Paralelné algoritmy, časť č. 5

František Mráz

Kabinet software a výuky informatiky, MFF UK, Praha

Paralelné algoritmy, 2011/2012

Obsah

Outline

Metóda vyvážených binárnych stromov

Pr. Počítanie minima z n čísel

- výška stromu = čas výpočtu $\lceil \log n \rceil$

Metóda vyvážených binárnych stromov

Pr. Počítanie minima z n čísel

- výška stromu = čas výpočtu $\lceil \log n \rceil$
- počet procesorov $\frac{n}{2}$

Metóda vyvážených binárnych stromov

Pr. Počítanie minima z n čísel

- výška stromu = čas výpočtu $\lceil \log n \rceil$
- počet procesorov $\frac{n}{2}$
- pole s $2n$ prvkami, $n = 2^m$

Metóda vyvážených binárnych stromov

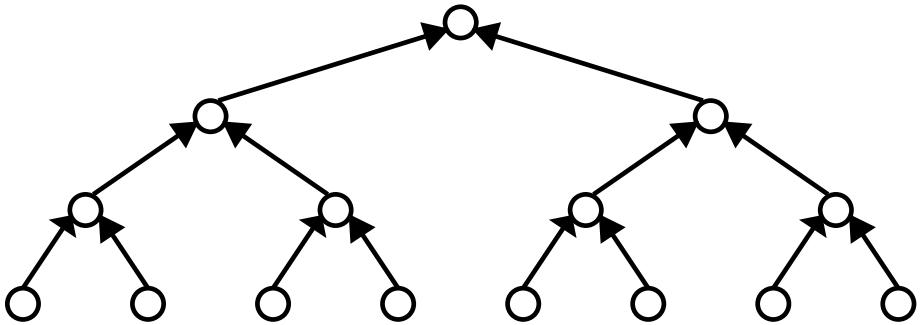
Pr. Počítanie minima z n čísel

- výška stromu = čas výpočtu $\lceil \log n \rceil$
- počet procesorov $\frac{n}{2}$
- pole s $2n$ prvkami, $n = 2^m$
- vstupné čísla uložené v $A[n], A[n+1], ..., A[2n-1]$

Metóda vyvážených binárnych stromov

Pr. Počítanie minima z n čísel

- výška stromu = čas výpočtu $\lceil \log n \rceil$
- počet procesorov $\frac{n}{2}$
- pole s $2n$ prvkami, $n = 2^m$
- vstupné čísla uložené v $A[n], A[n+1], ..., A[2n-1]$
- **for** $k := m - 1$ **downto** 0 **do**
  for all $j, 2^k \leq j < 2^{k+1}$ **in parallel do**
  $A[j] := \min(A[2j], A[2j+1])$



Technika zdvojovania

Pr. Počítanie minima z n čísel

- aplikuje sa na pole alebo na spojový zoznam

Technika zdvojovania

Pr. Počítanie minima z n čísel

- aplikuje sa na pole alebo na spojový zoznam
- v každom kroku sa spracovávajú prvky v pevnej vzdialenosti, táto vzdialenosť sa v každom kroku zdvojnásobuje

Technika zdvojovania

Pr. Počítanie minima z n čísel

- aplikuje sa na pole alebo na spojový zoznam
- v každom kroku sa spracovávajú prvky v pevnej vzdialenosti, táto vzdialenosť sa v každom kroku zdvojnásobuje
- Pr. Očíslovanie zoznamu prvkov L:

Technika zdvojovania

Pr. Počítanie minima z n čísel

- aplikuje sa na pole alebo na spojový zoznam
- v každom kroku sa spracovávajú prvky v pevnej vzdialenosti, táto vzdialenosť sa v každom kroku zdvojnásobuje
- Pr. Očíslovanie zoznamu prvkov L:
  - každý prvok k pozná následníka $next(k)$, ak je k posledný prvok zoznamu, tak $next(k) = k$

