Adatszerkezetek és algoritmusok

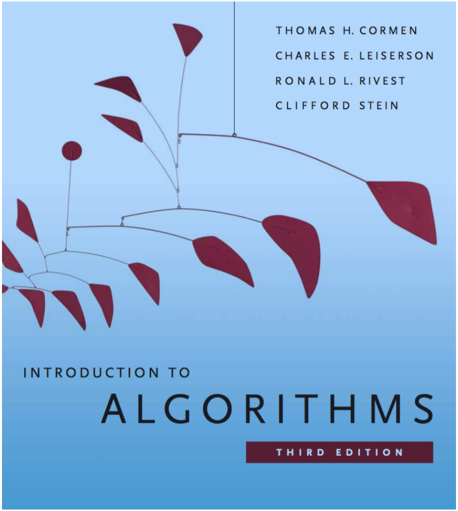
Horváth Géza

első előadás

Előadások témái

1. Az algoritmusokkal kapcsolatos alapfogalmak bevezetése egyszerű példákon keresztül.
2. Az algoritmusok futási idejének aszimptotikus korlátai.
3. Az adatszerkezetekkel kapcsolatos alapfogalmak. A halmaz, a multihalmaz és a tömb adatszerkezet bemutatása.
4. Az adatszerkezetek folytonos és szétszórt reprezentációja. A verem, a sor és a lista.
5. Táblázatok, önátrendező táblázatok, hash függvények és hash táblák, ütközéskezelés.
6. Fák, bináris fák, bináris keresőfák, bejárás, keresés, beszúrás, törlés.
7. Kiegyensúlyozott bináris keresőfák: AVL fák.
8. Piros-fekete fák.
9. B-fák.
10. Gráfok, bejárás, legrövidebb út megkeresése.
11. Párhuzamos algoritmusok.
12. Eldönthetőség és bonyolultság, a P és az NP problémaosztályok.
13. Lineáris idejű rendezés. Összefoglalás.

Szakirodalom: előadások + könyv



1+2+...+100 for ciklussal

Kimenet:

$$x = \sum_{i=1}^{100} i$$

Pszeudokód:

ÖSSZEADÁS()

1. x=0
2. for i=1 to 100
3. x=x+i
4. return x

C program:

```
#include <stdio.h>
int sum(){
    int i,x;
    x=0;
    for(i=1;i<=100;i++)
        x=x+i;
    return x;
}
int main(){
    printf("%d",sum());
}
```

A pszeudokód felépítése

ÖSSZEADÁS()

```
1 x=0
2 for i=1 to 100
3     x=x+i
4 return x
```

- Eltolással jelöljük a blokkokat, például a for ciklus magja csak a 3. sort tartalmazza.
- A használható ciklusok: for, while, és repeat-until.
- Feltételes utasítás: if-else.
- Megjegyzéseket a "//" után tudunk tenni.
- A részletek megtalálhatóak a könyv 20-22. oldalán a "Pseudocode conventions" részben.

1+2+...+100 for ciklussal

Kimenet:

$$x = \sum_{i=1}^{100} i$$

Pszeudokód:

ÖSSZEADÁS()

1. x=0
2. for i=1 to 100
3.   x=x+i
4. return x

Lépésszám: 100

1+2+...+n for ciklussal

Bemenet: $n \geq 1$

Kimenet:

$$x = \sum_{i=1}^{n} i$$

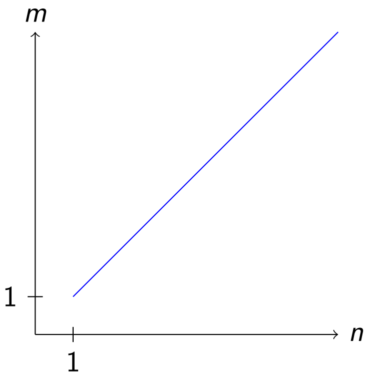
Pszeudokód:

ÖSSZEADÁS(N)

