1. feladat: Mikorra érhet oda (50 pont)

Az alábbi ábrán egy sétapálya látható, amelyről tudjuk, hogy egy ember melyik szakaszt mennyi idő alatt tudja bejárni. A pályán a nyilaknak megfelelő irányba lehet haladni.



A. Add meg, hogy merre kell menni A-ból G-be, hogy G-be a lehető leghamarabb érjünk!

B..G. Add meg, hogy az egyes pontokra (B..G) leghamarabb, illetve legkésőbb mennyi idő alatt lehet odaérni az A pontból indulva!

2. feladat: Társasjáték (60 pont)[[1]](#footnote-1)

Mint minden társasjátékban, itt is az egyes mezőkre csak bizonyos szabály szerint lehet ráugrani.

Ennél a játéknál minden mezőhöz tartozik egy szabály. Háromféle szabály van.

* nB: n mezőt ugrunk balra, 2B azt jelenti tehát, hogy 2 mezőt ugrunk balra:
* nJ: n mezőt ugrunk jobbra, 3J így azt jelenti, hogy 3 mezőt ugrunk jobbra:
* 0: nem lehet tovább ugrani.

A. Hányadik mezőn kell kezdeni ahhoz, hogy a játék végén minden mezőn egyszer járjunk? Rajzold az ábrára nyilakkal a bejárt mezők sorrendjét is!



B. Mely mezőkre nem lépünk, ha a legelső mezőn kezdjük a játékot?

C. Mely mezőkre nem lépünk, ha a legutolsó mezőn kezdjük a játékot?

D. Mely mezőkre nem lehet lépni sehonnan, ha az ötödik mezőn levő 3J helyére 1J-t írnánk?

E. Honnan lehetne eljutni a negyedik mezőre, ha az ötödik mezőn levő 3J helyére 2B-t írnánk és mi történne, ha más mezőkről indulnánk?

F. Melyik mezőn kell kezdeni az alábbi ábrán ahhoz, hogy a játék végén minden mezőn egyszer járjunk? Rajzold az alábbi ábrára a bejárt mezők sorrendjét is!



 **0 1J 2B 3J 1J 2J 5B 4B**

G. Melyik mezőn kell kezdeni az alábbi ábrán ahhoz, hogy a játék véget érjen a legtöbb mező érintésével, de ne kerüljünk végtelen ciklusba? Mely mezőkről kerülünk végtelen ciklusba? Rajzold az alábbi ábrára a bejárt mezők sorrendjét is, a végtelen ciklust is!



 **2J 0 5J 3B 3B 2B 2B 2B**

3. feladat: Mit csinál (70 pont)

Az alábbi algoritmus egy N elemű, 1-től N-ig indexelt X vektort dolgoz fel, amely 1 és M közötti egész számok zárt intervallumainak kezdetét és végét tartalmazza (X[i].k≤X[i].v), eredményét az U, V változókba írja. A D vektort 1-től M+1-ig indexeljük.

Legtöbb(V,U):
 D:=(0,…,0)
 Ciklus i=1-től N-ig
 D[X[i].k]:=D[X[i].k]+1
 D[X[i].v+1]:=D[X[i].v+1]-1
 Ciklus vége
 U:=1; A:=D[1]; V:=A
 Ciklus i=2-től M-ig
 A:=A+D[i]
 Ha A>V akkor U:=i; V:=A
 Ciklus vége
Eljárás vége.

A. Mi lesz U és V értéke, ha N=3, M=9, X=((4,6),(1,2),(9,9))?

B. Mi lesz U és V értéke, ha N=3, M=9, X=((4,6),(1,2),(5,9))?

C. Mi lesz U és V értéke, ha N=5, M=8, X=((4,8),(1,2),(6,7),(1,3),(7,7))?

D. Mi lesz a D vektor értéke, ha N=5, M=8, X=((4,8),(1,2),(6,7),(1,3),(7,7))?

E. Fogalmazd meg, mi a D vektor szerepe!

F. Fogalmazd meg, hogy U és V értéke hogyan függ az X vektortól!

4. feladat: Automata (55 pont) [[2]](#footnote-2)

Egy automata kezdetben A állapotban van, jeleket olvas és a jelek hatására az állapota megváltozhat. Ha A állapotban a bemenetére 0 jel érkezik, akkor marad A állapotban, ha 1 érkezik, akkor átkerül B állapotba. Ha B állapotban a bemenetére 1 érkezik, akkor marad B állapotban, ha pedig 0, akkor átkerül C állapotba. C állapotból 0 és 1 hatására is átkerül B állapotba

Az automata az alábbi rajzzal ábrázolható:

![[initial text=, ->, shorten >=1pt, >=latex, auto, bend angle=25,node distance=2cm]\tikzstyle{state}=[inner sep=1.5mm,minimum size=5mm, ,shape=circle,draw] \node[state, initial] (A) {$A$}; \node[state, accepting] (B) [right of=A] {$B$}; \node[state] (C) [right of=B] {$C$}; \path (A) edge [loop above] node {\mbox{\tt0}} (A) (A) edge node {\mbox{\tt1}} (B) (B) edge [loop above] node {\mbox{\tt1}} (B) (B) edge [out=20,in=160] node {\mbox{\tt0}} (C) (C) edge [in=340,out=200] node {\mbox{\tt0},\mbox{\tt1}} (B);]()

A. Milyen állapotban lesz az automata az 11010 jelsorozat hatására? Add meg az egyes jelek utáni állapotot is!

B. Milyen állapotban lesz az automata a 0101010011 jelsorozat hatására? Add meg az egyes jelek utáni állapotot is!

C. Milyen jelsorozatokra lesz az automata a legvégén A, B, illetve C állapotban?

5. feladat: Gépközeli programozás (50 pont)

Az alábbiakban látható egy gépközeli nyelven megírt programrészlet, amely az A és B címen levő 32 bites pozitív egész számokból számol ki egy 32 bites pozitív egész számot, amit a C címre tesz. A segédszámításokhoz a processzor három, 32-bites regiszterét (a processzor tárolóegységét) használja (eax, ebx, ecx).

Egy kettes számrendszerbeli szám balra léptetése azt jelenti, hogy a legmagasabb helyiértékű bitjét törli és a végére egy 0-s bitet tesz. A jobbra léptetés a legalacsonyabb helyiértékű bitet törli és az elejére tesz egy 0-s bitet.

 mov eax,A ; az eax regiszterbe tölt egy értéket az A címről
 mov ebx,B ; az ebx regiszterbe tölt egy értéket a B címről
 mov ecx,0 ; az ecx regiszterbe nullát tölt
xxx: cmp ebx,0 ; összehasonlítja ebx tartalmát a nullával
 je yyy ; az yyy címen folytatja, ha egyenlők voltak
 bt ebx,0 ; megnézi ebx legalacsonyabb helyiértékű bitjét
 jc zzz ; a zzz címen folytatja, ha 1-es értékű volt
 shl eax ; az eax regisztert 1 bittel balra lépteti
 shr ebx ; az ebx regisztert 1 bittel jobbra lépteti
 jmp xxx ; az xxx címen folytatja
zzz: add ecx,eax ; hozzáadja ecx-hez eax-et
 dec ebx ; eggyel csökkenti ebx-et
 jmp xxx ; az xxx címen folytatja
yyy: mov C,ecx ; a C címre teszi az ecx regiszter tartalmát

A. Mi lesz a C címen, ha A=42, B=1?

B. Mi lesz a C címen, ha A=3, B=8?

C. Mi lesz a C címen, ha A=3, B=10?

D. Hogyan függ a kiszámított C értéke A-tól és B-től?

E. Milyen esetben lesz az eredmény értelmezhetetlen (hibás)?

F. Hányszor lép a program a zzz címre és ez mitől függ?

G. Hányszor lép a program az xxx címre és ez mitől függ?

6. feladat: Hasító táblák (50 pont)

A hasító tábla egy olyan adatszerkezet, amely segítségével hatékonyan kereshetünk pozitív egész számokat egy adathalmazban. A kereséshez az egész számot előbb mindig egy K-nál kisebb egész számmá képezzük. A tárolásra egy egyszerű, 0-tól K-1-ig indexelt tömböt használunk. Beszúráskor egy N értékű adatot a tömb (N mod K). pozíciójába szeretnénk tárolni, azonban, ha az már foglalt, akkor sorra a következő pozíciókkal próbálkozunk, ha a tömb végére érünk, akkor az elején folytatjuk a szabad hely keresését.

Például egy 7 elemű hasító táblába szeretnénk a 44-öt beszúrni. Ez a 44 mod 7=2. pozícióra kerül.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 44 |  |  |  |  |

Ezután a 130-at szúrjuk be, ami a 130 mod 7=4 pozícióra kerül.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 44 |  | 130 |  |  |

Ezután a 2-es beszúrásával folytatjuk, ami a 2 mod 7=2 pozícióra kerülne, de ez foglalt, ezért próbálkozunk a következő pozícióval, ami szabad.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 44 | 2 | 130 |  |  |

A keresés a beszúráshoz hasonlóan működik. A kiszámított helyen keressük az adatot. Például, ha a 2-t keressük, akkor a 2 mod 7=2 pozícióban keressük először, ott másik elem van, ezért a következő pozíción folytatjuk, ahol meg is találjuk a keresett értéket, ehhez 2 tömbelem olvasására volt szükség. Ha például a 18-at keressük, azt a 4-es pozícióban kell keresnünk, ahol nem ez az elem szerepel, ezért a következő helyen próbálkozunk, az viszont üres, így 2 tömbelem olvasásával megtudtuk, hogy a keresett kulcs nem szerepel az adatok között.

