Merge sort—hlavný program

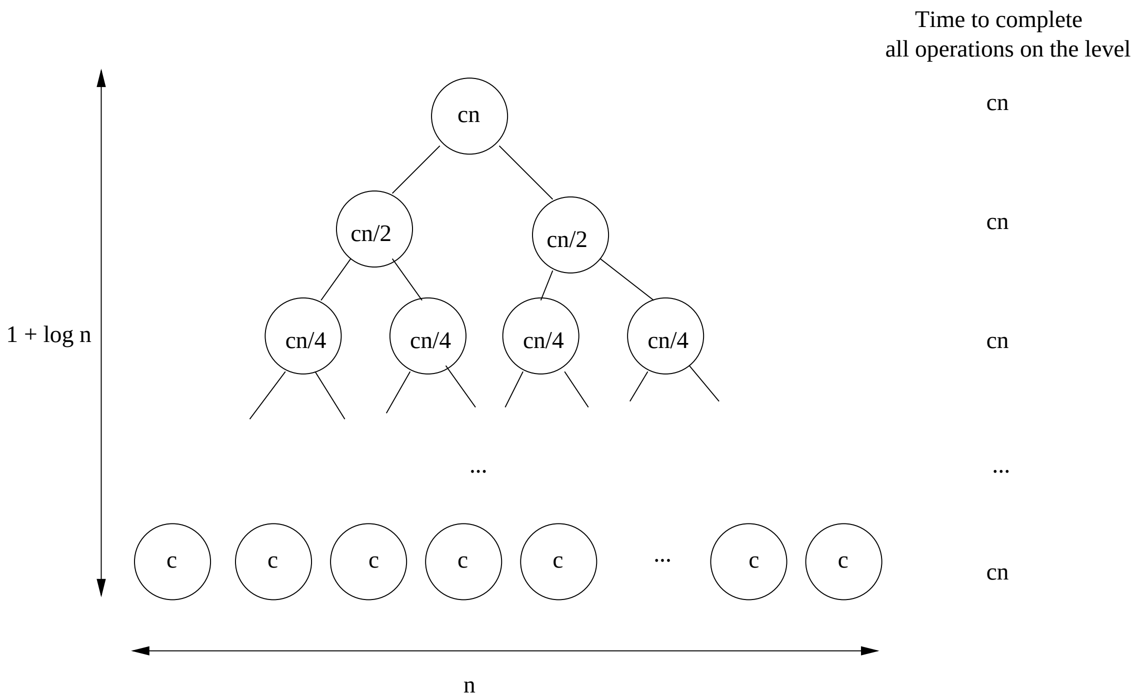
```
// sort sequence A[l..r]
function merge_sort(l,r)
  // base case - 1 element is always sorted
  if (l=r) then return;
  m=(l+r) div 2;
  // we need to sort sequences l..m, m+1..r
  merge_sort(l,m);
  merge_sort(m+1,r);
  // and finally merge two sorted sequences
  merge(l,m,r);
```

Merge sort—merge

```
//merge two sorted sequences l..m, m+1..r
function merge(l,m,r)
  copy A[l..m] to L; L[m-l+2]:=infinity;
  copy A[m+1..r] to R; R[r-m+1]:=infinity;

  i:=1; j:=1; k:=l;
  while (L[i]<infinity or R[i]<infinity) do
    if L[i]<=R[j] then
      A[k]:=L[i];
      i:=i+1; k:=k+1;
    else
      A[k]:=R[j];
      j:=j+1; k:=k+1;
```

Merge sort—časová zložitosť



Časová zložitosť: $O(n \log n)$

Rozdeľuj a panuj

- **Rozdeľuj.** Rozdeľ problém na niekoľko menších podproblémov.
- **Panuj.** Každý podproblém vyrieš samostatne rekurzívnym volaním. Ak sú podproblémy dostatočne malé, vyrieš ich priamočiaro.
- **Kombinuj.** Skombinuj riešenia menších podproblémov do riešenia pôvodného veľkého problému.

