
Eötvös Loránd Tudományegyetem
Informatikai Kar
Komputeralgebra Tanszék

# Véletlen sorozatok ellenőrzésének módszerei

dolgozat

*Témavezető:*
Dr. Iványi Antal Miklós
egyetemi tanár

*Készítette:*
Potempski Dániel
nappali tagozat
programtervező informatikus MSc

Budapest, 2011.

# 1. Bevezetés

A Latin négyzet, Sudoku négyzet és egyéb hasonló struktúrák ellenőrzésének több alkalmazása is ismert [1]. Ezért fontos ezen problémát megoldó algoritmusok ismerete, elemzése, esetleges javítása. A feladatot mélyebben megvizsgálva kiderül, hogy legtöbbször csak az első sor ellenőrzéséig jut el a vizsgálat, ezért a legtöbb forrás csak ennek a megoldó algoritmusaira koncentrál. Ezen cikk is ezt teszi. Jelen cikk ismerteti a pontos feladatot, az azt megoldó néhány algoritmust és azok már ismert főbb jellemzőit. Ezen túl foglalkozik egy eddig nem tárgyalt, általánosabb esettel, nevezetesen, amikor megengedett a cella üresen hagyása is. Ekkor a feladat annak eldöntése, hogy az adott sorban – leszámítva az üres cellákat – van-e ismétlődés. Ezen témakör eredményei a különböző algoritmusok összehasonlításából fakadnak.

# 2. Megoldó algoritmusok

A feladatot megoldó algoritmusok közül az ELŐRE, VISSZA, LINEÁRIS, SZEMETES, EDÉNY elnevezésűeket ismertetjük. A szakirodalom közöl egyéb megoldást is [2, 3].

## 2.1. ELŐRE

```
ELŐRE(n, s)
1 g ← TRUE
2 for i ← 1 to n − 1
3     for j ← i + 1 to n
4         if s[i] = s[j]
5             g ← FALSE
6             return g
7 return g
```

## 2.2. VISSZA

```
VISSZA(n, s)
1 g ← TRUE
2 for i ← 2 to n
3     for j ← i − 1 downto 1
4         if s[i] = s[j]
5             g ← FALSE
6             return g
7 return g
```

## 2.3. LINEÁRIS

```
LINEÁRIS(n, s)
1 g ← TRUE
2 for i ← 1 to n
3     v[i] ← 0
```

```
4 for i ← 1 to n
5     if v[s[i]] > 0
6         g ← False
7         return g
8     else v[s[i]] ← v[s[i]] + 1
9 return g
```

### 2.4. SZEMETES

Ennél az algoritmusnál fontos megjegyezni, hogy az $i$ ciklusváltozó csak nemnegatív értékkészlettel rendelkezhet, valamint az előforduló és-kapcsolat esetén csak a lusta kiértékelés lehet megengedett, különben könnyen kiindexelhetnénk az $s$ tömbből egy konkrét implementációban. Fontos még megemlíteni Monostori Gábort, aki ezt az algoritmust először megfogalmazta.

```
Szemetes(n, s)
1 g ← True
2 for i ← 1 to n
3     if v[s[i]] < i and s[v[s[i]]] = s[i]
4         g ← False
5         return g
6     else v[s[i]] ← i
7 return g
```

### 2.5. EDÉNY

```
Edény(n, s)
1 g ← True
2 m ← √n
3 for i ← 1 to m
4     c[i] ← 1
5 for i ← 1 to n
6     r ← ⌈s[i]/m⌉
7         for j ← 1 to c[r] − 1
8             if s[i] = Q[r, j]
9                 g ← False
10                return g
11        Q[r, c[r]] ← s[i]
12        c[r] ← c[r] + 1
13 return g
```

## 3. A kiterjesztett feladat

Felmerül a kérdés, hogyan lehetne az előbbi algoritmusokat átalakítani úgy, hogy megengedjük a bemenő sorozat celláinak kitöltetlenségét is. Azaz a véletlen sorozat egy elemének generálásakor megengedjük, hogy valamilyen valószínűséggel nem egy szám, hanem egy üres cella következzen. A mi esetünkben az ilyen cellát reprezentálja a -1 érték. Ez valóban értelmes, ha föltesszük, hogy az értékek csak nemnegatívak lehetnek.

