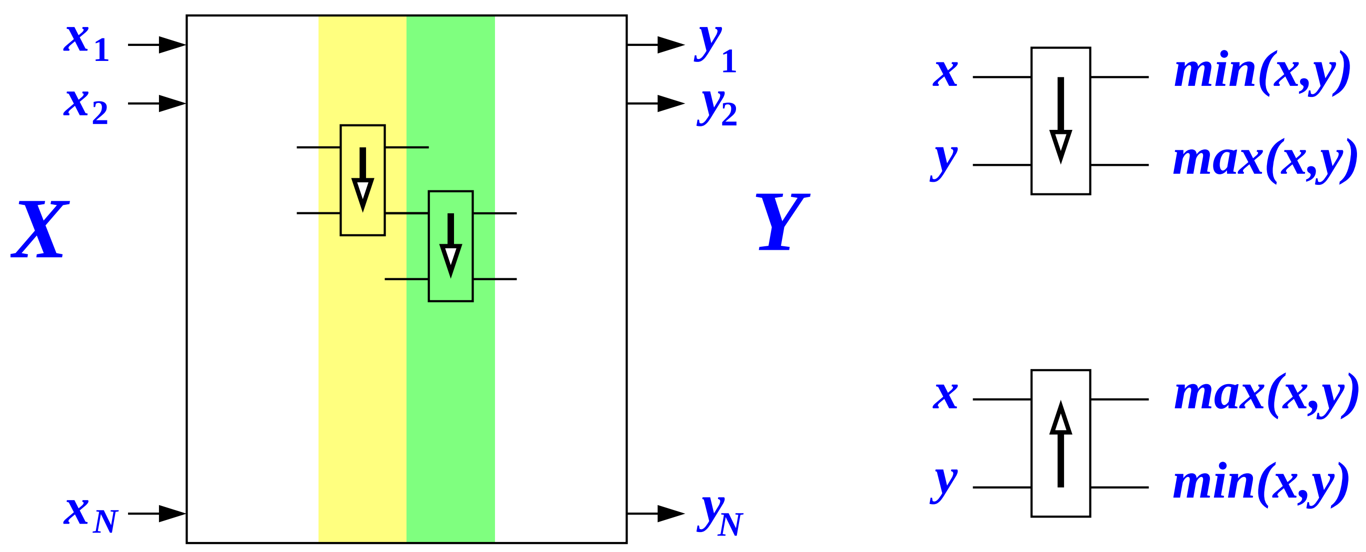
Přednáška #8: **Paralelní třídící algoritmy**

Úvod do paralelního třídění založeného na operaci porovnání

- Tato přednáška: Paralelní deterministické třídící algoritmy založené na operaci porovnej-a-vyměň (Compare-and-Exchange, C&E).
- N = délka vstupní posloupnosti.
- Spodní mez na počet C&E operací pro setřídění N čísel je $\Omega(N \log N)$.
- $\exists$ optimální sekvenční algoritmy (HeapSort, MergeSort, QuickSort (někdy), ...).
- $\exists$ časově a cenově optimální PRAM // třídící algoritmy: Coleův MergeSort. Třídí N čísel na N-uzlovém EREW PRAM v $O(\log N)$ // C&E krocích.
- $\exists$ asymptoticky optimální // třídící algoritmy na mřížkách: Třídí N čísel na N-uzlové 2-D mřížce v $k\sqrt{N}$ krocích, $2 < k < 3$.
- Nejpraktičtější // třídící algoritmus na hyperkubických sítích: Batcherův MergeSort. Třídí N čísel na N-uzlové síti v $O(\log^2 N)$ krocích.
- $\exists$ // třídící algoritmy pro třídění N čísel na N-uzlové hyperkubické síti v $O(\log N(\log\log N))$ krocích. Skrytá konstanta je velmi vysoká.
- $\exists$ asymptoticky cenově optimální // třídící sítě hloubky $O(\log N)$: expandéry. Velmi komplikované, obrovské skryté konstanty, zajímavé pouze teoreticky.

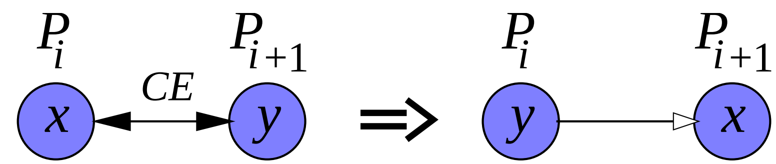
Nepřímé třídící sítě



(a) (b)

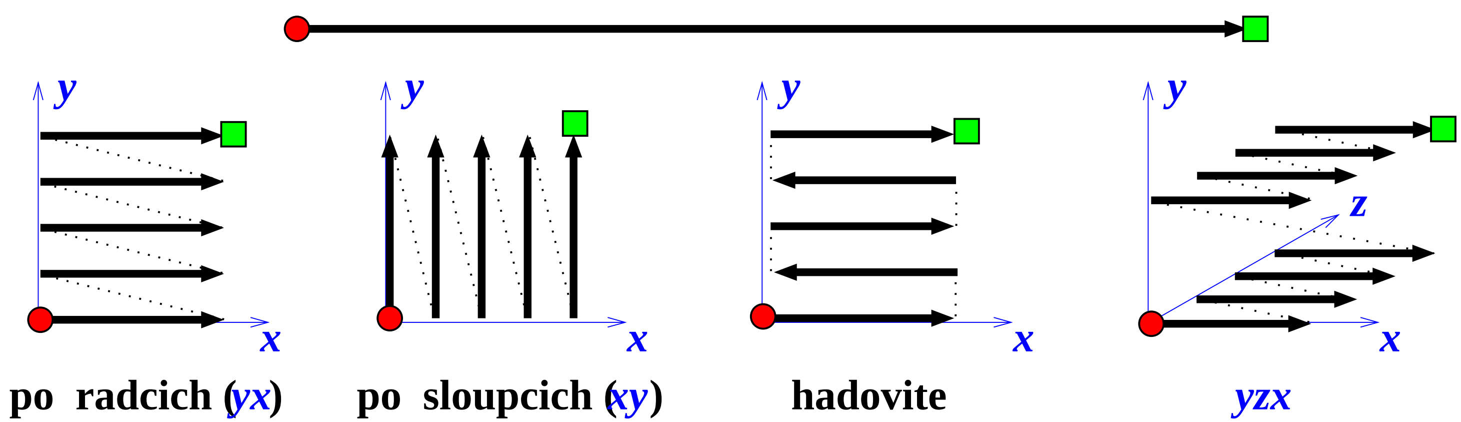
- (a) $=$ Třídící síť $=$ levo-pravá síť složená ze sloupců komparátorů (jako MIN).
- (b) $=$ Komparátor $=$ HW implementace operace C&E (vzestupně, sestupně).
- Nesetříděná vstupní posloupnost $X = [x_1, \ldots, x_N]$ je permutována na setříděnou výstupní posloupnost (klesající, rostoucí, bitonickou)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

Přímé třídící sítě

- Paralelní C&E v přímé propojovací síti při $x > y$: 

- Topologie: PRAM, hyperkrychle a hyperkubické sítě, mřížky.
- Výběr vhodného lineárního indexování procesorů $P_1, \ldots, P_N$:
  - Má vliv na složitost C&E třídění pro danou topologii.
  - Je triviální, pokud $\exists$ hamiltonovská cesta.
  - V ostatních případech $\exists$ indexování s dilatací nejvýše 3 (viz přednáška 5).

Různá schemata pro lineární indexování procesorů v mřížkách.



- Přímý třídící alg. $=$ posloupnost dokonalých párování procesorů (perfect matchings) odvozených z indexování.
  - 1 dokonalé párování se skládá z $\lfloor p/2 \rfloor$ disjunktních dvojic.
  - Datově necitlivé třídění $\Longrightarrow$ párování nejsou závislá na vstupních hodnotách.

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 třídící 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 třídících algoritmů

Věta 1. *Nechť $\tau(N) = T(N,N)$ je počet paralelních C&E kroků třídění N čísel na N-procesorové síti G. Pak počet par. C&E kroků třídění N čísel na p-procesorové síti G je*

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

Naivní PRAM třídící algoritmus

Algorithm $\textsc{NaivePRAMsort}(X = [x_1, \dots, x_N])$ 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 setříděním 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 $2N/p$.
$\dots$ atd.
($\log p$)-tá fáze: 1 zbývající P sloučí poslední 2 podposloup. o velikosti $N/2$.

- $T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(\frac{N}{p} + \frac{2N}{p} + \cdots + N\right) = O\left(\frac{N}{p}\log\frac{N}{p} + N\right)$
- $T_{\min}(N,p) = T(N,N) = O(N)$
- $E(N,p) = \Theta\left(\frac{\log N}{\log(N/p)+p}\right) = \Theta\left(\frac{\log N}{\log N+p}\right)$
- $\psi_1(p) = (2^p/p)^{O(1)} = \Theta(\exp(p))$ a $\psi_2(N) = \log N$ a $\psi_3(N) = \log N$

Závěr

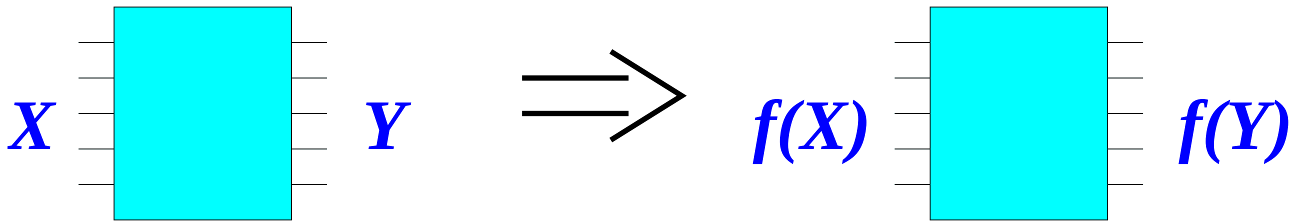
$\textsc{NaivePRAMSort}$ má špatnou škálovatelnost: čas je řádově stejný pro $p \in \{\log N, \dots, N\}$!!!!!!

