CZ.1.07/2.2.00/28.0041
Centrum interaktivních a multimediálních studijních opor pro inovaci výuky a efektivní učení

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Časť I

Návrh a analýza algoritmov

1 Rozdeľ a panuj

- Princípy
  - Binárne vyhľadávanie
  - Faktoriál
- Analýza rekurzívnych algoritmov
- Triedenie spájaním
- Maximálny a minimálny prvok
- Problém inverzií

Návrh algoritmov

ideálny svet návod (*algoritmus*) „ako konštruovať algoritmy“

realita osvedčené postupy

- iteratívny prístup
- „rozdeľ a panuj“ (*divide et impera*)
- dynamické programovanie
- hladové techniky
- heuristiky
- náhodnostné techniky
- aproximatívne techniky
- ......

Iteratívny prístup

- lineárny postup
- príklady: algoritmy pre vyhľadávanie prvku, algoritmus triedenia vkladaním

1 Rozdeľ a panuj

- Princípy
  - Binárne vyhľadávanie
  - Faktoriál
- Analýza rekurzívnych algoritmov
- Triedenie spájaním
- Maximálny a minimálny prvok
- Problém inverzií

Rozdeľ a panuj - princípy

Rozdeľ *(divide)* problém na podproblémy, ktoré majú menšiu veľkosť ako pôvodný problém.

Vyrieš *(conquer)* rekurzívne podproblémy. Ak veľkosť podproblému je malá, použi priame riešenie.

Skombinuj *(combine)* riešenie podproblémov a vyrieš pôvodný problém.

Vyhľadávanie prvku v utriedenej postupnosti

Vstup postupnosť prvkov $(a_1, \ldots, a_n)$ taká, že $a_1 < a_2 < \ldots < a_n$, prvok x

Výstup index i taký, že $a_i = x$, resp. hodnota $\textsc{No}$ ak prvok x sa v postupnosti nevyskytuje

- porovnáme prvok x s prvkom $A[q]$
- ak $A[q] = x$, našli sme hľadaný index
- ak $A[q] > x$, tak postupnosť $A[1 \ldots q]$ určite neobsahuje prvok x a v ďalšom výpočte stačí hľadať prvok x v postupnosti $A[q + 1 \ldots n]$
- ak $A[q] < x$, tak analogicky stačí hľadať prvok x v postupnosti $A[1 \ldots q - 1]$

Vyhľadávanie prvku v utriedenej postupnosti

Vstup postupnosť prvkov $(a_1, \ldots, a_n)$ taká, že $a_1 < a_2 < \ldots < a_n$, prvok x

Výstup index i taký, že $a_i = x$, resp. hodnota No ak prvok x sa v postupnosti nevyskytuje

- porovnáme prvok x s prvkom $A[q]$
- ak $A[q] = x$, našli sme hľadaný index
- ak $A[q] > x$, tak postupnosť $A[1 \ldots q]$ určite neobsahuje prvok x a v ďalšom výpočte stačí hľadať prvok x v postupnosti $A[q + 1 \ldots n]$
- ak $A[q] < x$, tak analogicky stačí hľadať prvok x v postupnosti $A[1 \ldots q - 1]$
- index q volíme tak, aby postupnosti $A[q + 1 \ldots n]$ a $A[1 \ldots q - 1]$ mali cca rovnakú dĺžku
- ak postupnosť $A[q + 1 \ldots n]$ resp. $A[1 \ldots q - 1]$ je dlhá, tak pre hľadanie prvku x použijeme ten istý princíp
- ak postupnosť je krátka, prvok x vyhľadáme priamo (napr. Linear Search)

Vyhľadávanie prvku v utriedenej postupnosti

Rozdeľ porovnaj $A[q]$ a x, podľa výsledku vyber postupnosť polovičnej dĺžky 5(prípadne skonči)

Vyrieš ak vybraná postupnosť má dĺžku > 1 tak rekurzívne (*tým istým postupom*) nájdi prvok x vo vybranej postupnosti

Skombinuj vráť odpoveď

Binárne vyhľadávanie - iteratívne

BINARY_SEARCH(A, n, x)

Vstup a Výstup rovnaké ako pre LINEAR SEARCH

$p \leftarrow 1$
$r \leftarrow n$
while $p \leq r$ **do**
    $q \leftarrow \lfloor (p + r)/2 \rfloor$
    if $A[q] = x$ **then return** q **fi**
    if $A[q] > x$ **then** $r \leftarrow q - 1$ **fi**
    if $A[q] < x$ **then** $p \leftarrow q + 1$ **fi od**
return NO

Binárne vyhľadávanie - korektnosť

Invariant cyklu

Na začiatku každej iterácie **while** cyklu platí, že ak x sa vyskytuje v poli A, tak sa vyskytuje (aj) v postupnosti $A[p \dots r]$

Inicializácia: Na začiatku je $p = 1$ a $q = n$ a preto invariant platí.

