Bonyolultságelmélet

Thursday 1st December, 2016, 23:20

A Cobham–Edmonds tézis, még egyszer

Egy problémát akkor tekintünk gyakorlatilag megoldhatónak, ha van rá polinom időigényű algoritmus.

A Cobham–Edmonds tézis, még egyszer

Egy problémát akkor tekintünk gyakorlatilag megoldhatónak, ha van rá polinom időigényű algoritmus.

Biztos?

A Cobham–Edmonds tézis, még egyszer

Egy problémát akkor tekintünk gyakorlatilag megoldhatónak, ha van rá polinom időigényű algoritmus.

Biztos?

Vannak esetek, mikor a lineáris időigény is soknak bizonyul. . .

A Cobham–Edmonds tézis, még egyszer

Egy problémát akkor tekintünk gyakorlatilag megoldhatónak, ha van rá polinom időigényű algoritmus.

Biztos?

Vannak esetek, mikor a lineáris időigény is soknak bizonyul...
...de a RAM-gépes kiszámítási modellben nem értelmezhető szublineáris időigény.

A Cobham–Edmonds tézis, még egyszer

Egy problémát akkor tekintünk gyakorlatilag megoldhatónak, ha van rá polinom időigényű algoritmus.

Biztos?

Vannak esetek, mikor a lineáris időigény is soknak bizonyul...
...de a RAM-gépes kiszámítási modellben nem értelmezhető szublineáris időigény.

A hatékony párhuzamosítás kérdése azonban gyakorlati szempontból lényeges, így szükség van olyan kiszámítási modellre, melyben pl. $\mathcal{O}(\log n)$-es időigény is értelmezhető.

Az elméleti párhuzamos számítógép

PRAM

Az elméleti párhuzamos számítógép

PRAM

Parallel, random-access machine.

Az elméleti párhuzamos számítógép

PRAM

Parallel, random-access machine.

- osztott memória

Az elméleti párhuzamos számítógép

PRAM

Parallel, random-access machine.

- osztott memória
- sok (de ,,csak" polinom sok) processzor

Az elméleti párhuzamos számítógép

PRAM

Parallel, random-access machine.

- osztott memória
- sok (de ,,csak" polinom sok) processzor
- minden processzor a memória bármelyik bitjét $\mathcal{O}(1)$ időben eléri

Az elméleti párhuzamos számítógép

PRAM

Parallel, random-access machine.

- osztott memória
- sok (de ,,csak" polinom sok) processzor
- minden processzor a memória bármelyik bitjét $\mathcal{O}(1)$ időben eléri
- a konfliktusok kezelésének módja (EREW, CREW, CRCW) nem rögzített, de nem is lényeges igazán

Az elméleti párhuzamos számítógép

PRAM

Parallel, random-access machine.

- osztott memória
- sok (de ,,csak" polinom sok) processzor
- minden processzor a memória bármelyik bitjét $\mathcal{O}(1)$ időben eléri
- a konfliktusok kezelésének módja (EREW, CREW, CRCW) nem rögzített, de nem is lényeges igazán

A matematikai modell

PRAM gépek helyett logikai hálózatcsaláddal fogjuk (ekvivalensen) definiálni a hatékony párhuzamosíthatóság fogalmát.

Hálózatcsaláddal való eldöntés

Hálózatcsalád

Hálózatcsaládon logikai hálózatoknak egy $\mathcal{C} = C_0, C_1, \ldots$ sorozatát értjük, ahol minden i-re C_i olyan logikai hálózat, melynek input kapuinak címkéje az $\{x_1, \ldots, x_i\}$ halmazba esik.

Hálózatcsaláddal való eldöntés

Hálózatcsalád

Hálózatcsaládon logikai hálózatoknak egy $\mathcal{C} = C_0, C_1, \ldots$ sorozatát értjük, ahol minden i-re C_i olyan logikai hálózat, melynek input kapuinak címkéje az $\{x_1, \ldots, x_i\}$ halmazba esik.