Datově necitlivé třídění a 0-1 Třídící Lemma

Lemma 2. *Nechť* f = *monotonně rostoucí* *funkce na lineárně uspořádané množině* $\mathcal{S}$,

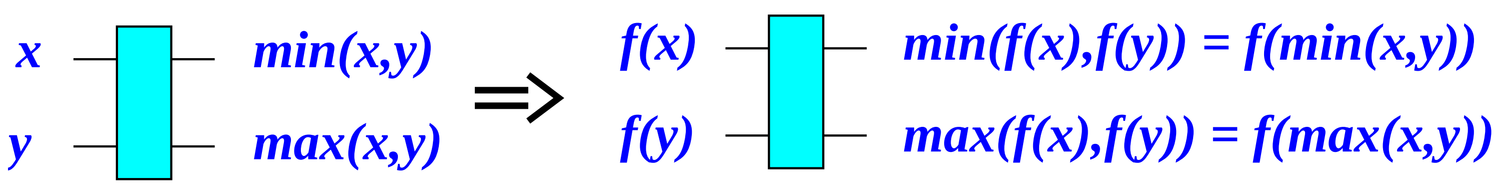
$$\text{\textit{čili}} \quad \forall\, x, y \in S;\ x \le y \Leftrightarrow f(x) \le f(y).$$

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 $f(y)$ $\iff$ vymění x a y)



2. **Indukční krok:**

> *Nese-li určitý vodič v třídící 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 Třídící Lemma

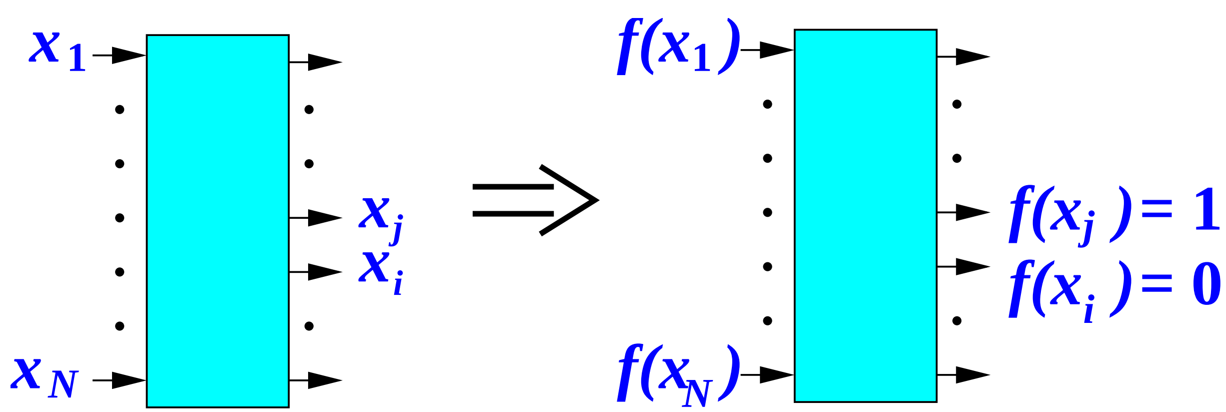
Lemma 3. *Jestliže datově necitlivý třídící algoritmus dokáže setřídit libovolnou binární vstupní posloupnost, pak dokáže setřídit libovolnou vstupní posloupnost.*

Důkaz. (Sporem.)

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

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

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



Sudo-lichá transpozice (paralelní bubble-sort) 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

Věta 4. $\textsc{EOTSort}$ *setřídí* N *čísel na mřížce* $M(N)$ *v* N *krocích.*
Důkaz. (Pomocí 0-1 Třídící Lemmy.)

- Předpokládejme jakoukoli vstupní posloupnost k hodnot 1 a $N-k$ hodnot 0, $1 \leq k \leq N-1$.
- 1. jednička zprava má napravo pouze nuly $\Longrightarrow$ začne se pohybovat doprava nejpozději v 2. kroku a setrvá v pohybu bez přerušení, dokud nedosáhne své konečné pozice.
- 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 $k + (N-k) = N$. ♣

Škálovatelnost

Důsledek 5. (vět 1 a 4.) EOTSort *setřídí* N *čísel na mřížce* $M(p)$, $p < N$, *v*

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

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

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

Závěr

- Stejně špatná škálovatelnost jako u 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 $p \in \{\log N, \ldots, N\}$.
- Nicméně: EOTSort je topologicky optimální!!!!

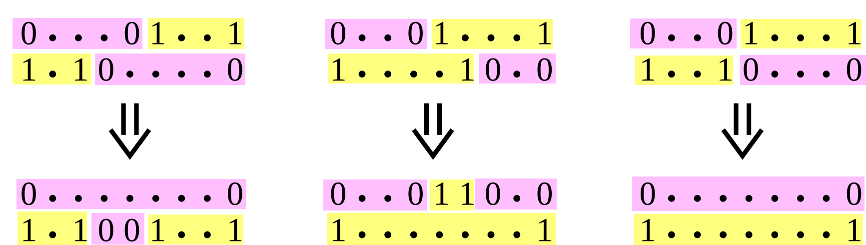
ShearSort na 2-D mřížce

Algorithm $\textsc{ShearSort}(X = \langle x_1, \ldots, x_N \rangle)$ na 2-D mřížce $M(n, m)$, kde $N = nm$

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 Třídící 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 případech klesne počet špinavých řádků ze 2 na 1 nebo na 0. ♣



(a) (b) (c)

Věta 7. $\textsc{ShearSort}$ *třídí hadovitě* nm *čísel na* $M(n, m)$
v $\lceil \log n \rceil + 1$ *řádkových fázích a* $\lceil \log n \rceil$ *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 $\Longrightarrow$ nutné ještě 1 třídění řádků. ♣

Pět fází ShearSortu na $M(4,4)$

$$\underset{(0)}{\begin{array}{cccc} 3 & 11 & 6 & 16 \\ 8 & 1 & 5 & 10 \\ 14 & 7 & 12 & 2 \\ 4 & 13 & 9 & 15 \end{array}} \quad \underset{(1)}{\begin{array}{cccc} 3 & 6 & 11 & 16 \\ 10 & 8 & 5 & 1 \\ 2 & 7 & 12 & 14 \\ 15 & 13 & 9 & 4 \end{array}} \quad \underset{(2)}{\begin{array}{cccc} 2 & 6 & 5 & 1 \\ 3 & 7 & 9 & 4 \\ 10 & 8 & 11 & 14 \\ 15 & 13 & 12 & 16 \end{array}} \quad \underset{(3)}{\begin{array}{cccc} 1 & 2 & 5 & 6 \\ 9 & 7 & 4 & 3 \\ 8 & 10 & 11 & 14 \\ 16 & 15 & 13 & 12 \end{array}} \quad \underset{(4)}{\begin{array}{cccc} 1 & 2 & 4 & 3 \\ 8 & 7 & 5 & 6 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 13 & 14 \end{array}} \quad \underset{(5)}{\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 14 & 13 \end{array}}$$

Poznámka 8. *Modifikovaný algoritmus s tříděním řádků stejným směrem nefunguje.*

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*

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

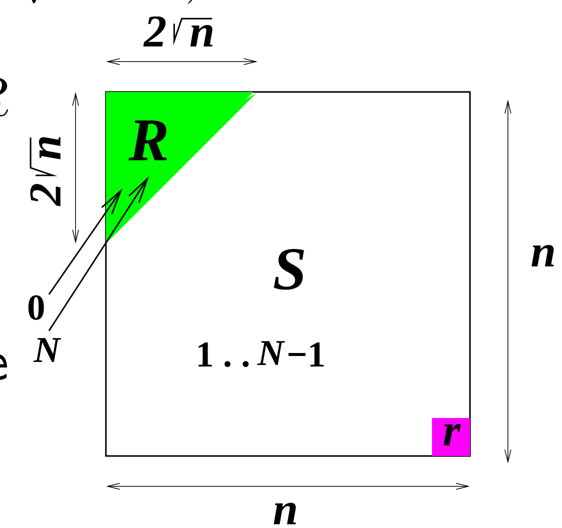
$$\psi_1(p) = p^{\alpha\sqrt{p}} \quad \textit{a} \quad \psi_2(N) = \left(\frac{\log N}{\log\log N}\right)^2 \quad \textit{a} \quad \psi_3(N) = N.$$

Spodní mez složitosti hadovitého třídění na 2-D mřížce

Věta 10. *Nechť* $N = n^2$*. Jakýkoliv datově necitlivý hadovitý třídící algoritmus na mřížce* $M(n,n)$ *vyžaduje*

$$\textit{nejméně } \max(2n-2, 3n-2\sqrt{n}-4) \textit{ kroků.}$$

Důkaz. (Konstruktivní) Rozděl $M(n,n)$ na oblasti R a S dle obrázku:
Oblast R obsahuje $2n$ čísel 0 nebo N.
Oblast S obsahuje $N-2n$ čísel $1,\dots,N-1$.
Vzdálenost mezi oblastí R a dolním pravým rohem je $2(n-1)-2\sqrt{n} = 2n-2\sqrt{n}-2$.



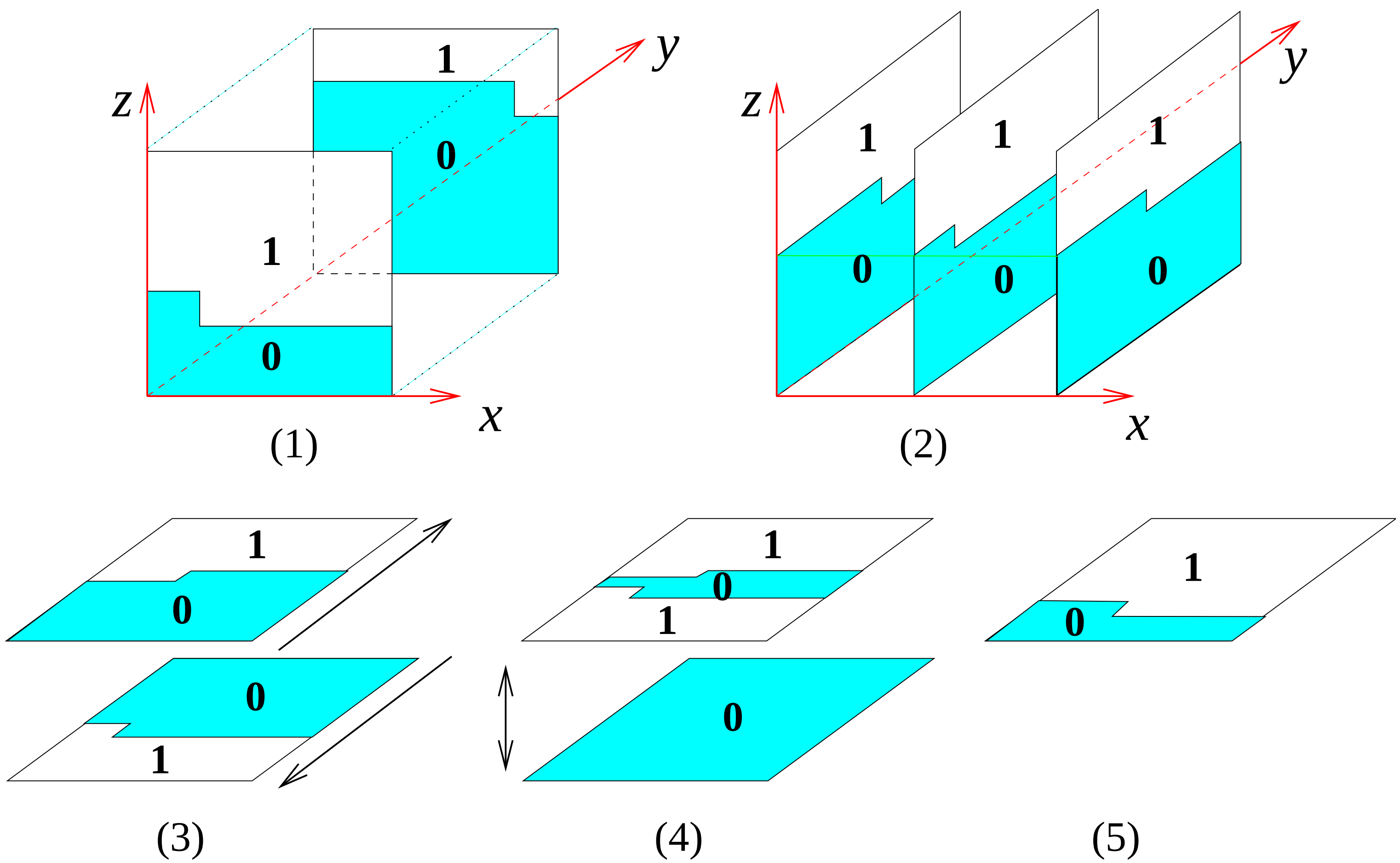
- r = obsah spodního pravého rohu mřížky v kroku $2n-2\sqrt{n}-3$ (nedotčený čímkoli z R).
- Nechť $C(q)$, $0 \le q \le 2n$, je číslo správného sloupce pro r, jestliže R obsahuje q nul a $2n-q$ hodnot N.
- Had bude končit buď v levém nebo v pravém spodním rohu, v závislosti na paritě n.
- q probíhá skrz $\{0,\dots,2n\}$ $\implies$ $C(q)$ probíhá skrz $\{1,\dots,n\}$ dvakrát
  $\implies$ pro n liché nebo sudé, můžeme vždy nalézt q' takové, že $C(q') = 1$.
- Je-li $q = q'$, musíme po kroku $2n-2\sqrt{n}-3$ provést nejméně $n-1$ dalších kroků. ♣

Důsledek 11. *Podobná spodní mez platí i pro řazení po řádkách čí sloupcích.*

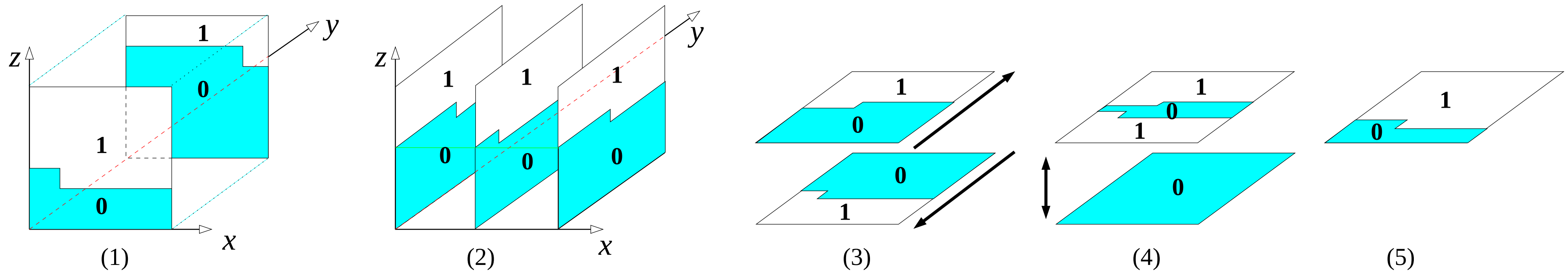
3DSort: Lexikografické třídění na 3-D mřížce $M(n,n,n)$

Algorithm $\textsc{3DSort}(X = [x_1, \ldots, x_N])$ 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.



Věta 12. *Alg.* $3\mathrm{D}\textsc{Sort}$ *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 Třídící 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 třídění. ♣

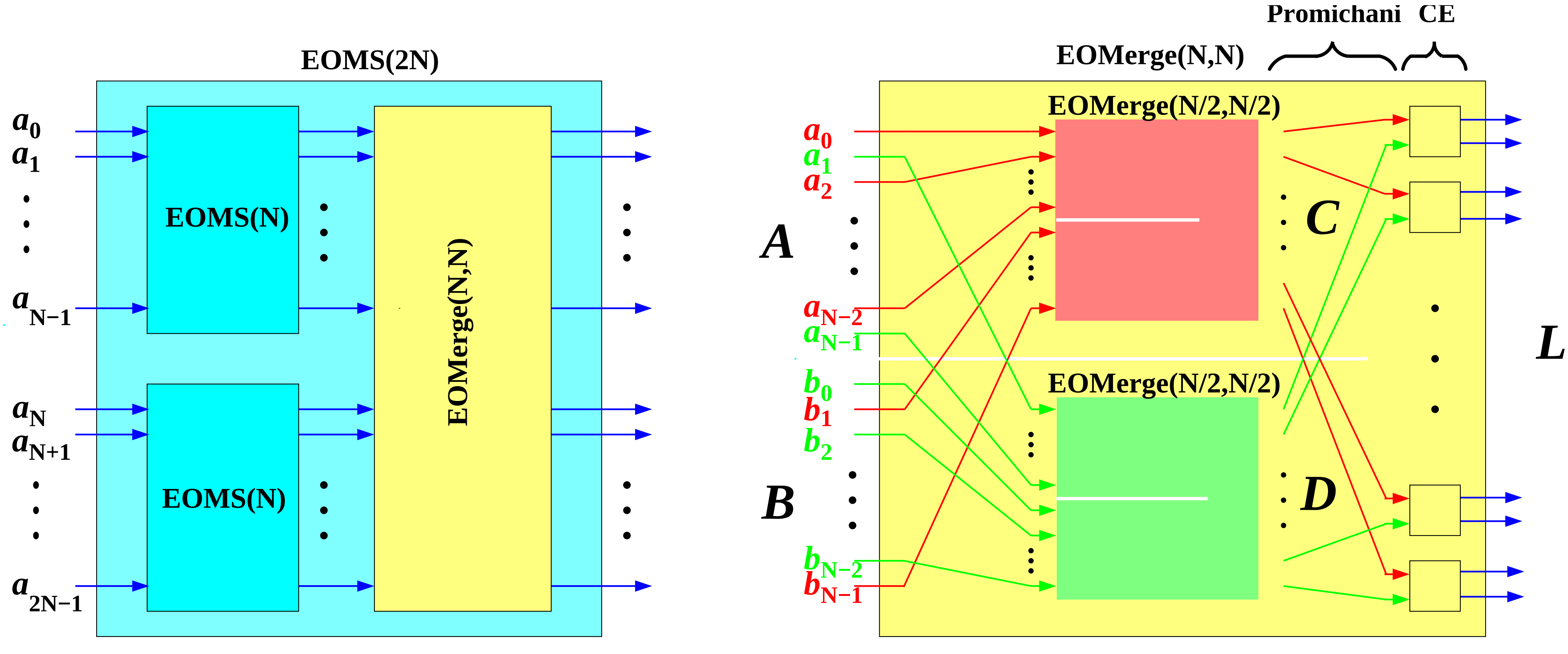
Škálovatelnost v $M(\sqrt[3]{p},\sqrt[3]{p},\sqrt[3]{p})$ (je-li v rovinách použit $\textsc{ShearSort}$)

$$T(N,p)=O\left(\frac{N}{p}\log\frac{N}{p}\right)+O\left(\frac{N}{\sqrt[3]{p^2}}\log p\right)\quad \text{a}\quad \psi_1(p)=p^{\alpha\sqrt[3]{p}}\quad \text{a}\quad \psi_2(N)=\left(\frac{\log N}{\log\log N}\right)^3.$$

Třídění na hyperkubických sítích

Batcherovy algoritmy: Sudo-Lichý MergeSort, Bitonický MergeSort.