1. x=0
2. for i=1 to n
3. x=x+i
4. return x

Lépésszám: n

1+2+...+n for ciklussal



n – **bemenet mérete**

m – **futási idő**

A bemenet mérete

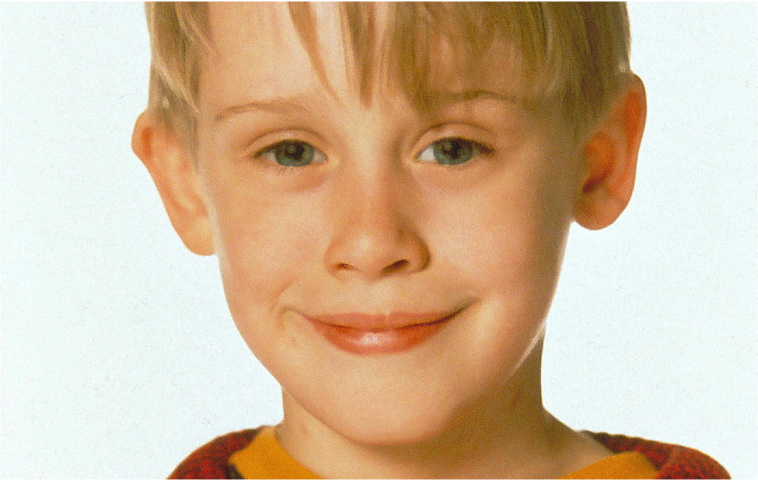
,,A **bemenet méretének** legjobb mértéke a vizsgált feladattól függ. Sok feladatra, mint például a rendezés vagy a diszkrét Fourier-transzformáltak kiszámítása, a legtermészetesebb mérték a bemenő elemek száma. Például rendezésnél a rendezendő tömb n elemszáma lehet a mérték. Sok egyéb feladatra, mint például két egész szám szorzása, a bemenet méretére a legjobb mérték a bemenet közönséges bináris jelölésben való ábrázolásához szükséges bitek teljes száma. Néha megfelelőbb a bemenet méretét egy szám helyett inkább kettővel leírni. Például, ha az algoritmus bemenete egy gráf, a bemenet méretét leírhatjuk a gráf éleinek és csúcsainak számával. Minden vizsgált feladatnál megadjuk, hogy a bemenet méretének melyik mértékét használjuk.” (könyv, 20-22. oldal)

A futási idő

,,Az algoritmusok **futási ideje** egy bizonyos bemenetre a végrehajtott alapműveletek vagy "lépések" száma. Kényelmes úgy definiálni a lépést, hogy minél inkább gépfüggetlen legyen. Egyelőre fogadjuk el a következőt: pszeudokódunk mindegyik sorának végrehajtásához állandó mennyiségű idő szükséges. Lehet, hogy az egyik sor tovább tart, mint a másik, de feltesszük, hogy az i-edik sor minden végrehajtása c_i ideig tart, ahol c_i állandó. Ez a nézőpont megfelel a RAM modellnek, és tükrözi azt is, ahogyan a pszeudokódot végrehajtaná a legtöbb jelenlegi számítógép."
(könyv, 20-22. oldal)

Carl Friedrich Gauss (1777-1855)

Carl Friedrich Gauss 8 évesen. . .



1+2+...+100 Gauss megoldása

$$\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & \dots & 46 & 47 & 48 & 49 & 50 \\
+ & + & + & + & + & \dots & + & + & + & + & + \\
100 & 99 & 98 & 97 & 96 & \dots & 55 & 54 & 53 & 52 & 51 \\
= & = & = & = & = & \dots & = & = & = & = & = \\
101 & 101 & 101 & 101 & 101 & \dots & 101 & 101 & 101 & 101 & 101
\end{array}$$

