**BABEŞ-BOLYAI TUDOMÁNYEGYETEM**

**MATEMATIKA-INFORMATIKA KAR**

**MATEK-INFO VERSENY – 2017, április 1**

**INFORMATIKA**

**I. TÉTELSOR**

**A versenyzők figyelmébe:**

1. A feladatok megoldásait *pszeudokódban* vagy egy *programozási nyelvben* (*Pascal/C/C*++) kell megadnotok.
2. A megoldások értékelésekor az első szempont az algoritmus ***helyessége***, majd a ***hatékonysága***, ami a *végrehajtási időt* és a *felhasznált memória méretét* illeti.
3. A tulajdonképpeni megoldások előtt, ***kötelezően leírjátok szavakkal az alprogramokat, és megindokoljátok a megoldásotok lépéseit***. Feltétlenül írjatok ***megjegyzéseket*** (kommenteket), amelyek segítik az adott meg­oldás technikai részleteinek megértését. Adjátok meg az azonosítok jelentését és a fölhasznált adatszer­keze­teket. Ha ez hiányzik, a tételre kapható pontszámotok 10%-kal csökken.
4. Ne használjatok különleges fejállományokat, előredefiniált függvényeket (például STL, karakterláncokat feldolgozó sajátos függvények stb.).

**I. Tétel (35 pont)**

1. **Fénysugár (20 pont)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 6 | | 1 | | 9 | | 4 | |  |
| 7 |  |  |  |  |  |  |  |  | 3 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 0 |  | 8 | | 5 | | 2 | |  |  |

Legyen egy tükrökből kialakított, téglalap alakú keret. Egy fénysugár elindul a téglalap bal alsó sarkából, 45o fokos szöget alkotva a téglalap alsó oldalával, és nekiütközik a téglalap felső vagy jobboldali oldalának. Itt tükröződik (elindul egy másik oldal felé, szintén 45o fokos szöget alkotva azzal az oldal­lal, amelybe beleütközött). Így folytatja az útját, amíg a keret valamelyik sar­kába nem ér.

Írjatok alprogramot, amely kiszámítja, hogy hányszor (***váltSz***) változtatja a tükröződés irányát a fénysugár, amíg leáll valamelyik sarokban. A kiindulási pontot nem számítjuk be ebbe a számba. Az alprogram bemeneti paraméterei a téglalap hossza (1 < ***a*** < 10 000) és szélessége (1 < ***b*** < 10 000), míg a ***váltSz*** kimeneti paraméter.

**1. *Példa:*** ha ***a*** = 8 és ***b*** = 3, akkor ***váltSz*** = 9.

**2. *Példa:*** ha ***a*** = 8 és ***b*** = 4, akkor ***váltSz*** = 1.

1. **„Erős” számok (15 pont)**

Egy nullától különböző ***sz*** természetes számnak az *erőssége* ***k***, ha bináris alakjában pontosan ***k*** darab 1-es szám­jegy található. Például, a 23 erőssége 4 (kettes számrendszerben felírva, 4 darab 1-es számjegye van). Adott szám­sorozat ***k*** *erősségű csoport*jának nevezzük azt a részsorozatot, amely a sorozat ***k*** erősségű elemeit tartalmazza, az elemek eredeti sorrendjében. Például, az ***s*** = (7, 12, 3, 13, 24, 19), sorozat ***k*** = 2 *erősségű* *csoportja* a (12, 3, 24) részsorozat.

Írjatok alprogramot, amely meghatároz minden *erősségi csoportot*, amelyek az adott ***x*** sorozat elemeiből létrehoz­hatók. Bemeneti paraméterek: a nem nulla, 30 000-nél kisebb, különböző természetes számokból álló ***x*** sorozat és a sorozat ***n*** hossza (1 < ***n*** < 100). Kimeneti paraméterek: a ***csSzáma*** (a csoportok száma) és a ***csoportok*** (a létrehozott csoportok, erősségük szerint növekvően rendezve a csoporton belül az elemek sorrendje tetszőleges).

***Példa:*** ha ***n*** = 6 és ***x*** = (12, 3, 24, 16, 15, 32), akkor ***csSzáma*** = 3, és ***csoportok***: (16, 32), (12, 3, 24), (15).

**II. Tétel (15 pont)**

Adott a következő alprogram, amelynek három, nem nulla természetes szám bemeneti paramétere van: ***a****,* ***b*** és ***sz***, amelyeknek értékei kisebbek, mint 10 000:

**Algoritmus** f(a, b, sz):

k ← 0

**Amíg** b < sz **végezd el:**

k ← k + 1

b ← a + b

a ← b - a

**vége(amíg)**

**térítsd** k

**Vége(algoritmus)**

* 1. Adjátok meg a feladat szövegét, amelyet ez az algoritmus old meg, ha ***a*** = 1 és ***b*** = 0 értékekre hívjuk meg.
  2. Mit térít az **F(1,0,10)** hívás?
  3. Írjátok le egy *rekurzív* változatát a fenti algoritmusnak, megőrizve az iteratív (nem rekurzív) változat fejlécét.