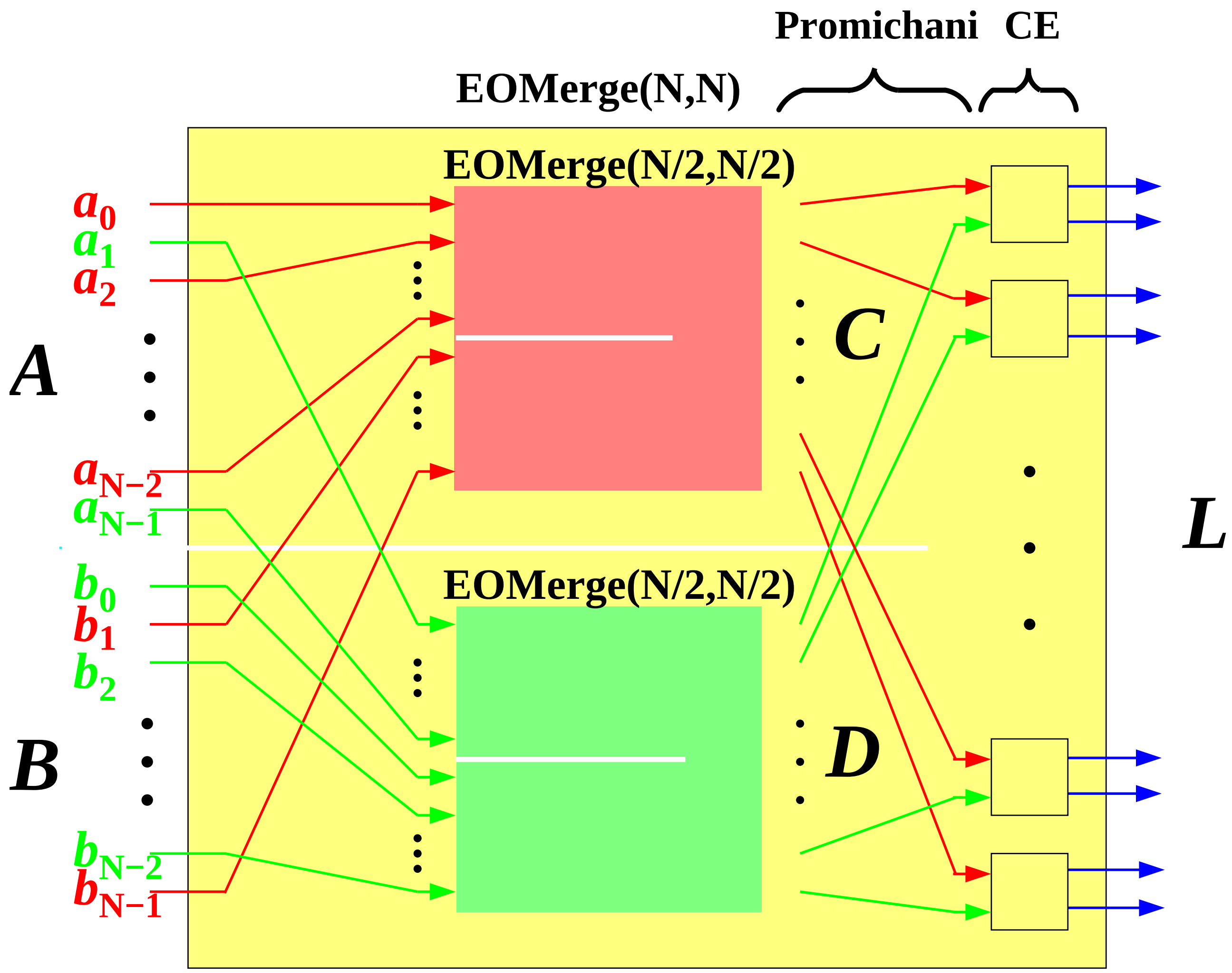
Sudo-Lichý MergeSort (EOMS)



(a) (b)

$$\mathrm{EOMS}(a_0,\ldots,a_{2N-1}) = \mathrm{EOM{\scriptsize ERGE}}(\mathrm{EOMS}(a_0,\ldots,a_{N-1}),\mathrm{EOMS}(a_N,\ldots,a_{2N-1}))$$

Sudo-Liché Sloučení (EOMerge)



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

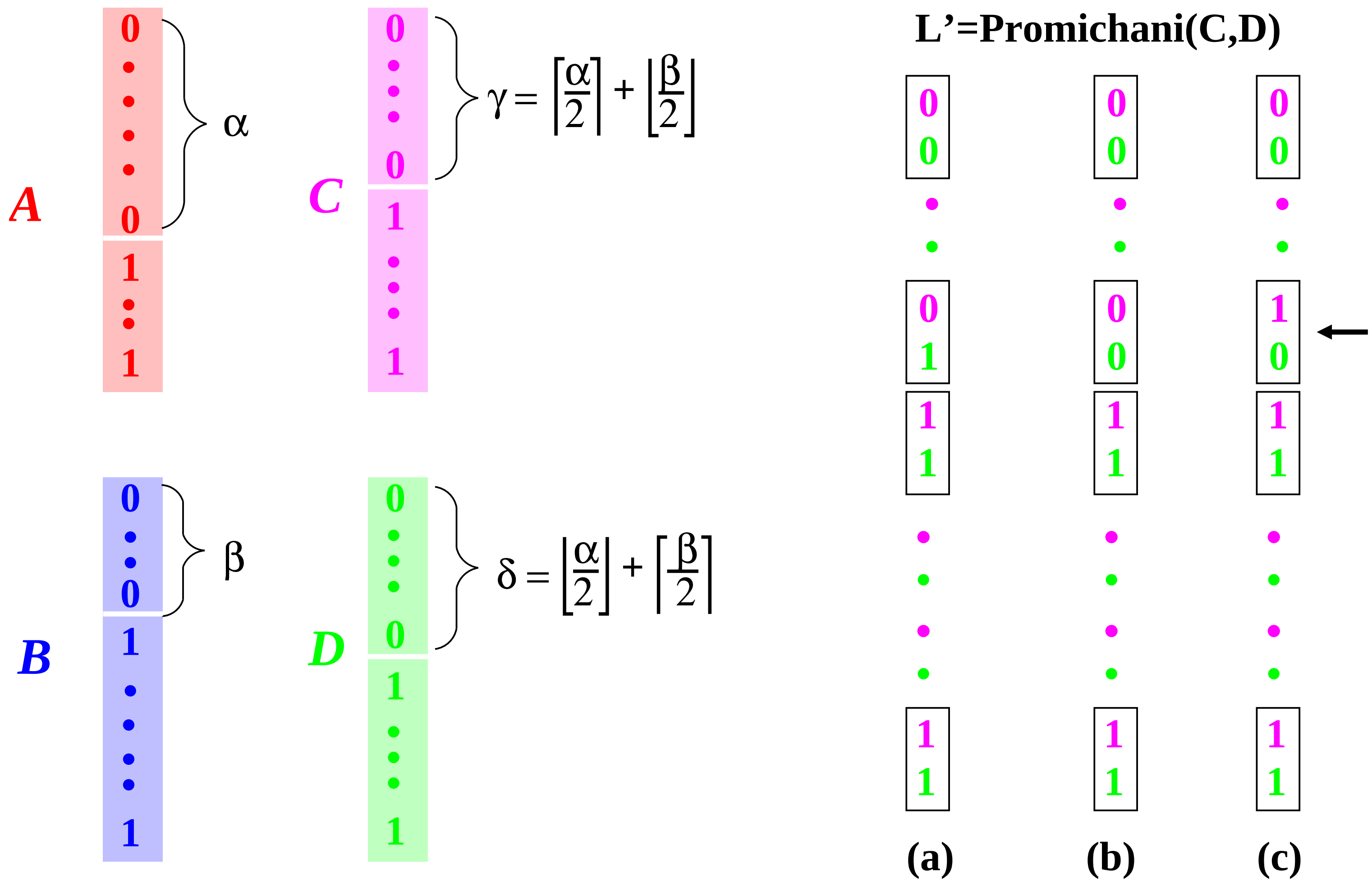
$$C = \text{EOMerge}(\text{even}(A), \text{odd}(B))$$
$$D = \text{EOMerge}(\text{odd}(A), \text{even}(B))$$
$$L' = \text{Promichani}(C, D)$$
$$L = \text{EOMerge}(A, B) = \text{Parovane\_CE}(L')$$

Důkaz správnosti Sudo-Lichého slučování

Věta 13. $\textsc{EOMerge}$ *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 Třídící Lemmy.) Věta platí pro $N = 1$. Nechť $N = 2^k$, $k \geq 1$.
Nechť $\gamma = \lceil \alpha/2 \rceil + \lfloor \beta/2 \rfloor$ a $\delta = \lfloor \alpha/2 \rfloor + \lceil \beta/2 \rceil \implies |\gamma - \delta| \leq 1$

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



Časová složitost $\textsc{EOMerge}$

Nechť

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

Potom

- $\textsc{EOMerge}(N,N)$ je rekurzivní a každý stupeň rekurze přidá právě 1 sloupec komparátorů:
$$d_\mathrm{m}(2N) = d_\mathrm{m}(N) + 1$$
$\Longrightarrow$
$$d_\mathrm{m}(2N) = \log N + 1 = \log(2N).$$

Cenová složitost $\textsc{EOMerge}$ = počet komparátorů

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

♣

Časová a cenová složitost EOMS

Věta 14. *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

- $d_\mathrm{s}(N)$ = hloubka EOMS(N),
- $c_\mathrm{s}(N)$ = počet komparátorů EOMS(N).

Potom

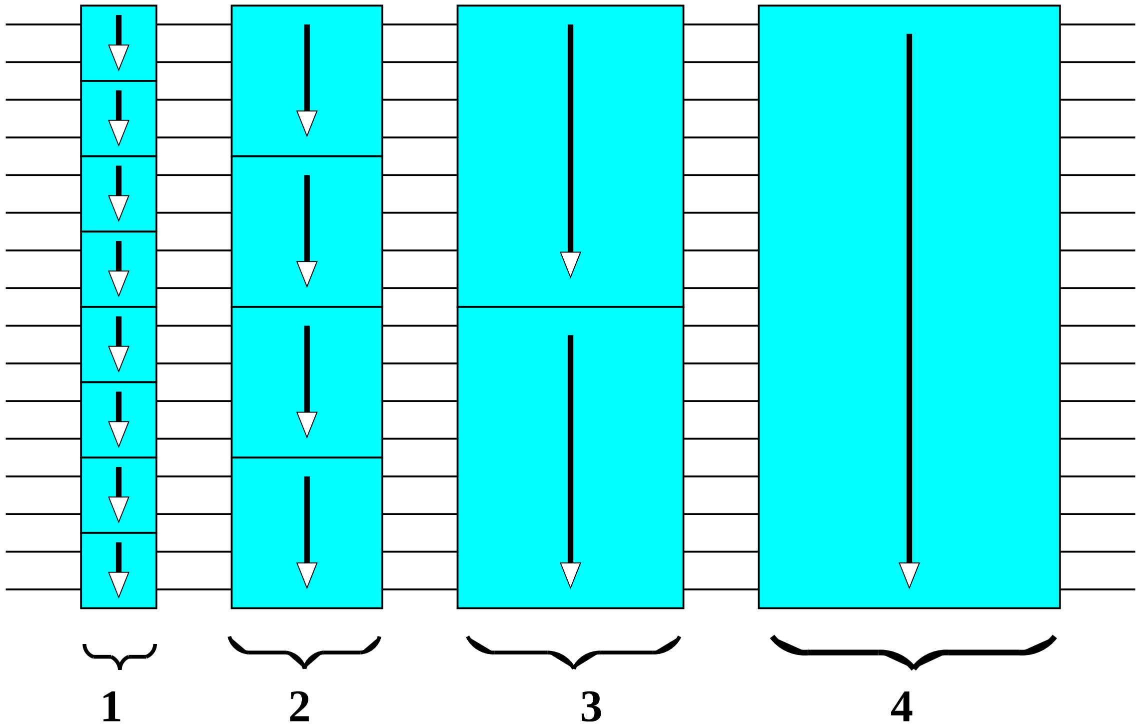
- $d_\mathrm{s}(2) = 1$
  & $d_\mathrm{s}(2N) = d_\mathrm{s}(N) + d_\mathrm{m}(2N) = d_\mathrm{s}(N) + \log(2N)$
  $\Longrightarrow$
  $$d_\mathrm{s}(N) = \log N(\log N + 1)/2 = O(\log^2 N)$$