# 4. A kiterjesztett algoritmusok

Az első három algoritmust úgy módosítjuk az új feladat elvégzése érdekében, hogy tárolnak egy címező tömböt, mely az üres cellák címeit már nem tartalmazza. Így miután egy ilyet felismertünk, többé azt nem használjuk. A maradék két algoritmusnál ez értelmetlen lenne, hisz egy-egy bemenő tömbelemet legfeljebb csak egyszer olvasnak, ezért ott minden lépésben először megnézik, vajon üres-e a cella. Ha igen, akkor nem foglalkoznak vele, ha nem, akkor az eredeti módom járnak el.

## 4.1. Kiterjesztett ELŐRE

Az első ciklus megkeresi az első nemüres cellát és frissíti ennek megfelelően az $a$ címező tömböt. A megtalált cella indexét $i$-ben tárolja. A második ciklus az eredeti program legelső iterációjának kiírása úgy, hogy az $i$ indexű cellát hasonlítja össze a többivel és ez alatt frissíti a címező tömböt. Ezek alatt karbantartottuk az $m$ változót is úgy, hogy a két ciklus végére a címező tömb hosszát, azaz a nemüres elemek számát tartalmazza. A harmadik ciklus az eredeti algoritmus csak a címező tömb segítségével már nem látjuk az üres cellákat, valamint a második nemüres elemtől indulunk, hiszen az elsőt már leellenőriztük.

ELŐRE2$(n, \mathbf{s})$
**1** $g \leftarrow$ TRUE
**2** $k \leftarrow 0$
**3** $i \leftarrow 0$
**4** $m \leftarrow n$
**5** **while** $k = 0$
**6** **if** $-1 = s[i]$
**7** $m \leftarrow m - 1$
**8** $i \leftarrow i + 1$
**9** **else** $a[k] \leftarrow i$
**10** $k \leftarrow k + 1$
**11** **for** $j \leftarrow i + 1$ **to** $n - 1$
**12** **if** $-1 = s[j]$
**13** $m \leftarrow m - 1$
**14** **else**
**15** **if** $s[i] = s[j]$
**16** $g \leftarrow$ FALSE
**17** **return** $g$
**18** $a[k] \leftarrow j$
**19** $k \leftarrow k + 1$
**20** **for** $i \leftarrow 1$ **to** $m - 2$
**21** **for** $j \leftarrow i + 1$ **to** $m - 1$
**22** **if** $s[a[i]] = s[a[j]]$
**23** $g \leftarrow$ FALSE
**24** **return** $g$
**25** **return** $g$

## 4.2. Kiterjesztett VISSZA

Az első ciklus megkeresi az első két nemüres cellát, miközben karbantartja az $a$ címező tömböt és annak $k$ hosszát. Ezután programunk az eredeti algoritmus szerint halad, de a címező tömbön keresztül, és annak frissítésével. Az $i$ változó megfelel az eredeti algoritmus ugyanilyen nevű ciklusváltozójának.

```
VISSZA2(n, s)
1 g ← TRUE
2 k ← 0
3 i ← 0
5 while k <= 1 and m > 0
6         if −1 = s[i]
7            m ← m − 1
8            i ← i + 1
9         else a[k] ← i
10              k ← k + 1
11              i ← i + 1
12 if k > 1 and s[a[k − 1]] = s[a[k − 2]]
13    g ← FALSE
14    return g
15 while i < n
16         if −1 = s[i]
17            a[k] ← i
18            for j ← k − 1 downto 0
19                if s[a[k]] = s[a[j]]
20                   g ← FALSE
21                   return g
22         else
23            n ← n − 1
24            i ← i + 1
25 return g
```