**III. Tétel (40 pont)**

**Előszelet**

*Sors-számjegynek* hívjuk azt a természetes számot, amelyet adott természetes számra a következőképpen számí­tunk ki: összeadjuk a szám számjegyeit, majd a kapott összeg számjegyeit, és így tovább, amíg a kapott összeg nem válik egyszámjegyű számmá. Például, a 182 *sors-számjegye* 2 (1 + 8 + 2 = 11, 1 + 1 = 2).

Egy pontosan ***k*** számjegyű ***p*** számot egy legkevesebb ***k*** számjegyű ***q*** szám *előszeleté*nek nevezünk, ha a ***q*** szám első ***k*** számjegyéből alkotott szám (balról jobbra tekintve) egyenlő ***p***-vel. Például, 17 előszelete 174-nek, és 1713 előszelete 1 713 242-nek.

Legyen az ***sz*** természetes szám (0 < ***sz*** ≤ 10 000) és az ***m*** sorral és ***n*** oszloppal (0 < ***m*** ≤ 100, 0 < ***n*** ≤ 100) rendelkező ***A*** mátrix (kétdimenziós tömb), amelynek elemei 30 000-nél kisebb természetes számok. Írjatok programot, amely meghatározza és kiírja az ***sz*** szám *leghosszabb előszeletét*, amelyet az adott mátrix elemeinek megfelelő *sors-számjegyeiből* fel lehet építeni. Egy ilyen sors-számjegyet akárhányszor fel lehet használni. Ha nem építhető fel előszelet, a program írja ki a „*nem létezik előszelet*” üzenetet.

|  |  |
| --- | --- |
| ***Példa*:** ha ***sz*** = 12319, ***m*** *=* 3, ***n*** *=* 4 és a mátrix: | , |
| akkor a leghosszabb előszelet 1231, a megfelelő sors-számjegyek pedig: |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Mátrixelem értéke | 182 | 12 | 274 | 22 | 1 | 98 | 56 | 5 | 301 | 51 | 94 |
| Sors-számjegy | 2 | 3 | 4 | 4 | 1 | 8 | 2 | 5 | 4 | 6 | 4 |

A megoldásban fölhasználjátok a következő alprogramokat:

* 1. a bemeneti adatok beolvasása billentyűzetről
  2. adott számhoz rendelhető sors-számjegy meghatározása
  3. a leghosszabb előszelet meghatározása
  4. a leghosszabb előszelet kiírása a képernyőre ha nincs előszelet, a megfelelő üzenet kiírása.

**Megjegyzések:**

1. Minden tétel kidolgozása kötelező.
2. A megoldásokat a vizsgalapokra írjátok, (a piszkozatokat nem vesszük figyelembe).
3. Hivatalból jár 10 pont.
4. Rendelkezésetekre áll 3 óra.

**MEGOLDÁSOK**

**I. 1. Fénysugár**

**1. Változat:** A fénysugár irányváltoztatásainak száma az ***a*** és ***b***, illetve *Lnko*(***a****,* ***b***) értékektől függ.

|  |  |
| --- | --- |
| **Algoritmus** lnko(a, b): { *ismételt kivonásokkal* }  **Ha** a = b **és** a = 0 akkor  **térítsd** 1  **vége(ha)**  **Ha** a \* b = 0 **akkor** { *legalább egyik* 0 }  **térítsd** a + b  **vége(ha)**  **Amíg** a ≠ b **végezd el:**  **Ha** a > b **akkor**  a ← a - b  **különben**  b ← b - a  **vége(ha)**  **vége(amíg)**  **térítsd** a  **Vége(algoritmus)** | **Algoritmus** lnko(a, b): { *ismételt osztásokkal* }  mar ← a **MOD** b  **Amíg** mar ≠ 0 **végezd el:**  a ← b  b ← mar  mar ← a **MOD** b  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** |
| **Algoritmus** sugár(a, b):  { *hányszor változtatja az irányt a fénysugár* }  d ← lnko(a, b)  **Ha** d = a **akkor** { *ha az a oldal osztója b-nek* }  váltSz ← b **DIV** d - 1  **különben**  **Ha** d = b **akkor** { *ha a b oldal osztja a-t* }  váltSz ← a **DIV** d - 1  **különben**  váltSz ← b **DIV** d + a **DIV** d – 2  **vége(ha)**  **vége(ha)**  **térítsd** váltSz  **Vége(algoritmus)** |  |

**2. Változat:** Ha megpróbáljuk szimulálni a fénysugár mozgását, elhelyezzük egy olyan koordináta rendszerbe, amelynek az origója a keret bal-alsó sarka, az *Ox* tengely az alsó oldal és az *Oy* tengely pedig a bal oldal.