- $c_\mathrm{s}(2) = 1$
  & $c_\mathrm{s}(2N) = 2c_\mathrm{s}(N) + c_\mathrm{m}(2N) = 2c_\mathrm{s}(N) + N\log(2N)$
  $\Longrightarrow$
  $$c_\mathrm{s}(N) = N d_\mathrm{s}(N)/2 = O(N \log^2 N).$$

♣

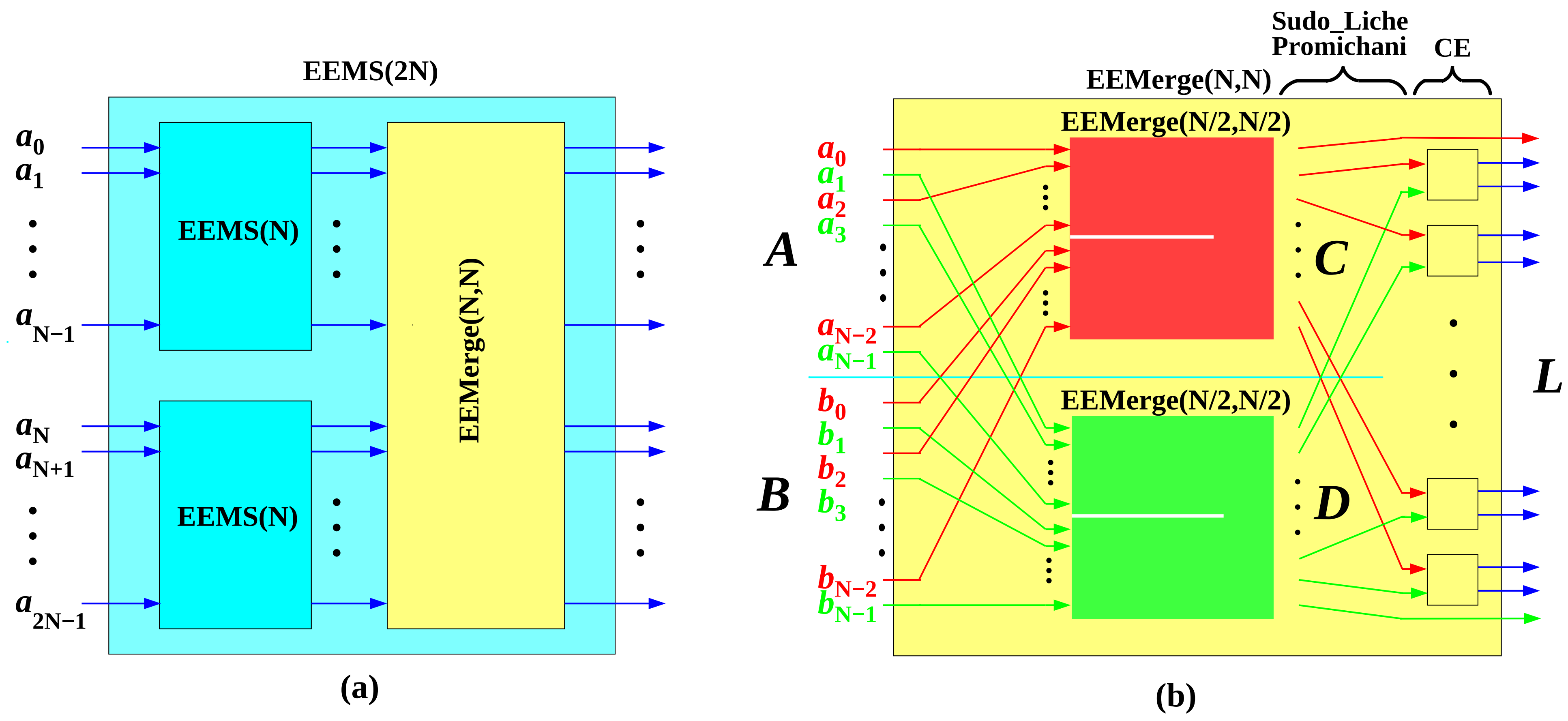
Rozvinutí sítě EOMS

- 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 $2N$.



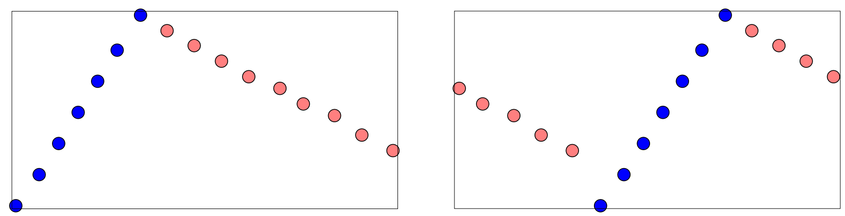
Sudo-Sudý MergeSort (EEMS)

$$\mathrm{EEMS}(a_0,\ldots,a_{2N-1}) = \mathrm{EEM{\small ERGE}}(\mathrm{EEMS}(a_0,\ldots,a_{N-1}),\mathrm{EEMS}(a_N,\ldots,a_{2N-1}))$$
$$\mathrm{EEM{\small ERGE}}(A,B) = \mathrm{Parovane\_CE}(\mathrm{Sudo\_Liche\_Promichani}(\mathrm{C},\mathrm{D}))$$
$$C = \mathrm{EEM{\small ERGE}}(\mathrm{even}(A),\mathrm{even}(B))$$
$$D = \mathrm{EEM{\small ERGE}}(\mathrm{odd}(A),\mathrm{odd}(B))$$



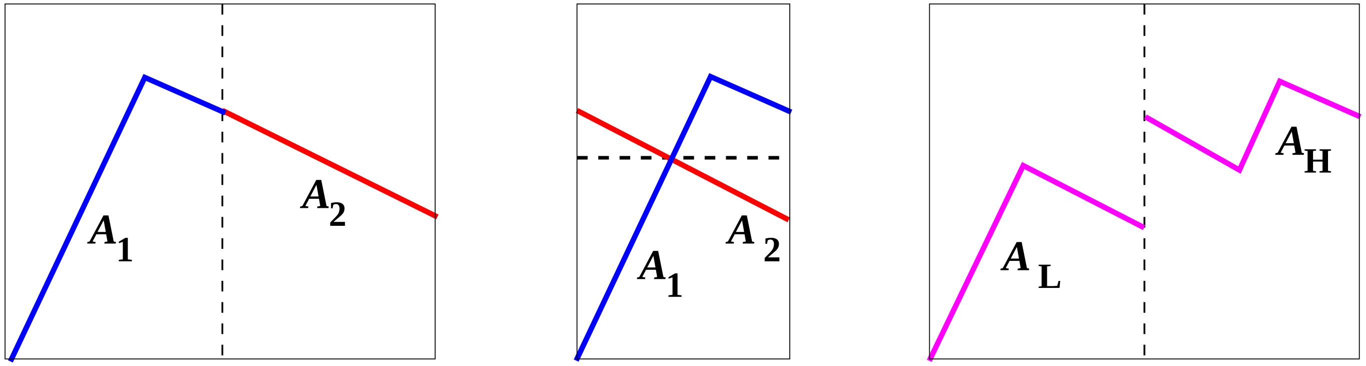
Bitonické posloupnosti

1 údolí a 1vrchol nezávisle na rotacích.



(a) (b)

Bitonické rozdělení



(a) (b) (c)

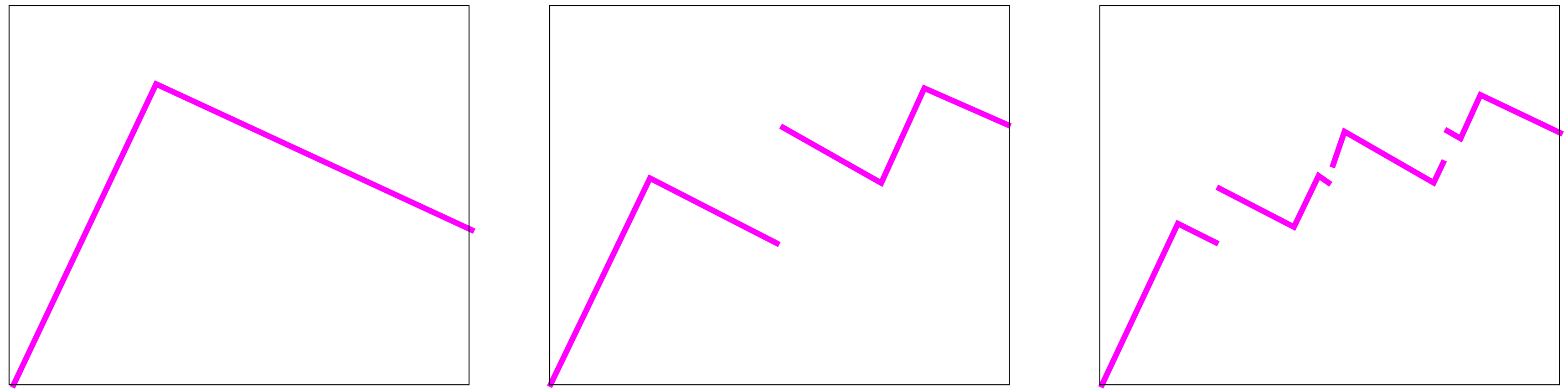
Lemma 15. *Je-li* $A = a_0, a_1, \ldots, a_{2N-1}$ *bitonická, její bitonické rozdělení je* $A' = A_{\mathrm{L}} A_{\mathrm{H}}$, *kde*

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

Pak: *(1)* A_{L} *a* A_{H} *jsou opět bitonické.*
(2) Každé číslo v A_{L} *je menší než libovolné číslo v* A_{H}.

