Programarea Calculatoarelor - Seria CC

Tema 1

Publicarea enuntului: 24.11.2020
Data ultimei modificari a enuntului: 25.11.2020

Termen de predare: 10.12.2020, ora 23:55

Nu se accepta teme trimise dupa termenul de predare !

Responsabili tema: *Bogdan Nutu, Constantin Raducanu, Gabriel Mocanu*

Profesor titular: *Carmen Odubasteanu*

Facultatea de Automatica si Calculatoare
Universitatea Politehnica din Bucuresti
Anul universitar 2020-2021

Problema 1 - Operatii cu intervale

Fiind date doua intervale inchise de numere intregi, de forma [a, b], definim urmatoarele operatii pe ele:

- 1 - intersectie
- 2 - reuniune extinsa - similara cu reuniunea, dar pentru intervale disjuncte va returna reuniunea celor doua intervale si a intervalului cuprins intre acestea. De exemplu, pentru intervalele [1, 3] si [7, 12] rezultatul va fi [1, 12]. In rest, pentru intervale nedisjuncte, rezultatul este ca cel al reuniunii clasice.

In continuare, vom defini doua tipuri de interogari pe care le putem efectua:

- **1 - interogare simpla**
  Acest tip de interogare primeste doua intervale si o operatie din cele definite anterior si va returna rezultatul.
  Inputul este de forma ***operatie x1 x2 x3 x4***, unde ***operatie*** are valoarea 1 sau 2, corespunzatoare intersectiei, respectiv reuniunii, iar output-ul este de forma ***x5 x6***, reprezentand rezultatul aplicarii operatiei peste intervalele [x1,x2] si [x3, x4].
  Exemple:
  - 1 2 5 3 9 ⇒ 3 5
  - 2 1 4 2 5 ⇒ 1 5
  - 2 1 2 3 4 ⇒ 1 4
  - 1 1 2 3 4 ⇒ 0

- **2 - interogare compusa**
  Acest tip de interogare presupune aplicarea succesiva a M operatii pe M+1 intervale date ca input.
  O astfel de interogare are urmatorul format:
  - Pe prima linie se citeste un numar natural M - nr de operatii
  - Pe a doua linie se citesc M numere separate prin spatiu (1 sau 2) - operatiile
  - Apoi se citesc M+1 linii de forma “a b” - cele M+1 intervale

Rezultatul se calculeaza in felul urmator: Se aplica prima operatie pe primele 2 intervale. Apoi se aplica a doua operatie pe rezultat si al treilea interval, si tot asa pana la ultima operatie, care se aplica pe ultimul rezultat partial si ultimul interval.

Exemplu:

- 5
  2 2 1 2 1
  0 3
  1 2
  4 5
  6 8
  1 4
  2 3

Output:
2 3

Explicatie:
[0, 3] 2 [1, 2] ⇒ [0, 3]
[0, 3] 2 [4, 5] ⇒ [0, 5]
[0, 5] 1 [6, 8] ⇒ 0
0 2 [1, 4] ⇒ [1, 4]
[1, 4] 1 [2, 3] ⇒ [2, 3]

Date de intrare:

- Pe prima linie se citeste N - numarul de interogari
- Apoi se citesc N interogari de forma:
  - Pe prima linie se citeste tipul interogarii (1-simpla, 2-compusa)
  - Pe urmatoarele linii se citeste interogarea, dupa cum am explicat mai sus:
    - O singura linie pentru cea simpla
    - M + 3 linii pentru cea compusa, unde M este numarul de operatii (o linie pentru M, o linie cu cele M operatii si M+1 linii pentru intervale)

Date de iesire:

- N linii cu cate doua numere separate prin spatiu, reprezentand rezultatele celor N interogari