**Algoritmus** szim(a, b): { *a tükröződések szimulálása* }

váltSz ← 0

posX ← 0

posY ← 0

irány ← jobbraFel { *felsorolás típus:* {jobbraFel, jobbraLe, balraLe, balraFel} }

vége ← *hamis* { *még nem érkezett meg egy sarokba* }

**Amíg** **nem** vége **végezd el:**

**Ha** irány = jobbraFel **akkor**

**Ha**a - posY **=** b – posX **akkor** { *sarokba ért* }

vége ← *igaz*

**különben**

**Ha**a - posY < b – posX **akkor** { *ütközik a felső oldalba* }

posX ← posX + a - posY

posY ← a

irány ← jobbraLe

váltSz ← váltSz + 1

**különben** { *ütközik a jobb oldalba* }

posY ← posY + b - posX

posX ← b

irány ← balraFel

váltSz ← váltSz + 1

**vége(ha)**

**vége(ha)**

**különben**

**Ha** irány = jobbraLe **akkor**

**Ha**posY **=** b – posX **akkor** { *sarokba ért* }

vége ← *igaz*

**különben**

**Ha**posY < b – posX **akkor** { *ütközik az alsó oldalba* }

posX ← posX + posY

posY ← 0

irány ← jobbraFel

váltSz ← váltSz + 1

**különben** { *ütközik a jobb oldalba* }

posY ← posY – (b – posX)

posX ← b

irány ← balraLe

váltSz ← váltSz + 1

**vége(ha)**

**vége(ha)**

**különben**

**Ha** irány = balraLe **akkor**

**Ha**posY **=** posX **akkor** { *sarokba ért* }

vége = *igaz*

**különben**

**Ha**posY < posX **akkor** { *ütközik az alsó oldalba* }

posX ← posX - posY

posY ← 0

irány ← balraFel

váltSz ← váltSz + 1

**különben** { *ütközik a bal oldalba* }

posY ← posY - posX

posX ← 0

irány ← jobbraLe

váltSz ← váltSz + 1

**vége(ha)**

**vége(ha)**

**különben**

**Ha** irány = balraFel **akkor**

**Ha**a - posY **=** posX **akkor** { *sarokba ért* }

vége ← *igaz*

**különben**

**Ha**a - posY < posX **akkor** { *ütközik a felső oldalba* }

posX ← posX – (a – posY)

posY ← a

irány ← balraLe

váltSz ← váltSz + 1

**különben** { *ütközik a bal oldalba* }

posY ← posY + posX

posX ← 0

irány ← jobbraFel

váltSz ← váltSz + 1

**vége(ha)**

**vége(ha)**

**vége(ha)**{ irány = balraFel

**vége(ha)**{irány = balraLe

**vége(ha)**{ irány = jobbraLe

**vége(ha)**{ irány = jobbraFel

**vége(amíg)**

**térítsd** váltSz

**Vége(algoritmus)**

**I. 2. „Erős” számok**

* 1. **A szám erősségének meghatározása (az 1-es számjegy darabszáma az ábrázolásban)**

**1. Változat:** a & (*és*) bitenkénti operátor alkalmazása:nr = nr & (nr – 1) (lépések száma = ahány 1-es számjegye van az ábrázolásnak).

**2. Változat:** a & (*és*) és a >> (*jobbra tolás*) bitenkénti operátorok alkalmazása: (lépések száma = ahány számjegye van összesen az ábrázolásnak).

**3. Változat:** osztást alkalmazunk: (lépések száma = ahány számjegye van összesen az ábrázolásnak).

|  |  |  |
| --- | --- | --- |
| **Algoritmus** erősség\_1(sz):  erő ← 0  **Ismételd**  sz ← sz & (sz – 1)  erő ← erő + 1  **ameddig** sz = 0 { *kilépünk, ha sz* = 0 }  **térítsd** erő  **Vége(algoritmus)** | **Algoritmus** erősség\_2(sz):  erő ← 0  **Amíg** sz > 0 **végezd el:**  **Ha** (sz & 1) ≠ 0 **akkor**  { *ha az utolsó számjegy* 1*-es* }  erő ← erő + 1  **vége(ha)**  sz ← sz >> 1  { *osztás* 2*-vel* (*jobbra tolással*) }  **vége(amíg)**  **térítsd** erő  **Vége(algoritmus)** | **Algoritmus** erősség\_3(sz){  erő ← 0  **Amíg** sz > 0 **végezd el:**  **Ha** sz **MOD** 2 = 1 **akkor**  erő ← erő + 1  sz ← sz **DIV** 2  **vége(ha)**  **vége(amíg)**  **térítsd** erő  **Vége(algoritmus)** |

* 1. **Az azonos erősségű számok csoportosítása**

Mivel a maximális erősség = 32, a csoportokat egy kétdimenziós tömbben tároljuk (32 sor és ***n*** oszlop, mivel előfordulhat, hogy minden szám egy bizonyos csoportba kerül). Egy-egy csoport hosszát a 0 oszlopindexű elemben tároljuk, a csoport elemei a mátrix sorában találhatók.

