Sor, Sorozat adattípusok

Horváth Gyula
horvath@inf.elte.hu

3. A Sor adattípus

3.1. Sor

Értékhalmaz: $Sor = \{\langle a_1, \ldots, a_n \rangle : a_i \in E, i = 1, \ldots, n, n \geq 0\}$
Műveletek argumentumai: $S : Sor, x : E$

Előfeltétel	Művelet	Utófeltétel
$S = \langle a_1, \ldots, a_n \rangle$	$SorBa(S,x)$	$S = \langle a_1, \ldots, a_n, x \rangle$
$S = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	SorBól(S,x)	$x = a_1 \wedge S = \langle a_2, \ldots, a_n \rangle$
$S = \langle a_1, \ldots, a_n \rangle$	Elemszám(S)	$= n \wedge S = Pre(S)$
$S = S$	ÜresE(S)	$= (Pre(S) = \langle\rangle) \wedge S = Pre(S)$
$S = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	Első(S)	$= a_1 \wedge S = Pre(S)$
$S = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	Töröl(S)	$S = \langle a_2, \ldots, a_n \rangle$
$S = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	Utolsó(S)	$= a_n \wedge S = Pre(S)$

3.2. A Sor C++ és C# műveletei

Művelet	C++	C#
Típusdef	$queue < E >$	$Queue < E >$
$SorBa(S,x)$	$S.push(x)$	$S.Enqueue(x)$
$SorBól(S,x)$	$x = S.front(); S.pop()$	$x := S.Dequeue()$
Elemszám	$S.size()$	$S.Count$
$ÜresE(S)$	$S.empty()$	$S.Count == 0$
$Első(S)$	$S.front()$	$S.Peek()$
$Töröl(S)$	$S.pop()$	$S.Dequeue()$
$Utolsó(S)$	$S.back()$	*nincs*

3.3. Feladat: Várólista

Járóbeteg ellátásra egy napon n beteg jelent meg. Minden beteg érkezésekor feljegyezték, hogy mikor érkezett, és azt is, hogy hány percig tartott a kezelése. A betegeket érkezési sorrendben veszik kezelésre. A kezelést folyamatosan végezte az orvos, tehát ha a p perckor kezdte meg egy beteg kezelését ami k percig tartott, akkor a következő beteg kezelését a $p+k$ perctől végezhette. Ki kell számítani, hogy legtöbben hányan várakoztak aznap egy időben kezelésre várva!

Bemenet

A standard bemenet első sora egy egész számot tartalmaz, a betegek n ($1 \leq n \leq 1000$) számát. A következő n sor mindegyike két egész számot tartalmaz, egy beteg e érkezési és k kezelési idejét ($0 \leq e \leq 720$, $1 \leq k \leq 40$). A bemenetben az adatok érkezési sorrendben vannak. Adott időpontban több beteg is érkezhetett.

Kimenet

A standard kimenet a nap során a legtöbb egy időben várakozó beteg számát kell írni! A várakozokba nem számít be az, akinek éppen a kezelését végzik.

Példa bemenet és kimenet

bemenet	kimenet
5 1 3 2 4 3 2 3 5 4 2	3

Korlátok

Időlimit: 0.2

Memória limit: 32 MiB

3.4. Megoldás

Jelölje $B[i] = (erk_i, kezeli_i)$ az i-edik beteg adata. Legen v_i az az időpont, amikor az i-edik beteg kezelése befejeződik, tehát ekkor kezdődhet a következő beteg kezelése. v_i a következőképpen számítható.

$$
\begin{aligned}
v_1 &= B[1].erk + B[1].kezeli \\
v_i &= \left\{ \begin{array}{ll} B[i].erk + B[i].kezeli, & \text{ha } v_{i-1} < B[i].erk \\ v_{i-1} + B[i].kezeli, & \text{ha } B[i].erk \leq v_{i-1} \end{array} \right\}
\end{aligned}
$$

Szimuláljuk a történéseket. Vegyük sorra, hogy adott v_i időben mi történik. Ehhez a következő adatokat számítjuk:

$$
\begin{aligned}
i &= \text{a következő érkező beteg sorszáma} \\
v &= \text{ekkor fejeződik be az aktuális beteg kezelése} \\
S &= \text{a várakozók sora (érkezési sorrendben)} \\
varok &= \text{az eddigi várakozók maximális száma}
\end{aligned}
$$

Szimulációs tevékenység a v időben:

- ha van várakozó (S nem üres), akkor vegyük S-ből az elsőt, egyébként a következő érkezőt
- aktualizáljuk a v értékét
- tegyük a várakozók S sorába a v időig beérkezőket
- aktualizáljuk a $varok$ értékét