## 4.3. Kiterjesztett LINEÁRIS

Az eredeti algoritmus első, nullázó ciklusát használjuk az $a$ indexelő tömb kitöltésére. A tömb méretét az $m$ változóban számoljuk ki. Ezek után a bejövő $s$ tömböt már $a$-n keresztül érjük el.

```
LINEÁRIS2(n, s)
1 g ← TRUE
2 k ← 0
3 m ← n TRUE
4 for i ← 0 to n − 1
5     if −1 = s[i]
6        a[k] ← i
7        v[s[i]] ← 0
8        k ← k + 1
9     else m ← m − 1
10 for i ← 0 to n − 1
11     if v[s[a[i]]] > 0
```

```
12          g ← False
13          return g
14      else v[s[a[i]]] ← v[s[a[i]]] + 1
15 return g
```

### 4.4. Kiterjesztett SZEMETES

Itt az eredeti ciklusmagban levő vizsgálatot csak akkor hajtjuk végre, ha nemüres elemnél tartunk.

```
Szemetes2(n, s)
1 g ← True
2 for i ← 0 to n − 1
3     if −1! = s[i]
4         if v[s[i]] < i and s[v[s[i]]] = s[i]
5             g ← False
6             return g
7         else v[s[i]] ← i
8 return g
```

### 4.5. Kiterjesztett Edény

Ebben az algoritmusban is csak a nemüres elemek esetén vizsgálunk.

```
Edény(n, s)
1 g ← True
2 m ← √n
3 for i ← 0 to m − 1
4     c[i] ← 1
5 for i ← 0 to n − 1
6     if −1! = s[i]
7         r ← ⌈s[i]/m⌉
8                 for j ← 1 to c[r] − 1
9                     if s[i] = Q[r, j]
10                        g ← False
11                        return g
12                Q[r, c[r]] ← s[i]
13                c[r] ← c[r] + 1
14 return g
```

## 5. Jó sorozat valószínűsége

Fontos kérdés, hogy adott $n$ sorozathossz és $p$ ürescella-valószínűség mellett milyen valószínűséggel kapunk jó sorozatot. A kérdést elemi valószínűségszámítási megfontolások segítségével meg lehet válaszolni. Könnyen látható, hogy annak a valószínűsége, hogy a sorozatban $k$ darab cella üres, binomiális eloszlású. Ezért ez a valószínűség

$$P(k \text{ üres cella van}) = \binom{n}{k} p^k (1-p)^{n-k} .$$

$k$ kiválasztott cella esetén a jó sorozat valószínűségét relatív gyakorisággal kapjuk meg. A jó sorozatok száma: $n!$. Az összes lehetséges sorozat száma: $n^{n-k}k!$. Így az említett valószínűség

$$P(\text{ jó sorozat}|k \text{ üres cella van}) = \frac{n!}{n^{n-k}k!} .$$

A teljes valószínűség tételét felhasználva adódik a jó sorozat valószínűsége adott $n$ és $p$ esetén:

$$P(\text{ jó sorozat }) = \sum_{k=0}^{n} P(\text{jó sorozat}|k \text{ üres cella van})P(k \text{ üres cella van }) =$$

$$\sum_{k=0}^{n} \frac{n!}{n^{n-k}k!}\binom{n}{k}p^k(1-p)^{n-k}$$

Ezt szimulációval is ellenőriztem, mely eredményei az 1., 2., 3. ábrákon láthatóak.

# 6. A futásidők összehasonlítása

A nem kiterjesztett algoritmusok elemzéséről több helyen is olvashatunk [1, 2, 3]. A kiterjesztett algoritmusokat elemezni kell. Vajon mennyit számítanak a módosítások és az általánosabb feladat?

