2. Rekurzió

Egy objektum definícióját rekurzívnak nevezünk, ha a definíció tartalmazza a definiálandó objektumot.
Egy P eljárást (vagy függvényt) rekurzívnak nevezünk, ha P utasításrészében előfordul magának a P eljárásnak a hívása.

2.1. Partíciószám

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

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

(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:

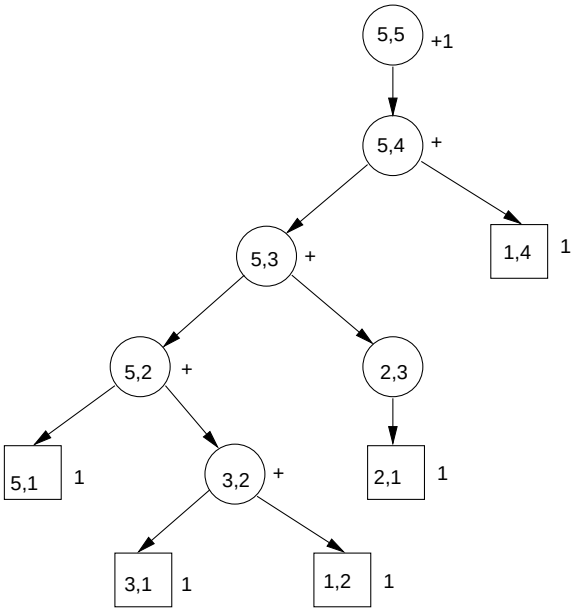
1. $P2(1,k) = 1, P2(n,1) = 1$
2. $P2(n,n) = 1 + P2(n,n-1)$
3. $P2(n,k) = P2(n,n)$ ha $n < k$
4. $P2(n,k) = P2(n,k-1) + P2(n-k,k)$ ha $k < n$

A megoldás: $P(n) = P2(n,n)$

```
public class Particio{
    public static long P(int n){
        return P2(n,n);
    }
    private static long P2(int n, int k){
        if (k==1 || n==1)
            return 1;
        if (k>=n)
            return P2(n,n-1)+1;
        return P2(n, k-1)+P2(n-k, k);
    }
}
```

Rekurziós fa fogalma. Olyan fa, amelynek minden pontja egy eljáráshívást jelent adott aktuális paraméterekre, úgy, hogy a pont fiai megfelelnek azoknak az eljáráshívásoknak, amelyek végrehajtódnak az aktuális paraméterek esetén.
Futási idő elemzése
Legyen $E(n,k)$ a P$2(n,k)$ eljáráshívás hatására végrehajtott eljárásutasítások száma.

Állítás. $P2(n,k) \leq E(n,k) \leq 2P2(n,k) - 1$.
$\Rightarrow P(n) \leq E(n,n) \leq 2P(n,n) - 1$.
Tehát a P(n) eljáráshívás futási ideje $\Theta(P(n))$
Biz.
1. $P2(n,k) \leq E(n,k)$
a. Ha $n = 1$ vagy $k = 1$: $P2(n,k) = 1 = E(n,k)$
b. $P2(n,n) = 1 + P2(n,n-1) \leq 1 + E(n,n-1) = E(n,n)$
c. $n > k$: $P2(n,k) = P2(n,k-1) + P2(n-k,k) \leq E(n,k-1) + E(n-k,k)$
$= E(n,k) - 1 < E(n,k)$
2. $E(n,k) \leq 2P2(n,k) - 1$.
a. Ha $n = 1$ vagy $k = 1$: $E(n,k) = 1, P2(n,k) = 1, 2 * 1 - 1 = 1$
b. $E(n,n) = 1 + E(n,n-1) \leq 1 + 2P2(n,n-1) - 1 = 2[P2(n,n) - 1]$
$= 2P2(n,n) - 2 < 2P2(n,n) - 1$



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

c. $n > k$: $E(n,k) = 1 + E(n,k-1) + E(n-k,k)$
$\leq 1 + 2P2(n,k-1) - 1 + 2P2(n-k,k) - 1 = 2[P2(n,k-1) + P2(n-k,k)] - 1$
$= 2P2(n,k) - 1$
Állítás. $P(n) = \Omega(2^{\sqrt{n}})$