3.4.1. C++ megvalósítás

```
1 #include <iostream>
2 #include <queue>
3 #define maxN 10000
4 using namespace std;
5 struct Beteg{
6     int erk,kezeli;
7 };
8 Beteg B[maxN];
```

```
 9  int main(){
10     queue<Beteg> S;
11     int n,v=0,varok=0,i=0;
12     cin>>n;
13     for(int i=0;i<n;i++){          //adatok be
14        cin>>B[i].erk>>B[i].kezeli;
15     }
16     Beteg ku;                      //az aktuális beteg
17     while(i<n || !S.empty()){      //amíg van beteg
18        if(S.size()>varok)
19           varok=S.size();
20        if(!S.empty()){             //van várakozó
21           ku=S.front(); S.pop();   //vegyük ez első várakozót
22        }else                       //nincs várakozó
23           ku=B[i++];               //vegyük a vövetkező érkezőt
24        if(v<ku.erk)
25           v=ku.erk+ku.kezeli;      //v-ig tart az aktuális kezelése
26        else
27           v+=ku.kezeli;            //eddig tart az aktuális kezelése
28        while(i<n && B[i].erk<v){   //v-ig beérkezők mennek a sorba
29           S.push(B[i]); i++;
30        }
31     }
32     cout<<varok<<endl;
33  return 0;}
```

3.4.2. C# megvalósítás

```
1  using System;
2  using System.Collections.Generic;
3
4  class Program{
5     public class Beteg{
6        public int erk; public int kezeli;
7        public Beteg(int x, int y){erk=x; kezeli=y;}
8     }
9     public static void Main(){
10       int n;
11       n=Convert.ToInt32(Console.ReadLine());
12       Queue<Beteg> S = new Queue<Beteg>();
13       Beteg[] B=new Beteg[n+1];
14       string[] sor;
15       int v=0,varok=0,i=0;
```

```
16          for(int ii=0;ii<n;ii++){          //adatok be
17              sor=Console.ReadLine().Split();
18              B[ii]=new Beteg(Convert.ToInt32(sor[0]), Convert.ToInt32(sor[1]));
19          }
20          Beteg ku;                         //az aktuális beteg
21          while(i<n || S.Count!=0){         //amíg van beteg
22              if(S.Count >varok)
23                  varok=S.Count;
24              if(S.Count!=0){               //van várakozó
25                  ku=S.Dequeue();           //vegyük ez első várakozót
26              }else                         //nincs várakozó
27                  ku=B[i++];                //vegyük a vövetkező érkezőt
28              if(v<ku.erk)
29                  v=ku.erk+ku.kezeli;       //v-ig tart az aktuális kezelése
30              else
31                  v+=ku.kezeli;             //eddig tart az aktuális kezelése
32              while(i<n && B[i].erk<v){     //v-ig beérkezők mennek a sorba
33                  S.Enqueue(B[i]); i++;
34              }
35          }
36          Console.WriteLine(varok);
37      }
38  }
```

3.5. Feladat: Elárasztás

Nagy vízhozamú forrás tört fel, amely elárasztja a környező területet. Azt szeretnénk megtudni, hogy hány nap alatt árasztja el a forrás a környékét. A probléma megoldáshoz a terület négyzetrácsos modelljét használhatjuk, ismerjük a forrás helyét és a forrást körülvevő, a forrás helyénél magasabban fekvő részeket. A víz egy nap alatt a négy (balra, jobbra, felfelé és lefelé lévő) szomszédos cella területét árasztja el, ha azok nem magasabbak a forrás helyénél.

Bemenet

A standard bemenet első sora négy egész számot tartalmaz, a négyzetrács sorainak $m(0 < m <= 1000)$ és oszlopainak $n(0 < n <= 1000)$ és a forrás fs és fo koordinátáit ($1 \leq fs \leq m\, 1 \leq fo \leq n$). A négyzetrács bal felső cellájának koordinátái $(1, 1)$. A következő m sor írja le a területet, azon mezők, amelyek magasabbak a forrás helyénél 1, a többi 0 értékkel jelölt.

Kimenet

A standard kimenet első sorába azt a legkisebb k számot kell írni, ahány nap alatt a forrás elárasztja a területet!

