PROGRAMAREA CALCULATOARELOR
Tema #1 Funcții, Vectori și Matrici

Deadline soft: 14.11.2021 23:55. Deadline hard 17.11.2021 23:55

Responsabili: Radu Nichita, Cristian Olaru, Sabina Horincar, Marius-Razvan Pricop, Alexandru Buzea, Bogdan Rizescu

Contents

Problema 1 - On the fly analysis

Enunț

În pădurea lui Ninel cei **N** copaci cresc pe un singur rând. Fiecare copac i ($0 \leq i \leq N-1$) are o înălțime x_i (exact ca în Figura 1). Facem abstracție de celelalte dimensiuni.

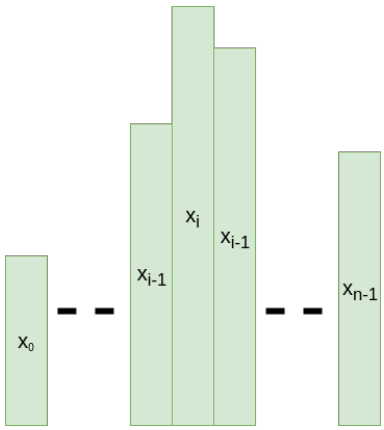

Figure 1: Pădurea lui Ninel.

Pădurarul dorește să realizeze câteva statistici pentru ocolul silvic, dacă s-ar dori tăierea unei mulțimi de copaci în vederea reîntineririi pădurii. Mai exact, el dorește să ia în calcul că va tăia doar acei copaci care sunt deja prea mari și umbresc copacii vecini.

Un copac i este numit **special** dacă și numai dacă sunt adevărate următoarele condiții:

1. există copacii $i-1$ și $i+1$;
2. $x_{i-1} < x_i$ și $x_i > x_{i+1}$.

În această problemă nu vom tăia copaci, ci vom calcula următoarele:

1. **S** = suma înălțimilor copacilor speciali;
2. **m_a** = media aritmetică a înălțimilor copacilor speciali;
3. **xmax_impar** = înălțimea maximă a unui copac special cu indice impar;
4. **xmin_par** = înălțimea minimă a unui copac special cu indice par.

Restricții și precizări

- $0 < N \leq 10^7$
- $0 < x_i < 10^6$

ATENȚIE! Șirul dat este foarte mare, astfel încât stocarea acestuia nu este posibilă folosind noțiunile învățate până acum. Din fericire, problema se poate rezolva și fără a memora șirul. O rezolvare care reține valorile într-un vector va obține maxim 4 puncte din 15 posibile, datorită restricțiilor de memorie de pe VMchecker. **Date de intrare**

Toate datele se vor citi de la **STDIN**.

```
N
x_0 x_1 x_2 ... x_{N-1}
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**. Numerele reale se vor afișa cu **7 zecimale** exacte.

```
S
m_a
xmax_impar
xmin_par
```

Exemplu

Date de intrare

```
9
3 1 3 2 2 5 2 4 2
```

Date de ieșire

```
12
4.0000000
5
3
```

Explicație

Pădurea descrisă are 9 copaci, numerotați de la 0 la 8.

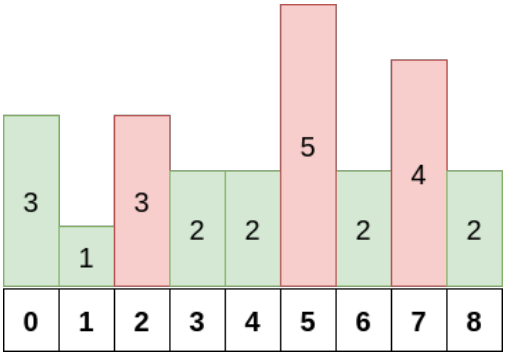

Figure 2: Copacii din exemplul 1.

Copaci sunt cei din Figura 2. Explicăm:

- 2 este special: deoarece înățimea acestuia este 3, iar a vecinilor 1 și 2;
- 5 este special: deoarece înățimea acestuia este 5, iar a vecinilor 2;
- 7 este special: deoarece înățimea acestuia este 4, iar a vecinilor 2;
- ceilalți nu sunt speciali:
  - 0,8: nu au câte 2 vecini;
  - 1, 3, 4, 6: au cel puțin un vecin cu înățime mai mare sau egală decât a copacului în cauză.

În consecință, valorile se calculează astfel:

```
S = x2 + x5 + x7 = 12
m_a = 12 / 3 = 4
xmax_impar = max({x5, x7}) = max({5, 4}) = 5
xmin_par = min({x2}) = min({3}) = 3
```

Problema 2 - Proprietăți ale numerelor naturale

Enunț

O proprietate generală a numerelor naturale spune că pentru orice număr natural **N**, făcând, în mod repetat, **diferența** dintre numărul format cu aceleași cifre, dar în ordine descrescătoare (n_descrescător) și numărul format din cifrele sale în ordine crescătoare (n_crescător), după un număr variabil (k) de pași se ajunge la o secvență de numere **(p1, p2, ..., pm)** de dimensiune **m** care se repeta la infinit în calculul diferențelor.

Această proprietate, se păstrează și pentru numerele în scrierea cărora o cifră poate apărea de orice număr de ori sau care conțin cifra 0 dacă:

- Numerele n_descrescător și n_crescator se formează din cu tot același număr de apariții ale cifrelor de la 1 la 9;
- Cifra 0 este eliminată complet din scrierea lui n_crescător.

Scrieți un program care, citind de la tastatură un număr natural N, urmărește (în mod repetat) algoritmul de calcul al secvenței de diferențe între n_descrescător și n_crescător și oferă ca output:

- Pe primul rând, numărul de diferențe realizate până la întâlnirea primului număr din secvența periodică;
- Pe al doilea rând, secvența de numere care se repetă, fiecare o singură dată. și în ordinea în care acestea sunt descoperite prin algoritmul propus.

Restricții și precizări

- $0 \leq N \leq 999.999.999$;
- Pentru oricare N din input, se garantează că $m + k \leq 100$.

Exemplul 1

Date de intrare

```
14325
```

Date de ieșire

```
1
82962 75933 63954 61974
```

Explicație

Conform algoritmului, diferențele calculate sunt:

- 54321 - 12345 = 41976
- 97641 - 14679 = 82962
- 98622 - 22689 = 75933
- 97533 - 33579 = 63954
- 96543 - 34569 = 61974
- 97641 - 14679 = 82962

Singurul număr calculat care nu apare în secțiunea periodică este 41976.

Exemplul 2

Date de intrare

```
5309
```

Date de ieșire

```
2
6174
```

Explicație

Conform algoritmului, diferențele calculate sunt:

- 9530 - 359 = 9171
- 9711 - 1179 = 8532
- 8532 - 2358 = 6174
- 7641 - 1467 = 6174

Primele 2 numere nu fac parte din perioadă, apoi 6174 se repetă la infinit.

Problema 3 - Perfecționarea punctajelor

Enunț

În Universitatea **CodeInVim**, bursele se acordă studenților cu cele mai multe puncte. Vă propunem să învățați să calculați câte puncte poate strânge un student.

Vom folosi un scenariu simplu în care presupunem că pe parcursul unui an un student are **N** materii. Fiecare materie **i** este caracterizată printr-o notă x_i (obținută de student) și un număr de credite c_i (care reflectă dificultatea materiei).

Punctajul îl vom defini astfel:

$$P = \sum_{i=0}^{N-1} (x_i * c_i) \tag{1}$$

Mihai tocmai și-a aflat toate notele din anul 1 și se gândește să păstreze la îndemână cursurile pentru materiile la care va merge la măriri, deoarece vrea neapărat să fie printre bursieri.

El se întreabă care este numărul minim **m** de materii (cu notă diferită de 10) la care ar fi trebuit să ia 10, astfel încât punctajul lui să fie mai mare sau egal cu un **p_min** dat.

Restricții și precizări

- $1 \leq N \leq 100$;
- $1 \leq x_i \leq 10$ (x_i întreg)
- $1 \leq c_i \leq 6$ (c_i întreg)

HINT: Ce ați alege dacă ați avea voie să modificați **o singură notă**?

Date de intrare Toate datele se vor citi de la **STDIN**.