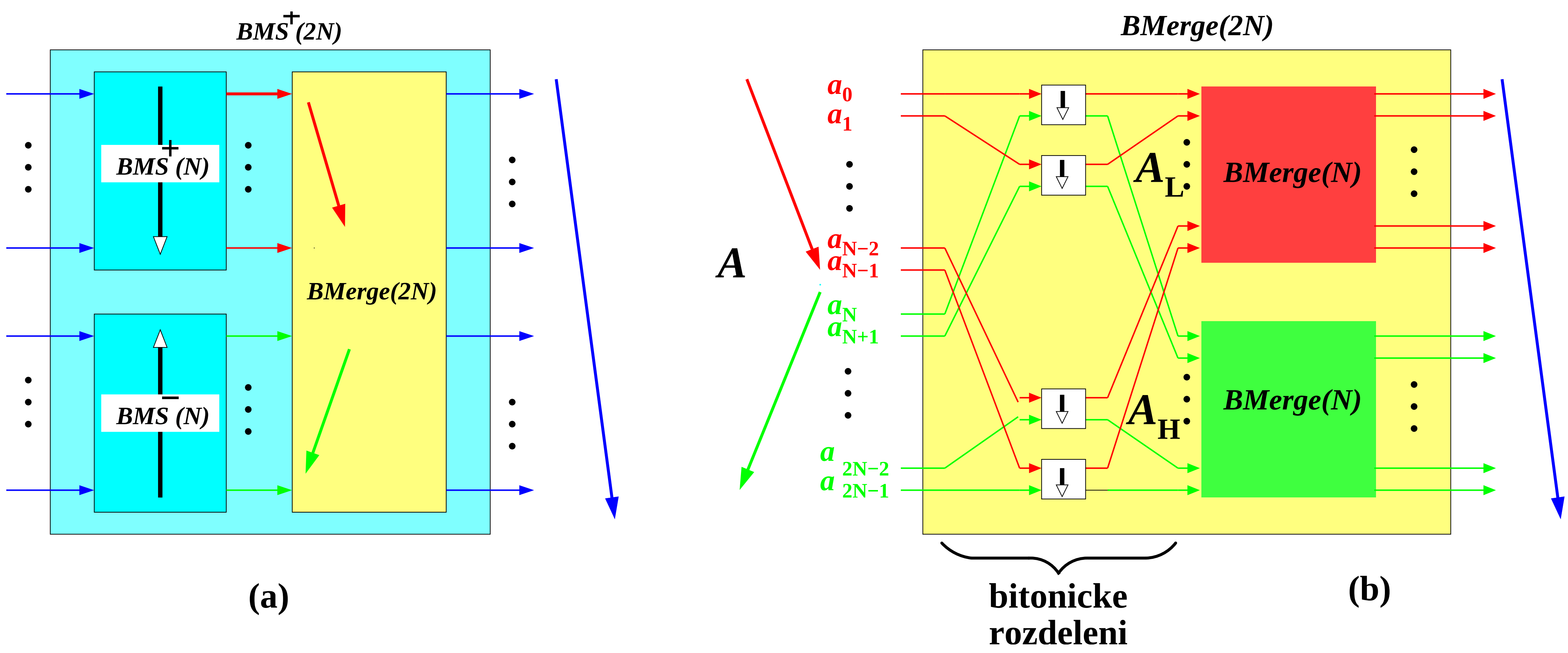
Pozorování

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



$\Longrightarrow$ Bitonické Sloučení $\Longrightarrow$ Bitonický MergeSort

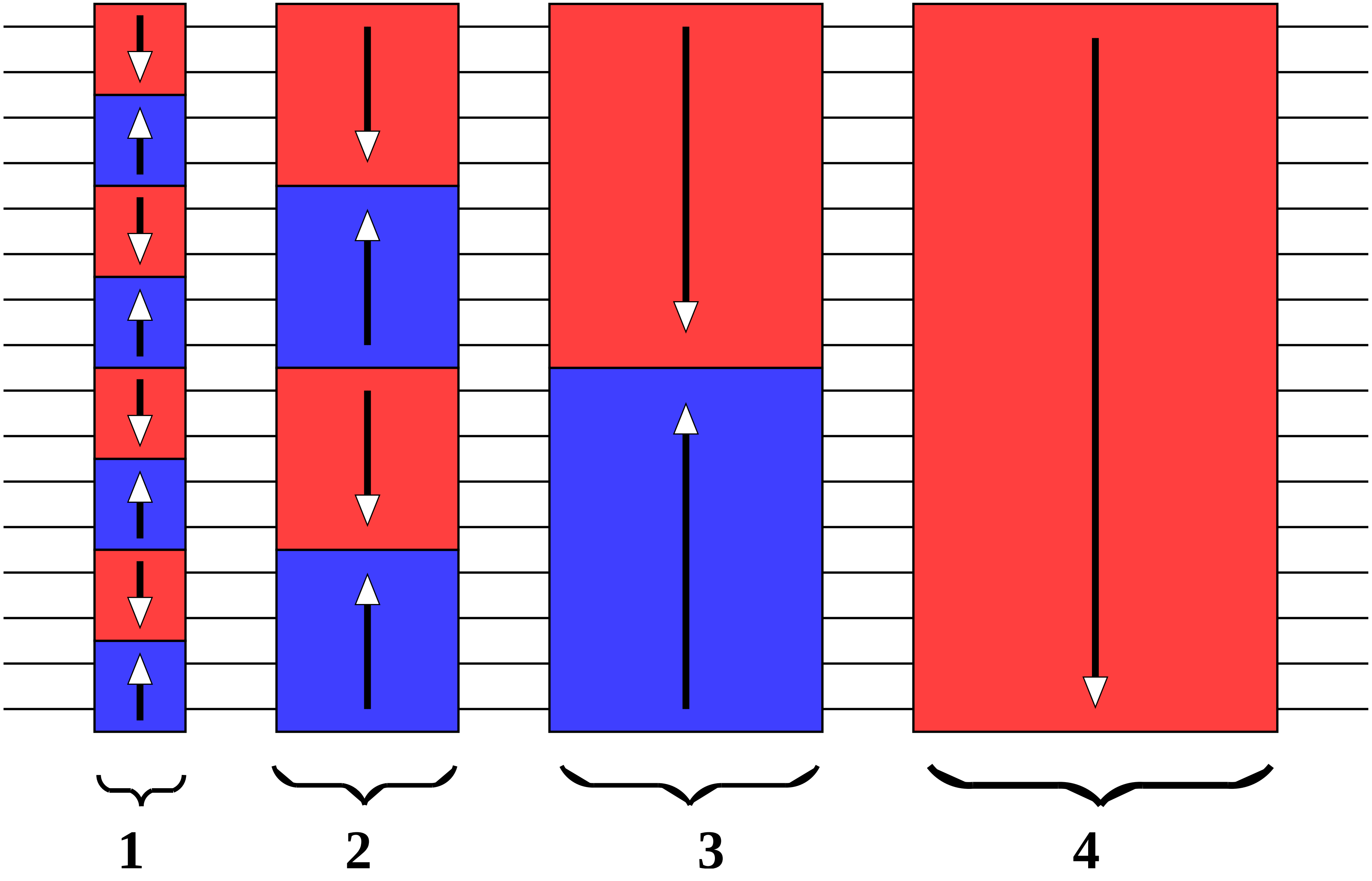
Bitonický MergeSort (BMSort)



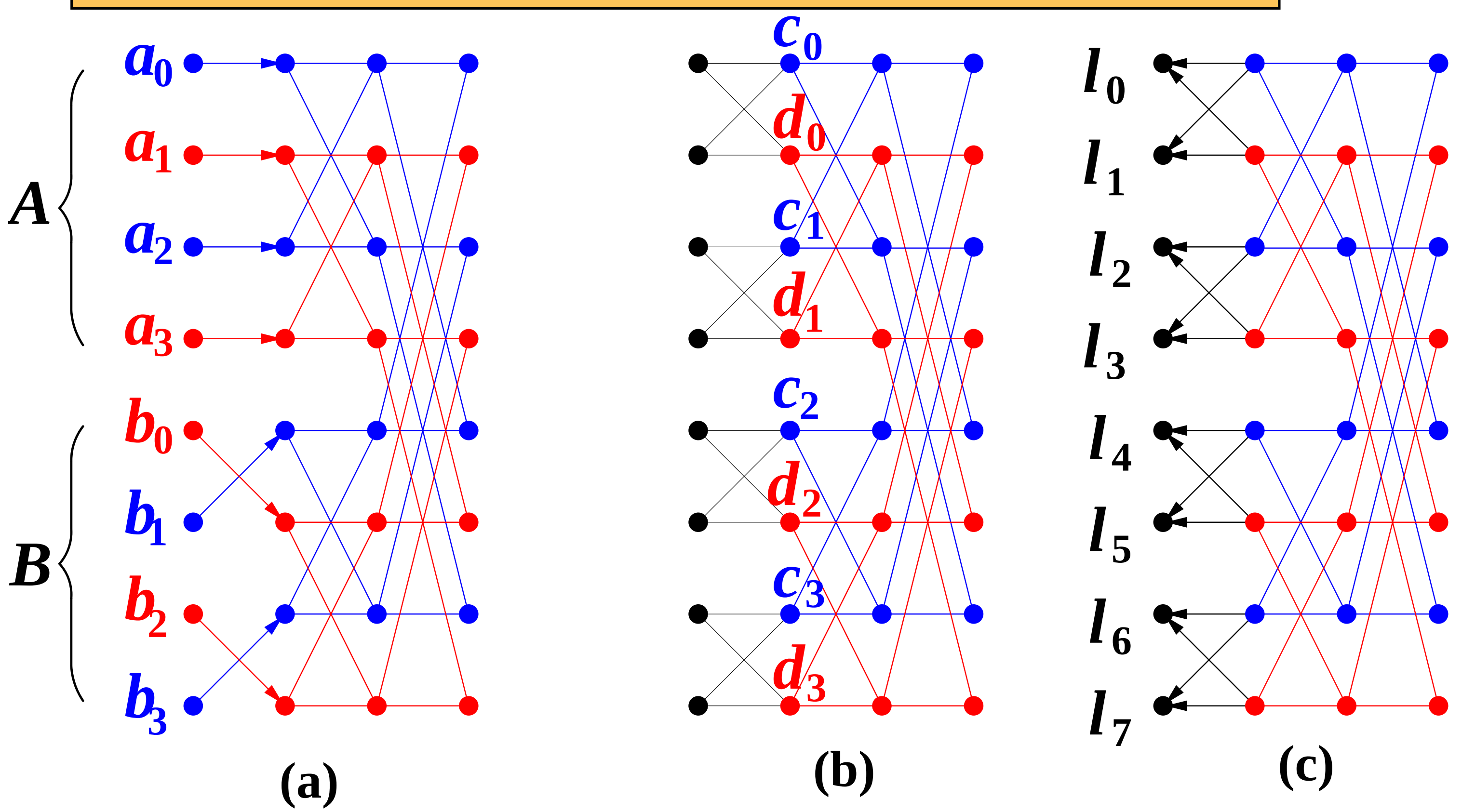
$\mathrm{BMS}^+(a_0,\ldots,a_{2N-1}) = \textsc{BMerge}(\mathrm{BMS}^+(a_0,\ldots,a_{N-1})\mathrm{BMS}^-(a_N,\ldots,a_{2N-1}))$

