Paralelní řadící algoritmy

prof. Ing. Pavel Tvrdík CSc.

Katedra počítačových systémů
Fakulta informačních technologií
České vysoké učení technické v Praze

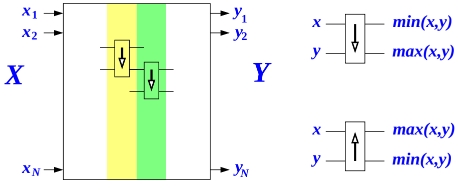
Paralelní algoritmy a systémy (MI-PAR)
ZS 2010/11, Přednáška 8

https://edux.fit.cvut.cz/MI-PAR/

Úvod do paralelního řazení založeného na operaci porovnání

- Tato přednáška: **Paralelní deterministické** řadící alg. založené na operaci **porovnej-a-vyměň** (Compare-and-Exchange, C&E).
- Dolní mez na počet C&E operací pro seřazení N čísel je $\Omega(N \log N)$.
- $\exists$ optimální sekvenční algoritmy (HeapSort, MergeSort, ...).
- $\exists$ časově a cenově optim. PRAM // řadící alg.: **Coleův MergeSort**. $T(N, N) = O(\log N)$ // C&E kroků na EREW PRAM.
- $\exists$ asymptoticky optimální // řadící algoritmy na mřížkách: $T(N, N) = O(k\sqrt{N})$ // C&E kroků na 2-D mřížce, $2 < k < 3$.
- Nejpraktičtější // řadící algoritmus na hyperkubických sítích: **Batcherův MergeSort** s $T(N, N) = O(\log^2 N)$ kroků na $oBF_{\log N}$ nebo $Q_{\log N}$.

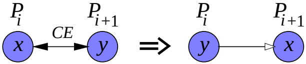
Nepřímé řadící sítě



(a) (b)

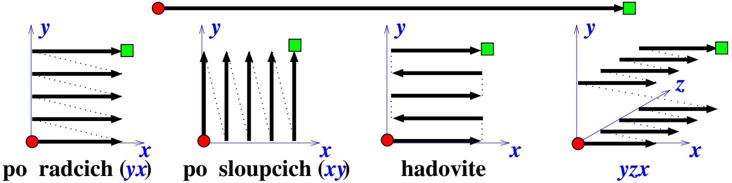
- (a) **Řadící síť** = síť složená ze **sloupců komparátorů** (jako MIN).
- (b) **Komparátor** = HW implementace operace C&E (vzestupně, sestupně).
- **Nesetříděná vstupní** posloupnost $X = [x_1, \ldots, x_N]$ je permutována na **setříděnou výstupní** posloupnost (klesající, rostoucí, bitonickou) $Y = [y_1, \ldots, y_N]$.
- **Statická** řadící síť = HW implementace **datově necitlivého** řadícího alg.
- Počet // C&E kroků = **hloubka** řadící sítě = délka nejdelší cesty ze vstupu na výstup.
- Je-li $N > \#$ vstupních vodičů $\Rightarrow$ operace **Sluč-a-Rozděl** (Merge-and-Split, M&S).

Přímé řadící sítě

- C&E operace mezi dvojicí procesorů: 

- Topologie: mřížky, toroidy, hyperkrychle, hyperkubické sítě,. . . .
- Přímý řadící alg. = posloupnost **dokonalých párování procesorů** (perfect matchings) odvozených z očíslování procesorů.
- 1 dokonalé párování se skládá z $\lfloor p/2 \rfloor$ disjunktních dvojic.
- **Datově necitlivé řazení**: způsob párování nezávisí na vstupních datech.
- Výběr vhodného **číslování (lineárního indexování)** procesorů

  má vliv na složitost C&E řazení pro danou topologii.
- Příklady číslování procesorů v mřížkách.



Škálovatelnost při $p < N$

- N/p čísel na 1 procesor + operace **Sluč-a-Rozděl** (M&S)
  - Každý procesor setřídí svých N/p čísel v $O((N/p)\log(N/p))$ krocích.
  - Všechny procesory provádějí přímý řadící algoritmus, kde používají M&S místo C&E.
- $M\&S(P_i, P_j)$, $i < j$, lze implementovat 2 asympt. ekvivalentními způsoby:

I. Plně-duplexní kanály:

(1) P_i a P_j si vymění své podposloupnosti.
(2) Každý provede M&S se svou a s obdrženou podposloup.
(3) Každý si ponechá svou polovinu a druhou zahodí.

II. Polo-duplexní kanály:

(1) P_i pošle svou podposloupnost P_j.
(2) P_j provede operaci M&S.
(3) P_j vrátí 1. polovinu výsledku P_i a ponechá si druhou.

Časová složitost řadících algoritmů

Věta 1

Nechť $\tau(N) = T(N, N)$ je počet // C&E kroků řazení N čísel na N-procesorové síti G. Pak počet // C&E kroků řazení N čísel na p-procesorové síti G, $p \leq N$, je

$$T(N, p) = O\left(\frac{N}{p} \log \frac{N}{p}\right) + O\left(\tau(p)\frac{N}{p}\right). \tag{1}$$

Naivní PRAM řadící algoritmus

Algorithm $\textsc{NaivePRAMSort}(X = [x_1, \ldots, x_N])$ on EREW PRAM(N, p)

(1) Každému P_i je přiřazena podposloup. N/p čísel v sdílené paměti.

(2) Každý P_i setřídí svých N/p čísel v $O((N/p)\log(N/p))$ C&E krocích.