Technika zdvojovania

Pr. Počítanie minima z n čísel

- aplikuje sa na pole alebo na spojový zoznam
- v každom kroku sa spracovávajú prvky v pevnej vzdialenosti, táto vzdialenosť sa v každom kroku zdvojnásobuje
- Pr. Očíslovanie zoznamu prvkov L:
  - každý prvok k pozná následníka $next(k)$, ak je k posledný prvok zoznamu, tak $next(k) = k$
  - Úloha: pre každý prvok spočítať jeho vzdialenosť od konca zoznamu

for all $k \in L$ **in parallel do**
begin $P(k) := next(k)$;
If $P(k) \neq k$ **then** $distance(k) := 1$
else $distance(k) := 0$
end;
Repeat $\log n$ **times**
For all $k \in L$ **in parallel do**
If $P(k) \neq P(P(k))$ **then**
begin $distance(k) := distance(k) + distance(P(k))$;
P(k):= P(P(k))
end;
For all $k \in L$ **in parallel do** $rank(k) := distance(k)$;

Technika rozdeľ a panuj

- podobné metóde binárneho stromu, ale má 2 fázy

Technika rozdeľ a panuj

- podobné metóde binárneho stromu, ale má 2 fázy
  1. rozdeľovanie od koreňa k listom

Technika rozdeľ a panuj

- podobné metóde binárneho stromu, ale má 2 fázy
  1. rozdeľovanie od koreňa k listom
  2. kombinovanie výsledkov od listov ku koreňu

Technika rozdeľ a panuj

- podobné metóde binárneho stromu, ale má 2 fázy
  1. rozdeľovanie od koreňa k listom
  2. kombinovanie výsledkov od listov ku koreňu
- príklady

Technika rozdeľ a panuj

- podobné metóde binárneho stromu, ale má 2 fázy
  1. rozdeľovanie od koreňa k listom
  2. kombinovanie výsledkov od listov ku koreňu
- príklady
  a) počítanie minima – čísla rozdelíme na 2 skupiny a hľadáme v nich minimá, z nich potom určíme minimum

Technika rozdeľ a panuj

- podobné metóde binárneho stromu, ale má 2 fázy
  1. rozdeľovanie od koreňa k listom
  2. kombinovanie výsledkov od listov ku koreňu
- príklady
  a) počítanie minima – čísla rozdelíme na 2 skupiny a hľadáme v nich minimá, z nich potom určíme minimum
  b) počítanie hodnoty polynómu $p(x)$ stupňa n v bode $x = x_0$; pre jednoduchosť predpokladáme, že $n = 2^m - 1$, pre vhodné celé m;

$$p(x) = r(x) + x^{\frac{n+1}{2}} q(x),$$

kde $r(x)$, $q(x)$ sú polynómy stupňa $2^{m-1} - 1$ rekurzívne spočítame $r(x_0)$, $q(x_0)$ a z nich výsledok

Technika kompresie

- Pr.: počítanie minima z množiny X:

Technika kompresie

- Pr.: počítanie minima z množiny X:
  - každý nepárny prvok nahradíme minimom z $X(i)$ a $X(i+1)$, tým dostaneme množinu polovičnej veľkosti s rovnakým minimom

$$[3,7,8,3,9,2,3,1] \rightarrow [3,3,2,1] \rightarrow [3,1] \rightarrow [1]$$

Technika kompresie

- Pr.: počítanie minima z množiny X:
  - každý nepárny prvok nahradíme minimom z $X(i)$ a $X(i+1)$, tým dostaneme množinu polovičnej veľkosti s rovnakým minimom

$$[3,7,8,3,9,2,3,1] \rightarrow [3,3,2,1] \rightarrow [3,1] \rightarrow [1]$$

- iné použitie: kompresia grafu

Redukcia počtu procesorov

- $T(n, p)$ - čas potrebný na riešenie úlohy veľkosti n pomocou p procesorov

Redukcia počtu procesorov

- $T(n,p)$ - čas potrebný na riešenie úlohy veľkosti n pomocou p procesorov
- ak $p < \frac{n}{2}$ potom

$$T(n,p) \leq T\left(\frac{n}{p}, 1\right) + T(p,p)$$

Redukcia počtu procesorov

- $T(n,p)$ - čas potrebný na riešenie úlohy veľkosti n pomocou p procesorov
- ak $p < \frac{n}{2}$ potom

$$T(n,p) \leq T\left(\frac{n}{p}, 1\right) + T(p,p)$$

- napr. počítanie minima z n čísel

Redukcia počtu procesorov

- $T(n,p)$ - čas potrebný na riešenie úlohy veľkosti n pomocou p procesorov
- ak $p < \frac{n}{2}$ potom

$$T(n,p) \leq T\left(\frac{n}{p}, 1\right) + T(p,p)$$

- napr. počítanie minima z n čísel
  - $T(\frac{n}{p}, 1) = \frac{n}{p}$ a $T(p,p) = \log n$

Redukcia počtu procesorov

- $T(n,p)$ - čas potrebný na riešenie úlohy veľkosti n pomocou p procesorov
- ak $p < \frac{n}{2}$ potom

$$T(n,p) \leq T\left(\frac{n}{p},1\right) + T(p,p)$$

- napr. počítanie minima z n čísel
  - $T(\frac{n}{p},1) = \frac{n}{p}$ a $T(p,p) = \log n$
  - ak vezmeme $p = \frac{n}{\log n}$, tak dostaneme optimálny algoritmus

Redukcia počtu procesorov

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

- pomocou p procesorov sa dajú vykonať v čase $\frac{m(i)}{p} + 1$

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

- pomocou p procesorov sa dajú vykonať v čase $\frac{m(i)}{p} + 1$
- celkový počet operácií $m = m(1) + m(2) + ... + m(t)$

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

- pomocou p procesorov sa dajú vykonať v čase $\frac{m(i)}{p} + 1$
- celkový počet operácií $m = m(1) + m(2) + ... + m(t)$
- Celkový čas: $O(\frac{m}{p} + t)$

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

- pomocou p procesorov sa dajú vykonať v čase $\frac{m(i)}{p} + 1$
- celkový počet operácií $m = m(1) + m(2) + ... + m(t)$
- Celkový čas: $O(\frac{m}{p} + t)$
- Implementačné problémy:

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

- pomocou p procesorov sa dajú vykonať v čase $\frac{m(i)}{p} + 1$
- celkový počet operácií $m = m(1) + m(2) + ... + m(t)$
- Celkový čas: $O(\frac{m}{p} + t)$
- Implementačné problémy:
  1. ako spočítať $m(i)$

Redukcia počtu procesorov

Veta (Brent)

Nech A je daný algoritmus s paralelným časom t. Predpokladajme, že A vyžaduje m výpočtových operácií. Potom sa A dá implementovať pomocou p procesorov v čase $O(\frac{m}{p} + t)$.

Dôkaz: $m(i) =$ počet operácií vykonaných algoritmom A v kroku i

- pomocou p procesorov sa dajú vykonať v čase $\frac{m(i)}{p} + 1$
- celkový počet operácií $m = m(1) + m(2) + ... + m(t)$
- Celkový čas: $O(\frac{m}{p} + t)$
- Implementačné problémy:
  1. ako spočítať $m(i)$
  2. ako rozdeliť $m(i)$ operácií do skupín – v poli to ide dobre, v zozname sa to dá ťažšie

Outline

Prefixové výpočty

- n čísel uložených v poli A – $A[n], A[n+1], \ldots, A[2n-1]$

Prefixové výpočty

- n čísel uložených v poli A – $A[n], A[n+1], \ldots, A[2n-1]$
- prefixový výpočet spočíta hodnoty $A[n]$, $A[n] \oplus A[n+1]$, $A[n] \oplus A[n+1] \oplus A[n+2]$, ..., kde $\oplus$ je asociatívna binárna operácia

Prefixové výpočty

- n čísel uložených v poli A – $A[n], A[n+1], \ldots, A[2n-1]$
- prefixový výpočet spočíta hodnoty $A[n]$, $A[n] \oplus A[n+1]$, $A[n] \oplus A[n+1] \oplus A[n+2]$, ..., kde $\oplus$ je asociatívna binárna operácia
- použije sa metóda binárneho stromu, nech $n = 2^m$, pre vhodné celé $m > 0$

Prefixové výpočty

- n čísel uložených v poli A – $A[n], A[n+1], \ldots, A[2n-1]$
- prefixový výpočet spočíta hodnoty $A[n]$, $A[n] \oplus A[n+1]$, $A[n] \oplus A[n+1] \oplus A[n+2]$, ..., kde $\oplus$ je asociatívna binárna operácia
- použije sa metóda binárneho stromu, nech $n = 2^m$, pre vhodné celé $m > 0$
- **for** $k := m-1$ **downto** 0 **do**
  for all j, $2^k \leq j \leq 2^{k+1}-1$ **in parallel do** $A[j] := A[2j] \oplus A[2j+1]$;
  $B[1] := A[1]$;
  for $k := 1$ **to** m **do**
  for all j, $2^k \leq j \leq 2^{k+1}-1$ **in parallel do**
  if $odd(j)$ **then** $B[j] := B[\frac{j-1}{2}]$
  else if $j = 2^k$ **then** $B[j] := A[j]$
  else $B[j] := B[\frac{j}{2}-1] \oplus A[j]$;