Exemplu:		Output:
3	//N - nr interogari	2 2
1	//prima interogare - simpla	2 5
1 0 2 2 4	//[0,2] 1 [2,4]	1 2
2	//a 2a interogare - compusa	
2	//M - nr de operatii ale interogarii 2	
1 2	//cele M operatii ale interogarii 2	
1 3	//M+1 intervale	
2 4		
2 5		
1	//a 3a interogare - simpla	
2 1 1 2 2	//[1,1] 2 [2,2]	

Restrictii si precizari:

- Pentru cele doua operatii, trebuie sa implementati doua functii cu urmatoarele antete:
  - **void intersectie (int x1, int x2, int x3, int x4, int *x5, int *x6);**
  - **void reuniune (int x1, int x2, int x3, int x4, int *x5, int *x6);**
- Toate intervalele cu care se lucreaza sunt inchise, de numere intregi.
- Este posibil ca anumite interogari sa aiba ca rezultat **multimea vida**. In cazul acesta, veti afisa la output valoarea constanta **0 (zero)**.
  - Exemple: 1 0 1 2 3 sau 1 -1 3 7 8 ⇒ afisati “0”
- Este posibil ca anumite interogari ca aiba ca rezultat un punct, a. In cazul acesta, rezultatul este considerat intervalul [a, a].
  - Exemple: 1 1 2 2 3 ⇒ 2 2 sau 2 1 1 1 1 ⇒ afisati “1 1”
- Functiile **intersectie** si **reuniune** vor returna rezultatul in parametri x5 si x6, reprezentand intervalul [x5, x6].

Problema 2 - Joc

Fie o matrice de dimensiune 4x4 ale carei elemente sunt 0 sau 1, pe care o codificam sub forma unui numar intreg pe 16 biti. Astfel, pentru a obtine matricea asociata unui anumit numar, va trebui sa il transformam in baza 2.

De exemplu, n = 12345. Reprezentarea lui n in baza 2 (pe 16 biti) este 0011000000111001. Asadar, matricea asociata lui n va fi:

0 0 1 1
0 0 0 0
0 0 1 1
1 0 0 1

Completarea matricei se face astfel incat bitii cei mai semnificativi sunt pe prima linie a sa.

In mod analog, daca pornim de la urmatoarea matrice:

0 1 1 0
1 0 0 0
1 1 0 1
0 0 1 0

Pentru a obtine numarul n, vom concatena liniile matricei si vom gasi reprezentarea in baza 2 a lui n (0110100011010010), dupa care vom transforma in baza 10 si obtinem n = 26834.

Valoarea 1 pe o anumita pozitie din matrice marcheaza faptul ca avem un soldat in pozitia respectiva, iar valoarea 0 arata ca nu avem nimic.

In continuare, ne dorim sa mutam acesti soldati din pozitiile lor in alte pozitii, iar daca doi soldati se vor intersecta, acestia se vor anihila reciproc si vor fi eliminati amandoi, punand valoarea 0 atat in pozitia initiala, cat si in cea finala. In schimb, daca un soldat va fi mutat intr-o pozitie in care nu avem nimic, pozitia initiala va deveni 0, iar cea finala 1. Daca in pozitia initiala nu avem nimic, nu se va efectua nicio transformare.

Date de intrare:

- Pe prima linie avem numarul intreg **n**, care reprezinta configuratia initiala a hartii.
- Pe a doua linie avem un numar intreg **m** - numarul de mutari.
- **m** linii de forma **xi yi xf yf**, reprezentand linia si coloana pozitiei initiale, respectiv ale pozitiei finale.

Date de iesire:

- Un numar intreg in baza 10 - configuratia finala a hartii.

Exemplu:	Output:
26834	146
4	
0 1 0 2	
1 0 0 1	Explicatie:
2 2 3 1	0000000010010010 in baza 10
2 1 0 1	

Configuratia hartii se va modifica in felul urmator:

0 1 1 0		0 0 0 0		0 1 0 0		0 1 0 0		0 0 0 0
1 0 0 0	⇒	1 0 0 0	⇒	0 0 0 0	⇒	0 0 0 0	⇒	0 0 0 0
1 1 0 1		1 1 0 1		1 1 0 1		1 1 0 1		1 0 0 1
0 0 1 0		0 0 1 0		0 0 1 0		0 0 1 0		0 0 1 0

Am marcat cu rosu pozitiile de start si cu albastru pe cele finale, pentru fiecare mutare in parte.

Restrictii si precizari:

- 0 <= n <= 65535 (2^16 - 1)
- Dupa cum am aratat si in exemplu, mutarile se efectueaza in ordinea primita. Fiecare mutare se va face pe harta obtinuta in urma mutarii anterioare.
- Trebuie sa implementati urmatoarele functii:
  - **void dec_to_bin (int n, char s[]);** - transforma numarul n din baza 10 in baza 2 si il intoarce ca sir de caractere (s) de ‘0’ sau ‘1’.
  - **int bin_to_dec(char s[]);** - primeste un numar in baza 2 ca sir de caractere si intoarce valoarea in baza 10
  - Alte doua functii care sa transforme un sir de caractere de ‘0’ si ‘1’ intr-o matrice 4x4 cu numerele 0 si 1 (pe care veti efectua ulterior modificarile), respectiv operatia inversa. Implementarile sunt la alegerea voastra.

Problema 3 - Filtre

Scopul aceste probleme este sa construiti un program care sa aplice anumite filtre pe o imagine. De data aceasta, vom simplifica foarte mult situatia reala si vom reprezenta o imagine ca o **matrice N x M de numere reale**, fiecare element al matricei fiind asociat cu un pixel din imagine.

Pentru a aplica un filtru peste o imagine, este nevoie sa actualizam valorile tuturor pixelilor, folosind valorile lor initiale si valorile din matricea de transformare. Vom explica acest procedeu in detaliu mai jos. Pentru cazul de fata vom folosi doar filtre 3x3.

Filtrele care pot fi aplicate pe imagine sunt *smooth* si *blur*.

Aplicarea unui filtru si calculul valorilor pixelilor dupa o transformare se fac in modul urmator:

- Sa presupunem ca avem un filtru f, cu urmatoarea matrice de transformare:

```
      f11   f12   f13
K =   f21   f22   f23
      f31   f32   f33
```

- Fie urmatoarea submatrice 3x3 din matricea initiala:

```
…     …             …             …             …
…     a[i-1][j-1]   a[i-1][j]     a[i-1][j+1]   …
…     a[i][j-1]     a[i][j]       a[i][j+1]     …
…     a[i+1][j-1]   a[i+1][j]     a[i+1][j+1]   …
…     …             …             …             …
```

- Ne dorim sa actualizam pixelul a[i][j].
- Valoarea pixelului de pe aceeasi pozitie din matricea (imaginea) rezultata in urma filtrarii va fi:

```
B[i][j]  =  f11 * a[i-1][j-1] + f12 * a[i-1][j] + f13 * a[i-1][j+1] +
            f21 * a[i][j-1] + f22 * a[i][j] + f23 * a[i][j+1] +
            f31 * a[i+1][j-1] + f32 * a[i+1][j] + f33 * a[i+1][j+1]
```

- Asadar, trebuie sa inmultiti valorile pixelului curent si pe cele ale vecinilor sai cu valorile de pe pozitiile corespunzatoare din matricea de filtrare.

Atentie! : Nu se efectueaza inmultirea clasica a doua matrice 3x3, ci valoarea noua a pixelului va fi o suma de 9 produse a[i][j] * f[i][j].

Dupa cum am spus, pentru aceasta tema vom considera doua filtre, ale caror matrice de transformare sunt urmatoarele:

- **Smooth**

$$K = 1/9 * \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

- **Blur**

$$K = 1/16 * \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$

Aceste doua filtre pot fi aplicate de oricate ori pe matricea initiala.