Biz. Tekintsük az összes olyan $\langle a_1, \cdots, a_k \rangle$ sorozatot, ahol $\sqrt{n} \geq a_1$ és $a_1 > a_2 > \ldots > a_k$. Ezek száma pontosan $2^{\lfloor\sqrt{n}\rfloor}$, mivel minden ilyen sorozat egyértelműen megadható $\langle b_{\lfloor\sqrt{n}\rfloor}, \ldots, b_1 \rangle$ bitvektorral, ahol $b_x = 1$ akkor és csak akkor, ha x eleme a sorozatnak.
Mivel $k \leq \sqrt{n}$, így $\sum_{i=1}^{k} a_i \leq \sqrt{n}\sqrt{n} \leq n$. Minden ilyen sorozathoz adjunk hozzá a végén annyi 1-est, hogy a számok összege n legyen, tehát n egy partícióját kapjuk. Ha két kiindulási sorozat különböző volt, akkor a kiegészítéssel különböző partíciót kapunk. Tehát n partícióinak száma $\geq 2^{\lfloor\sqrt{n}\rfloor}$
Következmény. A P algoritmus futási ideje $\Omega(2^{\sqrt{n}})$.
Megjegyzés.
1. Az is bizonyítható, hogy $P(n)$ futási ideje $O(2^{\sqrt{n}})$.
2. Nem ismert $P(n)$ kiszámítására zárt formula, de közelítő igen, az u.n. Hardy-Ramanujan formula:
$P(n) \sim \frac{1}{4\sqrt{3}n} e^{\frac{\pi}{3}\sqrt{6n}}$

2.2. 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 $P(x_1, \ldots, x_n)$ n-paraméteres rekurzív eljárás.
A $M(x_1, \ldots, x_n)$ kifejezés megállási feltétele a P rekurzív eljárásnak, ha

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

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.

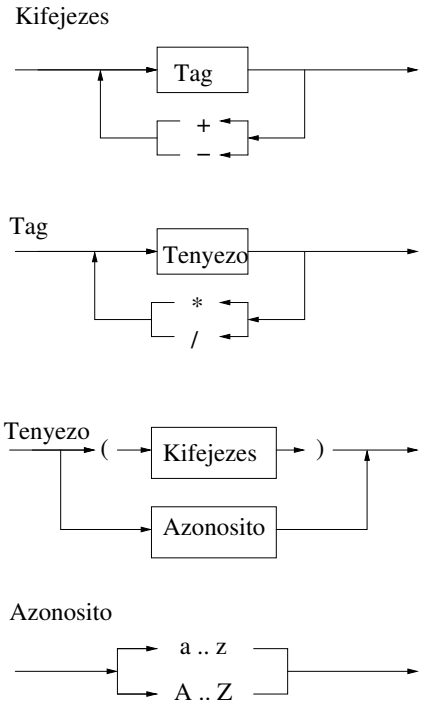
2. *Rekurzív lépés*. Feltéve, hogy minden rekurzív hívás helyes eredményt ad, annak bizonyítása, hogy a rekurzív hívások által adott értékekből az eljárás helyes eredményt számít ki.

Általános (szimultán) rekurzió fogalma
A $\{P_1, \ldots, P_n\}$ eljárásrendszer rekurzív, ha a hívási gráfjában van kör.

2.3. Postfix konverzió

Aritmetikai kifejezés szokásos írásmódja, hogy a műveleti jel az argumentumok között áll. Ezt az írásmódot infix jelölésnek is nevezzük. Mivel a multiplikatív műveletek (szorzás $*$ és osztás $/$) prioritása magasabb, mint az additív $(+,-)$ műveleteké, ezért zárójelezni kell.
Pl. $(a+b)*(c-d)+a$.

Lukasiewic lengyel logikatudós vette először észre, hogy ha a műveleti jeleket az argumentumok után írjuk, akkor nincs szükség zárójelekre. Ezért ezt az írásmódot *fordított lengyel jelölésnek*, vagy *postfix* alaknak hívjuk.
Jelölje $\phi(K)$ a K kifejezés postfix alakját.
Pl. $\phi((a+b)*(c-d)+a) = ab+cd-*a+$
Probléma: Postfix konverzió
Bemenet: K aritmetikai kifejezés infix jelölésben.
Kimenet: K postfix alakja.
A szabályos aritmetikai kifejezések megadhatók a következő (rekurzív) szintaxis diagramokkal (Az egyszerűség kedvéért az elemi kifejezések csak egybetűs azonosítok lehetnek).
Tehát minden kifejezés vagy egy *tag*, vagy tagok additív műveleti jelekkel elválasztott sorozata. Minden kifejezés egyértelműen



2. ábra. Kifejezés szintaxis diagramjai

felbontható
$K = t_1 \oplus_1 t_2 \ldots \oplus_m t_{m+1}$ alakban, ahol $\oplus_i \in \{+,-\}$ és t_i tag. Ekkor (a balról-jobbra szabály szerint)
$\phi(K) = \phi(t_1)\phi(t_2) \oplus_1 \ldots \phi(t_{m+1}) \oplus_m$.

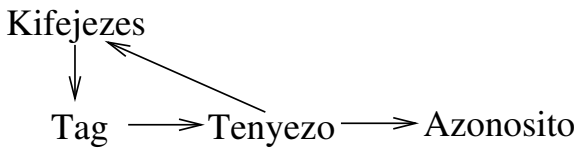
Minden tag vagy egy *tényező*, vagy tényezők multiplikatív műveleti jellel elválasztott sorozata. Minden tag egyértelműen felbontható
$T = t_1 \otimes_1 t_2 \ldots \otimes_m t_{m+1}$ alakban, ahol $\otimes_i \in \{*, /\}$ és t_i tényező. Ekkor (a balról-jobbra szabály szerint)
$\phi(T) = \phi(t_1)\phi(t_2) \otimes_1 \ldots \phi(t_{m+1}) \otimes_m$.

Minden tényező vagy zárójelbe tett kifejezés; (K) és $\phi((K)) = \phi(K)$, vagy azonosító; a és $\phi(a) = a$.
Tehát a konverzió algoritmusát megalkothatjuk úgy, hogy minden szintaktikus egységhez (Kifejezés, Tag, Tényező, Azonosító) egy eljárást adunk, amely balról jobbra haladva az aktuális karaktertől elolvassa és konvertálja a neki megfelelő (legszűkebb) karaktersorozatot.
A megvalósítás algoritmusa nem feltételezi, hogy a bemenet szabályos kifejezés, hibás kifejezés esetén jelzi az első hiba helyét.

```
public class Postfix{
    private static boolean Jo=true;
    private static String S;
    private static int i=0;
    private static char Jel;
    private static String Postform;

    private static  void KovJel(){
        Jel=S.charAt(i);
        i++;
    }

    public static void Kifejezes(){
        char M;
        Tag();
        while (Jo && Jel=='+' || Jel=='-'){
            M=Jel;
            KovJel();
            Tag();
            Postform=Postform+M;
        }
    }

    private static void Tag(){
        char M;
        Tenyezo();
        while (Jo && Jel=='*' || Jel=='/'){
            M=Jel;
            KovJel();
            Tenyezo();
            Postform=Postform+M;
        }
    }
    private static void Tenyezo(){
        if ('a'<=Jel && Jel<='z'){
            Postform=Postform+Jel;
            KovJel();
        }else if (Jel=='('){
            KovJel();
            Kifejezes();
            if (Jo && Jel==')')
                KovJel();
            else
                Jo=false;
        }else
            Jo=false;
```

```
    }

    public static String postfix(String K){
        S=K+'.';
        KovJel();
        Postform="";
        Kifejezes();
        return Postform;
    }

    public static void main (String[] args) {
    System.out.println(postfix("a+b*(a-b)/(x+y)"));
    }
}
```

Az eljárásrendszer hívási gráfja.



3. ábra. Az eljárások hívási gráfja