Algoritmusok és adatszerkezetek gyakorlat – 03

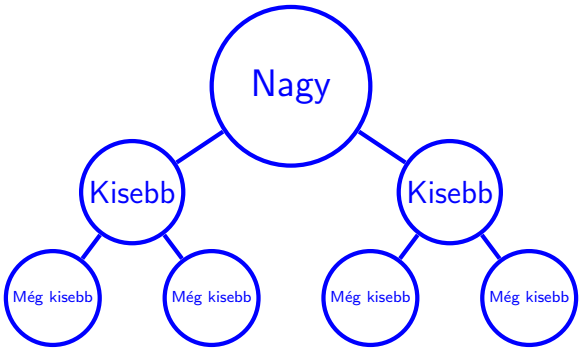
Oszd meg és uralkodj

Divide & Conquer („Oszd meg és uralkodj”) paradigma

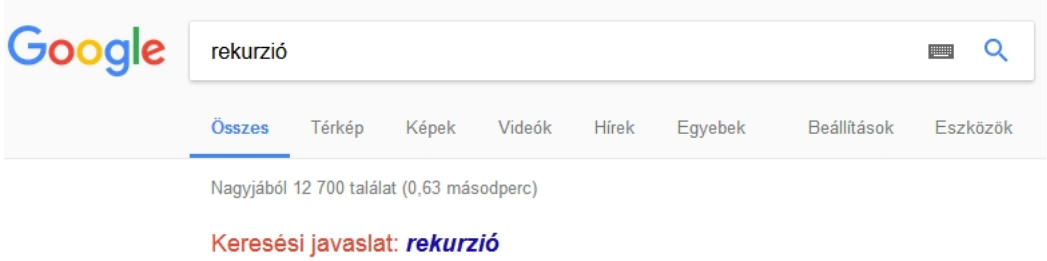
Divide: Osszuk fel az adott problémát kisebb problémákra.

Conquer: Oldjuk meg a kisebb problémákat (iteratívan vagy rekurzívan)

Combine: Ezek eredményei alapján oldjuk meg az adott „nagy” problémát.



Rekurzió



Olyan művelet, mely végrehajtáskor a saját maga által definiált műveletet, vagy műveletsort hajtja végre (általában a probléma egy kisebb példányára), ezáltal önmagát ismétli.

Egy rekurzív algoritmusnak két része van:

- **Alapeset**: a megoldást már előre tudjuk vagy könnyen kiszámítható (ekkor nincs rekurzív hívás)
- **Rekurzív eset:** a megoldást úgy kapjuk, hogy a probléma más, általában kisebb példányainak megoldását használjuk fel

Csináljunk belőle mindig kisebbet, amíg el nem érjük az alapesetet

Példa: Faktoriális számítás: $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$

- Rekurzív összefüggés: $n! = n \cdot (n-1)!$
- Alapeset: $n = 1$
- Rekurzív eset: $n > 1 \Rightarrow n \cdot (n-1)!$

1. Feladat Az előző órán tárgyalt bináris keresés egy ilyen Oszd meg és uralkodj típusú algoritmus. Mutassuk meg, hogy tényleg az és adjuk meg a rekurzív változatát!

Megoldás

Az oszd meg és uralkodj paradigma 3 lépésének megvalósulása:

1. **Divide:** megnézzük a középső elemet, és az értékétől függően döntünk
2. **Conquer:** Az egyik résztömbben keresünk ugyanezzel a módszerrel
3. **Combine:** Nem szükséges.

Írjuk fel a lehetséges eseteket:

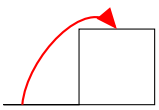
- **Alapeset:** két alapesetünk van:
  1. amikor megtaláljuk az elemet: $key == a[mid] \Rightarrow$ return mid
  2. és amikor nincs benne a keresett elem a tömbben: $r <= l \Rightarrow$ return -1
- **Rekurzív eset:** attól függően, hogy a középső elemtől merre van a keresett elem, keressük abban a résztömbben rekurzívan. return binarySearch(a, alsohatar, felsohatar, key);

2. Feladat Adjunk rekurzív algoritmust, ami meghatározza, hogy hányféleképpen mehetünk fel egy n lépcsőfokból álló lépcsőn, ha egyszerre csak 1 vagy 2 lépcsőfokot léphetünk!

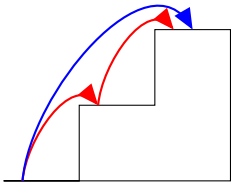
Megoldás

Jelölje $P(n)$ azt, hogy n lépcsőfokon hányféleképpen mehetünk fel. Ekkor a következő összefüggések állnak fent:

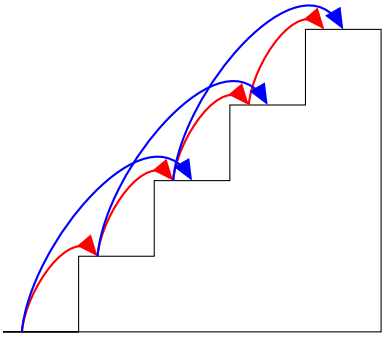
- $P(1) = 1$ (ha egy lépcső van, egyféleképpen mehetünk)