Példa bemenet és kimenet

bemenet

```
8 10 4 3
1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 1 0 0 1
1 0 0 1 0 0 1 0 0 1
1 0 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 1 0 1
1 0 0 0 0 1 0 0 0 1
1 0 1 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1
```

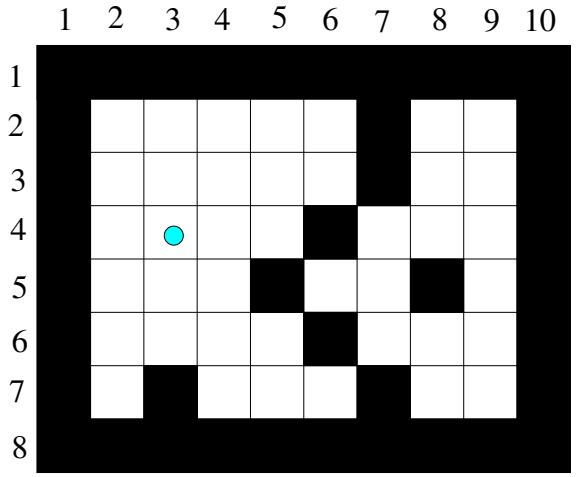
kimenet

```
7
```

Korlátok

Időlimit: 0.2

Memória limit: 32 MiB



1. ábra. A terület négyzetrácsos leírása

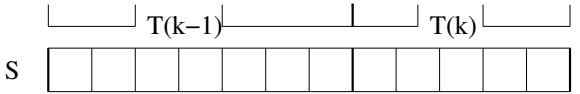
3.5.1. Megoldás

Tekintsük a celláknak a következő halmazait.