(3) Všechny P provedou **paralelní redukci sloučením** seřazením podposloup.:

1. fáze:	$p/2$ Ps sloučí $p/2$ párů podposloup. o velikosti N/p.
2. fáze:	$p/4$ Ps sloučí $p/4$ párů podposloup. o velikosti $2N/p$.
...	atd.
($\log p$)-tá fáze:	1 zbývající P sloučí poslední 2 podposloup. o velikosti $N/2$.

- $T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(\frac{N}{p} + \frac{2N}{p} + \cdots + N\right) = O\left(\frac{N}{p}\log\frac{N}{p} + N\right)$
- $T_{\min}(N,p) = T(N,N) = O(N)$
- $E(N,p) = \Theta\left(\frac{\log N}{\log(N/p)+p}\right) = \Theta\left(\frac{\log N}{\log N+p}\right)$
- $\psi_1(p) = (2^p/p)^{O(1)}$, $\psi_2(N) = \log N$, $\psi_3(N) = \log N$.
- $\textsc{NaivePRAMSort}$ je špatně škálovatelný: čas je řádově stejný pro $p \in \{\log N, \ldots, N\}$!!

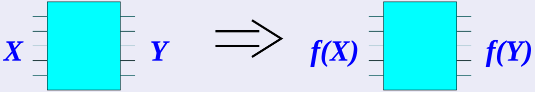
Datově necitlivé řazení a 0-1 Řadící Lemma

Lemma 2

Nechť f = **monotonně rostoucí** *funkce na lineárně uspořádané množině* $\mathcal{S}$,

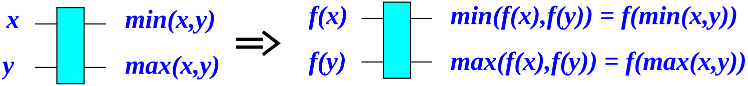
$$\text{čili} \quad \forall x, y \in S;\ x \le y \Leftrightarrow f(x) \le f(y).$$

Pak:



Důkaz. (Indukcí přes hloubku sítě.)

1. Samotný komparátor je datově necitlivý vzhledem k jakékoli m.r. funkci f (pro $x, y \in \mathcal{S}$, komparátor vymění $f(x)$ a $f(y)$ $\Leftrightarrow$ vymění x a y)



2. **Indukční krok:**

Nese-li určitý vodič v řadící síti hodnotu x_i, *když je na vstupu posloupnost* X, *pak tentýž vodič nese hodnotu* $f(x_i)$, *když je na vstupu* $f(X)$.

0-1 Řadící Lemma

Lemma 3

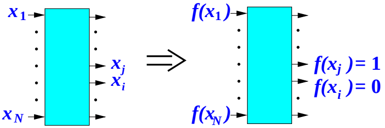
Jestliže datově necitlivý řadící algoritmus dokáže setřídit **libovolnou binární** *vstupní posloupnost, pak dokáže setřídit* **libovolnou** *vstupní posloupnost.*

Důkaz. (Sporem.)

- Předpokládejme, že řadící síť třídí správně všechny binární posloupnosti.
- Předpokládejme ale, že nesetřídí správně nebinární posloupnost $X = [x_1, \ldots, x_N]$
  $\Rightarrow \quad \exists\, x_i < x_j$ v X takové, že x_j je na výstupu umístěn před x_i.
- Definujme

$$f(z) = \begin{cases} 0, & \text{jestliže} z \leq x_i, \\ 1, & \text{jestliže} z > x_i. \end{cases}$$

- f je monotonně rostoucí & $f(X)$ = binární vstupní posloupnost & platí Lemma 2
  $\Rightarrow \quad f(x_j) = 1$ je na výstupu umístěn před $f(x_i) = 0$, je-li na vstupu $f(X)$
  $\Rightarrow \quad$ spor. $\blacksquare$



Sudo-liché transpozice (paralelní BubbleSort) na 1-D mřížce

Algorithm $\textsc{EOTSort}(X = \langle x_1, \ldots, x_N \rangle)$ on $M(N)$

---

for $j := 1 \ldots \lceil N/2 \rceil$ **do_sequentially**

  begin

    for all $i := 1, 3, .., 2\lfloor N/2 \rfloor - 1$ **do_in_parallel** $C\&E(x_i, x_{i+1})$;

    for all $i := 2, 4, .., 2\lfloor (N-1)/2 \rfloor$ **do_in_parallel** $C\&E(x_i, x_{i+1})$;

  end

Sudo-lichá transpozice na 1-D mřížce (pokr.)

Věta 4

$\textsc{EOTSort}$ setřídí N čísel na mřížce $M(N)$ v N krocích.

Důkaz. (Pomocí 0-1 Řadící Lemmy.)

- Uvažujme libovolnou binární posloupnost k hodnot 1 a $N-k$ hodnot 0, $1 \leq k \leq N-1$.
- 1. jednička zprava má napravo pouze nuly $\Rightarrow$ začne se pohybovat doprava nejpozději v 2. kroku a setrvá v pohybu bez přerušení, dokud nedosáhne cílové pozice N.
- Podobně, 2. jednička zprava se dá do pohybu doprava nejpozději v 3. kroku.
- Konečně, poslední k-tá jednička zprava se dá do pohybu směrem k pozici $N-k+1$ nejpozději v kroku $k+1$.
- Tudíž, v nejhorším případě je celkový počet kroků $k+(N-k)=N$. ∎

Sudo-lichá transpozice na 1-D mřížce (pokr.)

