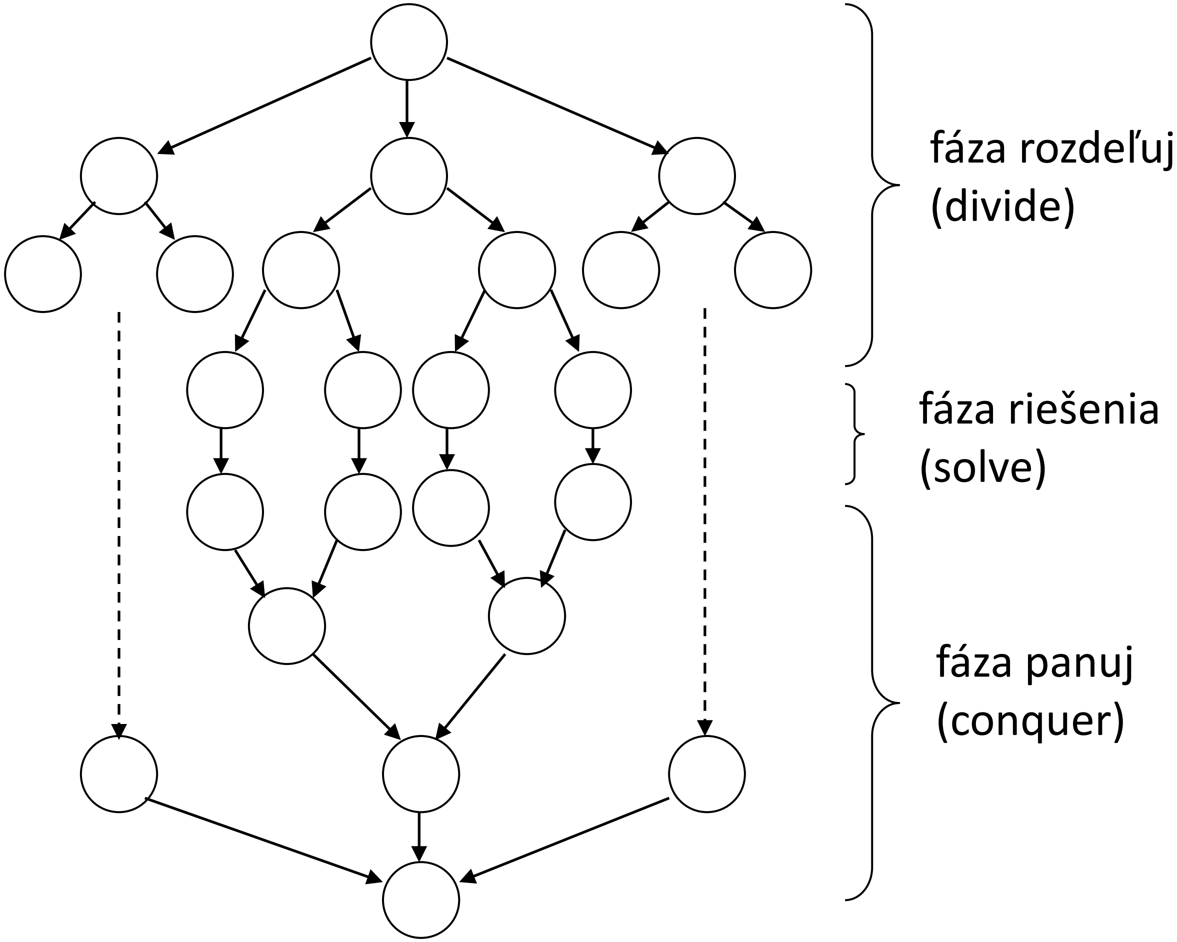
Rozdeľuj a panuj

Hľadanie maxima, druhého maxima

Triedenia

Násobenie matíc

Rozdeľuj a panuj (divide and conquer)



Hľadanie maxima

```
max = a[0];
for (int i = 1; i < n; i++) {
  if (a[i] > max) { max = a[i]; }
}
```

$T(n) = n - 1 = O(n)$

Maximum metódou rozdeľuj a panuj

```
int max(int z, int k) {
  int m1, m2, s;
  if (z == k) {
     return a[z];
  } else {
     s = (z + k) / 2;
     m1 = max(z, s);
     m2 = max(s + 1, k);
     if (m1 > m2) {
        return m1;
     } else {
        return m2;
     }
  }
}
```