- $P(2) = 2$ (ha két lépcső van, vagy kétszer egyet lépünk, vagy egyszer kettőt)



- $P(n) = P(n-1) + P(n-2)$, ha $n \geq 3$ (utolsó lépésként egyet vagy kettőt léphetünk)



Algoritmus:

```
int P(int n){
  if (n == 1)
    return 1;
  else if (n == 2)
    return 2;
  else
    return P(n-1) + P(n-2);
}
```

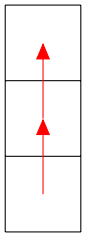
3. Feladat Adjunk rekurzív algoritmust, amely meghatározza, hogy hányféleképpen juthatunk el egy k sorból és n oszlopból álló sakktábla bal alsó sarkából a jobb felső sarkába, ha csak a jobbra vagy a felfelé szomszédos mezőre léphetünk!

Megoldás

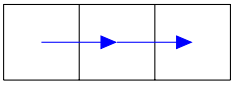
Jelölje $R(k, n)$ azt a számot, ahányféleképpen eljuthatunk a bal alsó sarokból a jobb felső sarokba.

Ekkor a következő összefüggések állnak fent:

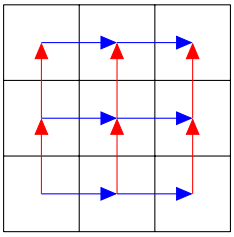
- $R(k,1) = 1$ (csak felfelé mehetünk)



- $R(1,n) = 1$ (csak jobbra mehetünk)



- $R(k,n) = R(k,n-1)+R(k-1,n)$, ha $n,k \geq 2$ (az első lépés jobbra vagy felfelé történhet)



Algoritmus:

```
int R( int k , int n){
  if ((k == 1) or (n == 1))
    return 1;
  else
    return R(k, n-1) + R(k-1, n);
}
```

4. Feladat Hajtsuk végre a következő rendezőalgoritmust a $8,4,1,5,3,2,6,7$ listán!

Összefésülő rendezés

- **Input**: Egy n elemet tartalmazó tömb
- **Output**: Az input tömb rendezve

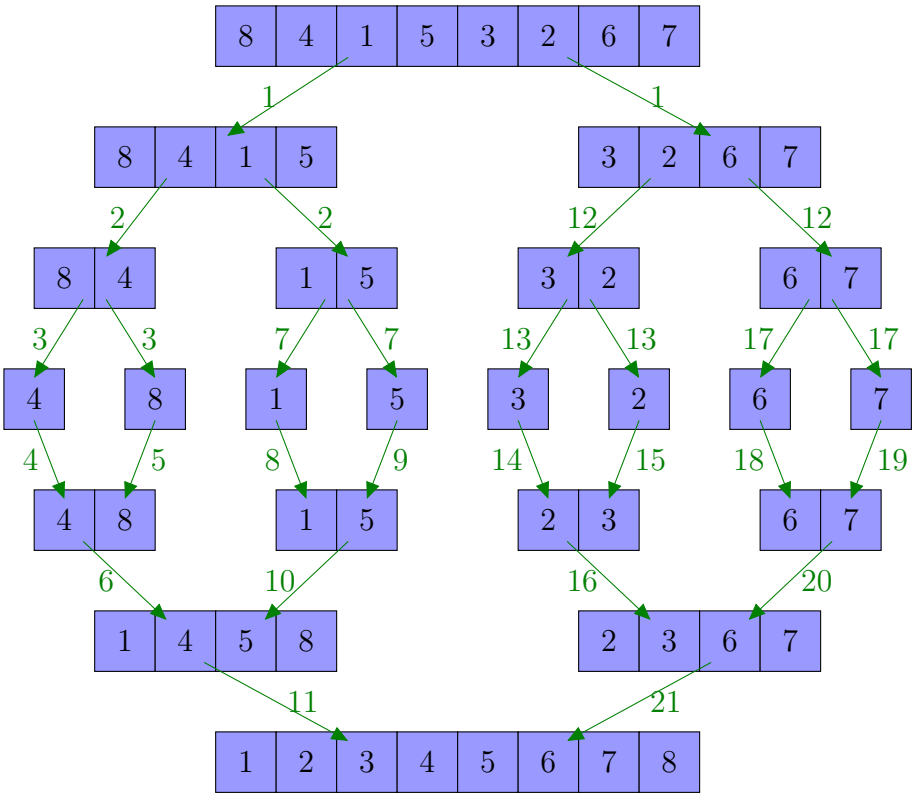
- **Algoritmus (D&C):**

  $MERGESORT(A[1,\ldots,n])$

  1. Ha $n = 1$ *return* A
  2. Rekurzívan alkalmazzuk a lista két felére: $MERGESORT(A[1,\ldots,\lceil n/2\rceil])$ és $MERGESORT(A[\lceil n/2\rceil + 1,\ldots,n])$
  3. A 2 listát ,,fésüljük össze"

Egy lehetséges megvalósítás (helyben rendezve) itt , és egy másik temp tömbbel itt.

Megoldás



1. Ábra.: A nyilakon a számok a rekurzív hívások sorrendjét jelölik.

Érdekesség: ha meg akarod tudni, hogyan működik a Divide & Conquer módszer a Fast Fourier transzformáció esetén, kattints a nyuszira: