

Algoritmizálás

Horváth Gyula

Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
horvath@inf.u-szeged.hu

3. Rekurzió

3.1. Feladat: Sorbaállítások száma

Hány féleképpen lehet sorbaállítani az osztály tanulóit?

Bemenet: a tanulók n száma.

Kimenet: ahány féleképpen az n tanuló sorbaállítható.

Megoldás

Jelölje $P(n)$ a megoldás értékét n tanuló esetén. A Tanulókat az $1, \dots, n$ számokkal azonosítjuk.

$P(1) = 1$

Visszavezés kisebb méretű, ugyanilyen probléma megoldására.

Tekintsük azokat a sorbaállításokat, amelyek esetén az n -edik tanuló a sorban az első helyen áll, és jelöljük ezek számát $S(1, n)$ -el. Általában, jelölje $S(i, n)$ azon sorbaállítások számát, ahol az n sorszámú tanuló a sorban az i -edik helyen áll. Tehát

$$P(n) = S(1, n) + S(2, n) + \dots + S(n, n)$$

Nyilvánvaló, hogy $S(1, n)$ megegyezik $n - 1$ tanuló összes lehetséges sorbaállításának számával, tehát $S(1, n) = P(n - 1)$. Általában, azon sorbaállítások száma, ahol az n -edik tanuló a i -edik helyen áll, $P(n - 1)$. Tehát, ha $n > 1$, akkor

$$P(n) = n * P(n - 1)$$

$$P(n) = n * (n - 1) * \dots * 2 * 1$$

```
1  P:=1;  
2  for i:=2 to n do  
3    P:=P*i;
```

3.2. Feladat: Zsebpénz

n Euro zsebpénzt kaptunk. Minden nap veszünk pontosan egy dolgot a következők közül (zárójelben az ár szerepel) percc (1 Eu), fagyalt (2 Eu), csoki (2 Eu).

Számítsuk ki, hogy hányféleképpen költhetjük el a zsebpénzünket!

Megoldás

Jelölje $K(n)$ az n Eu lehetséges elköltségeinek a számát. A következő összefüggések állnak fenn:

- $K(1) = 1$ (csak egy perccet vehetünk)
- $K(2) = 3$ (vagy két perccet, vagy egy csokit, vagy egy fagyit vehetünk)
- $K(n) = K(n - 1) + 2K(n - 2)$ ha $n \geq 3$, (első alkalommal perccet, csokit vagy fagyaltot vehetünk).

A következő algoritmus adja meg a $K(n)$ függvényt:

```

1 function K(n: integer): int64;
2 begin
3   if n=1 then
4     K:=1
5   else if n=2 then
6     K:=3
7   else
8     K:=K(n-1)+2^K(n-2);
9 end {K};

```

3.3. A rekurzió gyökerei: Peano axiómák

1. $0 \in \mathbb{N}$ (A 0 természetes szám)
2. $S(x) \neq 0$ (A 0 nem rákövetkezője egyetlen természetes számnak sem)
3. $S(x) = S(y) \Rightarrow x = y$
4. Ha $M \subseteq \mathbb{N}$ és $0 \in M$ és $\forall x (x \in M \Rightarrow S(x) \in M)$ akkor $M = \mathbb{N}$ (Indukció axióma)
5. $x + 0 = x$
6. $x + S(y) = S(x + y)$

Az $S(0) = 1$ jelölést használva, $x + 1 = x + S(0) = S(x + 0) = S(x)$

Az 5. és 6. axiómák egy rekurzív algoritmust adnak az 1-es számrendszerbeli összeadásra.

Az $a + b$ összeg kiszámításának (rekurzív) algoritmusai:

Vegyünk a darab kavicsot a bal kezünkbe, b darab kavicsot a jobb kezünkbe.

Ha a jobb kezünk üres, akkor az eredmény a bal kezünkben van (5. axióma).

Egyébként, $(b = S(\bar{b}) = \bar{b} + 1 \text{ valamely } \bar{b}\text{-ra})$

tegyünk félre 1 kavicsot a jobb kezünkéből

adjuk össze (ezen algoritmustal) a két kezünkben lévő kavicsokat

tegyük a bal kezünkbe az 1 félretett kavicsot

A szorzás rekurzív megadása

$$x \cdot 0 = 0$$

$$x \cdot S(y) = x + x \cdot y$$

3.4. Feladat: Eldöntendő, kinek van több birkája

Két szomszédos gazda vitatkozik, hogy kinek van több birkája. Adjunk algoritmust a vita eldöntésére!

