

PA - TEMA 1 - GREEDY & DP

Responsabili:

Barbu Miruna, Nan Mihai, Neațu Darius, Rotaru Alexandru, Popescu Silviu

Deadline soft: **12.04.2017**
Deadline hard: **16.04.2017**

Modificări

- 08.04.2017 22:30 - Modificare DEADLINE SOFT - prelungire cu 3 zile
- 08.04.2017 22:30 - Actualizare checker: testele NU au fost schimbate; s-a corectat un bug in checker care facea ca unele solutii gresite/cu hardcodari sa ia puncte la problema 2. Cei care au rezolvat corect problema 2 nu vor fi afectati.
- 28.03.2017 00:00 - Precizare timpi de rulare C/C++ && Java
- 28.03.2017 00:00 - [Adăugare secțiune checker](#) (pag. 12)
- 28.03.2017 00:00 - [Adăugare restricție pentru numele surselor](#) (pag. 13)
- 28.03.2017 00:00 - [Regula din Makefile pentru bonus se va numi run-p4](#) (pag. 13)
- 28.03.2017 00:00 - [Adăugare restricție pentru T](#) (pag. 6)
- 25.03.2017 20:20 - [Adăugare restricție pentru \$p_i\$](#) (pag. 8)
- 22.03.2017 18:35 - [Adăugare Obs.1 și Obs.2 la problema 2](#) (pag. 5)
- 22.03.2017 17:45 - [Corectare exemplu 1 problema 3](#) (pag. 9)
- 22.03.2017 12:00 - [Clarificare cerință problema 3](#) (pag. 7)

CUPRINS

0.1	Modificări	1
1	Problema 1: My points	3
1.1	Enunț	3
1.2	Date de intrare	3
1.3	Date de ieșire	3
1.4	Restricții și precizări	3
1.5	Testare și punctare	3
1.6	Exemple	4
1.6.1	Exemplu 1	4

2	Problema 2: Curiosity is in our DNA	5
2.1	Enunț	5
2.2	Date de intrare	5
2.3	Date de ieșire	5
2.4	Restricții și precizări	6
2.5	Testare și punctare	6
2.6	Exemple	6
2.6.1	Exemplu 1	6
2.6.2	Exemplu 2	6
3	Problema 3: Stropitorile lui Gigel	7
3.1	Enunț	7
3.2	Date de intrare	8
3.3	Date de ieșire	8
3.4	Restricții și precizări	8
3.5	Testare și punctare	8
3.6	Exemple	9
3.6.1	Exemplu 1	9
3.6.2	Exemplu 2	9
3.6.3	Exemplu 3	9
4	Bonus: Warriors	10
4.1	Enunț	10
4.2	Date de intrare	10
4.3	Date de ieșire	10
4.4	Restricții și precizări	10
4.5	Testare și punctare	10
4.6	Exemple	11
4.6.1	Exemplu 1	11
4.6.2	Exemplu 2	11
5	Punctare	12
5.1	Checker	12
6	Format arhivă	13
7	Links	14

PROBLEMA 1: MY POINTS

Enunț

Se dau N puncte pe axa Ox și M intervale. Să se determine numărul minim de intervale, din cele date, necesare ca să acopere în totalitate cele N puncte.

Date de intrare

Pe prima linie a fișierului **points.in** se află două numere întregi N și M .

Pe următoarea linie se află N numere întregi $x_1 < x_2 < x_3 < \dots < x_n$ reprezentând punctele.

Pe următoarele M linii se află câte două numere întregi x și y reprezentând capetele intervalelor.

Date de ieșire

În fișierul **points.out** se va scrie numărul minim de intervale necesare ca să acopere în totalitate cele N puncte.

Restricții și precizări

- $1 \leq N, M \leq 1000000$
- $0 \leq x_1 < x_2 < x_3 < \dots < x_n \leq 2 * 10^9$
- Se garantează că există mereu soluție.