Eldöntés

A $\mathcal{C} = C_0, C_1, \ldots$ hálózatcsalád elfogadja az $x = b_1 b_2 \ldots b_n \in \{0,1\}^*$ szót, ha $C_n(b_1, b_2, \ldots, b_n) = 1$.

Hálózatcsaládal való eldöntés

Hálózatcsalád

Hálózatcsaládon logikai hálózatoknak egy $\mathcal{C} = C_0, C_1, \ldots$ sorozatát értjük, ahol minden i-re C_i olyan logikai hálózat, melynek input kapuinak címkéje az $\{x_1, \ldots, x_i\}$ halmazba esik.

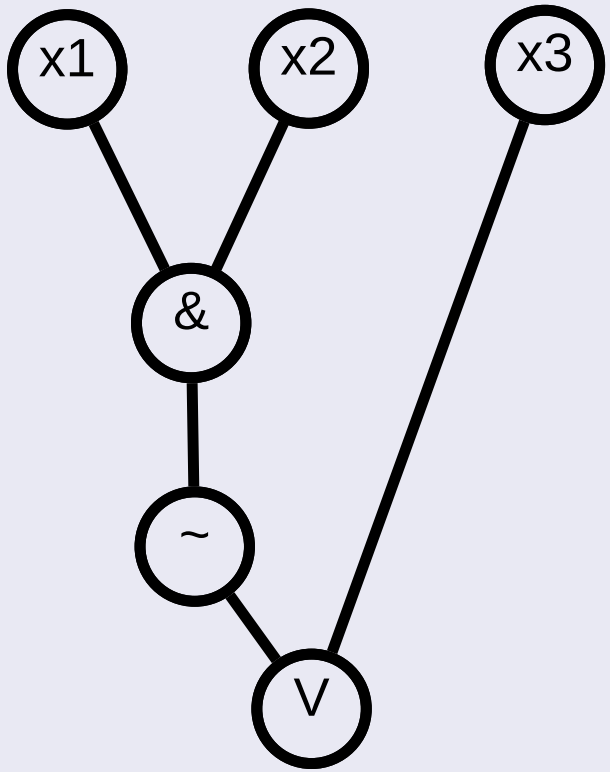
Eldöntés

A $\mathcal{C} = C_0, C_1, \ldots$ hálózatcsalád elfogadja az $x = b_1 b_2 \ldots b_n \in \{0,1\}^*$ szót, ha $C_n(b_1, b_2, \ldots, b_n) = 1$.
A $\mathcal{C}$ által eldöntött $L(\mathcal{C}) \subseteq \{0,1\}^*$ nyelvet a $\mathcal{C}$ által elfogadott szavak alkotják.

Példa

Elfogadás, elvetés

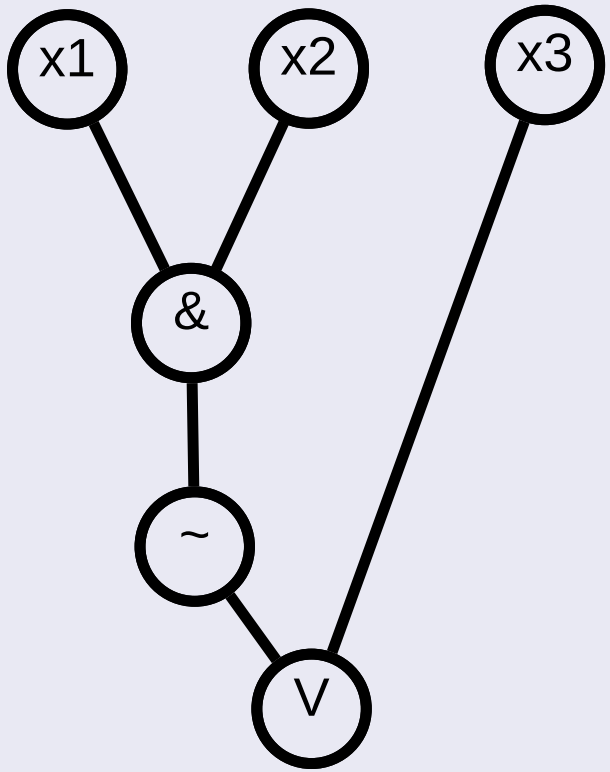
Ha C_3 a következő hálózat:



Példa

Elfogadás, elvetés

Ha C_3 a következő hálózat:



akkor $\mathcal{C} = C_0, \ldots, C_3, \ldots$ elfogadja a 001, 010 szavakat és elveti az 110-t.