A < (lineáris) rendezési reláció rekurzív megadása

$$0 < S(x)$$

$$\neg(x < 0)$$

$$S(x) < S(y) \Leftrightarrow x < y$$

3.5. Feladat: Partíciószám

Definíció. Az n természetes szám egy partíciója olyan $\pi = \langle a_1, \dots, a_k \rangle$ sorozat, amelyre:

- $a_1 \geq a_2 \geq \dots \geq a_k > 0$
- $\sum_{i=1}^k a_i = n$

(a_i a π partíció része.) Jelölje $P(n)$ n összes partíciójának számát.

Probléma: Partíciószám

Bemenet: n

Kimenet: n partícióinak száma, $P(n)$

Megoldás

Jelölje $P2(n, k)$ n azon partícióinak számát, amelyben minden rész $\leq k$.

Összefüggések:

1. $P2(1, k) = 1, P2(n, 1) = 1$
2. $P2(n, n) = 1 + P2(n, n-1)$
3. $P2(n, k) = P2(n, n)$ ha $n < k$
4. $P2(n, k) = P2(n, k-1) + P2(n-k, k)$ ha $k < n$

A megoldás: $P(n) = P2(n, n)$

```
1 function P(n:integer):int64;  
2   function P2(n,k:integer):int64;  
3   begin  
4     if (n=1) or (k=1) then  
5       P2:=1  
6     else if k>=n then  
7       P2:=1+P2(n,n-1)  
8     else  
9       P2:=P2(n, k-1)+P2(n-k, k);  
10  end{P2};  
11 begin  
12   P:=P2(n, n);  
13 end{P};
```

Rekurziós fa fogalma. Olyan fa, amelynek minden pontja egy eljáráshívást jelent adott aktuális paraméterekre, úgy, hogy a pont fiai megfelelnek azoknak az eljáráshívásoknak, amelyek végrehajthatók az aktuális paraméterek esetén.

Rekurzív algoritmus helyességének bizonyítása

I. Terminálás bizonyítása

Bebizonyítandó, hogy minden eljáráshívás végrehajtása véges lépésben befejeződik. (Az eljárás terminál.)

Terminálás bizonyítása megállási feltétellel.

Megállási feltétel

Legyen $P(x_1, \dots, x_n)$ n -paraméteres rekurzív eljárás.

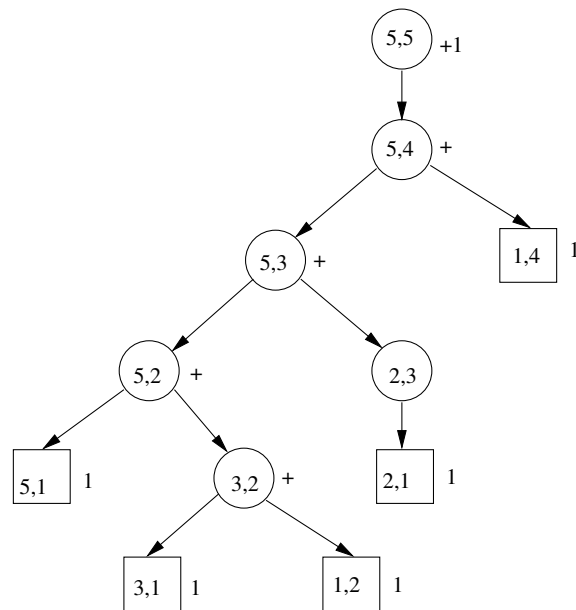
A $M(x_1, \dots, x_n)$ kifejezés megállási feltétele a P rekurzív eljárásnak, ha

1. $M(a_1, \dots, a_n) \geq 0$ minden megengedett $a_1, \dots, a_n$ aktuális paraméterre.
2. Ha $M(a_1, \dots, a_n) = 0$ akkor nincs rekurzív hívás $P(a_1, \dots, a_n)$ végrehajtásakor.
3. Ha van rekurzív hívás $P(a_1, \dots, a_n)$ végrehajtásakor valamely $b_1, \dots, b_n$ paraméterekre, akkor $M(b_1, \dots, b_n) < M(a_1, \dots, a_n)$

Állítás: $M(n, k) = (n-1) \times (k-1)$ megállási feltétel P2-re.