Počet porovnaní

```
int max(int z, int k) {
  int m1, m2, s;
  if (z == k) {
     return a[z];
  } else {
     s = (z + k) / 2;
     m1 = max(z, s);
     m2 = max(s + 1, k);
     if (m1 > m2) {
        return m1;
     } else {
        return m2;
     }
  }
}
```

$T(1) = 0$

$T(n) = T\left(\frac{n}{2}\right) + T\left(n - \frac{n}{2}\right) + 1$

T(2) = T(1)+T(1)+1 = 1

T(3) = T(1)+T(2)+1 = 2

T(4) = T(2)+T(2)+1 = 3

T(5) = T(2)+T(3)+1 = 4

T(6) = T(3)+T(3)+1 = 5

T(7) = T(3)+T(4)+1 = 6

T(8) = T(4)+T(4)+1 = 7

Počet porovnaní

$T(1) = 0$
$T(n) = T\left(\frac{n}{2}\right) + T\left(n - \frac{n}{2}\right) + 1$

$T(2^k) = 2 \cdot T(2^{k-1}) + 1$
$T(2^{k-1}) = 2 \cdot T(2^{k-2}) + 1$
$T(2^{k-2}) = 2 \cdot T(2^{k-3}) + 1$

Pre $n = 2^k$

$$T(2^k) = 2 \cdot (2 \cdot T(2^{k-2}) + 1) + 1 = 2^2 T(2^{k-2}) + 2 + 1 =$$
$$= 2^2(2 \cdot T(2^{k-3}) + 1) + 2 + 1 = 2^3 T(2^{k-3}) + 2^2 + 2 + 1 = \cdots =$$
$$= 2^k T(2^{k-k}) + 2^{k-1} + 2^{k-2} + \cdots + 2^1 + 2^0 =$$
$$= 2^{k-1} + 2^{k-2} + \cdots + 2^1 + 2^0 =$$
$$= 2^0 \cdot \frac{2^k - 1}{2 - 1} = 2^k - 1 = n - 1 = O(n)$$

POČET POROVNANÍ

- $T(n) = n - 1 = O(n)$
  - rovnako ako priamym prehľadávaním
- $S(n) = O(n + \log n)$
  - veľkosť poľa + veľkosť zásobníka (počet rekurzívnych vnorení)
  - horšie ako pri priamom prehľadávaní

Druhé maximum

- nech a je pole hodnôt veľkosti n, v ktorom hľadáme druhé maximum

```
imax = 0;
for (int i = 1; i < n; i++) {
   if (a[imax] < a[i]) { imax = i; }
}

p = a[n];
a[n] = a[imax];
a[imax] = p;

imax = 1;
for (int i = 1; i < n - 1 ; i++) {
   if (a[imax] < a[i]) { imax = i; }
}
```

- $\mathrm{T}(n) = (n-1)+(n-2) = 2n-3 = \mathrm{O}(2n)$

Druhé maximum metódou rozdeľuj a panuj

pseudokód:

```
metóda druhe_max (int z, int k) {
    if (k - z == 0) {                                      //dĺžka 1
        return (a[z],-∞);
   } else {
    if (k - z == 1) {                                      //dĺžka 2
        if (a[z]>=a[k])  return (a[z],a[k]);
        else return (a[k],a[z]);
    } else {                                               //dĺžka > 2
        s = (z + k) / 2;
        (m11, m21) = druhe_max(z,s);
        (m12, m22) = druhe_max(s+1,k);
        if (m11>=m12) {
            if (m21>=m12) return (m11,m21);
            else return (m11,m12);
        } else {
            if (m11>=m22) return (m12,m11);
            else return (m12,m22);
        }
    }
}
```

DRUHÉ MAXIMUM METÓDOU ROZDEĽUJ A PANUJ

- rekurzívne sa určí prvé a druhé maximum z prvej a druhej polovice poľa
  - *m11* je prvé maximum z prvej polovice poľa,
  - *m12* je prvé maximum z druhej polovice poľa
  - *m21* je druhé maximum z prvej polovice poľa,
  - *m22* je druhé maximum z druhej polovice poľa