Testare și punctare

- Punctajul maxim este de **22.5** puncte.
- Timpul de execuție:
 - C/C++: **2 s**
 - Java: **3 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **points.c**, **points.cpp** sau **Points.java**.

Exemple

Exemplu 1

Exemplu 1		
points.in	points.out	Explicație
2 2 1 4 1 4 2 7	1	Se va alege intervalul $[1, 4]$ care cuprinde ambele puncte: 1 și 4

PROBLEMA 2: CURIOSITY IS IN OUR DNA

Enunț

Robotul Curiosity se află pe Marte și explorează mediul. Acesta adună probe de sol și le analizează în speranța de a găsi secvențe organice de ADN. Din când în când se întâmplă ca robotul să descopere o astfel de secvență, însă nu poate să dea vestea mai departe până când nu este sigur că aceasta este într-adevăr extraterestră. De aceea el ar trebui să verifice dacă nu cumva secvența cercetată se poate forma din secvențele ADN cu care acesta a fost contaminat pe pământ.

Deoarece după lansare, oamenii au descoperit contaminarea, aceștia i-au transmis robotului informațiile despre toate secvențele de ADN în cauză.

Vi se cere să creați un program care să verifice pentru o secvență de ADN dată **dacă** aceasta **se poate** forma prin **întrepătrunderea** secvențelor ADN luate de pe pământ.

Obs.1: Procesul de întrepătrundere se realizează cu succes dacă s-au folosit **toate caracterele** din ADN-urile de intrare (se vor folosi **toate secvențele date**).

Obs.2: **Ordinea relativă** a caracterelor dintr-un ADN component este **aceeași** și în ADN-ul rezultat.

Date de intrare

Datele vor fi citite din fișierul **adn.in**.

Pe prima linie se va afla numărul **T** de teste din fișier.

Pe următoarele **T** linii se află câte un test.

Un test este format dintr-un întreg **N**, **N** șiruri de caractere reprezentând cele **N** secvențe de ADN și la final șirul de caractere target reprezentând ADN-ul ce se dorește testat.

Date de ieșire

Outputul va fi scris în fișierul **adn.out**.

Acesta va conține **T** linii.

Pe fiecare linie se va scrie 0 dacă ADN-ul testat nu se poate forma din celelalte **N** (deci este ADN extraterestru) sau 1 altfel.

Restricții și precizări

- lungimea oricărei secvențe de ADN ≤ 100
- $T \leq 205$
- 30% din teste au $N \leq 2$
- 70% din teste au $N \leq 3$
- 100% din teste au $N \leq 4$

Testare și punctare

- Punctajul maxim este de 35 puncte.
- Timpul de execuție:
 - C/C++: 1.2 s
 - Java: 1.2 s
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **adn.c**, **adn.cpp** sau **Adn.java**.

Exemple

Exemplu 1

Exemplu 1		
adn.in	adn.out	Explicație
1 2 xyyd xzzd xxyyzzdd	1	Răspunsul este 1, deoarece ADN-ul verificat poate fi obținut din cele două secvențe de ADN. Un mod prin care se poate obține este: se ia prima literă din ADN ₁ , prima literă din ADN ₂ , următoarele 2 litere din ADN ₁ , următoarele 2 litere din ADN ₂ , următoarea literă din ADN ₁ și ultima literă din ADN ₂

Exemplu 2

Exemplu 2		
adn.in	adn.out	Explicație
1 2 ab cd abdc	0	Răspunsul este 0 deoarece nu putem obține abdc, în ADN-ul component "cd", c este înaintea lui d, iar, în rezultat, ordinea este inversă.

PROBLEMA 3: STROPITORILE LUI GIGEL

Enunț

Gigel e într-o țară ciudată în care fiecare stadion are lungimea mult mai mare decât lățimea, astfel încât putem considera lățimea 1.

El a citit pe internet că lungimea maximă a unui stadion în această țară este mai mică sau egală decât $2 * 10^9$.

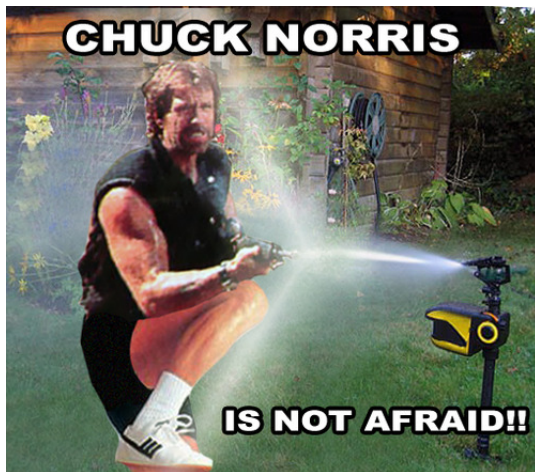
Pe fiecare stadion există un număr n de stropitori folosite pentru a uda gazonul înainte de meciuri.

Stropitoarea numărul i este plasată la coordonata x_i și are puterea p_i . O stropitoare poate uda un segment fix de iarbă: mai exact dacă udă la stânga va acoperi segmentul $[x_i - p_i, x_i]$, iar dacă udă la dreapta va acoperi segmentul $[x_i, x_i + p_i]$.

Gigel știe că stropitorile sunt ascunse în gazon, îngropate în pământ. La introducerea unei comenzi, o stropitoare poate ieși la suprafață și va uda fie la stânga, fie la dreapta (în funcție de comanda primită).

Dacă 2 stropitori udă în același punct, atunci una dintre ele este considerată redundantă și va fi blocată în poziție verticală. Considerăm că în această poziție de blocare, stropitoare i poate uda doar segmentul $[x_i, x_i]$ (deoarece se reduce puterea la 0).

Aceasta reprezintă o soluție de compromis, iar firma care se ocupă de administrarea stadionului o poate face gratis. Pentru orice altă operațiune, se percepe o taxă prea mare pentru a fi plătită de proprietar (preferă să cumpere un jucător decât să bage iar bani în stropitori).



Deoarece proprietarul stadionului probabil este supărat pentru că nu poate folosi toate stropitorile simultan la capacitate maximă (udă la stânga sau la dreapta), acesta se întreabă care este numărul maxim de stropitori folosite cu putere maximă.

Gigel dorește să îl înveselească pe patron și îți cere ajutorul pentru a determina **numărul maxim de stropitori de pe stadion care udă la putere maximă** (nu sunt blocate vertical). Pentru răspunsul corect îți oferă respect și punctaj maxim.

Dacă ești fată, îți dă și un **pupic**.

Date de intrare

TOATE datele de intrare se vor citi din fișierul **stropitori.in**. Acesta are următorul format:

- pe **prima linie** a fișierului stropitori.in se găsește **numele stadionului**.
- pe **a doua linie** se găsesc n și S , unde n este **numărul de stropitori** de pe stadion, iar S este **lungimea stadionului** (a porțiunii cu iarbă).
- pe **a treia linie** se găsesc n numere, al i -lea număr reprezentând **coordonata** x_i pentru stropitoarea cu numărul i (x_i exprimă distanța față de marginea din stânga a gazonului). Acest șir conține conține **elemente sortate și distincte**.
- pe **a patra linie** se găsesc n numere, al i -lea element reprezentând **puterea** p_i pentru stropitoarea i .

Date de ieșire

În fișierul **stropitori.out** se va scrie pe prima linie **DOAR numărul maxim de stropitori de pe stadion care udă la putere maximă**.