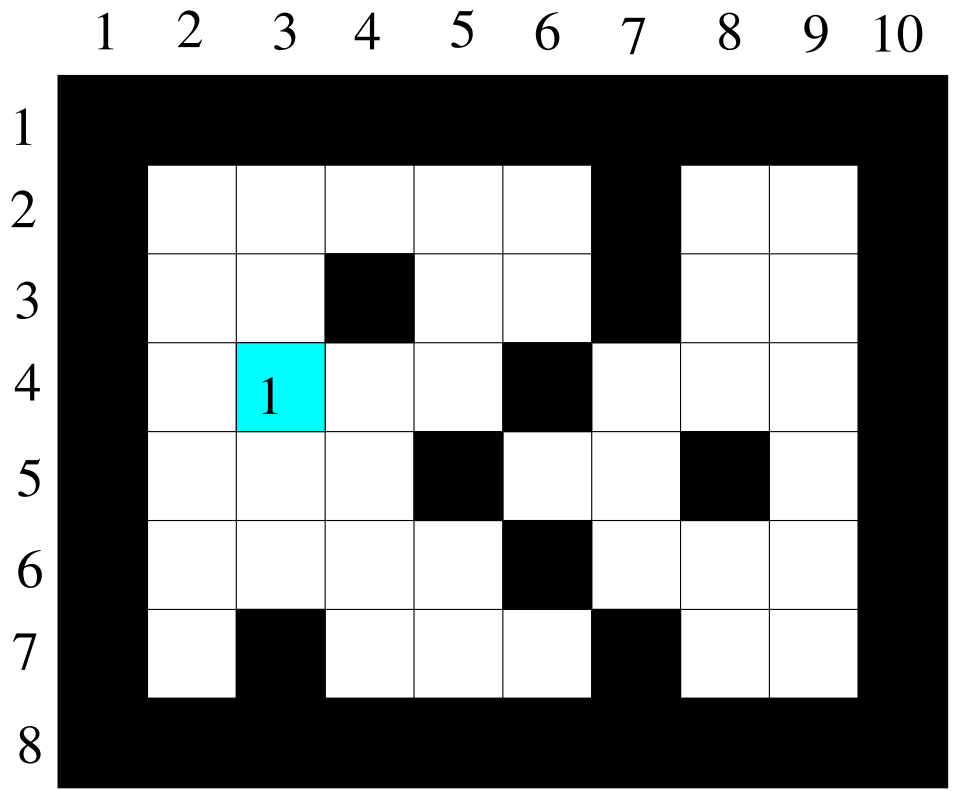
$$T(1) = \{(fs, fo)\}$$

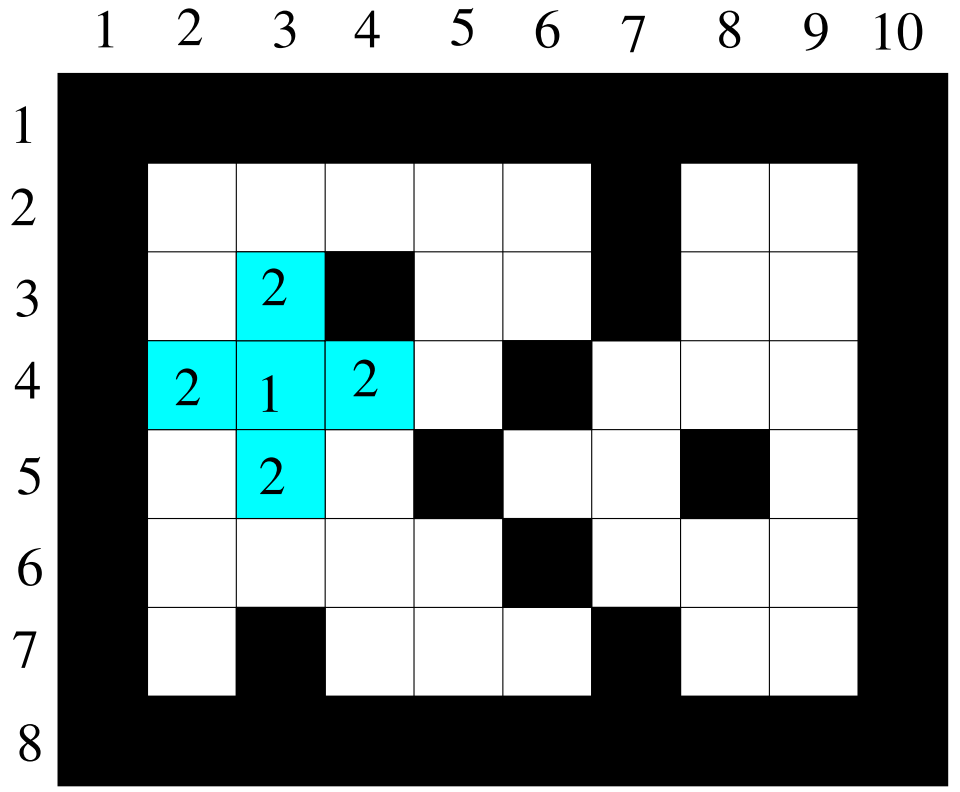
$T(k) = \{c : c \notin (T(1) \bigcup \ldots \bigcup T(k-1)) \wedge$ van olyan $cc \in T(k-1)$ cella, amely szonszéda c-nek$\}$

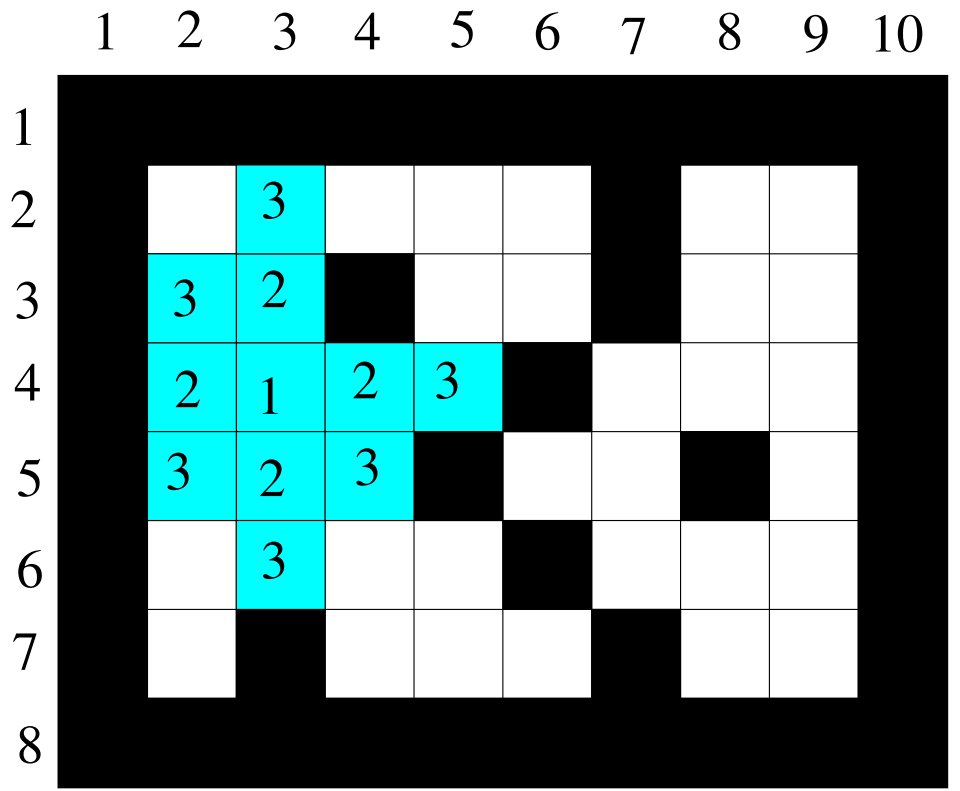
A feladat megoldása az a legnagyobb k, amelye $T(k) \neq \emptyset$