$m11 \geq m21$ $m12 \geq m22$

- vo fáze „panuj" zo **štyroch** hodnôt (*m11*, *m21*, *m12*, *m22*) na **dve** porovnania určí *max1*, *max2*

$m11 \geq m12$

$m21 \geq m12$ $m11 \geq m22$

$(m11, m21)$ $(m11, m12)$ $(m12, m11)$ $(m12, m22)$

Časová výpočtová zložitosť

$T(1) = 0$

$T(2) = 1$

$T(n) = T(\ n / 2) + T(n - n / 2) + 2$

pre $n = 2^k$

Časová výpočtová zložitosť

$T(2) = 1$

$T(n) = T(n / 2) + T(n - n / 2) + 2$

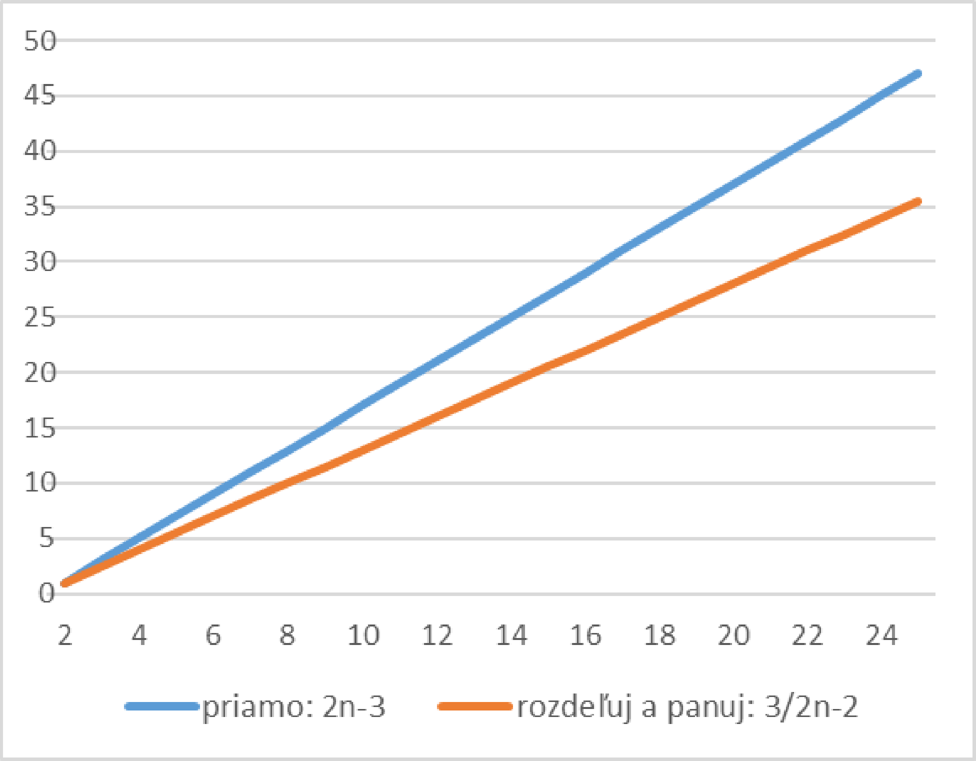
pre $n = 2^k$

$$
\begin{aligned}
T(2^k) &= 2\ T(2^{k-1}) + 2 = 2\ (2\ T(2^{k-2})+2) + 2 = \\
&= 2^2\ T(2^{k-2})+ 2^2 + 2 = 2^2\ (2\ T(2^{k-3})+2) + 2^2 + 2 = \\
&= 2^3\ T(2^{k-3})+ 2^3 + 2^2 + 2 = \\
&= \dots = \\
&= 2^{k-1}\ T(2^{k-(k-1)})+ 2^{k-1} + 2^{k-2} + \dots + 2 = \\
&= 2^{k-1} + 2^{k-1} + 2^{k-2} + \dots + 2 = \\
&= 2^{k-1} + 2\ (2^{k-1} -1) = 2^{k-1} + 2^k - 2 = 3/2\ .\ 2^k - 2 = \\
&=3/2\ n - 2 = O(3/2\ n)
\end{aligned}
$$