```
n
x_0 x_1 x_2 ... x_{n-1}
c_0 c_1 c_2 ... c_{n-1}
p_min
```

Date de ieșire

Toate datele se vor afișa la **STDOUT**. Se va scrie numărul căutat sau **-1** dacă problema nu are soluție.

```
m
```

Exemplul 1

Date de intrare

```
5
10 10 5 7 5
1 2 5 6 4
118
```

Date de ieșire

```
1
```

Explicație

Punctajul obținut este **10 * 1 + 10 * 2 + 5 * 5 + 7 * 6 + 5 * 4 = 117** Se pot mări:

- 5 (pondere 5) la 10 => punctajul crește cu 5 * (10 - 5) = 25;
- 7 (pondere 6) la 10 => punctajul crește cu 6 * (10 - 7) = 18;
- 5 (pondere 4) la 10 => punctajul crește cu 4 * (10 - 5) = 20.

Întrucât putem alege una dintre aceste variante, rezultă că numărul minim de măriri de care avem nevoie pentru a obține cel puțin 118 puncte este 1.

Exemplul 2

Date de intrare

```
5
10 10 5 7 5
1 2 5 6 4
160
```

Date de ieșire

```
2
```

Explicație

- Punctajul obținut și notele ce se pot mări sunt aceleași note ca în primul exemplu (deoarece notele și numărul de credite sunt identice).
- În acest caz, o singură mărire nu este suficientă (scor maxim 117 + 25 = 142). De aceea mai este necesară încă o mărire care să asigure diferența rămasă. Răspuns **2**.

Problema 4 - Nonogram checker

Enunț

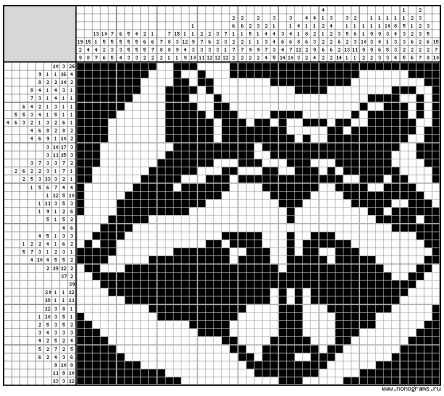

Figure 3: Grid de Nonogram.
Sursa imaginii: https://www.nonograms.org/nonograms/i/26645

Nonogram este un joc, similar Sudoku, care își propune completarea cu căsuțe albe și negre a unui grid de dimensiune variabilă (N linii x M coloane), pe baza unor restricții precizate pentru fiecare linie și coloană.

Astfel, pentru fiecare linie, respectiv fiecare coloană se precizează numărul de grupuri de căsuțe negre precum și dimensiunea acestora, în ordinea în care ele apar (de la stânga la dreapta pentru linii și de sus în jos pentru coloane). Două grupuri consecutive vor fi întotdeauna despărțite prin cel puțin o căsuță albă.

Gigel își dorește să creeze un program care să-l ajute să trișeze la acest joc, verificând, pentru el, dacă a completat corect un astfel de grid. El vă recomandă să încercați jocul, nu doar pentru că e fain, ci și pentru că așa înțelegeți cel mai ușor mecanismul de joc.

Restricții și precizări

- $1 \leq N, M \leq 100$
- Puteți juca Nonogram la adresa https://www.nonograms.org.

Date de intrare

Datele de intrare se vor citi de la **STDIN** altfel:

- Pe prima linie se va găsi un număr T, reprezentând numărul de puzzle-uri ce vor fi verificate;
- Pe următoarele linii puzzle-urile ce urmează a fi verificate.

Pentru fiecare puzzle:

- Pe prima linie, se vor găsi două numere naturale N și M reprezentând dimensiunile gridului;
- Pe următoarele N linii restricțiile pentru fiecare dintre linii, în ordinea crescătoare a indicelui liniei;
- Pe următoarele M linii restricțiile pentru fiecare dintre coloane, în ordinea crescătoare a indicelui coloanei;
  - Pentru o linie/coloană cu K grupuri colorate, restricțiile vor fi specificate prin intermediul a K+1 numere naturale reprezentând, în ordine: numărul de grupuri colorate de pe acea linie sau coloană (valoarea lui K), dimensiunea grupului pentru fiecare grup de căsuțe colorate.
- Pe următoarele N linii și M coloane, grid-ul completat de Gigel, dat ca o matrice de 1 și 0 în care o valoare de 0 reprezentă o căsuță albă, iar o valoare de 1 reprezintă o căsuță colorată.

Date de ieșire

Programul va afișa, pentru fiecare puzzle, la **STDOUT**, pe câte o linie distinctă, mesajul **Corect**, dacă gridul este corect sau **Eroare** dacă gridul nu respectă restricțiile date.

Exemplul 1

Date de intrare