Príklady metódy rozdeľuj a panuj

- **Quick sort.** Nie nutne "vyvážené" rozdelenie na podproblémy.
  V zlom prípade: $\Theta(n^2)$ Priemerná zložitosť: $\Theta(n \log n)$
  Existuje (nepraktická) $\Theta(n \log n)$ verzia.

- **Motivačný problém - Riešenie 3.**

- **Binárne vyhľadávanie.** Dané je utriedené pole $A[1..n]$
  Nájdite v poli A číslo x
  - Pozri sa na prostredný prvok m poľa A
  - Ak $x = m$, sme hotoví
    Ak $x < m$, hľadaj rekurzívne v ľavej časti
    Ak $x > m$, hľadaj rekurzívne v pravej časti

  Problém delíme na **jeden podproblém** polovičnej veľkosti

Rozdeľuj a panuj: Potenciálne problémy

- Príliš veľa podproblémov alebo príliš veľké podproblémy
  $\Rightarrow$ príliš veľa času na rekurzívne volania
- Príliš zložitá kombinácia
  $\Rightarrow$ kombinácia dominuje výpočet
- Podproblémy príliš rôznych veľkostí $\Rightarrow$ veľké podproblémy dominujú výpočet

Násobenie veľkých čísel

Úloha: Dané sú dve polia $X[1..n]$ a $Y[1..n]$, ktoré reprezentujú dve čísla pomocou ich desatinného zápisu:
$x = \sum_{i=1}^{n} X[i] \cdot 10^{i-1}, \quad y = \sum_{i=1}^{n} Y[i] \cdot 10^{i-1}$.
Vypočítajte $z = xy$.

Poznámka: Skutočná implementácia by použila inú reprezentáciu ako desatinný zápis: rozdelíme čísla na 64-bitové slová

Základnoškolský algoritmus

```
       9 8 1
x    1 2 3 4
-------------
     3 9 2 4
   2 9 4 3
 1 9 6 2
 9 8 1
-------------
1 2 1 0 5 5 4
```

Časová zložitosť: $\Theta(n^2)$

Rozdeľuj a panuj: prvý pokus

- Rozdeľ čísla na dve polovice veľkosti $k = n/2$:
  $x = a \cdot 10^k + b \quad y = c \cdot 10^k + d$
- $xy = (a \cdot 10^k + b)(c \cdot 10^k + d) = ac \cdot 10^{2k} + (ad + bc) \cdot 10^k + bd$
  - 4 násobenia veľkosti $n/2$
  - 3 sčítania veľkosti $2n$
- Násobenie počítaj **rekurzívne**
  Sčítanie môžeme urobiť v čase $\Theta(n)$

Prvý pokus: Časová zložitosť

- $T(n)$: časová zložitosť pre vstup veľkosti n
- $T(n) = 4T(n/2) + \Theta(n)$
- Koľko listov bude mať strom rekurzie?
  Na každej úrovni sa počet vrcholov zoštvornásobí $\Rightarrow 4^{\log_2 n} = n^2$
- **Už len na najnižšej úrovni potrebujeme čas $\Omega(n^2)$**

Rozdeľuj a panuj: druhý pokus

- $xy = ac \cdot 10^{2k} + (ad + bc) \cdot 10^k + bd$
- **Násobenia sú drahé operácie!**
- Vieme zredukovať na **3 násobenia**:

$$\begin{aligned} xy &= (a \cdot 10^k + b)(c \cdot 10^k + d) \\ &= ac \cdot 10^{2k} + (ad + bc) \cdot 10^k + bd \\ &= ac \cdot 10^{2k} + ((a + b)(c + d) - ac - bd) \cdot 10^k + bd \end{aligned}$$

- 3 násobenia (ac, bd, $(a + b)(c + d)$), 6 sčítaní
- $T(n) = 3T(n/2) + \Theta(n)$
- **Potrebujeme metódy na analýzu rekurencií!**

Hlavná veta (master theorem):

Nech $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$. Nech $k = \log_b a$. Potom:

1. Ak $\boxed{f(n) \in O(n^{k-\varepsilon})}$ pre niektoré $\varepsilon > 0$, potom $\boxed{T(n) \in \Theta(n^k)}$.
2. Ak $\boxed{f(n) \in \Theta(n^k)}$, potom $\boxed{T(n) \in \Theta(f(n) \log n)}$.
3. Ak $\boxed{f(n) \in \Omega(n^{k+\varepsilon})}$ pre niektoré $\varepsilon > 0$ a platí podmienka regularity, potom $\boxed{T(n) \in \Theta(f(n))}$.