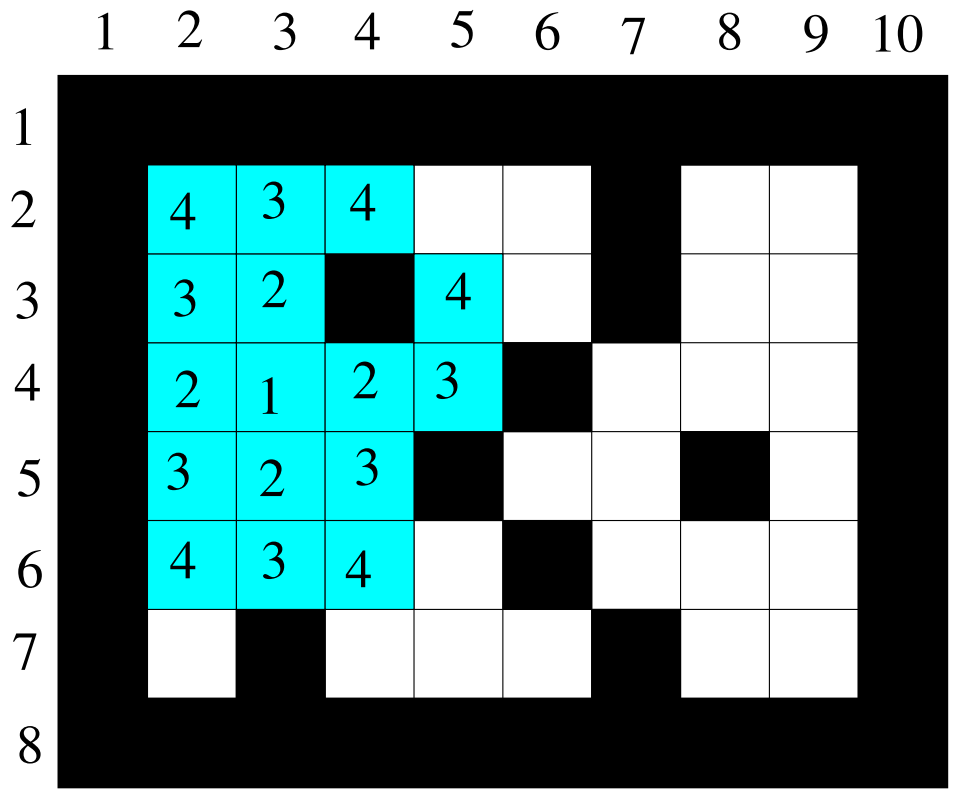


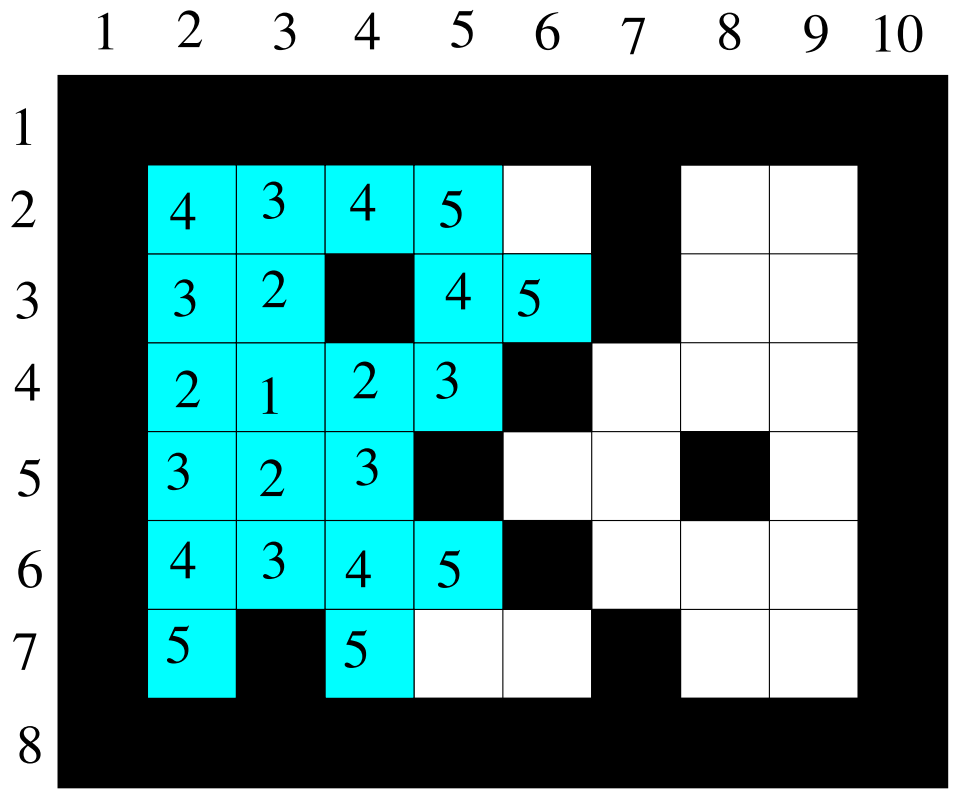
2. ábra. Az S sor tartalma

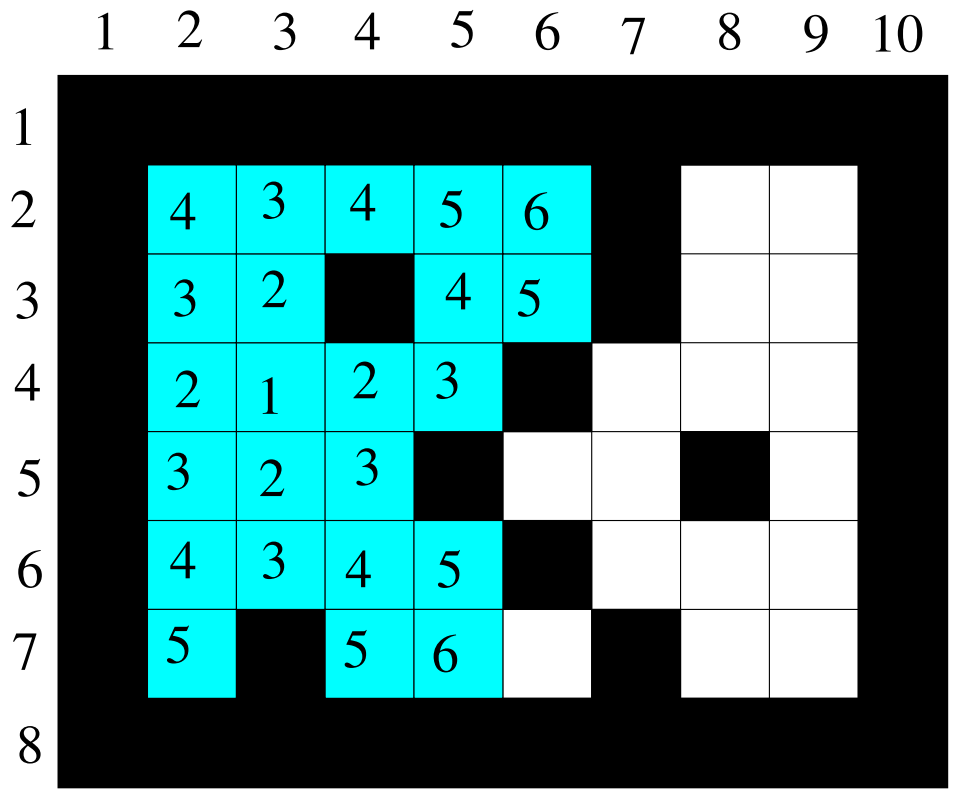

1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
1

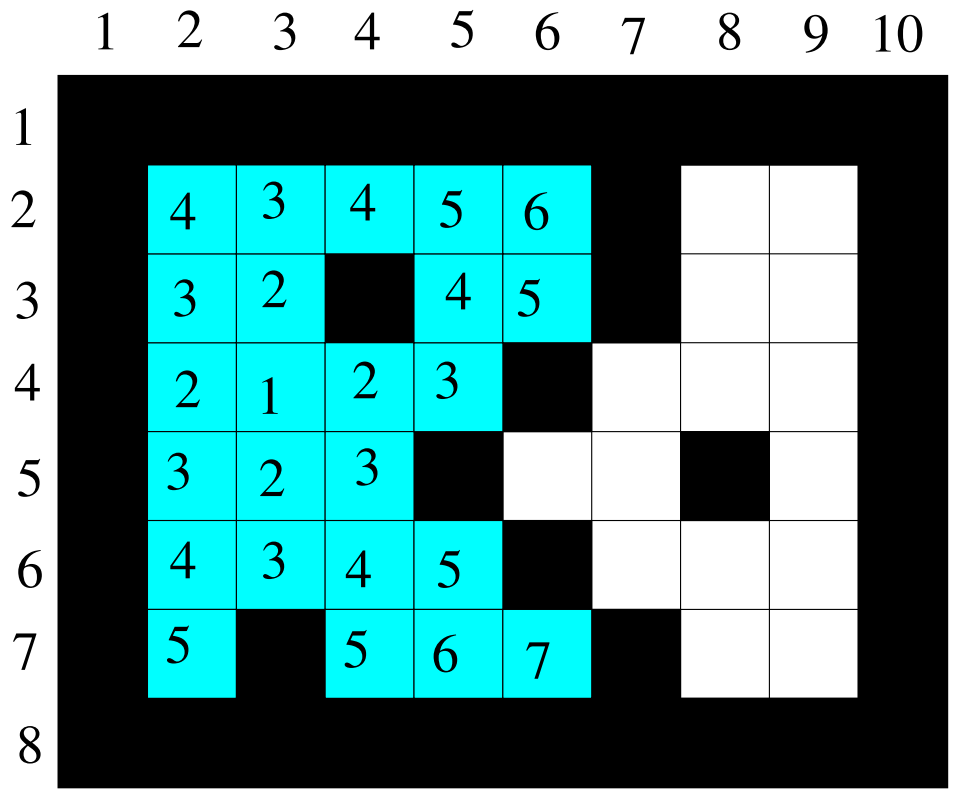

1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
2
2
1
2
2

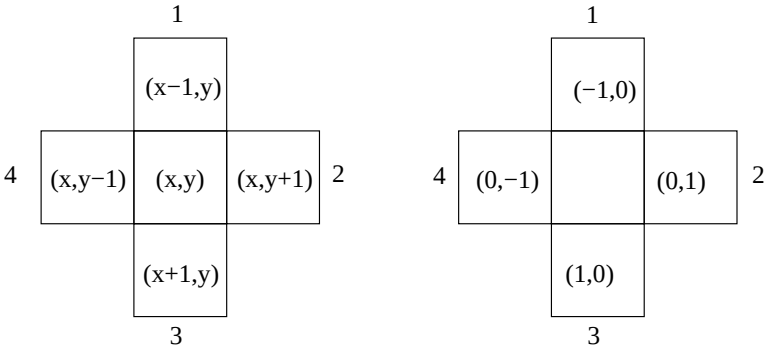

1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
3
3 2
2 1 2 3
3 2 3
3


1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
4 3 4
3 2 4
2 1 2 3
3 2 3
4 3 4


1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
4 3 4 5
3 2 4 5
2 1 2 3
3 2 3
4 3 4 5
5 5