Škálovatelnost

Důsledek 5 (vět 1 a 4)

EOTSORT setřídí N čísel na mřížce $M(p)$, $p < N$, v

$$T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O(N) \qquad \text{krocích.}$$

Důkaz. Věta 4 dává $\tau(N) = N$ a dle Věty 1 platí

$$T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(p\frac{N}{p}\right). \blacksquare$$

Sudo-lichá transpozice na 1-D mřížce (pokr.)

Závěr

- Stejně špatná škálovatelnost jako u $\textsc{NaivePRAMSort}$:
  - Cenová optimalita: $C(N,p) = O(N \log N)$, pouze je-li $p = O(\log N)$. Např.: $C(N,N) = O(N^2)$.
  - Paralelní čas je $O(N)$, pouze je-li $p = \Omega(\log N)$ čili pro $p \in \{\log N, \dots, N\}$.
- Nicméně: $\textsc{EOTSort}$ je topologicky optimální!!!!

ShearSort na 2-D mřížce $M(n,m)$, kde $N = nm$

Algorithm ShearSort($X = \langle x_1, \ldots, x_N \rangle$)

for $i = 1, \ldots, 2\log n + 1$ **do_sequentially**
  if (i je liché)
    then setřiď všechny řádky střídavými směry (* řádková fáze *)
    else setřiď všechny sloupce směrem dolů (* sloupcová fáze *)

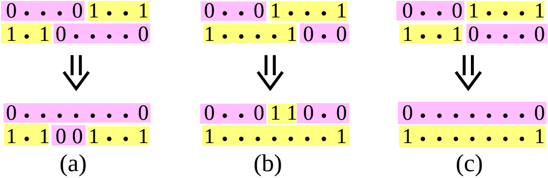
Výrok 6

1 řádková a 1 sloupcová fáze zmenší počet nečistých řádek na nejméně polovinu.

Důkaz. (Pomocí 0-1 Řadící Lemmy.) Uvažujme jakoukoli binární $n \times m$ matici.

- V matici obecně $\exists$ **čisté jedničkové**, **čisté nulové** a **nečisté** řádky.
- $\exists$ pouze 3 případy aplikace 1 řádkové + sloupcové fáze na 2 **sousední špinavé řádky**.
- Ve všech těchto 3 případech klesne počet špinavých řádků v každé dvojici na 1 nebo na 0. ∎

ShearSort na 2-D mřížce (pokr.)



Věta 7

ShearSort třídí hadovitě nm čísel na $M(n,m)$

$$\text{v } \lceil \log n \rceil + 1 \text{ řádkových fázích a } \lceil \log n \rceil \text{ sloupcových fázích.}$$

Důkaz. Na začátku může vstupní matice v nejhorším případě obsahovat n nečistých řádků. Po $\log n$ řádkových a $\log n$ sloupcových fázích, zbývá max. 1 nečistý řádek $\Rightarrow$ nutná ještě 1 řádková fáze. ∎

SHEARSORT na 2-D mřížce (pokr.)

Pět fází SHEARSORTu na $M(4,4)$

3	11	6	16
8	1	5	10
14	7	12	2
4	13	9	15

(0)

3	6	11	16
10	8	5	1
2	7	12	14
15	13	9	4

(1)

2	6	5	1
3	7	9	4
10	8	11	14
15	13	12	16

(2)

1	2	5	6
9	7	4	3
8	10	11	14
16	15	13	12

(3)

1	2	4	3
8	7	5	6
9	10	11	12
16	15	13	14

(4)

1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

(5)

Poznámka 8

Modifikovaný algoritmus s tříděním řádků stejným směrem nefunguje.

ShearSort na 2-D mřížce (pokr.)

Škálovatelnost ShearSortu na $M(\sqrt{p},\sqrt{p})$

Důsledek 9 (vět 1 a 7)

Algoritmus ShearSort setřídí N čísel na 2-D mřížce $M(\sqrt{p},\sqrt{p})$ v

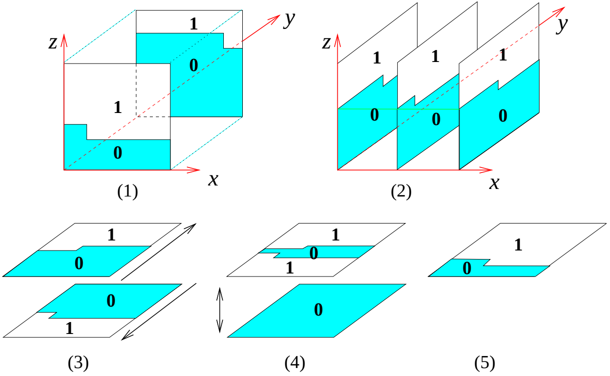
$$T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(\log p \frac{N}{\sqrt{p}}\right)$$ paralelních C&E krocích a

$$\psi_1(p) = p^{\alpha\sqrt{p}} \quad \text{a} \quad \psi_2(N) = \left(\frac{\log N}{\log\log N}\right)^2 \quad \text{a} \quad \psi_3(N) = N.$$

3DSort: Lexikografické řazení na 3-D mřížce $M(n,n,n)$

Algorithm $3\text{DS}\textsc{ort}(X=[x_1,\ldots,x_N])$ na 3-D mřížce $M(n,n,n)$, kde $N=n^3$

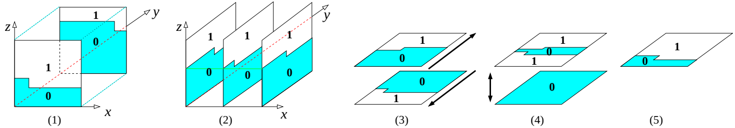
Fáze 1. Setřiď všechny xz-roviny v zx-pořadí.
Fáze 2. Setřiď všechny yz-roviny v zy-pořadí.
Fáze 3. Setřiď všechny xy-roviny v yx-pořadí **střídavě** ve směru y.
Fáze 4. Proveď jednu licho-sudou a jednu sudo-lichou transpozici ve všech sloupcích.
Fáze 5. Setřiď všechny xy-roviny v yx-pořadí.