Példa

$(01)^*$-ot eldöntő hálózatcsalád

Példa

$(01)^*$-ot eldöntő hálózatcsalád

- C_0 : 

Példa

$(01)^*$-ot eldöntő hálózatcsalád

- C_0 : 

- C_{2i+1}: 

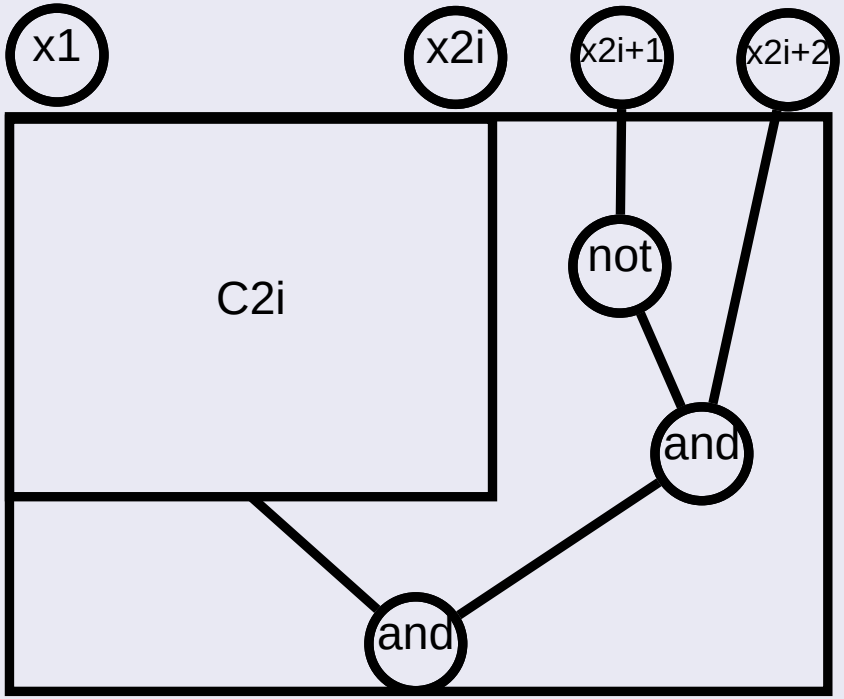
Példa

$(01)^*$-ot eldöntő hálózatcsalád

- C_0 : 

- C_{2i+1}: 

- C_{2i+2}:



Korlátok

Hálózatcsaláddal minden nyelv eldönthető

Korlátok

Hálózatcsaláddal minden nyelv eldönthető (mert minden Boole-függvényhez van őt reprezentáló hálózat),

Korlátok

Hálózatcsaláddal minden nyelv eldönthető (mert minden Boole-függvényhez van őt reprezentáló hálózat), így megszorításokat teszünk az ,,elfogadható" hálózatcsaládra n függvényében

Korlátok

Hálózatcsaláddal minden nyelv eldönthető (mert minden Boole-függvényhez van őt reprezentáló hálózat), így megszorításokat teszünk az ,,elfogadható" hálózatcsaládra n függvényében

- a C_n-beli kapuk száma szerint,

Korlátok

Hálózatcsaláddal minden nyelv eldönthető (mert minden Boole-függvényhez van őt reprezentáló hálózat), így megszorításokat teszünk az ,,elfogadható" hálózatcsaládra n függvényében

- a C_n-beli kapuk száma szerint,
- a C_n hálózat mélysége szerint,

Korlátok

Hálózatcsaláddal minden nyelv eldönthető (mert minden Boole-függvényhez van őt reprezentáló hálózat), így megszorításokat teszünk az ,,elfogadható'' hálózatcsaládra n függvényében