1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
4 3 4 5 6
3 2 4 5
2 1 2 3
3 2 3
4 3 4 5
5 5 6


1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
4 3 4 5 6
3 2 4 5
2 1 2 3
3 2 3
4 3 4 5
5 5 6 7



3. ábra. Egy cella szomszédai relatív koordinátákkal

3.5.2. C++ megvalósítás

```
1  #include <iostream>
2  #include <queue>
3  #define maxNM 1001
4  using namespace std;
5  int n,m,fs,fo;
6  int T[maxNM][maxNM];
7  struct Cella{
8     int x,y;
9     Cella(){};
10    Cella(int a, int b){x=a; y=b;}
11 };
12 Cella D[4]={Cella(-1,0),Cella(0,1),Cella(1,0),Cella(0,-1)};
13
14 void Beolvas(){
15    int h;
16    cin>>m>>n>>fs>>fo;
17    for(int x=1;x<=m;x++){
18       for(int y=1;y<=n;y++){
19          cin>>h;
20          T[x][y]=h;
21       }
22    }
23 }
```

```
24 int main (){
25    Beolvas();
26    queue<Cella> S;
27    T[fs][fo]=1;
28    Cella c=Cella(fs,fo);
29    Cella cc;
30    S.push(c);
31    int mego=1;
32    while(!S.empty()){
33       c=S.front(); S.pop();
34       for(int i=0;i<4;i++){
35          cc.x=c.x+D[i].x; cc.y=c.y+D[i].y;
36          if(T[cc.x][cc.y]==0){
37             T[cc.x][cc.y]=T[c.x][c.y]+1;
38             S.push(cc);
39             if(T[cc.x][cc.y]>mego) mego=T[cc.x][cc.y];
40          }
41       }
42    }
43    cout<<mego<<endl;
44    return 0;
45 }
```