```
1
10 10
2 2 1
2 1 1
1 1
1 7
1 10
2 8 1
2 8 1
2 8 1
1 10
1 7
1 7
1 7
1 8
2 1 7
1 7
2 1 7
2 1 7
2 1 5
3 1 1 1
1 5
0 0 0 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 0
```

Date de ieșire

```
Corect
```

Explicație Datele de intrare reprezintă grid-ul din Figura 4, care este completat corect.

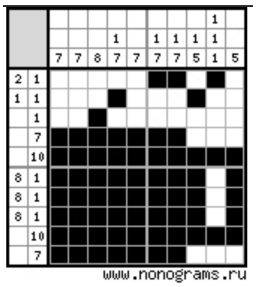

Figure 4: Grid de Nonogram din exemplul problemei 4.
Sursa imaginii: https://www.nonograms.org

Exemplul 2

Date de intrare

```
1
10 10
2 2 1
2 1 1
1 1
1 7
1 10
2 8 1
2 8 1
2 8 1
1 10
1 7
1 7
1 7
1 8
2 1 7
1 7
2 1 7
2 1 7
2 1 5
3 1 1 1
1 5
0 0 0 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 1
```

Date de ieșire

```
Eroare
```

Explicație: Input-ul reprezintă același grid ca în Exemplul 1, dar cu o căsuță neagră în plus.

Regulament

Regulamentul general al temelor se găsește pe ocw (Temele de casă). Vă rugăm să îl citiți integral înainte de continua cu regulile specifice acestei teme.

Arhivă

Soluția temei se va trimite ca o arhivă **zip**. Numele arhivei trebuie să fie de forma **Grupă_NumePrenume_TemaX.zip** - exemplu: `311CA_NichitaRadu_Tema1.zip`.

Arhiva trebuie să conţină în directorul **RĂDĂCINĂ** doar următoarele:

- Codul sursă al programului vostru (fișierele **.c** și eventual **.h**).
- Un fișier **Makefile** care să conțină regulile **build** și **clean**.
  - Regula **build** va compila codul vostru și va genera următoarele executabile:
    - **ninel** pentru problema 1
    - **vectsecv** pentru problema 2
    - **codeinvim** pentru problema 3
    - **nomogram** pentru problema 4
  - Regula **clean** va șterge **toate** fișierele generate la build (executabile, binare intermediare etc).
- Un fișier **README** care să conțină prezentarea implementării alese de voi. **NU** copiați bucăți din enunț.

Arhiva temei **NU** va conține: fișiere binare, fișiere de intrare/ieșire folosite de checker, checkerul, orice alt fișier care nu este cerut mai sus.

Numele şi extensiile fişierelor trimise **NU** trebuie să conţină spaţii sau majuscule, cu exceptia fișierului README (care este are nume scris cu majuscule și nu are extensie. Nerespectarea oricărei reguli din secțiunea **Arhivă** aduce un punctaj **NUL** pe temă.

Checker

Pentru corectarea aceste teme vom folosi scriptul **check** din arhiva **check_first_blood.zip** din secțiunea de resurse asociată temei. Vă rugăm să citiți **README.md** pentru a ști cum să instalați și utilizați checkerul.

Punctaj

Distribuirea punctajului:

- Problema 1: 15p
- Problema 2: 20p
- Problema 3: 20p
- Problema 4: 20p
- Claritatea și calitatea codului: 20p
- Claritatea explicaţiilor din README: 5p