1+2+...+100 Gauss megoldása

$$
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & \dots & 46 & 47 & 48 & 49 & 50 \\
+ & + & + & + & + & \dots & + & + & + & + & + \\
100 & 99 & 98 & 97 & 96 & \dots & 55 & 54 & 53 & 52 & 51 \\
= & = & = & = & = & \dots & = & = & = & = & = \\
101 & 101 & 101 & 101 & 101 & \dots & 101 & 101 & 101 & 101 & 101
\end{array}
$$

$$x = \sum_{i=1}^{100} i = 50 * 101$$

1+2+...+n Gauss megoldása

1	2	3	4	5	...
+	+	+	+	+	...
n	n-1	n-2	n-3	n-4	...
=	=	=	=	=	...
n+1	n+1	n+1	n+1	n+1	...

$$x = \sum_{i=1}^{n} i = \frac{n}{2} * (n+1) = \frac{n*(n+1)}{2}$$

1+2+...+n Gauss megoldása

Bemenet: $n \geq 1$

Kimenet:

$$x = \sum_{i=1}^{n} i$$

Pszeudokód:

ÖSSZEADÁS(N)

1. return n*(n+1)/2

C program:

```
#include <stdio.h>
int sum(int n){
    return n*(n+1)/2;
}
int main(){
    printf("%d",sum(100));
}
```

1+2+...+n Gauss megoldása

Bemenet: $n \geq 1$

Kimenet:

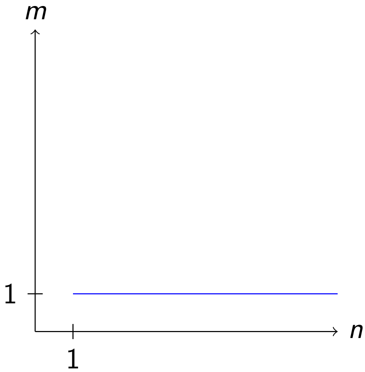
$$x = \sum_{i=1}^{n} i$$

Pszeudokód:
ÖSSZEADÁS(N)

1. return n*(n+1)/2

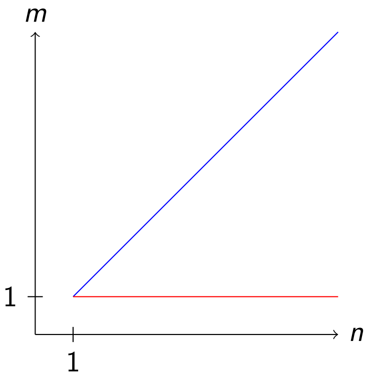
Lépésszám: 1

1+2+...+n Gauss megoldása



n – **bemenet mérete**
m – **futási idő**

1+2+...+n a futási idő összehasonlítása



n – **bemenet mérete**

m – **futási idő**

Faktoriális kiszámítása for ciklussal

Bemenet: $n \geq 1$

Kimenet: $x = n!$

Pszeudokód:

FAKTORIÁLIS(N)

1. x=1
2. for i=2 to n
3. x=x*i
4. return x

Lépésszám: n

C program:

```
#include <stdio.h>
int factorial(int n){
    int i,x;
    x=1;
    for(i=2;i<=n;i++)
        x=x*i;
    return x;
}
int main(){
    printf("%d",factorial(5));
}
```

Faktoriális kiszámítása rekurzívan

Bemenet: $n \geq 1$

Kimenet: $x = n!$

Pszeudokód:

FAKTORIÁLIS(N)

1. if n=1
2. return 1
3. else
4. return n*FACTORIAL(n-1)

Lépésszám: n

C program:

```
#include <stdio.h>
int factorial(int n){
    return n==1?1:n*factorial(n-1);
}
int main(){
    printf("%d",factorial(5));
}
```

Lineáris keresés

Bemenet: A[n], x

Kimenet: az x szám A vektorbeli pozíciója

Pszeudokód: gyakorlaton

Lépésszám: ???