Date de intrare:

- Pe prima linie se citesc **N** si **M** - dimensiunile imaginii/matricei initiale
- Apoi se citesc **N** linii a cate **M** numere reale - valorile initiale ale pixelilor imaginii
- Un numar natural **p** - numarul de filtre care se vor aplica pe imagine
- **p** siruri de caractere, fiecare pe cate o linie, reprezentand filtrele care se vor aplica succesiv peste imaginea initiala (“smooth” sau “blur”).

Date de iesire:

- N linii a cate M elemente reale - valorile finale ale pixelilor. Ele vor fi afisate cu 3 zecimale, pe 8 caractere.

Restrictii si precizari:

- $0 < N, M < 100$
- Dupa cum am precizat, in calculul unui pixel dupa o operatie de filtrare avem nevoie de valorile celor 8 vecini ai sai din matricea initiala, precum si de valoarea sa initiala. Asadar, apare intrebarea “Ce se intampla cu pixelii de pe margine, care nu sunt inconjurati de vecini?”. In problema aceasta vom lasa acele valori neschimbate.
- Atunci cand se efectueaza o filtrare, in calculul tuturor pixelilor din matricea rezultata trebuie sa folositi valorile initiale ale pixelilor/vecinilor de la filtrarea curenta (adica valorile din ultimul rezultat partial).

Altfel spus, trebuie sa retineti intr-o alta matrice noile valori, nu sa actualizati valorile in matricea initiala, pentru ca astfel veti pierde anumite valori initiale.

- Cele p filtre se aplica succesiv, deci veti avea nevoie de ultimul rezultat pentru a aplica urmatorul filtru. La final veti afisa doar matricea rezultata in urma aplicarii ultimului filtru.
- Trebuie sa implementati cate o functie care sa primeasca ca parametru o imagine (matrice) si sa aplice peste aceasta un filtru, pentru fiecare din cele 2 filtre. Veti apela apoi aceste functii in main.

Exemplu:

```
4 4
1 2.5 3 0
2.25 5 1 2
1.75 3 8 2
0 1 10 4.5
3
smooth
blur
smooth
```

Output:

```
1.000 2.500 3.000 0.000
2.250 2.566 2.455 2.000
1.750 3.066 3.566 2.000
0.000 1.000 10.000 4.500
```

Explicatie:

```
Initial
   1.000   2.500   3.000   0.000
   2.250   5.000   1.000   2.000
   1.750   3.000   8.000   2.000
   0.000   1.000  10.000   4.500
+ SMOOTH
   1.000   2.500   3.000   0.000
   2.250   3.056   2.944   2.000
   1.750   3.556   4.056   2.000
   0.000   1.000  10.000   4.500
+ BLUR
   1.000   2.500   3.000   0.000
   2.250   2.783   2.753   2.000
   1.750   3.071   3.986   2.000
   0.000   1.000  10.000   4.500
+ SMOOTH
   1.000   2.500   3.000   0.000
   2.250   2.566   2.455   2.000
   1.750   3.066   3.566   2.000
   0.000   1.000  10.000   4.500
```

Problema 4 - Soldati

Fiind data o matrice de dimensiuni N x M, cu elementele 1 (soldat) si 0 (civil), returnati indicii celor mai slabe k randuri din matrice.

Un rand i este mai slab decat alt rand j, daca numarul soldatilor din randul i este mai mic decat numarul soldatilor din randul j. Daca au acelasi numar de soldati, randul i este mai slab daca i < j.

Date de intrare:

- N, M - dimensiunile matricei
- N linii a cate M elemente (0 sau 1) - elementele matricei
- k

Date de iesire:

- Indicii celor mai slabe k linii, afisati pe o linie, cu spatiu intre ei

Exemplu:	Output:	Explicatie:
5 5	2 0 3	Nr de soldati de pe fiecare rand este:
1 1 0 0 0		0 ⇒ 2
1 1 1 1 0		1 ⇒ 4
1 0 0 0 0		2 ⇒ 1
1 1 0 0 0		3 ⇒ 2
1 1 1 1 1		4 ⇒ 5
3		Cele mai slabe 3 randuri sunt 2, 0, 3. Randul 0 e considerat mai slab decat randul 3 pentru ca au acelasi numar de soldati, iar 0 este mai mic decat 3.