3DSort: Lexikografické řazení na 3-D mřížce $M(n,n,n)$ (pokr.)

Věta 10

Alg. $\text{3DS}\textsc{ort}$ na mřížce $M(n,n,n)$ setřídí $N = n^3$ čísel lexikograficky v zyx pořadí v $O(\sqrt[3]{N}\log N)$ paralelních C&E krocích, je-li v rovinách použit $\textsc{ShearSort}$.



Důkaz. (Pomocí 0-1 Řadící Lemmy.) Uvažujme libovolnou binární vstupní posloupnost.

- Po fázi 1, v každé xz-rovině, existuje nejvýše 1 nečistý řádek, a proto:
  - jakékoli 2 yz-roviny se mohou lišit v **nejvýše** n nulách,
  - a všech n yz-rovin obsahuje ve svých nečistých řádcích souhrnně **nejvýše** n^2 prvků.
- Tudíž, po fázi 2, všechny nečisté řádky mohou překlenout **nejvýše** 2 xy-roviny.
- Je-li nečistá xy-rovina pouze jedna, jdeme přímo na fázi 5 a jsme hotovi.
- $\exists$-li 2 nečisté xy-roviny, fáze 3 a 4 vyčistí aspoň 1 z nich a fáze 5 dokončí řazení. ∎

3DSort: Lexikografické řazení na 3-D mřížce $M(n, n, n)$ (pokr.)

Škálovatelnost v $M(\sqrt[3]{p}, \sqrt[3]{p}, \sqrt[3]{p})$ (je-li v rovinách použit ShearSort)

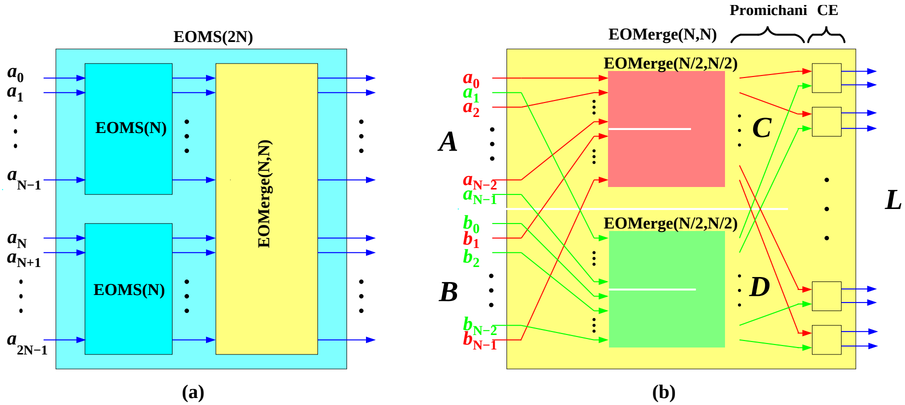
$$T(N, p) = O\left(\frac{N}{p} \log \frac{N}{p}\right) + O\left(\frac{N}{\sqrt[3]{p^2}} \log p\right)$$

$$\text{a} \quad \psi_1(p) = p^{\alpha \sqrt[3]{p}} \quad \text{a} \quad \psi_2(N) = \left(\frac{\log N}{\log \log N}\right)^3 .$$

Řazení na hyperkubických sítích

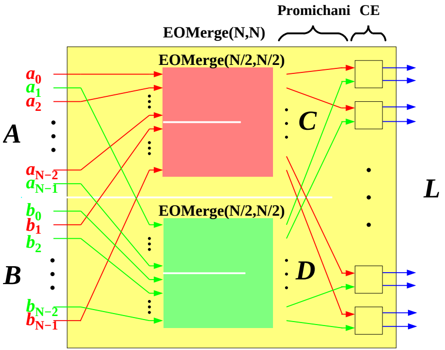
- Základem je klasický sekvenční MergeSort.
- Paralelizace = Batcherovy algoritmy:
    - Sudo-Lichý MergeSort,
    - Sudo-Sudý MergeSort,
    - Bitonický MergeSort.
- Realizace:
    - řadící nepřímá síť,
    - motýlek,
    - hyperkrychle,
    - simulace na mřížkách.

Sudo-Lichý MergeSort (EOMS)



$$\text{EOMS}(a_0, \ldots, a_{2N-1}) = \text{EOM}\textsc{erge}(\text{EOMS}(a_0, \ldots, a_{N-1}), \text{EOMS}(a_N, \ldots, a_{2N-1}))$$

Sudo-Liché Sloučení (EOMERGE)



$$L = \text{EOMERGE}(A, B) = \text{Parovane\_CE}(\text{Promichani}(\text{EOMERGE}(\text{even}(A), \text{odd}(B)), \text{EOMERGE}(\text{odd}(A), \text{even}(B)))).$$

$$C = \text{EOMERGE}(\text{even}(A), \text{odd}(B))$$
$$D = \text{EOMERGE}(\text{odd}(A), \text{even}(B))$$
$$L' = \text{Promichani}(C, D)$$
$$L = \text{EOMERGE}(A, B) = \text{Parovane\_CE}(L')$$

Důkaz správnosti Sudo-Lichého sloučení

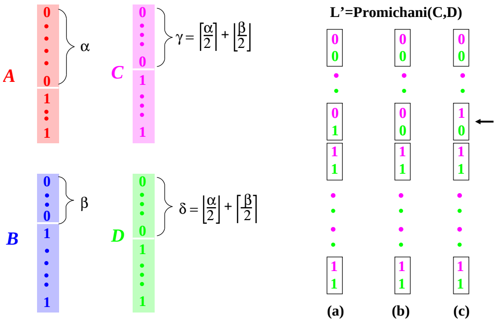
Věta 11

EOMERGE sloučí 2 setříděné posloupnosti A, B délky N

v $\log N + 1$ paralelních C&E krocích použitím $N(\log N + 1)$ komparátorů.

Důkaz. (Pomocí 0-1 Řadící Lemmy.) Věta platí pro $N = 1$. Nechť $N = 2^k$, $k \geq 1$.
Nechť $\gamma = \lceil \alpha/2 \rceil + \lfloor \beta/2 \rfloor$ a $\delta = \lfloor \alpha/2 \rfloor + \lceil \beta/2 \rceil \quad \Rightarrow \quad |\gamma - \delta| \leq 1$

$\Rightarrow$ **počet nul** v C a D se může lišit **nejvýše o jedna.**



Časová a cenová složitost EOMERGE

Nechť

- $d_{\mathrm{m}}(2N)$ = hloubka EOMERGE(N, N),
- $d_{\mathrm{m}}(2) = 1$ (1 komparátor).

Potom

- EOMERGE(N, N) je rekurzivní a každý stupeň rekurze přidá právě 1 sloupec komparátorů:

$$d_{\mathrm{m}}(2N) = d_{\mathrm{m}}(N) + 1$$

$\Rightarrow$

$$d_{\mathrm{m}}(2N) = \log N + 1 = \log(2N).$$

Cenová složitost EOMERGE = počet komparátorů

$$c_{\mathrm{m}}(2N) = N d_{\mathrm{m}}(2N) = N \log(2N)$$

■

Časová a cenová složitost EOMS

Věta 12

EOMS(N) dokáže setřídit N čísel

v $O(\log^2 N)$ paralelních C&E krocích s použitím $O(N\log^2 N)$ komparátorů.

Důkaz. Nechť

- $d_s(N)$ = hloubka EOMS(N),
- $c_s(N)$ = počet komparátorů EOMS(N).

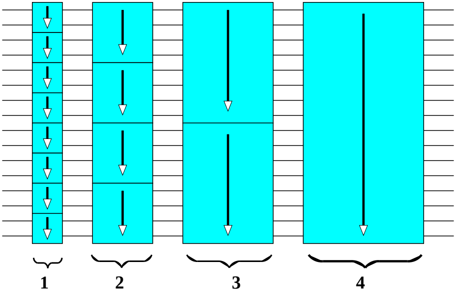
Potom

- $d_s(2) = 1$
  & $d_s(2N) = d_s(N) + d_m(2N) = d_s(N) + \log(2N)$
  $\Rightarrow$
  $$d_s(N) = \log N(\log N + 1)/2 = O(\log^2 N)$$
- $c_s(2) = 1$
  & $c_s(2N) = 2c_s(N) + c_m(2N) = 2c_s(N) + N\log(2N)$
  $\Rightarrow$
  $$c_s(N) = Nd_s(N)/2 = O(N\log^2 N).$$

■

Rozvinutí sítě EOMS

- EOMS zachází se vstupní posloupností jako s posloupností N dvojic.
- Po průchodu komparátorem, každá dvojice se stane setříděnou podposloupností délky 2.
- Tyto podposloupnosti se pak sloučí do $N/2$ setříděných podposloupností délky 4,
  pak do $N/4$ podposloupností délky 8, atd.
- V posledním slučovacím kroku, 2 setříděné rostoucí posloupnosti délky N jsou sloučeny do výsledné posloupnosti délky $2N$.



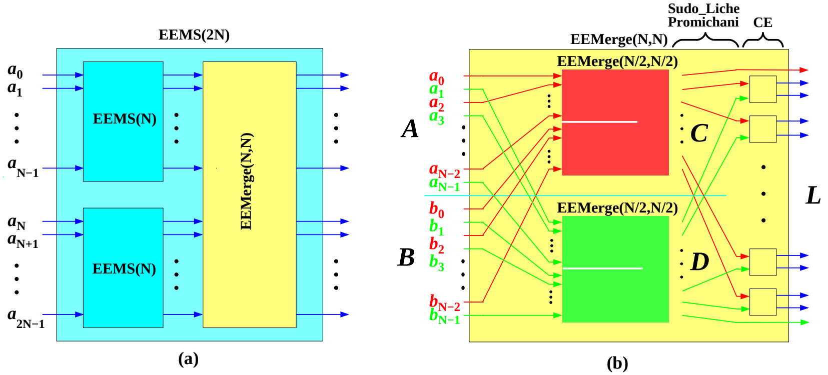
Sudo-Sudý MergeSort (EEMS)

$\mathrm{EEMS}(a_0,\ldots,a_{2N-1}) = \textsc{EEMerge}(\mathrm{EEMS}(a_0,\ldots,a_{N-1}),\mathrm{EEMS}(a_N,\ldots,a_{2N-1}))$

$\textsc{EEMerge}(A,B) = \text{Parovane\_CE}(\text{Sudo\_Liche\_Promichani}(\mathrm{C},\mathrm{D}))$

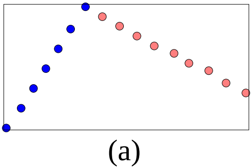
$C = \textsc{EEMerge}(\mathrm{even}(A),\mathrm{even}(B))$

$D = \textsc{EEMerge}(\mathrm{odd}(A),\mathrm{odd}(B))$

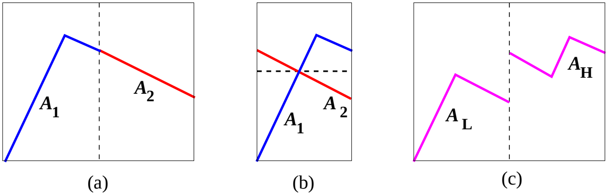


Bitonické posloupnosti

1 **údolí** a 1**vrchol** nezávisle na rotacích.



Bitonické rozdělení



Bitonické posloupnosti (pokr.)

Lemma 13

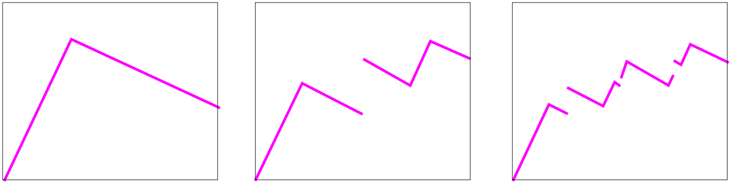
Je-li $A = a_0, a_1, \ldots, a_{2N-1}$ *bitonická, její* **bitonické rozdělení** *je* $A' = A_\mathrm{L} A_\mathrm{H}$, *kde*

$$\begin{aligned} A_\mathrm{L} &= \min(a_0, a_N), \min(a_1, a_{N+1}), \ldots, \min(a_{N-1}, a_{2N-1}), \\ A_\mathrm{H} &= \max(a_0, a_N), \max(a_1, a_{N+1}), \ldots, \max(a_{N-1}, a_{2N-1}). \end{aligned}$$

Pak: *(1)* A_L *a* A_H *jsou opět bitonické.*
(2) Každé číslo v A_L *je menší než libovolné číslo v* A_H.

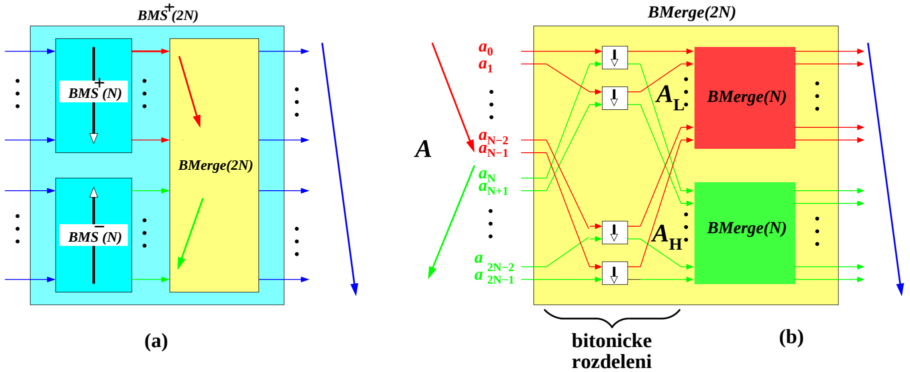
Pozorování

Rekurzivní aplikace bitonického rozdělení na **bitonickou** A ji změní na **monotonní**!!



$\Rightarrow$ Bitonické Sloučení $\Rightarrow$ Bitonický MergeSort

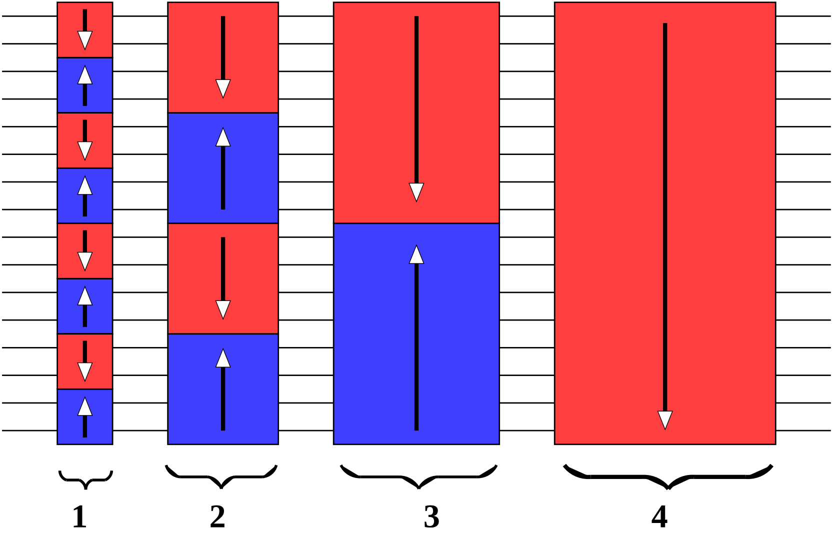
Bitonický MergeSort (BMSort)



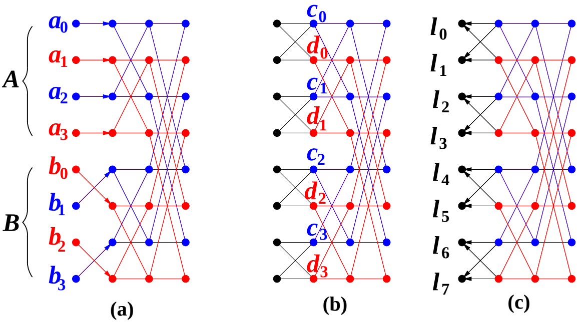
$$\mathrm{BMS}^{+}(a_0,\ldots,a_{2N-1}) = \mathrm{BMERGE}(\mathrm{BMS}^{+}(a_0,\ldots,a_{N-1})\mathrm{BMS}^{-}(a_N,\ldots,a_{2N-1}))$$

$\mathrm{BMERGE}(A) = \mathrm{BMERGE}(A_{\mathrm{L}})\mathrm{BMERGE}(A_{\mathrm{H}})$, kde $(A_{\mathrm{L}}A_{\mathrm{H}}) =$ Bitonicke_Rozdeleni(A)

Rozvinutí bitonické řadící sítě



Implementace EOMS na topologii motýlek oBF_n



Realizace $L = \text{EOMERGE}(A, B)$ na oBF_n =

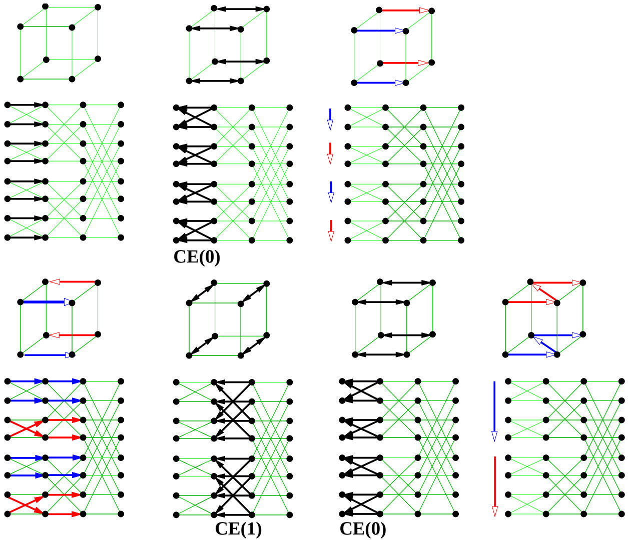
(a) Přenos 1.stupněm A v horní půlce rovně a B v dolní půlce křížem.

(b) Rekurzivní konstrukce:

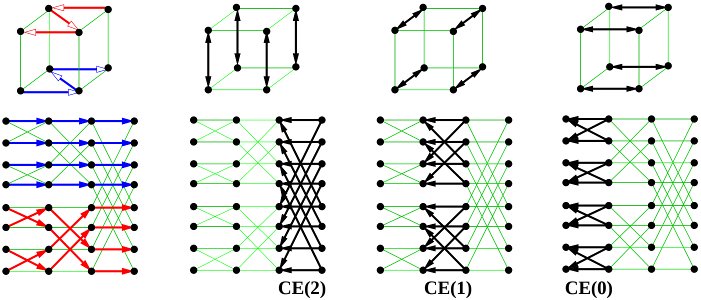
- $C = \text{EOMERGE}(\text{even}(A), \text{odd}(B))$ v MODRÉM oBF_{n-1},
- $D = \text{EOMERGE}(\text{odd}(A), \text{even}(B))$ v ČERVENÉM oBF_{n-1}.

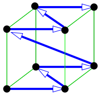
(c) Konstrukce $L = \text{Parovane\_CE}(\text{Promichani}(C, D))$ zpětným průchodem **prvním** stupněm motýlka.

Sudo-Lichý MergeSort 8 čísel na Q_3



Sudo-Lichý MergeSort 8 čísel na Q_3 (pokr.)



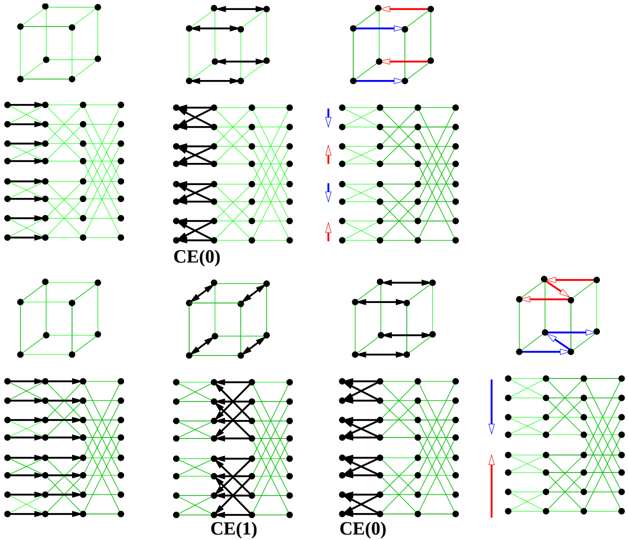


Pozorování:

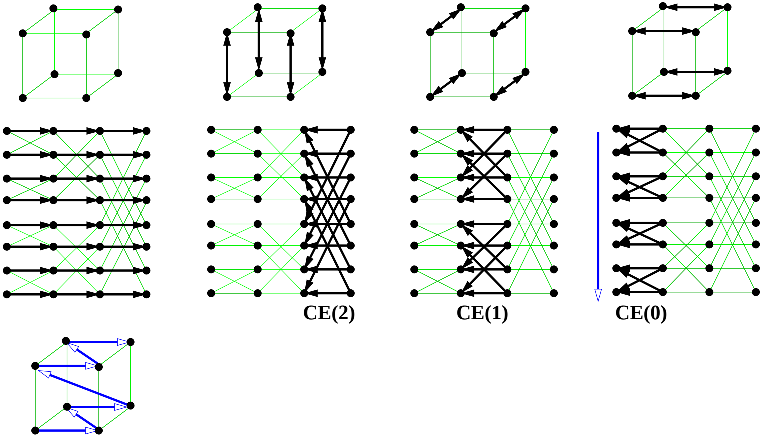
Každá druhá podposloupnost je otočena (rostoucí $\rightarrow$ klesající)

ALE to je přesně to, co Bitonický MergeSort dělá zadarmo!!!!

CubeBMS: Bitonický MergeSort 8 čísel na Q_3



CubeBMS: Bitonický MergeSort 8 čísel na Q_3 (pokr.)



Časová složitost a škálovatelnost CubeBMS

Věta 14

Algoritmus CubeBMS pro $N = 2^n$ čísel na Q_n vyžaduje $T(N, N) = O(\log^2 N)$ paralelních C&E kroků. Pro $p = 2^k$, $k < n$,

$$T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(\frac{N}{p}\log^2 p\right)$$

a $\quad \psi_1(p) = p^{\alpha \log p} \quad$ a $\quad \psi_2(N) = 2^{\sqrt{\alpha' \log N}}$

Implementace CubeBMS na PRAM

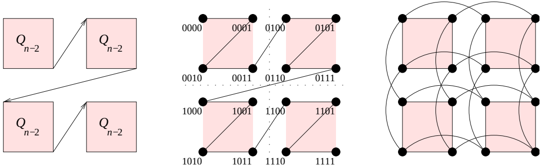
Triviální.

MeshBMS: Simulace CubeBMS na 2-D mřížce

- **Peanova křivka** indukuje vnoření $(\varphi, \xi) : Q_n \xrightarrow{\text{emb}} M_n$, kde
  1. $M_n = M(2^{\frac{n}{2}}, 2^{\frac{n}{2}})$ pro sudá n,
  2. $M_n = M(2^{\frac{n-1}{2}}, 2^{\frac{n+1}{2}})$ pro lichá n,

  a dilatace hyperkubické hrany dimenze i je $2^{\lfloor \frac{i}{2} \rfloor}$, $0 \leq i \leq n-1$, protože

  mřížková vzdálenost mezi $\varphi(u)$ a $\varphi(u \oplus 2^i)$ je $d_i = 2^{\lfloor \frac{i}{2} \rfloor}$ pro všechna $u \in \mathcal{B}^n$.



- CubeBMS na M_n vyžaduje celkově $\frac{n(n+1)}{2}$ komunikací na vzdálenosti d_i pro realizaci operací C&E (nebo M&S), kde i probíhá posloupnost

$$[0, 1, 0, 2, 1, 0, 3, 2, 1, 0, \ldots, n-1, n-2, \ldots, 1, 0].$$

MeshBMS: Simulace CubeBMS na 2-D mřížce (pokr.)

Věta 15

Pro $N = 2^n$, MeshBMS na M_n setřídí N čísel v pořadí Peanova indexování a celkový počet paralelních komunikací **mezi sousedy** je $T(N,N) \approx 7\sqrt{N}$.

Důkaz. Předpokládejme, že n je sudé (důkaz pro liché n je velmi podobný).

Protože $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$, dostáváme

$$\begin{aligned} T(N,N) &= \sum_{i=0}^{n-1}\sum_{j=0}^{i} d_j = \sum_{i=0}^{n-1}\sum_{j=0}^{i} 2^{\lfloor \frac{j}{2} \rfloor} = \sum_{i=1}^{n/2} 4(2^i - 1) - (2^{\frac{n}{2}} - 1) \\ &= 4 \cdot 2^{\frac{n}{2}+1} - 8 - 2n - 2^{\frac{n}{2}} + 1 = 7\sqrt{N} - O(\log N). \blacksquare \end{aligned}$$

MESHBMS: Simulace CUBEBMS na 2-D mřížce (pokr.)

Lemma 16

Permutace *Peanova indexování na řádkové vyžaduje kolem* $4\sqrt{N}/3 - \sqrt[4]{N}$ *paralelních komunikací mezi sousedy na plně-duplexní SF mřížce s* XY *přepínáním.*

Důsledek 17 (vět 1 a 15)

Nechť $p = 2^{2k}$, $2k < n$, a $N = 2^n$. Předpokládejme, že komunikační latence výměny k čísel mezi sousedními procesory je téhož řádu jako časová složitost operace M&S 2 setříděných posloupností velikosti k. Pak algoritmus MESHBMS setřídí N čísel na 2-D mřížce $M(\sqrt{p}, \sqrt{p})$ v

$$T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(\frac{N}{\sqrt{p}}\right) \quad \text{paralelních C\&E krocích}$$

$$\psi_1(p) = 2^{\sqrt{p}} \quad \text{a} \quad \psi_2(N) = \log^2 N.$$