Podmienka regularity: Existuje $c < 1$ také, že pre všetky dostatočne veľké n platí $af(n/b) \leq cf(n)$.

Pre náš príklad: $T(n) = 3T(n/2) + \Theta(n)$

$k = \log_2 3 = 1.5849...$, $\Theta(n) \in O(n^{1.5849...-\varepsilon})$

$\Rightarrow T(n) \in \Theta(n^{1.5849...})$

Hlavná veta (master theorem):

Nech $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$. Nech $k = \log_b a$. Potom:

1. Ak $\boxed{f(n) \in O(n^{k-\varepsilon})}$ pre niektoré $\varepsilon > 0$, potom $\boxed{T(n) \in \Theta(n^k)}$.
2. Ak $\boxed{f(n) \in \Theta(n^k)}$, potom $\boxed{T(n) \in \Theta(f(n) \log n)}$.
3. Ak $\boxed{f(n) \in \Omega(n^{k+\varepsilon})}$ pre niektoré $\varepsilon > 0$ a platí podmienka regularity, potom $\boxed{T(n) \in \Theta(f(n))}$.

Podmienka regularity: Existuje $c < 1$ také, že pre všetky dostatočne veľké n platí $af(n/b) \leq cf(n)$.

Merge sort: $T(n) = 2T(n/2) + \Theta(n)$

$k = \log_2 2 = 1$, $\Theta(n) \in \Theta(n^1)$

$\Rightarrow T(n) \in \Theta(n \log n)$

Hlavná veta (master theorem):

Nech $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$. Nech $k = \log_b a$. Potom:

1. Ak $\boxed{f(n) \in O(n^{k-\varepsilon})}$ pre niektoré $\varepsilon > 0$, potom $\boxed{T(n) \in \Theta(n^k)}$.

2. Ak $\boxed{f(n) \in \Theta(n^k)}$, potom $\boxed{T(n) \in \Theta(f(n)\log n)}$.

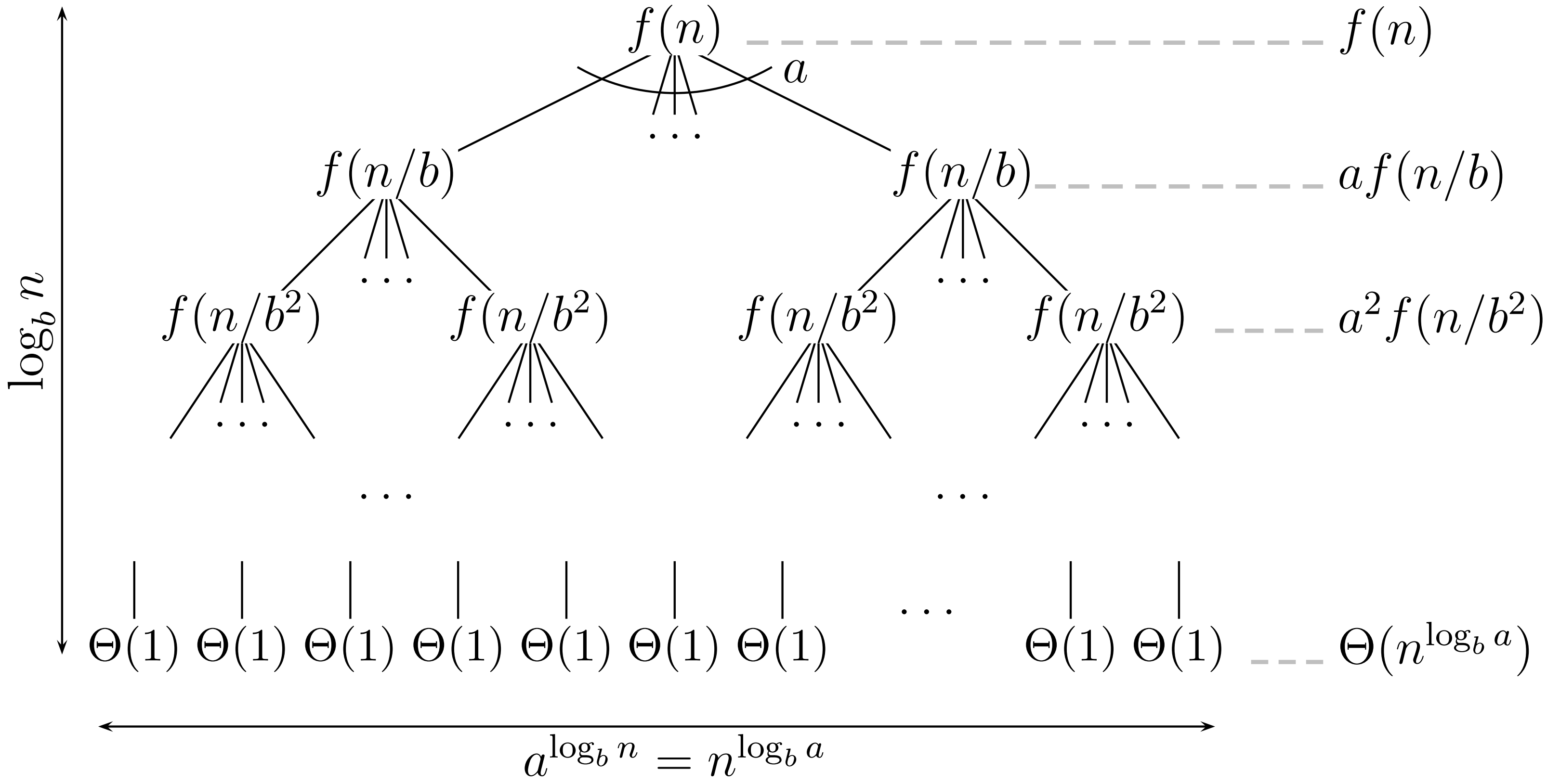
3. Ak $\boxed{f(n) \in \Omega(n^{k+\varepsilon})}$ pre niektoré $\varepsilon > 0$ a platí podmienka regularity, potom $\boxed{T(n) \in \Theta(f(n))}$.

Podmienka regularity: Existuje $c < 1$ také, že pre všetky dostatočne veľké n platí $af(n/b) \leq cf(n)$.

Násobenie čísel, prvý pokus: $T(n) = 4T(n/2) + \Theta(n)$

$k = \log_2 4 = 2$, $\Theta(n) \in O(n^{2-\varepsilon})$

$\Rightarrow T(n) \in \Theta(n^2)$


$f(n)$
a
$f(n/b)$
$f(n/b)$
$f(n/b^2)$
$f(n/b^2)$
$f(n/b^2)$
$f(n/b^2)$
$\Theta(1)$
$f(n)$
$af(n/b)$
$a^2 f(n/b^2)$
$\Theta(n^{\log_b a})$
$\log_b n$
$a^{\log_b n} = n^{\log_b a}$

Zhrnutie

- Metóda rozdeľuj a panuj
- Merge Sort, Quick Sort, Binárne vyhľadávanie
- Je dôležité správne vytipovať operáciu, ktorá bude dominovať celému výpočtu a tú optimalizovať
- Násobenie veľkých čísel v čase $\Theta(n^{1.5849\ldots})$
- Časová zložitosť algoritmov rozdeľuj a panuj:
  $T(n) = aT(n/b) + f(n)$
- Hlavná veta (master theorem) na riešenie takýchto rekurencií
- Dôkaz hlavnej vety na budúce