Egymás után végezd el a következő műveleteket egy kezdetben üres, **13 elemű** tömböt használó hasítótáblán! A beszúrásoknál add meg, hogy melyik pozícióba kerül az adott elem, a kereséseknél pedig, hogy hány tömbelemet kellett olvasni, ahhoz hogy megtaláljuk azt, vagy megbizonyosodjunk a keresett elem hiányáról!

A. Beszúr: 134

B. Beszúr: 39

C. Beszúr: 6

D. Beszúr: 265

E. Beszúr: 12

F. Beszúr: 388

G. Beszúr: 5

H. Keres: 265

I. Keres: 22

J. Keres: 5

K. Keres: 135

L. Keres: 11

7. feladat: Szenzorhálózat (65 pont)

Repülőgépről ledobálva telepítünk szenzorokat egy területen. A szenzoroknak szükségük van egymás közötti kommunikációra, azonban a jeladók teljesítménye miatt két szenzor csak akkor képes egymással kommunikálni, ha a távolságuk **legfeljebb 5 egység**. Távolabbi szenzorpárok ezért csak úgy tudnak egymásnak üzenni, ha más szenzorokon keresztül teszik ezt meg.

Két pont távolsága a koordinátáik különbségei abszolút értékének összege (Manhattan távolság). Például az (5,3) és a (7,2) pontok távolsága abs(5-7)+abs(3-2)=2+1=3.

Kétféle mohó stratégiával választhatják a szenzorok a csomagok továbbításakor a következő szenzort:

* Távolság: Az elérhető szenzorok közül – akár önmagát választva – annak továbbítjuk a csomagot, amely a célponthoz legközelebb van.
* Irány: Az elérhető szenzorok közül – önmagát kivéve – annak továbbítjuk a csomagot, amely leginkább a cél irányában van, azaz amelynek az aktuális pontból nézve a célhoz képest vett szögelfordulása minimális.

Mindkét stratégiára igaz, hogy ha több megfelelő szenzor lenne, akkor bármelyik választható. A megoldásban elég egyet megadni.

A képen látható pozíciókba sikerült a repülőről telepítenünk a szenzorokat.



A. Add meg a csomag által bejárt útvonalat a K szenzorból az I szenzorba küldéskor **távolság** alapú útválasztás esetén!

B. Add meg a csomag által bejárt útvonalat az U szenzorból a W szenzorba küldéskor **távolság** alapú útválasztás esetén!

C. Add meg a csomag által bejárt útvonalat a P szenzorból a V szenzorba küldéskor **távolság** alapú útválasztás esetén! Mi történik?

D. Add meg a csomag által bejárt útvonalat a W szenzorból a M szenzorba küldéskor **irány** alapú útválasztás esetén!

E. Add meg a csomag által bejárt útvonalat a P szenzorból a D szenzorba küldéskor **irány** alapú útválasztás esetén!

F. Add meg a csomag által bejárt útvonalat a B szenzorból a R szenzorba küldéskor **irány** alapú útválasztás esetén! Mi történik?

|  |
| --- |
| *A Neumann János Számítógép-tudományi Társaság az Eötvös Loránd Tudományegyetem Informatikai Kara közreműködésével a 2019/2020-as tanévre meghirdeti a Nemes Tihamér Online Programozási Versenyt (*[*http://tehetseg.inf.elte.hu/nemes-online/nemes\_aktualis.html*](http://tehetseg.inf.elte.hu/nemes-online/nemes_aktualis.html)*). A versenyt öt fordulóban rendezzük meg. Az egyes fordulókban a versenyzőknek 3-5 programozási feladatot kell megoldaniuk C++, Pascal, C#, Python, Visual BASIC vagy Java nyelven, a beküldött megoldásokat Linux környezetben fordítjuk újra és az így kapott programokat értékeljük. Azoknak is lehetőséget nyújt a versenyzésre, akik nemrég kezdték a programozás tanulást. A versenyre nevezni a verseny honlapján lehet, a szükséges adatok (név, település, iskola, évfolyam, email) megadásával a https://biro.inf.elte.hu/faces/versenyre.xhtml oldalon.* |

1. http://e-hod.elte.hu/archiv/feladatok/HOD\_osszes\_2018\_senior.pdf [↑](#footnote-ref-1)
2. A feladat Friedl Katalin, Csima Judit: Nyelvek és automaták című könyvéből származik: https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0064\_58\_nyelvek\_es\_automatak/ar01s02.html [↑](#footnote-ref-2)