Porovnanie

- Hľadanie 2. maxima priamo: $T(n) = 2n - 3 = O(2n)$
- Hľadanie 2. maxima rozdeľuj a panuj:

  $T(n) = 3/2\ n - 2 = O(3/2\ n)$



Triedenia metódou rozdeľuj a panuj

- úloha sa rozdelí na nezávislé podúlohy menšieho rozsahu
- z výsledkov podúloh sa vytvorí riešenie celej úlohy
- quicksort, mergesort
- zložitosti $O(n \log n)$ – lepšie ako priame metódy so zložitosťou $O(n^2)$

Násobenie matíc

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix}$$

$$C = A.B$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

počet násobení $M(n)=n^3$

počet sčítaní $S(n)=n^2(n-1)$

Strassenov algoritmus

- Pre matice rozmeru 2×2 platí:

$$\boxed{\begin{aligned}
u_0 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\
u_1 &= (a_{21} + a_{22})\, b_{11} \\
u_2 &= a_{11}(b_{12} - b_{22}) \\
u_3 &= a_{22}\, (-b_{11} + b_{21}) \\
u_4 &= (a_{11} + a_{12})\, b_{22} \\
u_5 &= (-a_{11} + a_{21})(b_{11} + b_{12}) \\
u_6 &= (a_{12} - a_{22})(b_{21} + b_{22})
\end{aligned}}$$

$$\boxed{\begin{aligned}
c_{11} &= u_0 + u_3 - u_4 + u_6 \\
c_{21} &= u_1 + u_3 \\
c_{12} &= u_2 + u_4 \\
c_{22} &= u_0 + u_2 - u_1 + u_5
\end{aligned}}$$

$\mathrm{M}(n) = 7$

$\mathrm{S}(n) = 18$

Strassenov algoritmus

- Pre väčšie matice rozmeru $n=2^k$
  - rozdelíme na podmatice A_{11}, A_{12}, A_{21}, A_{22}, B_{11}, B_{12}, B_{21}, B_{22}
  - vypočítame pomocné matice U_0, U_1, ..., U_6 a z nich časti výslednej matice C_{11}, C_{12}, C_{21}, C_{22} tak, že všetky operácie sú **maticové**

$$\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array} \qquad \begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}$$

Strassenov algoritmus

- T(n) pre n=2^k
  - Počet násobení

$$
\begin{aligned}
M(1) &= 1 \\
M(2^k) &= 7\,M(2^{k-1}) = \\
&= 7^2\,M(2^{k-2}) = \\
&= 7^3\,M(2^{k-3}) = \\
&\ldots \\
&= 7^k\,M(1) = \\
&= 7^k = \\
&= 7^{\log n} = \ n^{\log 7} = \\
&= n^{2,81}
\end{aligned}
$$

Strassenov algoritmus

T(n) pre n=2^k

Počet sčítaní:

S(2) = 18

S(n) = 18 $(n/2)^2$ + 7 S(n/2)

Dôkaz indukciou: Ukážeme, že pre všetky prirodzené k platí:

$S(2^k) = 6\,(7^k - 4^k)$

Platí pre k=1

Predp., že platí pre nejaké $k \geq 1$, potom platí aj pre k+1

$$
\begin{aligned}
S(2^{k+1}) &= 18\ .\ 2^{2k} + 7.\ S(2^k) = \\
&= 18\ .\ 4^k + 7.\ 6\ (7^k - 4^k) = \\
&= 3\ .\ 6\ .\ 4^k - 7.\ 6\ .4^k + 7.\ 6\ .7^k = \\
&= -\ 4.\ 6\ .4^k + 7.\ 6\ .7^k = \\
&= 6\ (7.7^k - 4.4^k\ ) = \\
&= 6\ (7^{k+1} - 4^{k+1}\ )
\end{aligned}
$$

Strassenov algoritmus

- T(n) pre n=2^k
  - Počet sčítaní

$$\begin{aligned} T(n) &= 6\,(7^k - 4^k) = \\ &= 6\,(7^{\log n} - 4^{\log n}) = \\ &= 6\,(n^{\log 7} - n^{\log 4}) = \\ &= 6\,(n^{2,81} - n^2) = \\ &= O(n^{2,81}) \end{aligned}$$