Iterácia: Ak $A[p \dots r]$ a $A[q] > x$, tak nutne musí x obsahovať $A[q + 1 \dots n]$ (symetricky pre $A[q] < x$).

Ukončenie: Cyklus skončí nájdením prvku x alebo ak $p > r$. V druhom prípade z platnosti invariantu vyplýva, že A nemôže obsahovať prvok x.

Binárne vyhľadávanie - rekurzívne

RECURSIVE_BINARY_SEARCH(A, p, r, x)

Vstup a Výstup rovnaké ako pre LINEAR SEARCH, p, r vymedzujú postupnosť, v ktorej hľadáme prvok x

if $p > r$ **then return** NO **else**
    $q \leftarrow \lfloor (p + r)/2 \rfloor$
    if $A[q] = x$ **then return** q **fi**
    if $A[q] > x$ **then return**
        RECURSIVE_BINARY_SEARCH$(A, p, q - 1, x)$ **fi**
    if $A[q] < x$ **then return**
        RECURSIVE_BINARY_SEARCH$(A, q + 1, r, x)$ **fi**
fi

iniciálne volanie je RECURSIVE_BINARY_SEARCH$(A, 1, n, x)$

Binárne vyhľadávanie - zložitosť

- veľkosť postupnosti, v ktorej hľadáme prvok x, sa po každej iterácii cyklu (*iteratívny prístup*) resp. po každom rekurzívnom volaní zmenší na polovicu
- presnejšie: ak na začiatku iterácie má postupnosť dĺžku s, tak na začiatku nasledujúcej iterácie má dĺžku $\lfloor s/2 \rfloor$ alebo $s/2 - 1$ podľa toho, či s je sudé alebo liché a podľa toho, či $A[q]$ je väčšie alebo menšie ako x
- výpočet končí keď dĺžka postupnosti klesne na 1
- počet iterácií je $\lfloor \log n \rfloor + 1$
- celková zložitosť je $\mathcal{O}(\log n)$

Faktoriál

$\textsc{Factorial}(n)$

if $n = 0$ **then return** 1
else return $n \cdot \textsc{Factorial}(n-1)$ **fi**

$\textsc{Bad Factorial}(n)$

if $n = 0$ **then return** 1
else return $\frac{\textsc{Factorial}(n+1)}{n+1}$ **fi**

1 Rozdeľ a panuj

- Princípy
  - Binárne vyhľadávanie
  - Faktoriál
- Analýza rekurzívnych algoritmov
- Triedenie spájaním
- Maximálny a minimálny prvok
- Problém inverzií

Zložitosť rekurzívneho algoritmu

- ak program obsahuje rekurzívne volanie seba samého, jeho zložitosť obvykle vyjadrujeme pomocou *rekurentnej rovnice*, ktorá vyjadruje zložitosť výpočtu na vstupe veľkosti n pomocou zložitosti výpočtu na menších vstupoch
- na riešenie rekurentnej rovnice použijeme štandartné nástroje

Rozdeľ a panuj - princípy

Rozdeľ problém na podproblémy, ktoré majú menšiu veľkosť ako pôvodný problém.

Vyrieš rekurzívne podproblémy. Ak veľkosť podproblému je malá, použi priame riešenie.

Skombinuj riešenie podproblémov a vyrieš pôvodný problém.

Rozdeľ a panuj - zložitosť

- označme $T(n)$ časovú zložitosť výpočtu na vstupe dĺžky n
- ak veľkosť problému je dostatočne malá, $n \leq c$ pre nejakú konštantu c, tak priame riešenie problému si vyžiada konštantný čas $\Theta(1)$
- predpokladajme, že problém rozdelíme na a podproblémov z ktorých každý má veľkosť $1/b$ veľkosti pôvodného problému
- riešenie každého podproblému si vyžiada čas $T(n/b)$
- označme $D(n)$ čas potrebný na konštrukciu podproblémov a $C(n)$ čas potrebný na skombinovanie riešení podproblémov a nájdenie riešenia pôvodného problému

$$T(n) = \begin{cases} \Theta(1) & \text{ak } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{inak} \end{cases}$$

Riešenie rekurentných rovníc

Substitučná metóda „uhádneme" riešenie a dokážeme jeho správnosť matematickou indukciou.

Metóda rekurzívneho stromu konštruujeme strom, ktorého vrcholy vyjadrujú zložitosť jednotlivých rekurzívnych volaní, výslednú zložitosť vypočítame sumáciou.

Hlavná metóda (*master method*) „vzorec" pre riešenie rovnice tvaru $T(n) = aT(n/b) + f(n)$.