II. Helyesség bizonyítása

1. *Alaplépés.* Annak bizonyítása, hogy az eljárás helyes eredményt számít ki, ha az aktuális paraméterek esetén nincs rekurzív hívás.
2. *Rekurzív lépés.* Feltéve, hogy minden rekurzív hívás helyes eredményt ad, annak bizonyítása, hogy a rekurzív hívások által adott értékekből az eljárás helyes eredményt számít ki.



1. ábra. A $P2(5,5)$ eljárás hívás rekurziós fája

3.6. Feladat: Hanoi tornyai

A hanoi torony probléma: Három pálcá egyiken n korong van a többi üres. A korongok nagyság szerinti sorrendben helyezkednek el, alul van a legnagyobb. Át akarjuk helyezni a korongokat egy másik pálcára a következő szabályok alapján. Egyszerre csak egy korong mozgatható. A korong vagy üres pálcára vagy egy nála nagyobb korongra helyezhető. Oldjuk meg a feladatot egy rekurzív algoritmussal! Határozzuk meg a korongmozgatások számát!

Megoldás

Legyen a hanoi eljárás egy olyan algoritmus, amelynek három argumentuma van, az első argumentum azt adja meg, hogy hány korongot helyezünk át, a második megadja, hogy melyik toronyról a harmadik, hogy melyik toronyra. Ekkor az eljárás az $(n, 1, 2)$ argumentummal megoldja a feladatot. Amennyiben $i - 1$ korongot már át tudunk helyezni, i korongot a következőképpen helyezhetünk át. Elsőként $i - 1$ korongot áthelyezünk az oszlopról egy másik oszlopra. Utána az i -edik korongot ráadjuk a kimaradó üres oszlopra. Végül ezen korong tetejére felrakjuk az $i - 1$ korongot. Ezt a rekurziót írja le a következő eljárás (a megengedett lépést a mozgat függvény írja le, az argumentumai, hogy honnan hova)

```

1  procedure mozgat(rol, ra : Integer);
2  begin { mozgat }
3      writeln(rol, ' → ', ra);
4  end { mozgat };
5
6  procedure hanoi(n, rol, ra : integer);
7  begin { hanoi }
8      if (n=1) then
9          mozgat(rol, ra)
10     else begin
11         hanoi(n-1, rol, 6-rol-ra);
12         mozgat(rol, ra);
13         hanoi(n-1, 6-rol-ra, ra);
14     end
15 end { hanoi };

```

Lépések száma: Az i -edik korong átrakásához kétszer kell $i - 1$ korongot áthelyezni és egy további mozgatás szükséges. Tehát $T(i)$ -vel jelölve az i korong átrakásához szükséges mozgatások számát a $T(i) = 2T(i - 1) + 1$ rekurzív összefüggés áll fenn. $T(1) = 1$, így $T(2) = 3$, $T(3) = 7$. Azt sejtethetjük, hogy $T(i) = 2^i - 1$, amely egyenlőség teljes indukcióval egyszerűen igazolható.

3.7. Feladat: Pénzváltás

Bemenet: $P = \{p_1, \dots, p_n\}$ pozitív egészek halmaza, és E pozitív egész szám.

Eldöntendő, hogy van-e olyan $S \subseteq P$, hogy $\sum_{p \in S} p = E$.

Megjegyzés: A pénzek tetszőleges címletek lehetnek, nem csak a szokásos 1, 2, 5, 10, 20, stb., és minden pénz csak egyszer használható a felváltásban.

Megoldás

A megoldás szerkezetének elemzése.

Tegyük fel, hogy

$$E = p_{i_1} + \dots + p_{i_k}, \quad i_1 < \dots < i_k$$

egy megoldása a feladatnak. Ekkor

$$E - p_{i_k} = p_{i_1} + \dots + p_{i_{k-1}}$$

megoldása lesz annak a feladatnak, amelynek bemenete a felváltandó $E - p_{i_k}$ érték, és a felváltáshoz legfeljebb a első $i_k - 1$ ($p_1, \dots, p_{i_k-1}$) pénzeket használhatjuk.

Részproblémákra bontás.

Bontsuk részproblémákra a kiindulási problémát: Minden $(X, i) (1 \leq X \leq E, 1 \leq i \leq N)$ számpárra vegyük azt a részproblémát, hogy az X érték felváltható-e legfeljebb az első $p_1, \dots, p_i$ pénzzel. Jelölje $V(X, i)$ az (X, i) részprobléma megoldását, ami logikai érték; $V(X, i) = \text{Igaz}$, ha az X összeg előállítható legfeljebb az első i pénzzel, egyébként Hamis .

Összefüggések a részproblémák és megoldásaik között.

Nyilvánvaló, hogy az alábbi összefüggések teljesülnek a részproblémák megoldásaira:

1. $V(X, i) = (X = p_i)$, ha $i = 1$
2. $V(X, i) = V(X, i-1) \vee (X > p_i) \wedge V(X - p_i, i-1)$ ha $i > 1$

Rekurzív megoldás.

Mivel a megoldás kifejezhető egy $V(X, i)$ logikai értékű függvénnyel, ezért a felírt összefüggések alapján azonnal tudunk adni egy rekurzív függvényeljárást, amely a pénzváltás probléma megoldását adja.

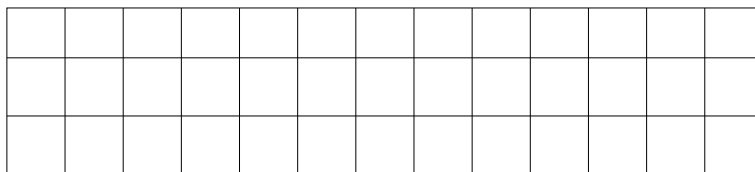
```
1 function V(X, i : integer)
2 // Globális:P
3 begin
4   V:= (X==P[i]) or
5     (i>1) and ( V(X,i-1) or (X>P[i]) and V(X-P[i],i-1) );
6 end{V};
```

Ez a megoldás azonban igen lassú, legrosszabb esetben a futási idő $\Omega(2^n)$.

3.8. Feladat: Járdakövezés

Számítsuk ki, hogy hányféleképpen lehet egy $3 \times n$ egység méretű járdát kikövezni 1×2 méretű lapokkal!

Megoldás

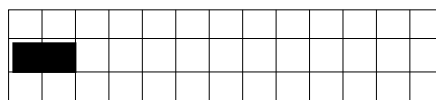


2. ábra.

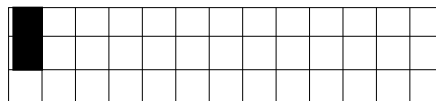
Jelölje $A(n)$ a megoldás értékét $3 \times n$ egység méretű járda esetén.

Az első oszlop középső négyzete háromféleképpen fedhető le.

Az egyes esetek csak az alábbi módon folytathatók:



3. ábra. 1. eset

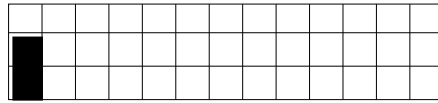


4. ábra. 2. eset

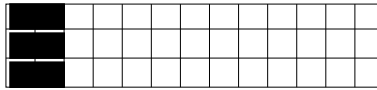
Jelölje $B(n)$ azt, hogy hányféleképpen fedhető le egy $3 \times n$ egység méretű járda, amelynek a bal alsó sarka már le van fedve. Szimmetria miatt a jobb felső sarok lefedettsége esetén is $B(n)$ -féle lefedés van.

$$A(n) = \begin{cases} 0 & \text{ha } n = 1 \\ 3 & \text{ha } n = 2 \\ A(n-2) + 2B(n-1) & \text{ha } n > 2 \end{cases} \quad (1)$$

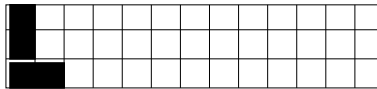
$$B(n) = \begin{cases} 1 & \text{ha } n = 1 \\ 0 & \text{ha } n = 2 \\ A(n-1) + B(n-2) & \text{ha } n > 2 \end{cases} \quad (2)$$



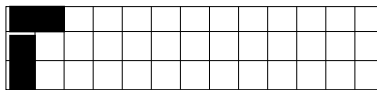
5. ábra. 3. eset



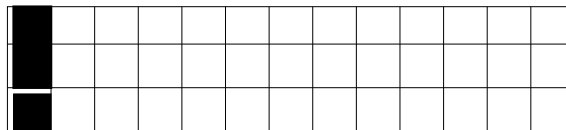
6. ábra. Az 1. eset csak így folytatható



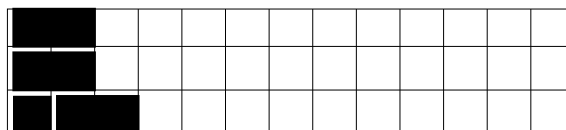
7. ábra. A 2. eset csak így folytatható



8. ábra. A 3. eset csak így folytatható



9. ábra. Az 1. eset csak így folytatható

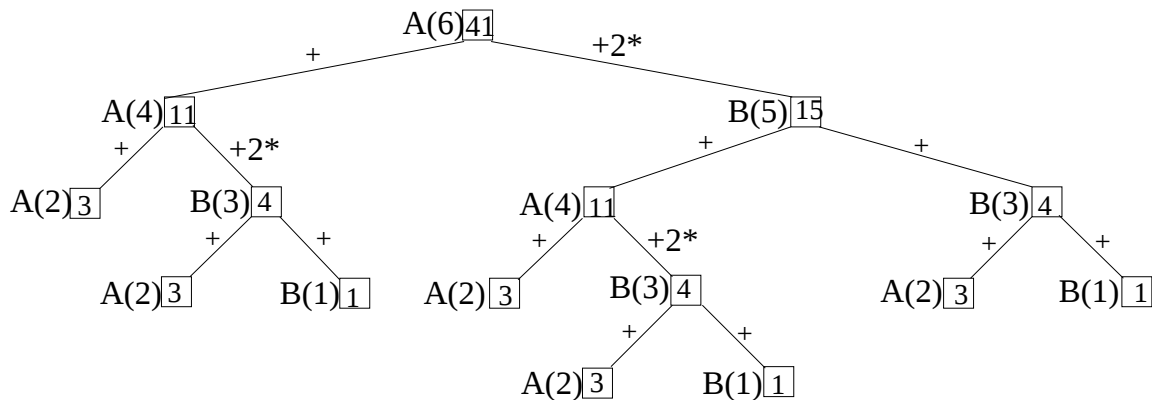


10. ábra. Az 2. eset csak így folytatható

```

1  program jarda;
2  function B(n: integer): longint; forward;
3  function A(n: integer): longint;
4  begin
5      if (n=1) then
6          A := 0
7      else if (n=2) then
8          A := 3
9      else
10         A := A(n-2)+2*B(n-1);
11 end{A};
12
13 function B(n: integer): longint;
14 begin
15     if (n=1) then
16         B := 1
17     else if (n=2) then
18         B := 0
19     else
20         B := A(n-1)+B(n-2);
21 end{B};
22 begin
23     writeln(A(32));
24 end.

```



11. ábra. Rekurziós fa

3.9. Feladat: Ördöglakat kinyitása

Az ördöglakat fém gyűrűkből összeállított szerkezet. Minden gyűrűnek van szára, amelyet körbefog a sorrendben következő gyűrű. Zárt állapotban a szárat körbefogja egy fémből készült hurok. Az a cél, hogy a lakatot kinyissuk, azaz a hurkot eltávolítsuk.

A gyűrűket balról-jobbra 1-től n -ig sorszámozzuk. Minden lépésben egy gyűrű vehető le, vagy tehető fel az alábbi két szabály betartásával.

1. Az első gyűrű bármikor levehető, illetve felrakható.
 2. Minden $i > 1$ sorszámú gyűrű akkor és csak akkor vehető le, illetve tehető fel, ha az $i - 1$ -edik gyűrű fent van, és minden $i - 1$ -nél kisebb sorszámú gyűrű lent van. A lakat akkor van kinyitva, ha minden gyűrű lent van.
- Írjon olyan rekurzív eljárást, amely megadja lépések olyan sorozatát, amely kinyitja a lakatot!

Megoldás

Részproblémákra bontás:

MindLe(m) Leveszi az első m gyűrűt (tetszőleges sorrendben), a többit változatlanul hagyja.

Le(i) Leveszi az i . gyűrűt, minden $j > i$ sorszámút változatlanul hagy. (A $j < i$ sorszámúak helyzete akármilyen lehet a művelet után.)

Fel(i) Felteszi az i . gyűrűt, minden $j > i$ sorszámút változatlanul hagy. (A $j < i$ sorszámúak helyzete akármilyen lehet a művelet után.)

```
1  program ordoglakat;
2  const
3      maxN=32;
4  var
5      lakat : array [1..maxN] of boolean;
6      i,n:integer;
7  procedure Le(i :integer) forward;
8  procedure Fel(i :integer)forward;

9  procedure MindLe(m : integer);
10 var i:integer;
11 begin
12     for i:=m downto 1 do
13         if lakat[i] then Le(i);
14 end{MindLe};
15
16 procedure Le(i :integer);
17 begin
18     if (i>1) and not lakat[i-1] then
19         Fel(i-1);
20     if (i>2) then MindLe(i-2);
21     lakat[i]:=false;
22     writeln(i, '._Le_');
23 end{Le};
24
25 procedure Fel(i :integer);
26 begin
27     if (i>1) and not lakat[i-1] then
28         Fel(i-1);
29     if (i>2) then MindLe(i-2);
30     lakat[i]:=true;
31     writeln(i, '._Fel_');
32 end{Fel};
33
34 begin
35     n:=5;
36     for i:=1 to n do lakat[i]:=true;
37     MindLe(n);
38 end.
```

3.10. Rekurzív görbék

3.11. Feladat: Hópihe görbe rajzolás

Görbe leírása Lindenmayer rendszerrel

Mag=F++F++F,

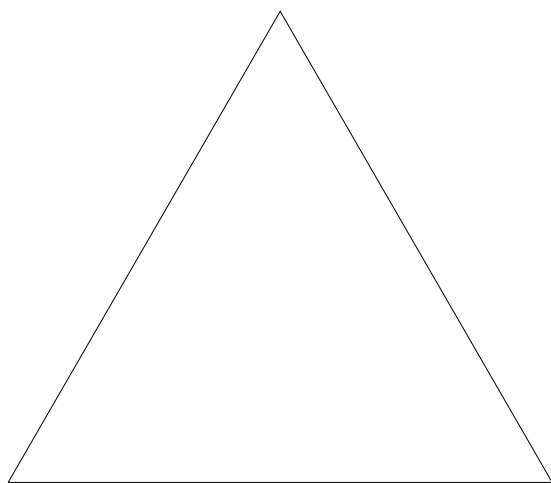
F => F-F++F-F,

alfa=0,

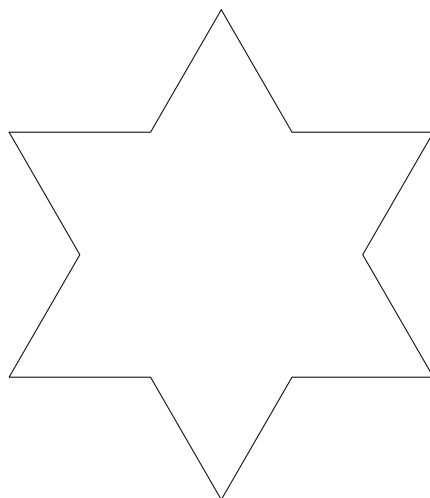
delta=120 fok

Megvalósítás teknőc grafikával

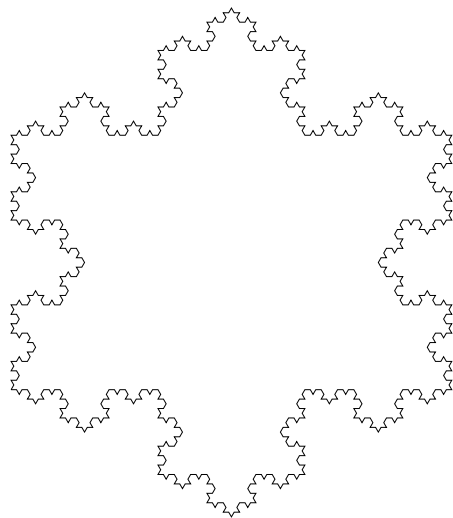
Teknőc állapota:



12. ábra. 1. rendű hópihe



13. ábra. 2. rendű hópihe



14. ábra. 5. rendű hópihe.

tartózkodási helye : (x,y) koordinátájú pont a síkon

irány : a vízszintes egyenessel bezárt α szög

elmozdulás mértéke : d (konstans)

elfordulás szöge : δ (konstans)

Műveletek:

- E : előremegy az adott irányítással d távolságot
- $+$ (B): balra fordul δ szöggel
- $-$ (J): jobbra fordul δ szöggel

```
1 program Hopihe;
2 uses PSteknoc;
3 //Mag=F++F++F,
4 //F => F-F++F-F,
5 // alfa=0,
6 // delta=120 fok
7 procedure F(n: integer);
8 begin
9     if n=1 then
10         E
11     else begin
12         F(n-1);J;F(n-1);B;B;F(n-1);J;F(n-1);
13     end;
14 end;
15 procedure Mag(n: integer);
16 begin
17     F(n);B;B;F(n);B;B;F(n);
18 end;
19 begin
20     kezd(100,200,0,5, Pi/3, 'hopihe.ps');
21     Mag(5);
22     zar;
23 end.

1 unit PSteknoc;
2 interface
3 const
4 //A4-es lapméret postscript pontokban, 1 pt=25,4/72 (=0,3528)mm
5     maxX=595;
6     maxY=842;
7 procedure kezd(x0,y0, alfa0 ,d0,delta0: double; fnev:string);
8 procedure E;
9 procedure J;
10 procedure B;
11 procedure zar;
12 procedure M;
13 procedure V;
14 procedure tollSzin(r,g,b: single);

15 implementation
16 const
17     Pi2=2*Pi;
18     Vmax=10000;
19 var
```

```

20   x,y,alfa,d,delta :double;
21   psf:text;
22   Vt:array[1..Vmax] of double;
23   Vm:integer;
24
25 procedure kezd(x0,y0,alfa0,d0,delta0: double; fnev:string);
26 begin
27   x:=x0; y:=y0; alfa:=alfa0;
28   d:=d0; delta:=delta0;
29   Vm:=0;
30   assign(psf, fnev); rewrite(psf);
31   writeln(psf, '%!PS-Adobe-2.0');
32   writeln(psf, '0.5_setlinewidth');
33   writeln(psf, x:8:3, '_', y:8:3, '_moveto');
34 end{kezd};
35
36 procedure E;
37 begin
38   x := x + d  $\wedge$  cos(alfa);
39   y := y + d  $\wedge$  sin(alfa);
40   writeln(psf, x:8:3, '_', y:8:3, '_lineto');
41 end;
42 procedure J;
43 begin
44   alfa:=alfa-delta;
45   if (alfa<0) then alfa:=alfa+Pi2;
46 end;
47 procedure B;
48 begin
49   alfa:=alfa+delta;
50   if (alfa>Pi2) then alfa:=alfa-Pi2;
51 end;
52 procedure zar;
53 begin
54   writeln(psf, 'stroke');
55   close(psf)
56 end{zar};
57
58 procedure M;
59 begin
60   inc(Vm); Vt[Vm]:=alfa;
61   inc(Vm); Vt[Vm]:=y;
62   inc(Vm); Vt[Vm]:=x;
63 end;
64 procedure V;
65 begin
66   x := Vt[Vm]; dec(Vm);
67   y := Vt[Vm]; dec(Vm);
68   alfa:=Vt[Vm]; dec(Vm);
69   writeln(psf, x:8:3, '_', y:8:3, '_moveto');
70 end;
71 procedure tollSzin(r,g,b:single);
72 begin
73   writeln(psf, r:8:3, '_', g:8:3, b:8:3, '_setrgbcolor');
74 end;

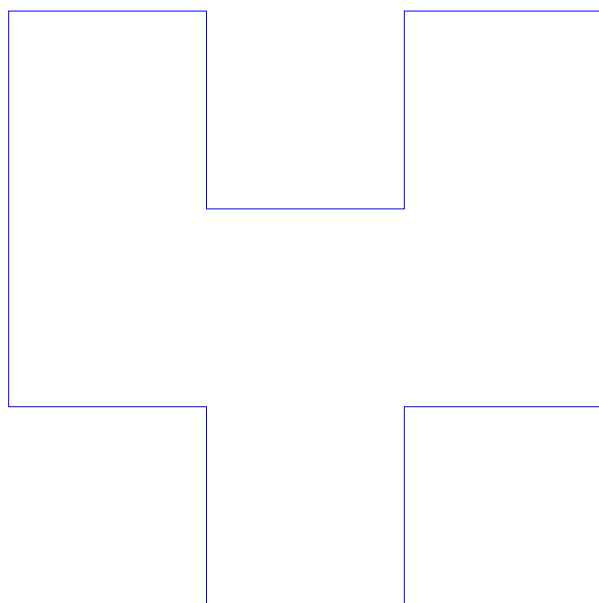
```

3.12. Feladat: Hilbert-görbe rajzolás



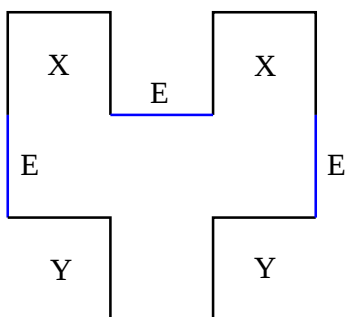
15. ábra. 1. rendű Hilbert-görbe

```
1 program Hilbert;
2 uses PSteknoc;
3 //X => +YE-XEX-Y+
4 //Y => -XE+YFY+Y-
5 procedure Y(n:integer) forward;
6 procedure X(n:integer);
7 begin
8   if n=0 then exit;
9   B; Y(n-1);E; J; X(n-1); E; X(n-1); J; E; Y(n-1); B;
10 end{X};
11 procedure Y(n:integer);
12 begin
13   if n=0 then exit;
14   J; X(n-1); E; B; Y(n-1); E; Y(n-1); B; E; X(n-1); J;
15 end{Y};
16
17 procedure Mag(n:integer);
18 begin
19   X(n);
20 end;
21 var
22   n:integer;
23   h,d:double;
```



16. ábra. 2. rendű Hilbert-görbe

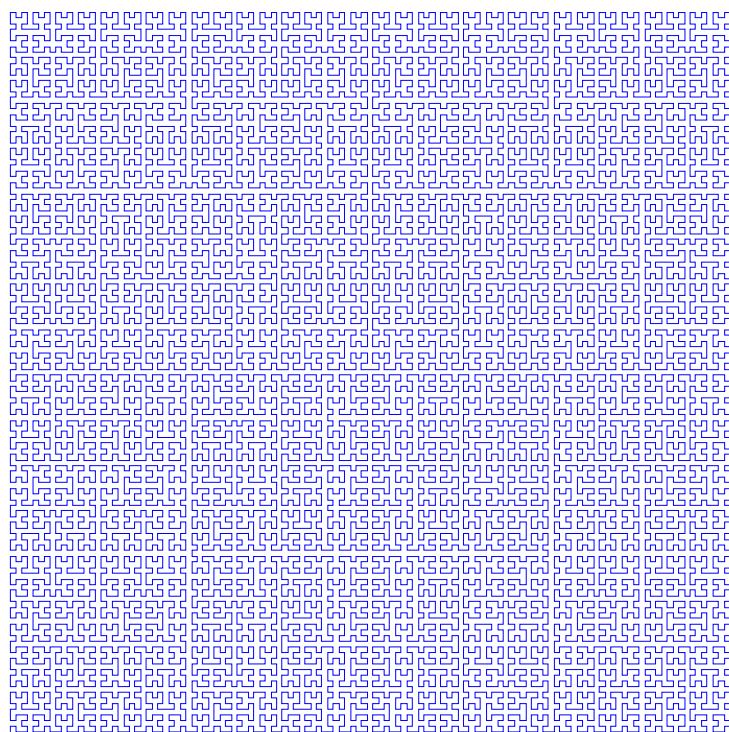
Hilbert görbe



1. ábra.

$$\begin{aligned} X &= +YE - XEX - EY + \\ Y &= -XE + YEY + EX - \end{aligned}$$

17. ábra.



18. ábra. 7. rendű Hilbert-görbe