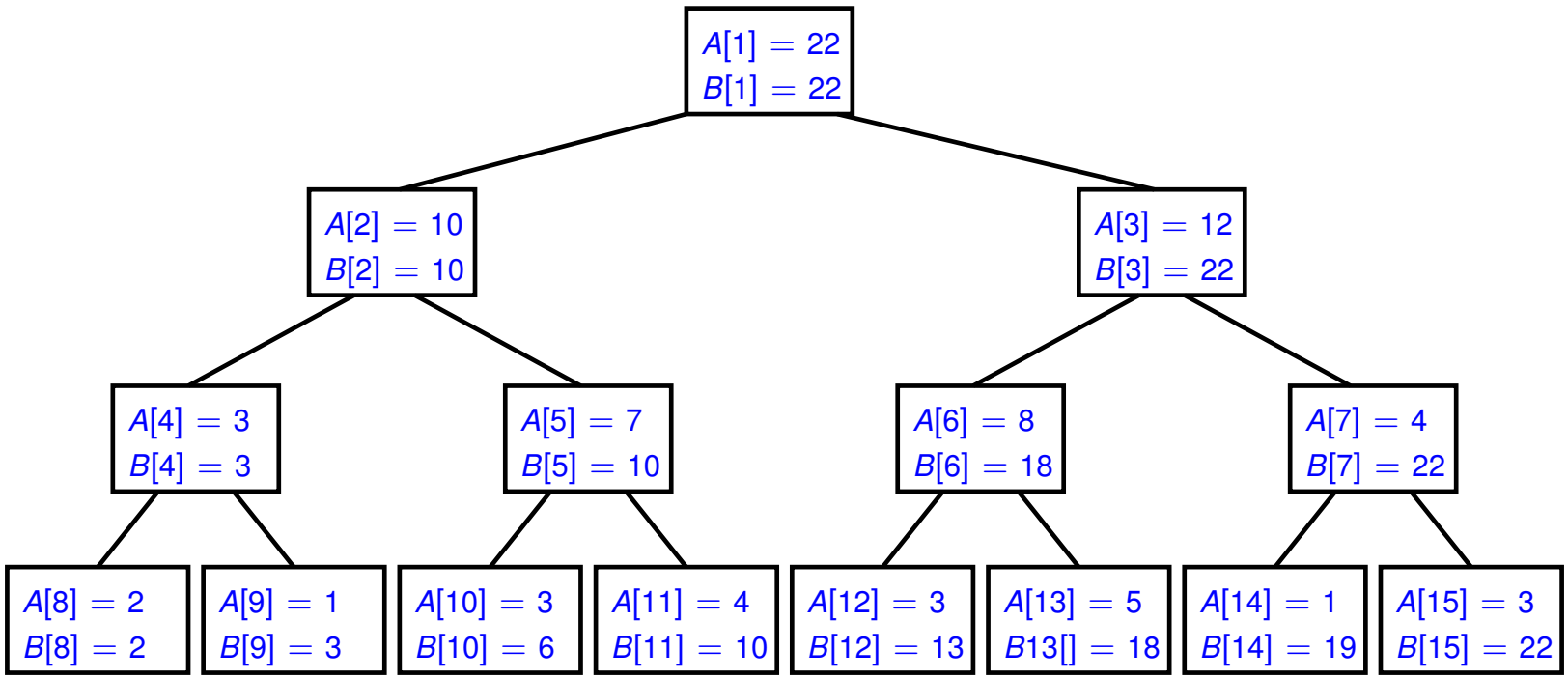
Prefixové výpočty

- n čísel uložených v poli A – $A[n], A[n+1], \ldots, A[2n-1]$
- prefixový výpočet spočíta hodnoty $A[n]$, $A[n] \oplus A[n+1]$, $A[n] \oplus A[n+1] \oplus A[n+2]$, ..., kde $\oplus$ je asociatívna binárna operácia
- použije sa metóda binárneho stromu, nech $n = 2^m$, pre vhodné celé $m > 0$
- **for** $k := m-1$ **downto** 0 **do**
  for all j, $2^k \leq j \leq 2^{k+1}-1$ **in parallel do** $A[j] := A[2j] \oplus A[2j+1]$;
  $B[1] := A[1]$;
  for $k := 1$ **to** m **do**
  for all j, $2^k \leq j \leq 2^{k+1}-1$ **in parallel do**
  if $odd(j)$ **then** $B[j] := B[\frac{j-1}{2}]$
  else if $j = 2^k$ **then** $B[j] := A[j]$
  else $B[j] := B[\frac{j}{2}-1] \oplus A[j]$;

- Pr. prefixové súčty čísel 2, 1, 3, 4, 3, 5, 1, 3.

Príklad prefixových súčtov

Prefixové súčty čísel 2, 1, 3, 4, 3, 5, 1, 3.



Minimá intervalov

Je daný vstupný vektor celých čísel $x(1), \ldots, x(n)$ a vektor intervalov $int(i) = (l(i)..r(i))$, obidva veľkosti n.
Úloha: Pre každé i spočítať $\min\{x(k); k \in int(i)\}$.

- metóda binárneho stromu; n mocnina dvojky, $x(i)$ uložené v listoch

Minimá intervalov

Je daný vstupný vektor celých čísel $x(1), \ldots, x(n)$ a vektor intervalov $int(i) = (l(i)..r(i))$, obidva veľkosti n.
Úloha: Pre každé i spočítať $\min\{x(k); k \in int(i)\}$.