**Algoritmus** erők(n, x, csoportok):

csSzáma ← 0

**Minden** i = 1, maxCif – 1 **végezd el:**

csoportok[i][0] ← 0 { *a csoportok hosszúságainak inicializálása* }

**vége(minden)**

**Minden** i = 1, n **végezd el:**

erő ← erősség\_1(x[i]) { *az x*[*i*] *elem erőssége* }

**Ha** csoportok[erő][0] = 0 **akkor** { *x*[*i*]*-vel új csoport kezdődik* }

csSzáma ← csSzáma + 1 { *nő a csoportok száma* }

**vége(ha)**

csoportok[erő][0] ← csoportok[erő][0] + 1 { *nő az "erő" indexű sor hossza* }

csoportok[erő][csoportok[erő][0]] ← x[i] { *beszúrjuk x*[*i*]*-t az erősségének megfelelő sorba* }

**vége(minden)**

**térítsd** csSzáma

**Vége(algoritmus)**

**II.**

**Algoritmus** f(a, b, sz):

k ← 0

**Amíg** b < sz **végezd el:**

k ← k + 1

b ← a + b

a ← b - a

**vége(amíg)**

**térítsd** k

**Vége(algoritmus)**

1. Adjátok meg a feladat szövegét, amelyet ez az algoritmus old meg, ha ***a*** = 1 és ***b*** = 0 értékekre hívjuk meg.

Válasz: *Határozzuk meg az* ***sz*** *számnál szigorúan kisebb Fibonacci számok darabszámát.*

1. Mit térít az F(1, 0, 10) hívás? 7-et
2. Írjátok le egy *rekurzív* változatát a fenti algoritmusnak, megőrizve az iteratív (nem rekurzív) változat fejlécét.

**Algoritmus** fRekurziv(a, b, n):

**Ha** b < n **akkor**

k ← fRekurziv(b, a + b, n) + 1

**különben**

k ← 0

**vége(ha)**

**térítsd** k

**Vége(algoritmus)**

**III. Előszelet**

**Algoritmus** sorsSzámjegy(x): { *az x szám sorsszámjegye* }

**Amíg** x > 9 **végezd el:**

y ← x

s ← 0

**Amíg** y > 0)**végezd el:**

s ← s + y **MOD** 10

y ← y **DIV** 10

**vége(amíg)**

x ← s

**vége(amíg)**

**térítsd** x

**Vége(algoritmus)**

**Algoritmus** maxElőszelet(sz, m, n, A, szJegyek, szJegyekSzáma): { *a legnagyobb előszelet meghatározása* }

{ *a mátrixot egyszer járjuk be és meghatározzuk a mátrix elemeinek megfelelő sorsszámjegyekről, hogy előfordultak-e* }

{ *a vizsgált szám számjegyeit megőrizzük egy vektorban bejárjuk ezt a vektort (a nagyobb helyértéktől* }

{ *és csökkenő sorrendben) vizsgáljuk, hogy megjelent-e, mint sorsszámjegy* }

**Minden** i = 0, 9 **végezd el**: { *az előfordulások tömbjének inicializálása* }

előfordulások[i] ← 0

**vége(minden)**

**Minden** i = 1, m **végezd el:**

**Minden** j = 1, n **végezd el:**

előfordulások[sorsSzámjegy(A[i][j])] ← 1 { *előfordul az* A[i][j] *elem sorsszámjegye* }

**vége(minden)**

**vége(minden)**

szJegyekSzáma ← 0 { *az* sz *szám számjegyeinek száma* }

**Amíg** sz > 0 **végezd el:**

szJegyekSzáma ← szJegyekSzáma + 1

szJegyek[szJegyekSzáma] ← sz % 10 { *az* sz *szám számjegyei* }

sz ← sz **DIV** 10

**vége(amíg)**

i ← szJegyekSzáma { *bejárjuk a számjegyek tömbjét* }

**Amíg** i > 0 **és** előfordulások[szJegyek[i]] ≠ 0 **végezd el**: { *a sorsszámjegy* = *az* sz *szám aktuális számjegyével* }

i ← i – 1

**vége(amíg)**

**térítsd** szJegyekSzáma - i

**Vége(algoritmus)**