Restrictii si precizari:

- 0 < N, M < 100
- 0 < k <= N

Observatii si precizari

★ Separati logica programelor in mai multe functii, asa cum vi s-a cerut in enunturi! Nu se vor puncta sursele in care tot programul este scris in main!

★ **Este interzisa folosirea variabilelor globale.**

★ Tema va fi rezolvata **obligatoriu** in limbajul **C**. Nu folositi sintaxa sau instructiuni specifice limbajului C++.

★ Veti incarca o arhiva **zip** numita <grupa>_<nume>_<prenume>_Tema1.zip. De exemplu, 313CC_Popescu_Andrei_Tema1.zip

★ Arhiva va contine fisierele sursa, Makefile si README. Fisierele trebuie sa se regaseasca direct in radacina arhivei, nu in alte subdirectoare !!!.

★ Fiecare problema va fi scrisa intr-un fisier sursa separat, numit problemaX.c, X = {1,2,3,4}.

★ **Toate citirile se fac de la tastatura. Nu folositi fisiere.**
Datele de intrare sunt descrise pentru fiecare cerinta individual si vor fi citite de la tastatura folosind scanf. Pentru a nu fi nevoiti sa scrieti de fiecare data inputul, puteti redirecta un fisier in care sa scrieti datele pe care de obicei le introduceati de la tastatura, si sa rulati folosind formatul urmator:
./nume_executabil < numele_fisierului_de_intrare

★ Precizati in **README** ce probleme ati rezolvat, cat timp v-a luat fiecare si explicati, pe scurt, implementarea problemelor (comentariile din cod vor documenta mai amanuntit rezolvarea).

★ Este recomandat ca liniile de cod si cele din fisierul README sa nu depaseasca 80 de caractere.

★ Folositi un coding style astfel incat codul sa fie usor de citit si de inteles. De exemplu:

- ➢ Dati nume corespunzatoare variabilelor si functiilor.
- ➢ Nu adaugati prea multe linii libere sau alte spatii goale unde nu este necesar (exemplu: nu terminati liniile in spatii libere, trailing whitespaces; nu adaugati prea multe linii libere intre instructiuni sau la sfarsitul fisierului). Principalul scop al spatiilor este indentarea.
- ➢ Fiti consecventi. Coding style-ul are o natura subiectiva.
- ➢ Pentru sugestii de coding style, puteti intra aici.

★ **Temele sunt strict individuale. Copierea temelor va fi sanctionata. Persoanele cu portiuni de cod identice nu vor primi niciun punctaj pe tema.**

★ **Nu copiati cod de pe Internet! Se poate ajunge la situatia in care doi studenti sa aiba acelasi cod, preluat de pe Internet, caz in care ambele teme vor fi depunctate, desi studentii nu au colaborat direct intre ei.**

★ Pentru intrebari despre tema se va folosi in mod exclusiv **forumul** temei, pe care va recomandam sa il vizitati chiar si daca nu aveti intrebari, intrucat este posibil sa aflati informatii noi din intrebarile puse de colegii vostri, respectiv din raspunsurile date de noi.

★ **Temele trebuie sa fie incarcate pe vmchecker. NU se accepta teme trimise pe e-mail sau altfel decat prin intermediul vmchecker-ului.**

★ **Temele trimise dupa deadline nu vor fi luate in considerare.**

Notare

Total = 200p:

- **40p** - Problema 1
- **50p** - Problema 2
- **50p** - Problema 3
- **40p** - Problema 4
- **20p** - Coding style + README
    - **15p** - Coding style
        - Comentarii
        - Folosirea mai multor functii
        - Logica clara
        - Spatiere corecta
        - Stil consecvent
        - Variabile si functii denumite adecvat
    - **5p** - Completarea fisierului README