# 7. Eredmények összegzése???

# Hivatkozások


[1] Iványi Antal, Kátai Imre, *Testing of random matrices* Acta Univ. Sapientiae, Informatica, 3, 1 (2011) 99–126

[2] Iványi Antal, Balázs Novák *Testing of sequences by simulation* Acta Univ. Sapientiae, Informatica, 2, 2 (2010) 135–153

[3] Novák Balázs *S*udoku algoritmusok elemzése, Diplomamunka, ELTE, Budapest, 2010

| $p$ | $n$ | szimulált érték | formula |
|---|---|---|---|
| 0 | 3 | 0.246914 | 0.222222 |
| 0 | 4 | 0.0976562 | 0.09375 |
| 0 | 5 | 0.03296 | 0.0384 |
| 0 | 6 | 0.0154321 | 0.0154321 |
| 0 | 7 | 0.0057714 | 0.0061199 |
| 0 | 8 | 0.00241089 | 0.00240326 |
| 0 | 9 | 0.000846754 | 0.000936657 |
| 0 | 10 | 0.00036 | 0.00036288 |
| 0 | 11 | 0.00015523 | 0.000139906 |
| 0 | 12 | 5.62629e-05 | 5.37232e-05 |
| 0.1 | 3 | 0.329218 | 0.352 |
| 0.1 | 4 | 0.213867 | 0.211009 |
| 0.1 | 5 | 0.12928 | 0.127592 |
| 0.1 | 6 | 0.0807613 | 0.0775113 |
| 0.1 | 7 | 0.0467662 | 0.0472046 |
| 0.1 | 8 | 0.0284424 | 0.0287884 |
| 0.1 | 9 | 0.0174601 | 0.0175721 |
| 0.1 | 10 | 0.01092 | 0.0107319 |
| 0.1 | 11 | 0.00674321 | 0.00655683 |
| 0.1 | 12 | 0.00388616 | 0.00400714 |
| 0.2 | 3 | 0.489712 | 0.473778 |
| 0.2 | 4 | 0.358398 | 0.3344 |
| 0.2 | 5 | 0.23968 | 0.23721 |
| 0.2 | 6 | 0.159851 | 0.168632 |
| 0.2 | 7 | 0.119653 | 0.120012 |
| 0.2 | 8 | 0.0860291 | 0.0854643 |
| 0.2 | 9 | 0.0619655 | 0.0608863 |
| 0.2 | 10 | 0.04249 | 0.0433883 |
| 0.2 | 11 | 0.0310647 | 0.030925 |
| 0.2 | 12 | 0.022252 | 0.0220449 |
| 0.3 | 3 | 0.596708 | 0.586222 |
| 0.3 | 4 | 0.482422 | 0.459009 |
| 0.3 | 5 | 0.35968 | 0.360399 |
| 0.3 | 6 | 0.284851 | 0.283312 |
| 0.3 | 7 | 0.216933 | 0.222854 |
| 0.3 | 8 | 0.17218 | 0.175364 |
| 0.3 | 9 | 0.140189 | 0.138027 |
| 0.3 | 10 | 0.10816 | 0.108657 |
| 0.3 | 11 | 0.084141 | 0.0855467 |
| 0.3 | 12 | 0.0670292 | 0.0673576 |
| 0.4 | 3 | 0.691358 | 0.688 |
| 0.4 | 4 | 0.570312 | 0.58015 |
| 0.4 | 5 | 0.48064 | 0.49 |
| 0.4 | 6 | 0.418467 | 0.41416 |
| 0.4 | 7 | 0.353781 | 0.350195 |