$\textsc{BMerge}(A) = \textsc{BMerge}(A_\mathrm{L})\textsc{BMerge}(A_\mathrm{H})$, kde $(A_\mathrm{L}A_\mathrm{H}) = \text{Bitonicke\_Rozdeleni}(A)$

Rozvinutí bitonické třídící sítě



Implementace EOMS na topologii motýlek oBF_n



Realizace $L = \mathrm{EOM{\small ERGE}}(A, B)$ na $oBF_n =$

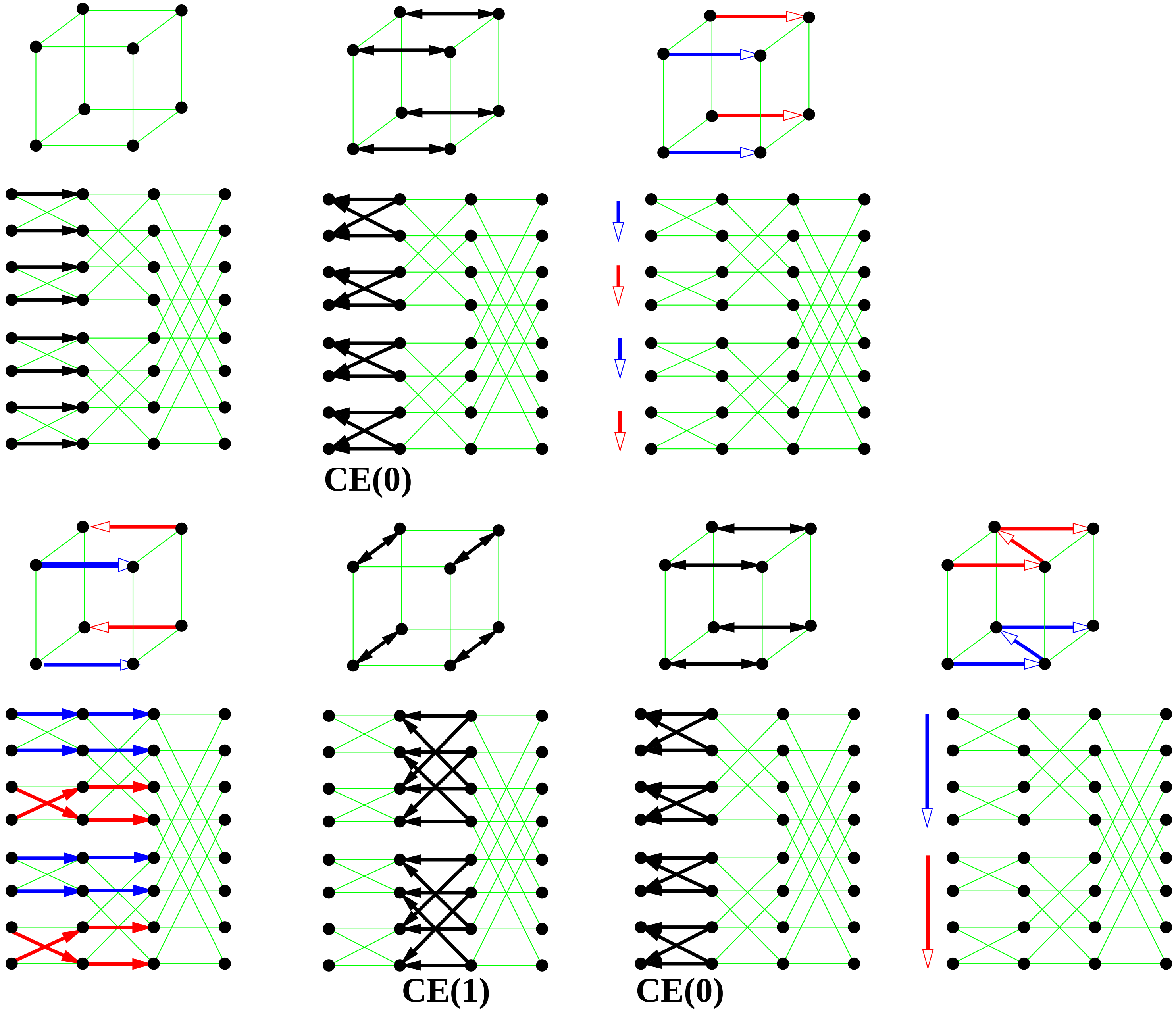
(a) Přenos 1.stupněm A v horní půlce rovně a B v dolní půlce křížem.

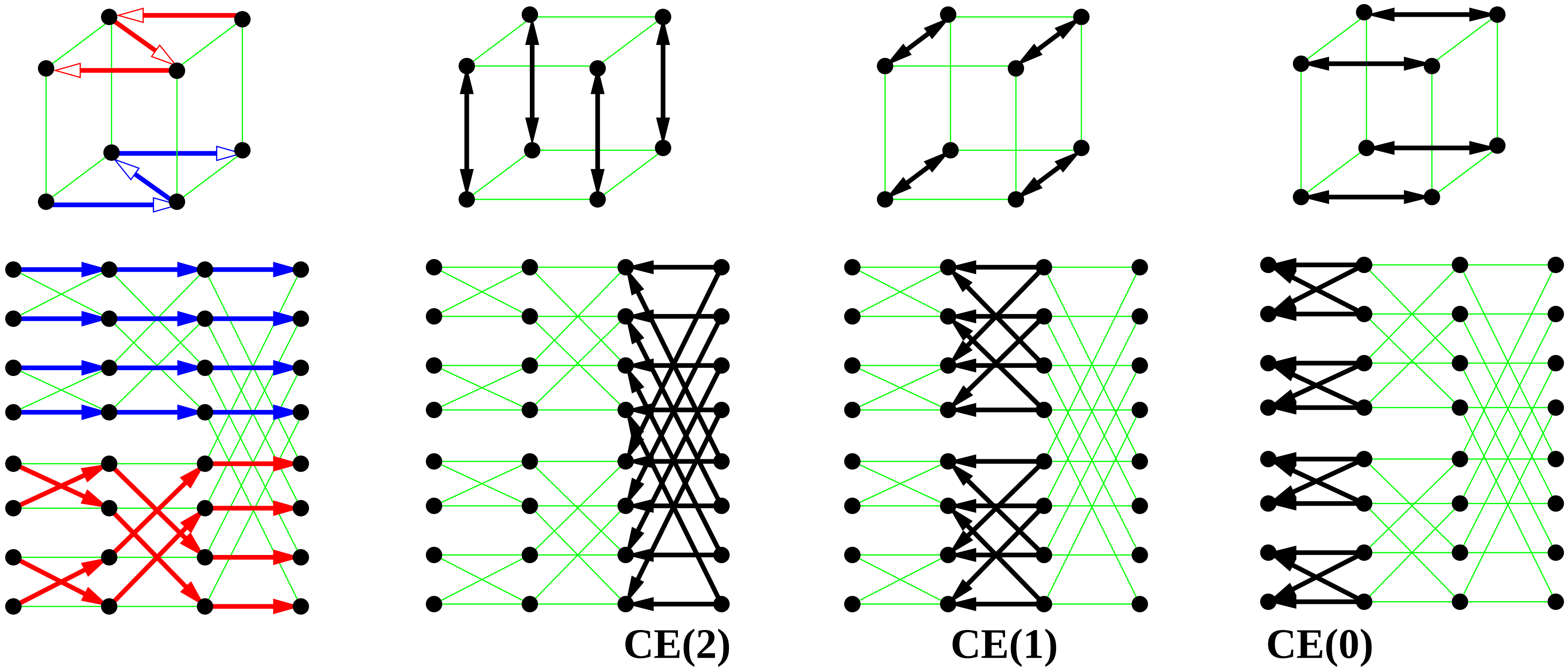
(b) Rekurzivní konstrukce:

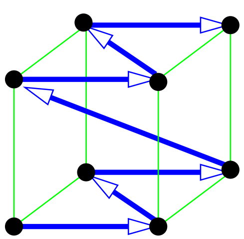
- $C = \mathrm{EOM{\small ERGE}}(\mathrm{even}(A), \mathrm{odd}(B))$ v MODRÉM oBF_{n-1},
- $D = \mathrm{EOM{\small ERGE}}(\mathrm{odd}(A), \mathrm{even}(B))$ v ČERVENÉM oBF_{n-1}.