Restricții și precizări

- $1 \leq n \leq 1000007$
- $0 \leq x_1 < x_2 < x_3 < \dots < x_n \leq S \leq 2 * 10^9$
- $0 < p_i \leq S$, oricare ar fi $i = 1 : n$
- Gazonul este reprezentat de segmentul $[0, S]$. O stropitoare nu ar trebui să ude înafara gazonului.
- Numele stadionului poate conține litere ale alfabetului englez și (eventual) spații. Lungimea maximă este 128 de caractere.
- **Gigel își caută iubită!**

Testare și punctare

- Punctajul maxim este de **55** puncte.
- Timpul de execuție:
 - C/C++: **1 s**
 - Java: **1.5 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **stropitori.c**, **stropitori.cpp** sau **Stropitori.java**.

Exemple

Exemplu 1

Exemplu 1		
stropitori.in	stropitori.out	Explicație
Ghencea 4 10 1 2 5 7 1 2 1 4	3	<p>Pe stadionul Ghencea, cu lungimea 10, avem 4 stropitori.</p> <p>Configurație posibilă L, R, R, B cu semnificația: stropitoarea 1 udă spre stânga, stropitoarea 2 udă spre dreapta, 3 spre dreapta, 4 este blocată.</p> <p>Alte configurații: B, R, R, B B, B, L, B L, B, L, B.</p> <p>Configurația în care avem un număr maxim de stropitori este L, R, R, B (rezultatul este 3).</p>

Exemplu 2

Exemplu 2		
stropitori.in	stropitori.out	Explicație
Camp Nou 3 7 1 2 5 100 2 10	1	<p>Pe stadionul Camp Nou, cu lungimea 7, avem 3 stropitori.</p> <p>În orice configurație stropitoarea 1 este oprită, deoarece nu se îndeplinește niciuna dintre condițiile $1 - 100 \geq 0$ (udă la stânga, dar nu trebuie să ude dincolo de limita stadionului) sau $1 + 100 < 2$ (udă la dreapta, dar nu trebuie să ude în punctul 2 deoarece mai este o altă stropitoare acolo).</p> <p>Analog și pentru stropitoarea 3. Dacă ar uda spre stânga s-ar suprapune cu stropitoarea 2. Dacă ar uda spre dreapta, jetul de apă ar ajunge înafara stadionului.</p> <p>Singurele configurații posibile sunt: B, B, B și B, R, B.</p> <p>Răspunsul este 1.</p>

Exemplu 3

Exemplu 3		
stropitori.in	stropitori.out	Explicație
Old Trafford 3 7 1 2 5 1 2 2	3	<p>Pe stadionul Old Trafford, cu lungimea 7, avem 3 stropitori (doar în România se cumpără foarte multe degeaba).</p> <p>Numărul maxim este 3 și se obține pentru configurația L, R, R.</p>

BONUS: WARRIORS

Enunț

Noul joc pe calculator "Warriors" i-a captat pe toți studenții din Regie. Cum tu ești un mare campion de PvP, ai început să-l joci. Jocul decurge în felul următor: la începutul fiecărui meci, ți se dă un campion cu un level random din intervalul $[1, N]$ care (pentru motive încă nedezvăluite de developeri) nu ți se arată și K vieți.

Jocul decurge în felul următor: La începutul fiecărei runde, poți să îți alegi nivelul campionului cu care vrei să faci PvP. Dacă nivelul campionului tău este cel puțin egal cu nivelul campionului inamicului, câștigi runda, altfel vei pierde o viață din cele K .

Tu fiind un algoritmician de clasă (dacă nu erai nu mai dadeai la Poli, nu-i așa?) îți propui să găsești numărul minim de meciuri pe care le poți juca pentru a-ți afla level-ul.

Date de intrare

Pe prima și unica linie a fișierului **warriors.in** se vor găsi două numere întregi N și K .

Date de ieșire

În fișierul **warriors.out** se va scrie numărul minim de meciuri pe care le poți juca pentru a-ți afla level-ul.