Substitučná metóda

1. „uhádni" riešenie
2. matematickou indukciou dokáž jeho korektnosť

Príklad

$$T(n) = \begin{cases} 1 & \text{pre } n = 1 \\ 2T(\lfloor n/2 \rfloor) + n & \text{inak} \end{cases}$$

1. $T(n) = \mathcal{O}(n \log n)$
2. indukciou dokážeme, že $T(n) \leq cn \log n$ pre vhodne zvolenú konštantu c

Substitučná metóda - príklad

Indukčný krok Predpokladajme, že tvrdenie platí pre všetky $m \leq n$, tj. špeciálne pre $m = \lfloor n/2 \rfloor$ platí $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)$. Potom

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \log(\lfloor n/2 \rfloor)) + n \\ &\leq cn \log(n/2) + n \\ &= cn \log n - cn \log 2 + n \\ &= cn \log n - cn + n \\ &\leq cnlogn \end{aligned}$$

Substitučná metóda - príklad

Indukčný základ $n = 1$
potrebujeme dokázať $T(1) \leq c1 \log 1 = 0$, čo je ale v spore s tým, že $T(1) = 1$

Substitučná metóda - príklad

Indukčný základ $n = 1$: spor s $T(1) = 1$

Indukčný základ $n = 2$ a $n = 3$

- zo znalosti $T(1) = 1$ odvodíme, že $T(2) = 4$ a $T(3) = 5$
- zvolíme konštantu $c \geq 1$ tak, aby platilo $T(n) \leq cn \log n$
- ak zvolíme $c \geq 2$, tak platí $T(2) \leq c2 \log 2$ aj $T(3) \leq c3 \log 3$
- platí indukčný základ

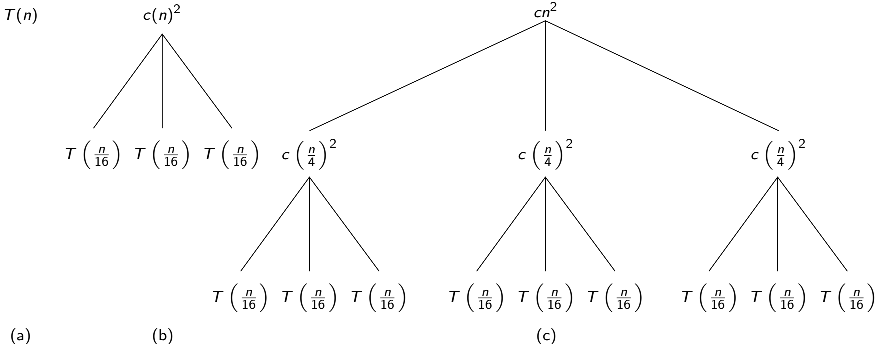
Metóda rekurzívneho stromu

- konštruujeme strom, ktorého vrcholy vyjadrujú zložitosť jednotlivých rekurzívnych volaní
- vypočítame súčet zložitostí na každej úrovni stromu
- sčítaním jednotlivých úrovní dostávame výseldnú zložitosť

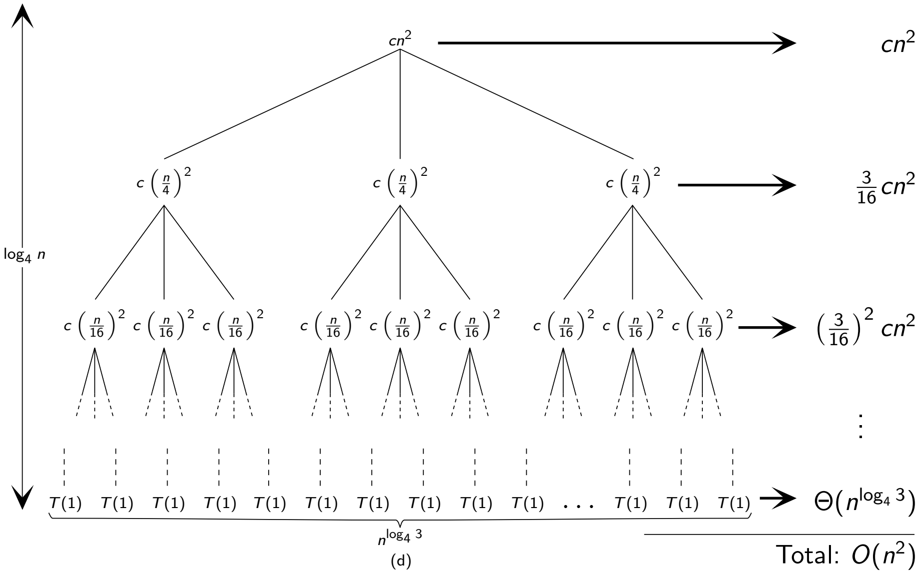
Príklad

$$T(n) = \begin{cases} 1 & \text{pre } n = 1 \\ 3T(\lfloor n/4 \rfloor) + \Theta(n^2) & \text{inak} \end{cases}$$

Metóda rekurzívneho stromu - príklad



Metóda rekurzívneho stromu - príklad



Metóda rekurzívneho stromu - príklad

- hĺbka stromu: $n = 1$ práve ak $n/4^i = 1$, tj. ekvivalentne ak $i = \log_4 n$
- zložitosť na úrovni i je pre $i = 0, 1, \ldots, \log_4 n - 1$ rovná $3^i c(n/4^i)^2 = (3/16)^i cn^2$
- úroveň $i = \log_4 n$ má $3^{\log_4 n} = n^{\log_4 3}$ vrcholov, z ktorých každý má zložitosť $T(1)$; zložitosť na úrovni $i = \log_4 n$ je preto $\Theta(n^{\log_4 3})$
- sumáciou cez všetky úrovne dostávame

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \cdots + (\frac{3}{16})^{\log_4 n - 1}cn^2 + \Theta(n^{\log_4 3})$$

Metóda rekurzívneho stromu - príklad

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1 - (3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= \mathcal{O}(n^2)
\end{aligned}
$$

hodnotu $T(n) = \mathcal{O}(n^2)$ použijeme ako odhad pre substitučnú metódu

Master method

Nech $a \geq 1$ a $b > 1$ sú konštanty, $f(n)$ je funkcia a nech $T(n)$ je definovaná na nezáporných číslach rekurentnou rovnicou

$$T(n) = \begin{cases} \Theta(1) & \text{pre } n = 1 \\ aT(n/b) + f(n) & \text{inak} \end{cases}$$

Potom platí

1. ak $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ pre nejakú konštantnu $\epsilon > 0$, tak $T(n) = \Theta(n^{\log_b a})$.
2. ak $f(n) = \Theta(n^{\log_b a - \epsilon})$ tak $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. ak $f(n) = \mathcal{O}(n^{\log_b a + \epsilon})$ pre nejakú konštantnu $\epsilon > 0$ a ak $af(n/b) \leq cf(n)$ pre nejakú konštantu $c < 1$ a pre všetky dostatočne veľké n, tak $T(n) = \Theta(f(n))$.

Master method - jednoduchšia varianta

Nech $a \geq 1$, $b > 1$ a $D \geq 0$ sú konštanty a nech $T(n)$ je definovaná na nezáporných číslach rekurentnou rovnicou

$$T(n) = \begin{cases} \Theta(1) & \text{pre } n = 1 \\ aT(n/b) + \mathcal{O}(n^d) & \text{inak} \end{cases}$$

Potom platí

$$T(n) = \begin{cases} \mathcal{O}(n^d) & \text{ak } a < b^d \\ \mathcal{O}(n^d \log n) & \text{ak } a = b^d \\ \mathcal{O}(n \log_b a) & \text{ak } a > b^d \end{cases}$$

a počet podproblémov, b faktor redukcie veľkosti problému, d obtiažnosť rozdelenia a kombinovania; a, b, d sú konštanty nezávislé na n

1 Rozdeľ a panuj

- Princípy
  - Binárne vyhľadávanie
  - Faktoriál
- Analýza rekurzívnych algoritmov
- Triedenie spájaním
- Maximálny a minimálny prvok
- Problém inverzií

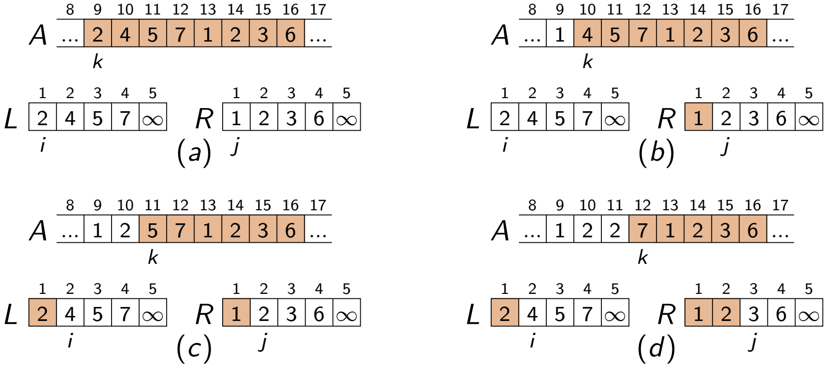
Triedenie spájaním

Rozdeľ Rozdeľ postupnosť n prvkov na dve postupnosti polovičnej veľkosti.

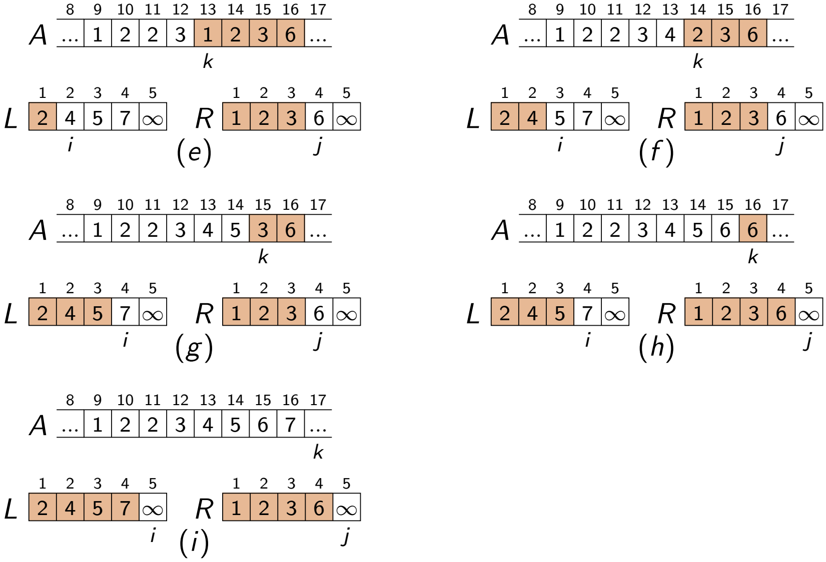
Vyrieš Obidve postupnosti (rekurzívne) utrieď.

Skombinuj Spoj dve utriedené postupnosti do jednej utriedenej postupnosti.

- konečnosť je daná faktom, že postupnosť dĺžky 1 je utriedená
- hlavnou časťou algoritmu je spájanie dvoch utriedených postupností do jednej utriedenej postupnosti
- pri spájaní porovnávame vedúce prvky oboch postupností, menší z porovnávaných prvkov presunieme do výslednej postupnosti



8 9 10 11 12 13 14 15 16 17
A … 2 4 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(a)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 4 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(b)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 2 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(c)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 2 2 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(d)
1 2 3 4 5
R 1 2 3 6 ∞
j


8 9 10 11 12 13 14 15 16 17
A … 1 2 2 3 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(e)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 2 2 3 4 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(f)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 2 2 3 4 5 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(g)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 2 2 3 4 5 6 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(h)
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 2 2 3 4 5 6 7 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
(i)
1 2 3 4 5
R 1 2 3 6 ∞
j

Spájanie utriedených postupností - MERGE

- procedúra MERGE má 4 parametre
  - A pole
  - p, q, r indexy poľa A také, že $p \leq q \leq r$
  - predpokladáme, že postupnosti prvkov $A[p \dots q]$ a $A[q+1 \dots r]$ sú utriedené
- výsledkom procedúry je pole $A[p \dots r]$, ktorého prvky sú utriedené
- pre zjednodušenie kódu používame *sentinel*

Merge

PROCEDURE MERGE(A, p, q, r)

$n_1 \leftarrow q - p + 1$
$n_2 \leftarrow r - q$
//nech $L[1 \dots n_1 + 1]$ a $R[1 \dots n_2 + 1]$ sú nové polia
for $i = 1$ **to** n_1 **do** $L[i] \leftarrow A[p + i - 1)$ **od**
for $j = 1$ **to** n_2 **do** $R[j] \leftarrow A[q + j]$ **od**
$L[n_1 + 1] \leftarrow \infty$
$R[n_2 + 1] \leftarrow \infty$
$i \leftarrow 1$
$j \leftarrow 1$
for $k = p$ **to** r **do**
    if $L[i] \leq R[j]$ **then** $A[k] \leftarrow L[i]$
        $i \leftarrow i + 1$
    else $A[k] \leftarrow R[j]$
        $j \leftarrow j + 1$ **fi**
od

Koreknosť procedúry MERGE

Invariant

Na začiatku každej iterácie cyklu **for** v riadkoch 10 - 15 postupnosť $A[p \ldots k-1]$ obsahuje $k-p$ najmeších prvkov z $L[1 \ldots n_1+1]$ a $R[1 \ldots n_2+1]$ a to v poradí podľa veľkosti. Naviac, $L[i]$ a $R[j]$ sú najmenšie prvky v svojich postupnostiach spomedzi tých, ktoré ešte neboli skopírované do A.

Inicializácia Na začiatku je $k = p$ a pole $A[p \ldots k-1]$ je prázdne. A obsahuje $k-p=0$ najmeších prvkov z L a R. Naviac, $i = j = 1$ a $L[i]$ a $R[j]$ sú najmenšie prvky v L a R.

Koreknosť procedúry MERGE

Invariant

Na začiatku každej iterácie cyklu **for** v riadkoch 10 - 15 postupnosť $A[p \dots k-1]$ obsahuje $k-p$ najmeších prvkov z $L[1 \dots n_1+1]$ a $R[1 \dots n_2+1]$ a to v poradí podľa veľkosti. Naviac, $L[i]$ a $R[j]$ sú najmenšie prvky v svojich postupnostiach spomedzi tých, ktoré ešte neboli skopírované do A.

Iterácia Predpokladajme, že $L[i] \leq R[j]$. Potom $L[i]$ je najmenší prvok z tých, ktoré ešte neboli prekopírované do A. Pretože $A[p \dots k-1]$ obsahuje $k-p$ najmeších prvkov, pole $A[p \dots k]$ bude obsahovať $k-p+1$ najmeších prvkov. Zvýšením k a i zaručíme platnosť invariantu aj po ukonční iterácie.

Koreknosť procedúry MERGE

Invariant

Na začiatku každej iterácie cyklu **for** v riadkoch 10 - 15 postupnosť $A[p \dots k-1]$ obsahuje $k-p$ najmeších prvkov z $L[1 \dots n_1+1]$ a $R[1 \dots n_2+1]$ a to v poradí podľa veľkosti. Naviac, $L[i]$ a $R[j]$ sú najmenšie prvky v svojich postupnostiach spomedzi tých, ktoré ešte neboli skopírované do A.

Ukončenie Cyklus skončí keď $k = r+1$. Z platnosti invariantu postupnosť $A[p \dots k-1] = A[p \dots r]$ obsahuje $k-p = r-p+1$ najmenších prvkov z $L[1 \dots n_1+1]$ a $R[1 \dots n_2+1]$ v usporiadanom poradí. Polia L a R obsahujú v súčte $n_1+n_2+2 = r-p+3$ prvkov. Všetky prvky s výnimkou dvoch najväčších boli prekopírované do A. Najväčšie dva prvky sú sentinely.

Zložitosť procedúry MERGE

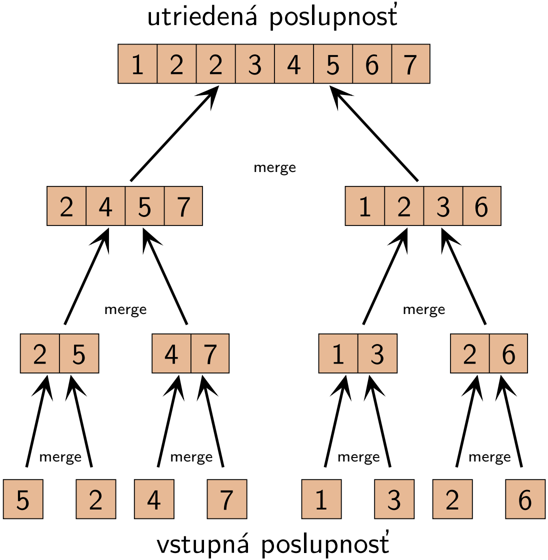
- riadky 1 - 2 a 6 - 9 majú konštantnú zložitosť
- **for** cykly na riadkoch 4 a 5 majú v súčte zložitosť $\Theta(n_1 + n_2) = \Theta(n)$, kde $n = r - p + 1$
- **for** cyklus na riadkoch 10 - 15 iteruje n krát pričom všetky príkazy na riadkoch 11 - 14 majú konštantnú zložitosť
- zložitosť procedúry MERGE je $\Theta(n)$

Triedenie spájaním - MERGE SORT

- algoritmus „rozdeľ a panuj“
- využíva procedúru MERGE
- pre utriedenie celej postupnosti voláme MERGE SORT$(A, 1, A.length)$

PROCEDURE MERGE SORT(A, p, r)

if $p \leq q$ **then** $q \leftarrow \lfloor (p+q)/2 \rfloor$
MERGE SORT(A, p, q)
MERGE SORT$(A, q+1, r)$
MERGE(A, p, q, r) **fi**



Obrázek: Definition of sth. *equation*

Zložitosť algoritmu Merge Sort

predpokladáme[1], že n je mocninou 2

Rozdeľ rozdelenie spočíva vo výpočte indexu, preto $D(n) = \Theta(1)$

Vyrieš rekurzívne triedime dve postupnosti veľkosti $n/2$, časová zložitosť je $2T(n/2)$

Skombinuj zložitosť procedúry Merge je $\Theta(n)$ a preto $C(n) = \Theta(n)$

$$T(n) = \begin{cases} \Theta(1) & \text{ak } n = 1 \\ 2T(n/2) + \Theta(n) & \text{inak} \end{cases}$$

riešením rekurentnej rovnice (*viz rekurzívny strom*) dostávame, že časová zložitosť Merge Sort je $T(n) = \Theta n \log n)$

[1] v časti o asymptotickej notácii ukážeme, že tento predpoklad nezmení riešenie rekurentnej rovnice

1. Rozdeľ a panuj
   - Princípy
     - Binárne vyhľadávanie
     - Faktoriál
   - Analýza rekurzívnych algoritmov
   - Triedenie spájaním
   - Maximálny a minimálny prvok
   - Problém inverzií

Príklad - maximálny a minimálny prvok

- problém nájdenia maximálneho a minimálneho prvku postupnosti $S[1 \dots n]$
- zložitostné kritérium - počet porovnaní prvkov

MAX(S)

```
max ← S[1]
for i = 2 to n do
    if S[i] > max then max ← S[i] fi
od
```

minimum nájdeme medzi zvyšnými $n-1$ prvkami podobne

celkove $(n-1)+(n-2)$ porovnaní

Maximálny a minimálny prvok – Prístup Rozdeľ a panuj

1. pole rozdeľ na dve (rovnako veľké) podpostunosti
2. nájdi minimum a maximum oboch podpostupností
3. maximálny prvok postupnosti je väčší z maximálnych prvkov oboch podpostupností
   podobne minimálny prvok

$\textsc{MaxMin}(x, y)$

//nájdi minimálny a maximálny prvok v postupnosti $S[x \dots y]$
if $y = x$ **then return** $(S[x], S[x])$ **fi**
if $y = x + 1$ **then return** $(\max(S[x], S[y]), \min(S[x], S[y]))$ **fi**
if $y > x + 1$ **then** $(A1, B1) \leftarrow \textsc{MaxMin}(x, \lfloor (x+y)/2 \rfloor)$
$(A2, B2) \leftarrow \textsc{MaxMin}(\lfloor (x+y)/2 \rfloor + 1, y)$
return $(\max(A1, A2), \min(B1, B2))$ **fi**

Maximálny a minimálny prvok – analýza

Korektnosť indukciou vzhľadom k $n = y - x + 1$ ukážeme, že $\textsc{MaxMin}(x, y)$ vráti maximálnu a minimálnu hodnotu postupnosti

Zložitosť

$$T(n) = \begin{cases} 1 & \text{pre } n = 2 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 2 & \text{inak} \end{cases}$$

Indukciou k n overíme, že $T(n) \leq \frac{5}{3}n - 2$

1. pre $n = 2$ platí $\frac{5}{3} \cdot 2 - 2 > 1 = T(2)$
2. predpokladajme platnosť nerovnosti pre všetky hodnoty $2 \leq i < n$, dokážeme jej platnosť pre n

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 2 && \text{indukčný predp.} \\ &\leq \frac{5}{3}\lfloor n/2 \rfloor - 2 + \frac{5}{3}\lceil n/2 \rceil - 2 + 2 = \frac{5}{3}n - 2 \end{aligned}$$

1. Rozdeľ a panuj
   - Princípy
     - Binárne vyhľadávanie
     - Faktoriál
   - Analýza rekurzívnych algoritmov
   - Triedenie spájaním
   - Maximálny a minimálny prvok
   - Problém inverzií

Príklad - problém inverzií

motivácia

porovnanie zoznamu preferencií

formulácia problému

- je daná postupnosť vzájomne rôznych čísel $a_1, \ldots, a_n$
- inverziou v postupnosti je dvojica indexov i, j takých, že $i < j$ a zároveň $a_i > a_j$
- úlohou je nájsť všetky inverzie v danej postupnosti čísel

príklad

postupnosť 1, 4, 6, 8, 2, 5 má 5 inverzií

naivný algoritmus

otestuje všetky dvojice indexov, zložitosť $\mathcal{O}(n^2)$

Problém inverzií — prístup Rozdeľ a panuj

1. postupnosť rozdelíme na dve podspostupnosti $a_1, \dots, a_{\lceil n/2 \rceil}$ a $a_{\lceil n/2 \rceil + 1}, \dots, a_n$
2. v každej z podpostupností spočítame inverzie
3. spočítame inverzie medzi prvkami rôznych podpostupností

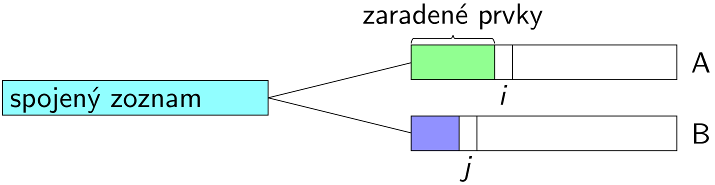
- ak chceme, aby časová zložitosť algoritmu bola $T(n) = \mathcal{O}(n \log n)$, tak musí platiť $T(n) \leq 2T(n/2) + cn$ *(pre vhodnú konštantu c)*
- ako vyriešiť úlohu 3. v čase cn ?
- pri riešení úlohy 2. súčasne s počítaním inverzií utriedime podpostupnosti
- úlohu 3. vyriešime spojením dvoch utriedených podpostupností do jednej utriedenej postupnosti, pričom zároveň počítame inverzie medzi prvkami podpostupností

Problém inverzií — riešenie úlohy 3.

- $A = a_1, a_2, \ldots, a_k$
- $B = b_1, b_2, \ldots, b_l$
- prepokladáme, že
  - prvky v oboch postupnostiach sú utriedené vzostupne
  - všetky prvky v postupnosti A majú vo vstupnej postupnosti menší index než prvky postupnosti B
- prvky $b_1, \ldots, b_{i-1}$ a $c_1, \ldots, c_{j-1}$ sú už zatriedené
- porovnávame prvok a_i s prvkom b_j

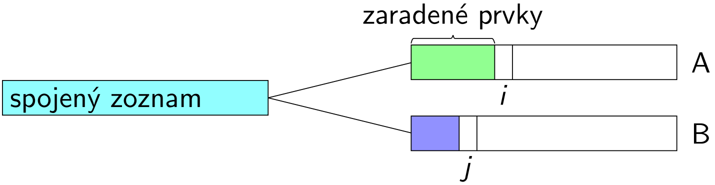
	menší z porovnávaných prvkov zaradíme do výstupnej postupnosti
$a_i < b_j$	a_i nie je v inverzii so žiadnym z prvkov $b_j, b_{j+1}, \ldots, b_l$
$b_j < a_i$	b_j je v inverzii so všetkými prvkami $a_i, a_{i+1}, \ldots, a_k$ a preto k počtu inverzií pripočítame $k - i + 1$

Problém inverzií - spájanie zoznamov



inverzie pre zaradené prvky sú už započítané

Problém inverzií - spájanie zoznamov



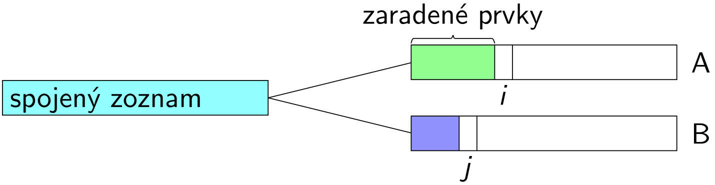
inverzie pre zaradené prvky sú už započítané

$a_i < b_j$ do spojeného zoznamu zaradíme a_i

$a_i < b_j < b_{j+1} < b_{j+2} < \dots$

a_i nie je v inverzii so žiadnym z $b_j, b_{j+1}, \dots$

Problém inverzií - spájanie zoznamov



inverzie pre zaradené prvky sú už započítané

$a_i < b_j$ do spojeného zoznamu zaradíme a_i

$a_i < b_j < b_{j+1} < b_{j+2} < \dots$

a_i nie je v inverzii so žiadnym z $b_j, b_{j+1}, \dots$

$b_j < a_i$ do spojeného zoznamu zaradíme b_j

$b_j < a_i < a_{i+1} < a_{i+2} < \dots$

b_j je v inverzii s každým z $a_i, a_{j+1}, \dots$

Problém inverzií - spájanie zoznamov

MERGE_AND_COUNT(A, B)

```
i ← 1; j ← 1
//i, j sú indexy prvých nezaradených prvkov zo zoznamov A resp. B
Count ← 0
//Count je počet nájdených inverzií
while  zoznamy A, B sú neprázdne  do
        porovnaj a_i a b_j
        menší z prvkov zaraď do výsledného zoznamu
        if b_j < a_i
           then  zvýš Count o počet nezaradených prvkov z A fi
        zvýš index i resp. j od
if  jeden zoznam je prázdny
   then  zaraď zvyšné prvky druhého zoznamu do výsledného zoznam
return Count a výsledný zoznam
```

Problém inverzií

Sort_and_Count(L)

if L má jeden prvok
  then $r \leftarrow 0$
  else rozdeľ L na dve postupnosti
      A obsahuje prvých $\lceil n/2 \rceil$ prvkov
      B obshuje zvyšných $\lfloor n/2 \rfloor$ prvkov
      $(r_A, A) \leftarrow \textsc{Sort\_and\_Count}(A)$
      $(r_B, B) \leftarrow \textsc{Sort\_and\_Count}(B)$
      $(r, L) \leftarrow \textsc{Merge\_and\_Count}(A, B)$
      $r \leftarrow r_A + r_B$ **fi**
return r a utriedený zoznam L

zložitosť algoritmu je $T(n) = 2T(n/2) + \Theta(n)$ a preto $T(n) = \mathcal{O}(n \log n)$