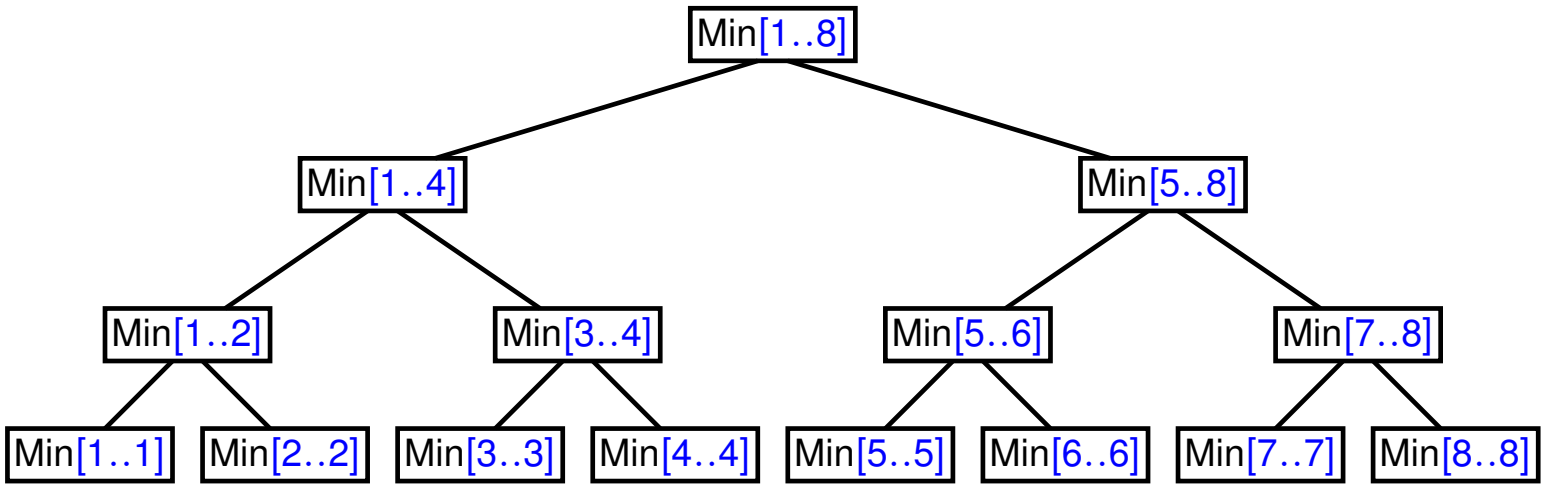
- metóda binárneho stromu; n mocnina dvojky, $x(i)$ uložené v listoch

Minimá intervalov

Je daný vstupný vektor celých čísel $x(1), \ldots, x(n)$ a vektor intervalov $int(i) = (l(i)..r(i))$, obidva veľkosti n.
Úloha: Pre každé i spočítať $\min\{x(k); k \in int(i)\}$.

- metóda binárneho stromu; n mocnina dvojky, $x(i)$ uložené v listoch

Fáza a) v každom uzle spočítame minimum z jeho potomkov – dostaneme minimá v "dobrých" intervaloch



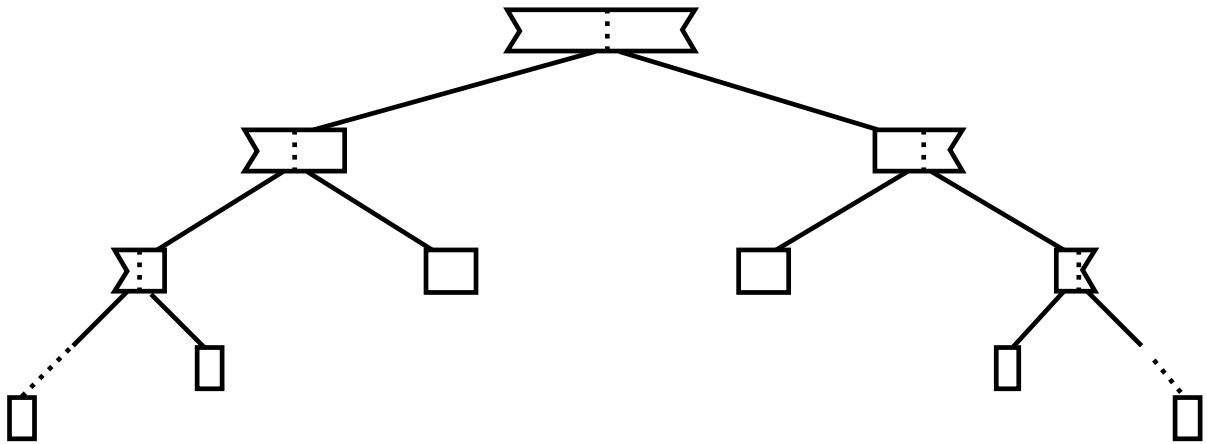
Minimá intervalov

Je daný vstupný vektor celých čísel $x(1), \ldots, x(n)$ a vektor intervalov $int(i) = (l(i)..r(i))$, obidva veľkosti n.
Úloha: Pre každé i spočítať $\min\{x(k); k \in int(i)\}$.