Restricții și precizări

- Pentru 5p: $1 \leq N, K \leq 1000$
- Pentru 20p: $1 \leq N, K \leq 2 * 10^9$

Testare și punctare

- Punctajul maxim este de **25** puncte.
- Timpul de execuție:
 - C/C++: **0.5 s**
 - Java: **1 s**
- Sursa care conține funcția **main** trebuie obligatoriu denumită: **warriors.c**, **warriors.cpp** sau **Warriors.java**.

Exemple

Exemplu 1

Exemplu 1		
warriors.in	warriors.out	Explicație
10 1	10	În cel mai rău caz pot avea level 10, dar (pentru că trebuie să fim precauți) strategia optimă va fi să luăm nivelele în ordinea unul câte unul.

Exemplu 2

Exemplu 2		
warriors.in	warriors.out	Explicație
10 2	4	<p>Strategia este următoarea:</p> <p>Încerci levele-le 4, 7 și 9 în această ordine până pierzi cu unul dintre ele.</p> <p>Dacă pierzi cu 4: încerci pe rând 1, 2, 3</p> <p>Dacă pierzi cu 7: încerci pe rând 5, 6</p> <p>Dacă pierzi cu 9: îl încerci pe 8</p> <p>Dacă nu pierzi, îl încerci pe 10</p>

PUNCTARE

- Punctajul temei este de 125 puncte, distribuit astfel:
 - Problema 1: 22.5p
 - Problema 2: 35p
 - Problema 3: 55p
 - 5 puncte vor fi acordate pentru coding style
 - 7.5 puncte vor fi acordate pentru comentarii și README

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

Se poate obține un bonus de 25p rezolvând problema Warriros. Acordarea bonusului **NU** este condiționată de rezolvarea celorlate 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 mai jos și pe pagina cu enunțul). Dacă execuția programului pe un test al acelei probleme va dura mai mult decât limita de timp, veți primi automat o puncte pe testul respectiv și execuția va fi întreruptă.
- În fișierul README 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.

Checker

- Arhiva se va trimite pe **vmchecker**, unde tema se va testa folosind un set de teste private.
- Pentru testarea locala, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu **resurse** a temei.
- **Punctajul pe teste** este cel de pe vmchecker și se acordă ruland tema doar cu testele private.
- Checkerul verifică doar existența unui README cu denumire corectă și conținut nenul. **Punctajul final pe README și comentarii** se acordă la corectarea manuală a temei.
- La corectare 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.

FORMAT ARHIVĂ

- Temele pot fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++ și Java. Dacă doriți să realizați tema în alt limbaj, trebuie să-i trimiteți un email lui Traian Rebedea (traian.rebedea@cs.pub.ro), în care să îi cereți explicit acest lucru.
- Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema1.zip** (ex: 399CX_PuiuGigel_Tema1.zip sau 399CX_BucurGigel_Tema1.zip) și va conține:
 - Fișierul/ fișierele sursă
 - Fișierul **Makefile**
 - Fișierul **README** (fără extensie)
- Fișierul pentru make trebuie denumit obligatoriu **Makefile** și trebuie să conțină următoarele reguli:
 - **build**, care va compila sursele și va obține 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
 - **clean**, care va șterge executabilele generate
 - **run-p4**, care va rula executabilul pentru problema bonus (**doar dacă** ați implementat și bonusul)
- **ATENȚIE!** Funcția **main** din rezolvarea unei probleme se va găsi într-o sursă ce trebuie obligatoriu denumită astfel:
 - **points.c, points.cpp** sau **Points.java** - pentru problema 1
 - **adn.c, adn.cpp** sau **Adn.java** - pentru problema 2
 - **stropitori.c, stropitori.cpp** sau **Stropitori.java** - pentru problema 3
 - **warriors.c, warriors.cpp** sau **Warriors.java** - pentru problema 4
- **ATENȚIE!** Tema va fi compilată și testată **DOAR pe Linux**.
- **ATENȚIE!** Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.
- **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).

LINKS

- [Regulament general teme PA](#)
- [Google C++ Style Guide](#)
- [Google Java Style Guide](#)
- [Debugging și Structuri de Date](#)