale, sau " V " pentru dreptunghiuri verticale.

2.5 Testare și punctare

- Punctajul maxim este de 25 puncte.
- Timpul de execuție:
 - C/C++: 1 s
 - Java: 2 s
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **colorare.c**, **colorare.cpp** sau **Colorare.java**.

2.6 Exemple

colorare.in	colorare.out	Explicație
2 1 H 1 V	6	<p>Toate picturile realizabile sunt desenate mai jos.</p> 

3 PROBLEMA 3: COMPRESIE

3.1 Enunț

Fie următoarea operație asupra unui șir de numere: se alege o subsecvență din șir și se înlocuiește cu suma elementelor din subsecvență. De exemplu dacă avem șirul [1, 6, 2, 4, 5] și alegem subsecvența [6, 2, 4], obținem [1, 12, 5].

Pentru un șir A de lungime n și un șir B de lungime m, determinați dacă putem obține două **șiruri egale** prin 0 sau mai multe aplicări ale operației definite mai sus, asupra lui A sau B, și aflați care e **lungimea maximă** pe care o poate avea șirul obținut.

3.2 Date de intrare

Pe prima linie din fișierul **compresie.in** va fi un număr natural, n.

Pe a doua linie vor fi n numere naturale, reprezentând șirul A.

Pe a treia linie va fi un număr natural, m.

Pe a patra linie vor fi m numere naturale, reprezentând șirul B.

3.3 Date de ieșire

În fișierul **compresie.out** veți scrie un singur număr întreg, lungimea maximă a șirului rezultat, sau "−1", în cazul în care nu putem obține două șiruri egale.

3.4 Restricții și precizări

- $1 \leq N, M \leq 3 \cdot 10^5$
- $1 \leq A_i, B_i \leq 10^9$

3.5 Testare și punctare

- Punctajul maxim este de 25 puncte.
- Timpul de execuție:
 - C/C++: 1 s
 - Java: 2 s
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **compresie.c**, **compresie.cpp** sau **Compresie.java**.

3.6 Exemple

compresie.in	compresie.out	Explicație
6 11 2 2 1 8 6 7 3 8 2 1 2 11 3	4	<p>Putem aplica următoarele operații pe șirul A:</p> <ul style="list-style-type: none"> • din [11,2,2,1,8,6] compresăm [2,1] • din [11,2,3,8,6] compresăm [8,6] • obținem [11,2,3,14] <p>Modificăm șirul B astfel:</p> <ul style="list-style-type: none"> • din [3,8,2,1,2,11,3] compresăm [3,8] • din [11,2,1,2,11,3] compresăm [1,2] • din [11,2,3,11,3] compresăm [11,3] • obținem [11,2,3,14] <p>Ambele șiruri sunt acum egale cu [11,2,3,14], care are 4 elemente.</p>

4 PROBLEMA 4: CRIPTAT

4.1 Enunț

Ne dorim să construim un generator de parole puternice. Din acest motiv, vom încerca să construim parole cât mai lungi, pentru a fi mai greu de spart. Dar, pentru a ne diferenția generatorul de celelalte din piață, mai avem câteva criterii pe care vrem să le îndeplinim.

Avem la dispoziție o listă de N cuvinte care ne plac, și vrem să formăm parola doar concatenând unele dintre aceste cuvinte (fiecare cuvânt poate fi folosit cel mult o dată). În plus, vrem ca toate parolele pe care le producem să aibă o literă dominantă (o literă care are strict mai multe apariții decât jumătate din lungimea parolei).

În aceste condiții, care este lungimea maximă a parolei pe care o putem construi?

4.2 Date de intrare

Pe prima linie din fișierul **criptat.in** va fi numărul de cuvinte care ne plac, N . Pe următoarele N linii se va afla câte un cuvânt.

4.3 Date de ieșire

În fișierul **criptat.out** veți scrie un singur număr, lungimea maximă a parolei.

4.4 Restricții și precizări

- $1 \leq N \leq 10^3$, numărul de cuvinte
- $1 \leq L \leq 10^4$, suma lungimilor cuvintelor din listă
- Cuvintele sunt formate doar din litere mici ale alfabetului englez.
- Numărul de **litere distincte** folosite într-un test va fi **cel mult egal cu 8**.
- Pentru testele acestei probleme vom putea mereu să formăm cel puțin o parolă.

4.5 Testare și punctare

- Punctajul maxim este de 30 puncte.
- Timpul de execuție:
 - C/C++: 2 s
 - Java: 3 s
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **criptat.c**, **criptat.cpp** sau **Criptat.java**.

4.6 Exemple

criptat.in	criptat.out	Explicație
4 too otter tote oo	9	Concatenând primul, al treilea, și al patrulea cuvinte putem obține parola "toototeoo". Această parolă are 9 litere și conține litera "o" de 5 ori. Litera "o" este dominantă pentru că $5 > 9/2$.
4 abbc cbb aa ab	9	Alegem cuvintele "cbb", "abbc", și "ab", și formăm parola "cbbabbcab". Lungimea parolei este 9, și conține litera "b" de 5 ori.

5 PROBLEMA 5: OFERTA

5.1 Enunț

Sunteți organizator pentru prima ediție a hackathonului de algoritmică. Întrucât vreți ca toată lumea să aibă o experiență cât mai frumoasă, v-ați propus să cumparați consumabile (snacks, apă, sucuri etc.) de la un supermarket. După ce ați ajuns la casă și ați așezat cele N produse pe banda rulantă, casierul v-a înștiințat că astăzi magazinul are două oferte atractive:

1. Dacă grupați două produse, veți beneficia de o reducere de 50% **pentru produsul mai ieftin**.
2. Dacă grupați trei produse, veți beneficia de o reducere de 100% **pentru produsul cel mai ieftin** (va fi gratis).

Întrucât deja ați pus toate produsele pe bandă, din lipsă de timp puteți să **grupați doar produsele adiacente**. În schimb, nu sunteți obligați să grupați toate produsele, pe unele le puteți cumpăra individual (dar trebuie să le cumpărați pe toate).

- Care este prețul **minim** pe care îl puteți obține prin gruparea produselor de pe bandă? (20p)
- (*bonus*) Care este **al K-lea cel mai mic preț unic** pe care îl puteți obține prin gruparea produselor de pe bandă? (15p)

5.2 Date de intrare

Pe prima linie din fișierul **oferta.in** vor fi numărul N de produse de pe bandă, și numărul K . Pentru prima cerință vom avea $K = 1$.

Pe a doua linie vor fi prețurile celor N produse, în ordinea în care apar pe bandă.

5.3 Date de ieșire

În fișierul **oferta.out** veți scrie un singur număr, prețul cerut. Veți afișa prețul cu **exact o zecimală**. Puteți găsi indicații pentru afișarea cu o zecimală la problema **Servere** ([Secțiunea 1.3.1](#)).

5.4 Restricții și precizări

Pentru 20 de puncte:

- $1 \leq N \leq 10^4$, numărul de produse
- $K = 1$, deci trebuie să găsim cel mai mic preț posibil
- Suma tuturor prețurilor nu va depăși $2^{31} - 1$.

Pentru alte 15 puncte:

- $1 < K \leq 10^4$
- În cazul în care nu există soluție veți afișa “-1”.

5.5 Testare și punctare

- Punctajul maxim este de 35 puncte.
- Timpul de execuție:
 - C/C++: 3.5 s
 - Java: 4 s
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **oferta.c**, **oferta.cpp** sau **Oferta.java**.

5.6 Exemple

oferta.in	oferta.out	Explicație
5 1 80 27 10 20 300	413.5	În acest exemplu $K = 1$, deci ne dorim să aflăm cel mai mic preț pe care îl putem obține. Alegem să cumpărăm primele două produse împreună, cu costul $(80 + 27/2)$, și ultimele trei împreună, cu costul $(20 + 300)$, obținând un total de 413.5.
5 3 80 27 10 20 300	418.5	De această dată ne dorim să aflăm al $K = 3$ -lea cel mai mic preț unic pe care l-am putea obține. Cele mai mici 5 prețuri (de exemplu) pe care le putem obține sunt 413.5, 417.0, 418.5, 422.0, 423.5. Așadar, răspunsul este 418.5.

6 PUNCTARE

- Punctajul temei este de 150 puncte, distribuit astfel:
 - Problema 1: 25p
 - Problema 2: 25p (dintre care 10p sunt bonus)
 - Problema 3: 25p
 - Problema 4: 30p
 - Problema 5: 35p (dintre care 15p sunt bonus)
 - 5 puncte vor fi acordate pentru comentarii și README.
 - 5 puncte vor fi acordate automat de checker pentru coding style. Totuși, la corectarea manuala se pot aplica **depunctări de până la 20 de puncte** pentru **coding style neadecvat**.

Punctajul pe README, comentarii și coding style este condiționat de obținerea unui punctaj strict pozitiv pe cel puțin un test.

Se poate obține un bonus de 25p rezolvând testele bonus din cadru problemelor **Colorare** și **Oferta**, care reprezintă variante mai dificile ale problemelor inițiale. Acordarea bonusului **NU** este condiționată de rezolvarea celorlalte teste/probleme. În total se pot obține 150 de puncte (**NU** se trunchiază).

Pentru detalii puteți să vă uitați și peste **regulile generale** de trimitere a temelor.

- O temă care **NU** compilează va fi punctată cu 0.
- O temă care **NU** trece niciun test pe vmchecker va fi punctată cu 0.
- Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.
- Fiecare problemă va avea o limită de timp pe test (precizată pe pagina cu enunțul). 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.

6.1 Checker

- Arhiva se va trimite pe **VMchecker**, unde tema se va testa folosind un set de teste private.
- Pentru testarea locală, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu **resurse** a temei.
- Arhiva pe care o primiți (cu scheletul temei) conține fișierul README.checker.md, referitor la funcționarea checker-ului. Vă recomandăm să îl citiți. Checkerul este strict cu lucruri precum numele surselor voastre, numele regulilor din Makefile etc.
- **Punctajul pe teste** este cel de pe VMchecker și se acordă rulând tema doar cu testele private.

- Checkerul verifică doar existența unui README cu denumire corectă (README.md) și conținut nenul. **Punctajul final pe README și comentarii** se acordă la corectarea manuală a temei.
- La corectarea manuală se poate depuncta pentru **erori de coding style** care nu sunt semnalate de checker.
- Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă găsită în implementare, dacă vor considera acest lucru necesar.
- Pentru citirea în Java se recomandă folosirea **BufferedReader**.

7 FOLOSIRE CHATGPT

- Folosirea ChatGPT, Copilot sau a oricărui model de limbaj sau tool (denumit în continuare **LLM**) ce vă poate ajuta la rezolvarea temei, cu idei sau cod, este puternic descurajată, dar nu interzisă.
- În cazul în care folosiți LLM-uri, trebuie să specificați acest lucru în README, precum și modul în care acestea au fost folosite (ex: ce prompt-uri ați folosit), pentru fiecare problemă pentru care au fost folosite tool-uri.
- Pentru fiecare problemă rezolvată folosind un LLM, pentru care a fost specificat în README acest lucru, se va aplica o penalizare de 33% din punctajul acelei probleme.
- În cazul în care o problemă este rezolvată folosind un LLM, dar nu este specificat acest lucru în README, acest lucru se va considera **încercare de copiere** și se va sancționa conform **regulamentului**.

8 FORMAT ARHIVĂ

- Temele pot fi testate automat pe VMchecker. Acesta suportă temele rezolvate în C/C++ și Java.
- Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema1.zip** (ex: 399CX_PuiuGigel_Tema1.zip) și va conține:
 - Fișierul/fișierele sursă
 - Fișierul **Makefile**
 - Fișierul **README.md**
- **ATENȚIE!** Tema va fi compilată și testată **DOAR pe Linux**.
- **ATENȚIE!** Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (-O1, -O2, etc.).
- **ATENȚIE!** Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).

9 LINKS

- [Regulament general PA](#)
- [Google C++ Style Guide](#)
- [Google Java Style Guide](#)
- [Debugging și Structuri de Date](#)
- [Coding Tips pentru teme la PA \(OCW\)](#)