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

Sudo-Lichý MergeSort 8 čísel na Q_3





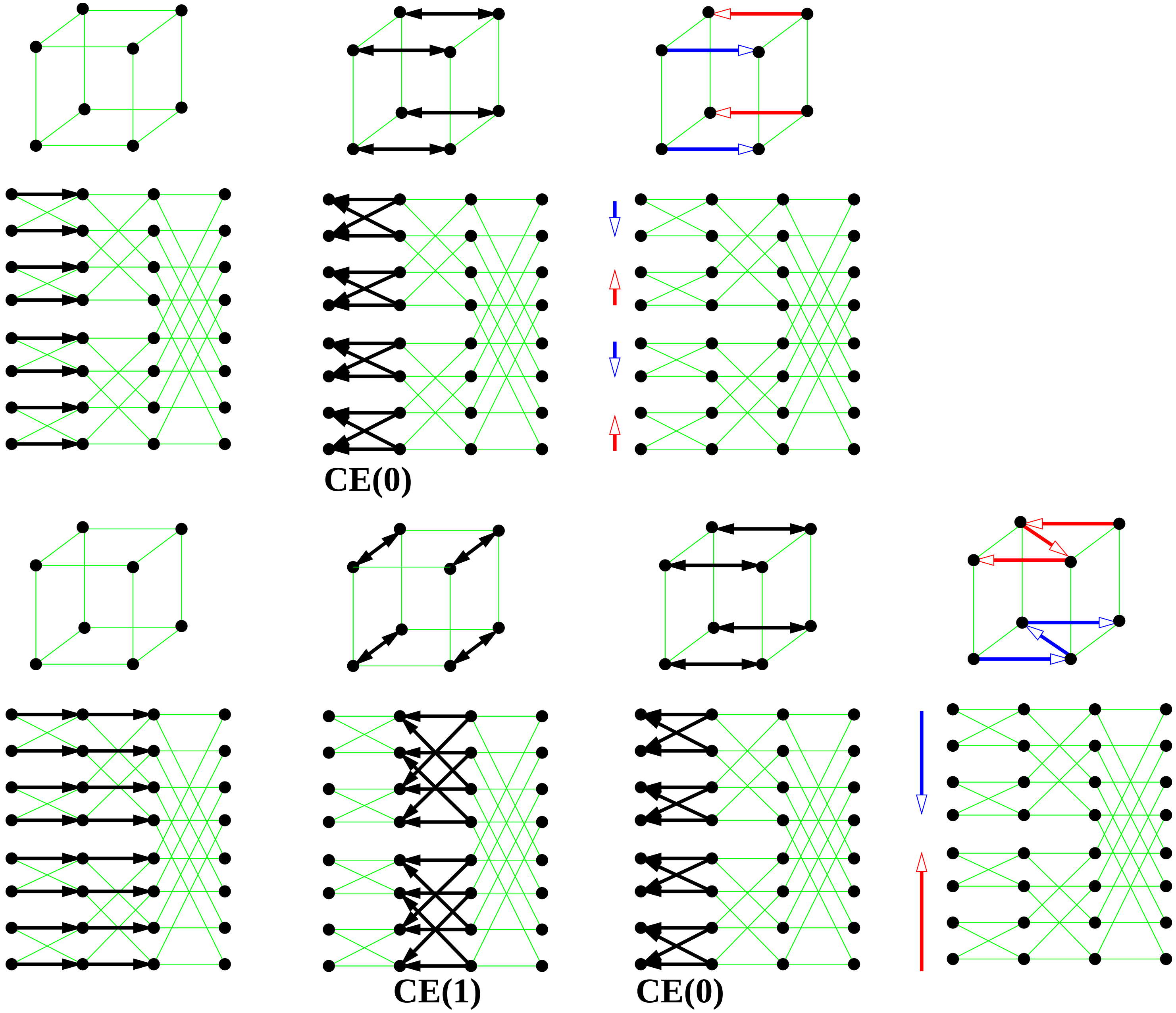


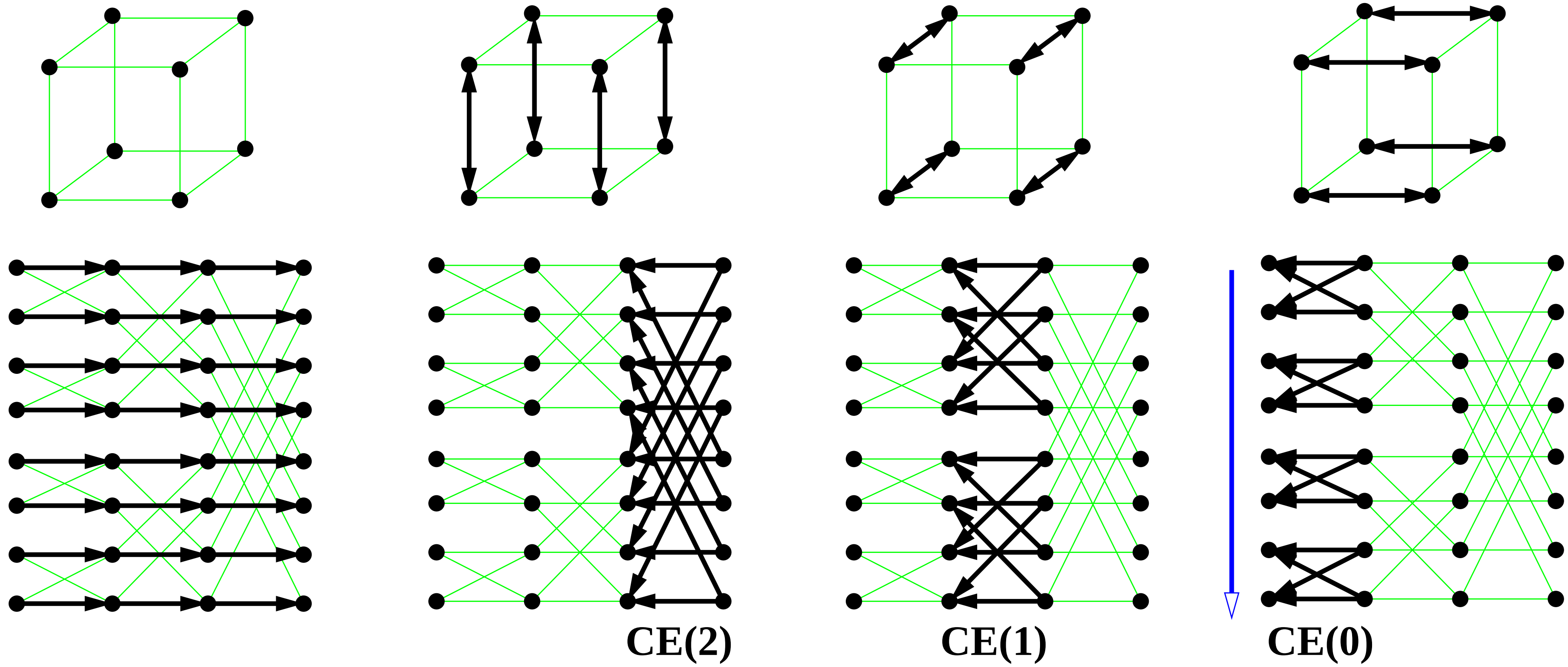
Pozorování:

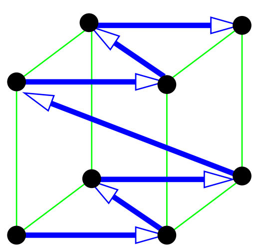
Každá druhá podposloupnost je otočena (rostoucí → klesající)

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

CUBEBMS: Bitonický MergeSort 8 čísel na Q_3







Časová složitost a škálovatelnost CubeBMS

Věta 16. *Algoritmus* CubeBMS *pro* $N = 2^n$ *čísel na* Q_n *vyžaduje* $T(N,N) = O(\log^2 N)$ *paralelních C&E kroků. Pro* $p = 2^k$, $k < n$,

$$T(N,p) = O\left(\frac{N}{p}\log\frac{N}{p}\right) + O\left(\frac{N}{p}\log^2 p\right) \quad a \quad \psi_1(p) = p^{\alpha\log p} \quad a \quad \psi_2(N) = 2^{\sqrt{\alpha'\log N}}$$

Implementace CubeBMS na PRAM

Triviální.

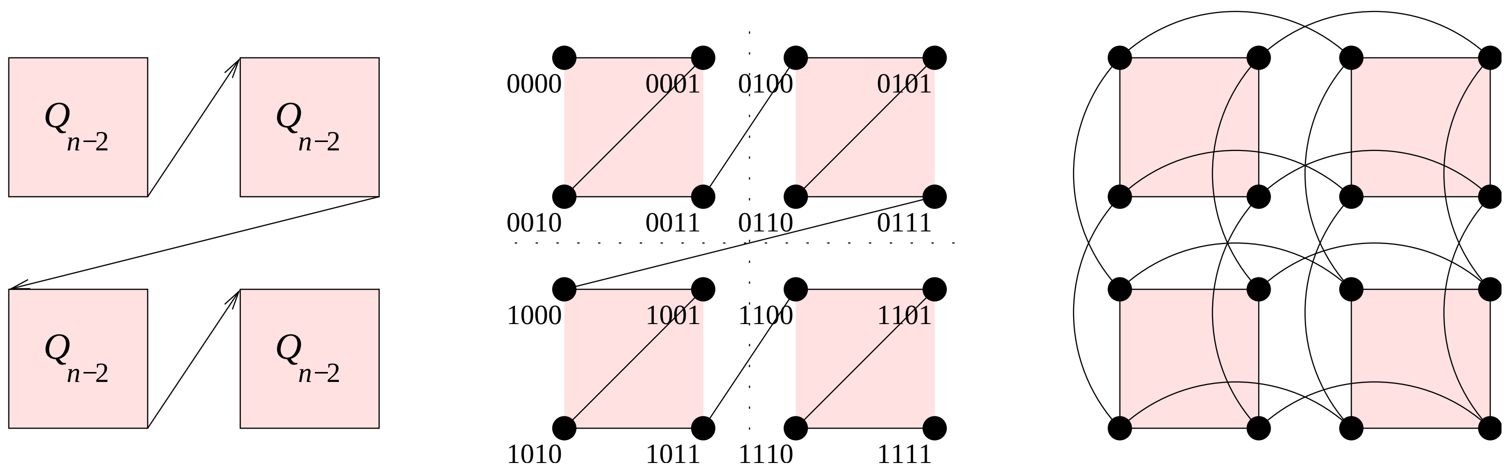
MeshBMS: Simulace CubeBMS na 2-D mřížce

- 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 $2^{\lfloor \frac{i}{2} \rfloor}$, $0 \le i \le n-1$, 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

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

Věta 17. *Pro $N = 2^n$, $\textsc{MeshBMS}$ 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

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

Lemma 18. **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 19. **(vět 1 a 17.)** *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 $\textsc{MeshBMS}$ setřídí N čísel na 2-D mřížce $M(\sqrt{p}, \sqrt{p})$ v*

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

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