C program:

```
#include <stdio.h>
int linear_search(int A[], int n, int x){
    int i;
    for(i=0;i<n;i++)
    if(A[i]==x)
        return i;
    return -1;
}
int main(){
    int A[5]={1,2,3,4,5};
    printf("%d",linear_search(A,5,3));
}
```

Lineáris keresés

Bemenet: A[n], x

Kimenet: az x szám A vektorbeli pozíciója

Pszeudokód: gyakorlaton

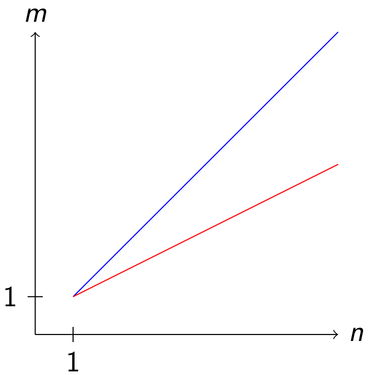
Lépésszám:

- legjobb eset: nem érdekes
- átlagos: $(n+1)/2$
- legrosszabb eset: n

C program:

```
#include <stdio.h>
int linear_search(int A[], int n, int x){
    int i;
    for(i=0;i<n;i++)
    if(A[i]==x)
        return i;
    return -1;
}
int main(){
    int A[5]={1,2,3,4,5};
    printf("%d",linear_search(A,5,3));
}
```

Lineáris keresés



n – **bemenet mérete**

m – **futási idő**

Bináris keresés

Bemenet: A[n], x

Kimenet: az x szám A vektorbeli pozíciója

Pszeudokód: gyakorlaton

C program: gyakorlaton

Lépésszám:

- legrosszabb eset: ???

Bináris keresés

Bemenet: A[n], x

Kimenet: az x szám A vektorbeli pozíciója

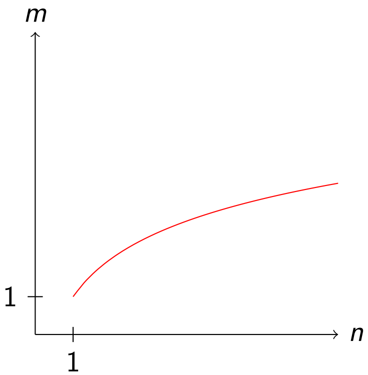
Pszeudokód: gyakorlaton

C program: gyakorlaton

Lépésszám:

- legrosszabb eset: $\log_2(n)+1$

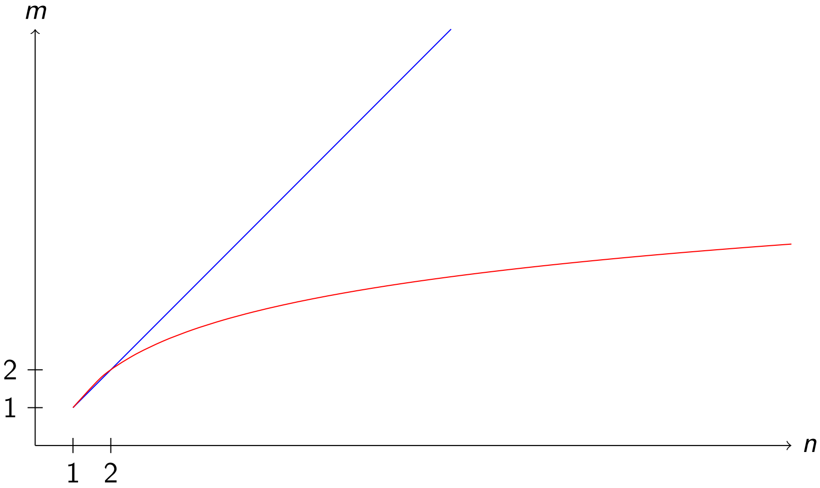
Bináris keresés



n – **bemenet mérete**

m – **futási idő**

Lineáris és bináris keresés összehasonlítása



n – **bemenet mérete**

m – **futási idő**

Irodalomjegyzék