Minimá intervalov

Je daný vstupný vektor celých čísel $x(1), \ldots, x(n)$ a vektor intervalov $int(i) = (l(i)..r(i))$, obidva veľkosti n.
Úloha: Pre každé i spočítať $\min\{x(k); k \in int(i)\}$.

Fáza b) každému intervalu $int(i) = (l(i)..r(i))$, sa pridelí 1 procesor, ten urobí rozklad na "dobré" intervaly a spočíta z nich minimum



Revidovaná technika zdvojovania

- namiesto dĺžky počítame *value*, namiesto $+$ nejaká binárna, asociatívna operácia $\oplus$

Revidovaná technika zdvojovania

- namiesto dĺžky počítame *value*, namiesto $+$ nejaká binárna, asociatívna operácia $\bigoplus$
- **Repeat** $\log n$ **times**
  For all $k \in L$ **in parallel do**
  If $P(k) \neq P(P(k))$ **then**
  Begin $value(k) := value(k) \bigoplus value(P(k))$;
  $P(k) := P(P(k))$
  End

Revidovaná technika zdvojovania

- namiesto dĺžky počítame *value*, namiesto $+$ nejaká binárna, asociatívna operácia $\bigoplus$
- **Repeat** $\log n$ **times**
  For all $k \in L$ **in parallel do**
  If $P(k) \neq P(P(k))$ **then**
  Begin $value(k) := value(k) \bigoplus value(P(k))$;
  $P(k) := P(P(k))$
  End

- $\bigoplus$ môže byť max, min, ...

Výber podzoznamu

- nech $P(k)$ je následník prvku k spojového zoznamu; ak je k posledný prvok zoznamu, tak $P(k) = k$

Výber podzoznamu

- nech $P(k)$ je následník prvku k spojového zoznamu; ak je k posledný prvok zoznamu, tak $P(k) = k$
- prvky k spojového zoznamu L obsahujú položku $k.Color \in \{\text{red}, \text{green}\}$. Chceme vybrať podzoznam obsahujúci červené prvky

Výber podzoznamu

- nech $P(k)$ je následník prvku k spojového zoznamu; ak je k posledný prvok zoznamu, tak $P(k) = k$
- prvky k spojového zoznamu L obsahujú položku $k.Color \in \{\text{red}, \text{green}\}$. Chceme vybrať podzoznam obsahujúci červené prvky

$P'(k) := P(k)$
Repeat $\log n$ **times**
  for all $k \in L$ **in parallel do**
    If $(P'.Color \neq \text{red})$**and**$(P'(k) \neq P'(P'(k))$ **then**
      $P'(k) := P'(P'(k))$
if $L.First.Color = \text{red}$ **then** $L'.First := L.First$
  else $L'.First := P'(L.First)$
for all $k \in L$ **in parallel do**
if $P'(k).Color \neq \text{red}$ **then** $P'(k) := k$

Výber podzoznamu

- nech $P(k)$ je následník prvku k spojového zoznamu; ak je k posledný prvok zoznamu, tak $P(k) = k$
- prvky k spojového zoznamu L obsahujú položku $k.Color \in \{\text{red}, \text{green}\}$. Chceme vybrať podzoznam obsahujúci červené prvky

$P'(k) := P(k)$
Repeat $\log n$ **times**
  for all $k \in L$ **in parallel do**
    If $(P'.Color \neq \text{red})$**and**$(P'(k) \neq P'(P'(k))$ **then**
      $P'(k) := P'(P'(k))$
if $L.First.Color = \text{red}$ **then** $L'.First := L.First$
  else $L'.First := P'(L.First)$
for all $k \in L$ **in parallel do**
if $P'(k).Color \neq \text{red}$ **then** $P'(k) := k$

- dá sa použiť i na výber význačného prvku zo zoznamu – bude to ten prvý vo vybranom podzozname