3.5.3. C# megvalósítás

```
1  using System;
2  using System.Collections.Generic;
3
4  class Program{
5     public class Cella{
6        public int x, y;
7        public Cella(int a, int b){x=a; y=b;}
8     }
9     public static void Main(){
10       string[] sor=Console.ReadLine().Split();
11       int m,n,fs,fo,h;
12       m=Convert.ToInt32(sor[0]);
13       n=Convert.ToInt32(sor[1]);
14       fs=Convert.ToInt32(sor[2]);
15       fo=Convert.ToInt32(sor[3]);
16       int[,]  T=new int[m+2,n+2];
17       Queue<Cella> S = new Queue<Cella>();
18
19       for(int x=1;x<=m;x++){
20          sor=Console.ReadLine().Split();
21          for(int y=1;y<=n;y++){
22             h=Convert.ToInt32(sor[y-1]);
23             T[x,y]=h;
```

```
24              }
25          }
26          T[fs, fo]=1;
27          Cella[] D={new Cella(-1,0), new Cella(0,1), new Cella(1,0), new Cella
28          Cella c=new Cella(fs,fo);
29          Cella cc=new Cella(0,0);
30          S.Enqueue(c);
31          int mego=0;
32          while(S.Count!=0){
33              c=S.Dequeue();
34              for(int i=0;i<4;i++){
35                  cc=new Cella(c.x+D[i].x, c.y+D[i].y);
36                  if(T[cc.x, cc.y]==0){
37                      T[cc.x, cc.y]=T[c.x, c.y]+1;
38                      S.Enqueue(cc);
39                      if(T[cc.x, cc.y]>mego) mego=T[cc.x, cc.y];
40                  }
41              }
42          }
43          Console.WriteLine(mego);
44      }
45  }
```

3.6. Sorozat (kétvégű sor)

Értékhalmaz: $Sorozat = \{\langle a_0, \ldots, a_{n-1} \rangle : a_i \in E, i = 1, \ldots, n, n \geq 0\}$
Műveletek argumentumai: $A : Sorozat, x : E$

Előfeltétel	Művelet	Utófeltétel
$A = \langle a_0, \ldots, a_{n-1} \rangle$	Elejére(A,x)	$A = \langle x, a_0, \ldots, a_{n-1} \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle$	Végére(A,x)	$A = \langle a_1, \ldots, a_{n-1}, x \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	Első(A)	$= a_0 \wedge A = Pre(A)$
$A = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	Utolsó(A)	$= a_{n-1} \wedge A = Pre(A)$
$A = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	ElsőTöröl(A)	$A = \langle a_1, \ldots, a_{n-1} \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	UtolsóTöröl(A)	$A = \langle a_0, \ldots, a_{n-2} \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle$	Elemszám(A)	$= n \wedge A = Pre(A)$
$A = A$	ÜresE(A)	$= (Pre(A) = \langle\rangle) \wedge A = Pre(A)$
–	–	–
$A = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	$Elem(A,i)$	$a_i \wedge A = Pre(A)$
$A = \langle a_0, \ldots, a_{n-1} \rangle \wedge 0 \leq i \wedge i < n$	Módosít(A,i,x)	$A = \langle x, a_0, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_{n-1} \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle$	Beszúr(A,i,x)	$A = \langle x, a_0, \ldots, a_{i-1}, x, a_i, \ldots, a_{n-1} \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle$	Töröl(A,i)	$A = \langle x, a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{n-1} \rangle$
$A = \langle a_0, \ldots, a_{n-1} \rangle$	Üresít(A)	$A = \langle\rangle$

3.7. A Sorozat C++ műveletei

Művelet	C++	C#
Típusdef	$deque < E >$	
Elejére(A,x)	$A.push\_front(x)$	$A.Insert(0,x)$
Végére(A,x)	$A.push\_back(x)$	$A.Add(x)$
Első(A)	$A.front()$	$A[0]$
Utolsó(A)	$A.back()$	$A[A.Count-1]$
ElsőTöröl(A)	$A.pop\_front()$	$A.RemoveAt(0)$
UtolsóTöröl(A)	$A.pop\_back()$	$A.RemoveAt(A.Count-1)$
Elemszám(A)	$A.size()$	$A.Count$
ÜresE(A)	$A.empty()$	$A.Count == 0$
$Elem(A,i)$	$A.item(i)$	$-$
$Elem(A,i)$	$A[i]$	$A[i]$
Módosít(A,i,x)	$A[i] = x$	$A[i] = x$
Beszúr(A,i,x)	$A.insert(A.begin()+i,x)$	$A.Insert(i,x)$
Töröl(A,i)	$A.erase(A.begin()+i)$	$A.RemoveAt(i)$
Üresít(A,i)	$A.clear()$	$A.Clear()$