ATENȚIE! Punctajul maxim obținut pe temă este de 100p. Acesta reprezintă 0.5p din nota finală la această materie.

Reguli și precizări

- Punctajul pe teste este cel acordat de **check**, rulat pe **vmchecker**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).
- Punctajul pe calitatea explicațiilor și a codului se acordă în mai multe etape:
  - **corectare automată**
    - Checkerul va încerca să detecteze în mod automat probleme legate de coding style și alte aspecte de organizare a codului.
    - Acesta va puncta cu maxim 20p dacă nu sunt probleme detectate.
    - Punctajul se va acorda proporțional cu numărul de puncte acumulate pe teste din cele 80p.
    - Checkerul poate să aplice însă și penalizări (exemplu pentru warninguri la compilare) sau alte probleme descoperite la runtime.
  - **corectare manuală**
    - Tema va fi corectată manual și se vor verifica și alte aspecte pe care checkerul nu le poate prinde. Recomandăm să parcurgeți cu atenție tutorialul de coding-style de pe ocw.cs.pub.ro.
    - Codul sursă trebuie să fie însoțit de un fișier README care trebuie să conțină informațiile utile pentru înțelegerea funcționalității, modului de implementare și utilizare a programului. Acesta evaluează, de asemenea, abilitatea voastră de a documenta complet și concis programele pe care le produceți și va fi evaluat, in mod analog CS, de către echipa de asistenți. In funcție de calitatea documentației, se vor aplica depunctări sau bonusuri.
    - Deprinderea de a scrie cod sursă de calitate, este un obiectiv important al materiei. Sursele greu de înteles, modularizate neadecvat sau care prezintă hardcodări care pot afecta semnificativ mentenabilitatea programului cerut, pot fi depunctate adițional.
    - În această etapă se pot aplica depunctări mai mari de 20p.
    - O temă care **NU** compilează cu -Wall -Wextra este depunctată la corectarea manuală cu 5p (punctajul echivalent pentru warnings).

Alte precizări

- Tema trebuie trimisă sub forma unei arhive pe site-ul cursului curs.upb.ro si pe vmchecker.
- Tema poate fi submisă de oricâte ori fără depunctări până la deadline. Mai multe detalii se găsesc în regulamentul de pe ocw.
- O temă care **NU** compilează pe vmchecker **NU** va fi punctată.
- O temă care compilează, dar care **NU** trece niciun test pe vmchecker, NU va fi punctată.
- Punctajul pe teste este cel acordat de **check** rulat pe **vmchecker**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).
- Ultima temă submisă pe vmchecker poate fi rulată de către responsabili de mai multe ori în vederea verificării faptului că nu aveți buguri în sursă. Vă recomandăm să verificați **local** tema de mai multe ori pentru a verifica că punctajul este mereu același, apoi să încărcați tema.