1. ábra. Szimulációval kapott értékek jó sorozatok valószínűségére

| $p$ | $n$ | szimulált érték | formula |
|---|---|---|---|
| 0.4 | 8 | 0.297119 | 0.296178 |
| 0.4 | 9 | 0.253383 | 0.250532 |
| 0.4 | 10 | 0.21142 | 0.211942 |
| 0.4 | 11 | 0.179322 | 0.17931 |
| 0.4 | 12 | 0.151138 | 0.151711 |
| 0.5 | 3 | 0.740741 | 0.777778 |
| 0.5 | 4 | 0.681641 | 0.693359 |
| 0.5 | 5 | 0.62272 | 0.6187 |
| 0.5 | 6 | 0.550926 | 0.552324 |
| 0.5 | 7 | 0.486583 | 0.493189 |
| 0.5 | 8 | 0.445618 | 0.440449 |
| 0.5 | 9 | 0.393351 | 0.393386 |
| 0.5 | 10 | 0.35133 | 0.351375 |
| 0.5 | 11 | 0.313565 | 0.313865 |
| 0.5 | 12 | 0.280868 | 0.280369 |
| 0.6 | 3 | 0.839506 | 0.854222 |
| 0.6 | 4 | 0.775391 | 0.7944 |
| 0.6 | 5 | 0.74464 | 0.739171 |
| 0.6 | 6 | 0.684671 | 0.687957 |
| 0.6 | 7 | 0.640328 | 0.64038 |
| 0.6 | 8 | 0.593353 | 0.596144 |
| 0.6 | 9 | 0.555572 | 0.554995 |
| 0.6 | 10 | 0.51495 | 0.516706 |
| 0.6 | 11 | 0.481661 | 0.481071 |
| 0.6 | 12 | 0.447885 | 0.447903 |
| 0.7 | 3 | 0.91358 | 0.916 |
| 0.7 | 4 | 0.878906 | 0.879259 |
| 0.7 | 5 | 0.84768 | 0.844221 |
| 0.7 | 6 | 0.806199 | 0.810681 |
| 0.7 | 7 | 0.781996 | 0.778529 |
| 0.7 | 8 | 0.74939 | 0.747684 |
| 0.7 | 9 | 0.719741 | 0.718081 |
| 0.7 | 10 | 0.689 | 0.689664 |
| 0.7 | 11 | 0.660679 | 0.66238 |
| 0.7 | 12 | 0.635549 | 0.636183 |
| 0.8 | 3 | 0.946502 | 0.961778 |
| 0.8 | 4 | 0.949219 | 0.94415 |
| 0.8 | 5 | 0.93152 | 0.926937 |
| 0.8 | 6 | 0.913709 | 0.910081 |
| 0.8 | 7 | 0.892366 | 0.893555 |
| 0.8 | 8 | 0.87561 | 0.877343 |
| 0.8 | 9 | 0.861081 | 0.861434 |
| 0.8 | 10 | 0.84535 | 0.84582 |

2. ábra. Szimulációval kapott értékek jó sorozatok valószínűségére (folytatás)

| $p$ | $n$ | szimulált érték | formula |
|---|---|---|---|
| 0.8 | 11 | 0.829383 | 0.830493 |
| 0.8 | 12 | 0.814256 | 0.815448 |
| 0.9 | 3 | 0.99177 | 0.990222 |
| 0.9 | 4 | 0.987305 | 0.985509 |
| 0.9 | 5 | 0.98304 | 0.980835 |
| 0.9 | 6 | 0.977623 | 0.97619 |
| 0.9 | 7 | 0.970012 | 0.971572 |
| 0.9 | 8 | 0.96817 | 0.966977 |
| 0.9 | 9 | 0.962506 | 0.962407 |
| 0.9 | 10 | 0.95714 | 0.957859 |
| 0.9 | 11 | 0.953096 | 0.953334 |
| 0.9 | 12 | 0.948098 | 0.94883 |
| 1 | 3 | 1 | 1 |
| 1 | 4 | 1 | 1 |
| 1 | 5 | 1 | 1 |
| 1 | 6 | 1 | 1 |
| 1 | 7 | 1 | 1 |
| 1 | 8 | 1 | 1 |
| 1 | 9 | 1 | 1 |
| 1 | 10 | 1 | 1 |
| 1 | 11 | 1 | 1 |
| 1 | 12 | 1 | 1 |

3. ábra. Szimulációval kapott értékek jó sorozatok valószínűségére (folytatás)