- a C_n-beli kapuk száma szerint,
- a C_n hálózat mélysége szerint,
- a hálózatcsalád tagjaiban megengedett kaputípusok szerint,

Korlátok

Hálózatcsaláddal minden nyelv eldönthető (mert minden Boole-függvényhez van őt reprezentáló hálózat), így megszorításokat teszünk az ,,elfogadható'' hálózatcsaládra n függvényében

- a C_n-beli kapuk száma szerint,
- a C_n hálózat mélysége szerint,
- a hálózatcsalád tagjaiban megengedett kaputípusok szerint,
- uniformitás szerint (egy hálózatcsalád akkor (logtár-)uniform, ha van olyan algoritmus, mely C_n-t előállítja $\mathcal{O}(\log n)$ tár felhasználása mellett).

A hatékonyan párhuzamosítható problémák osztálya

Definíció

Ha $k \geq 0$ konstans, akkor az $\boldsymbol{NC}^k$ osztályba azok a $\{0,1\}$ fölötti nyelvek tartoznak, melyek eldönthetők

A hatékonyan párhuzamosítható problémák osztálya

Definíció

Ha $k \geq 0$ konstans, akkor az $\boldsymbol{NC}^k$ osztályba azok a $\{0,1\}$ fölötti nyelvek tartoznak, melyek eldönthetők

- valamely c konstansra $\mathcal{O}(n^c)$, vagyis polinom sok kaput tartalmazó,

A hatékonyan párhuzamosítható problémák osztálya

Definíció

Ha $k \geq 0$ konstans, akkor az $\boldsymbol{NC}^k$ osztályba azok a $\{0,1\}$ fölötti nyelvek tartoznak, melyek eldönthetők

- valamely c konstansra $\mathcal{O}(n^c)$, vagyis polinom sok kaput tartalmazó,
- $\mathcal{O}(\log^k n)$ mélységű,

A hatékonyan párhuzamosítható problémák osztálya

Definíció

Ha $k \geq 0$ konstans, akkor az $\boldsymbol{NC}^k$ osztályba azok a $\{0,1\}$ fölötti nyelvek tartoznak, melyek eldönthetők

- valamely c konstansra $\mathcal{O}(n^c)$, vagyis polinom sok kaput tartalmazó,
- $\mathcal{O}(\log^k n)$ mélységű,
- legfeljebb kettő befokú kapukat tartalmazó,

A hatékonyan párhuzamosítható problémák osztálya

Definíció

Ha $k \geq 0$ konstans, akkor az $\boldsymbol{NC}^k$ osztályba azok a $\{0,1\}$ fölötti nyelvek tartoznak, melyek eldönthetők

- valamely c konstansra $\mathcal{O}(n^c)$, vagyis polinom sok kaput tartalmazó,
- $\mathcal{O}(\log^k n)$ mélységű,
- legfeljebb kettő befokú kapukat tartalmazó,
- uniform

hálózatcsaláddal.

A hatékonyan párhuzamosítható problémák osztálya

Definíció

Ha $k \geq 0$ konstans, akkor az $\boldsymbol{NC}^k$ osztályba azok a $\{0,1\}$ fölötti nyelvek tartoznak, melyek eldönthetők

- valamely c konstansra $\mathcal{O}(n^c)$, vagyis polinom sok kaput tartalmazó,
- $\mathcal{O}(\log^k n)$ mélységű,
- legfeljebb kettő befokú kapukat tartalmazó,
- uniform

hálózatcsaláddal.
$\boldsymbol{NC} = \bigcup_{k \geq 0} \boldsymbol{NC}^k$, a hatékonyan párhuzamosítható problémák osztálya.

NC^1: logaritmikus mélységű hálózatcsalád

Példa: 1^n

Az 1^* nyelv eldönthető logaritmikus mélységű hálózattal:

NC^1: logaritmikus mélységű hálózatcsalád

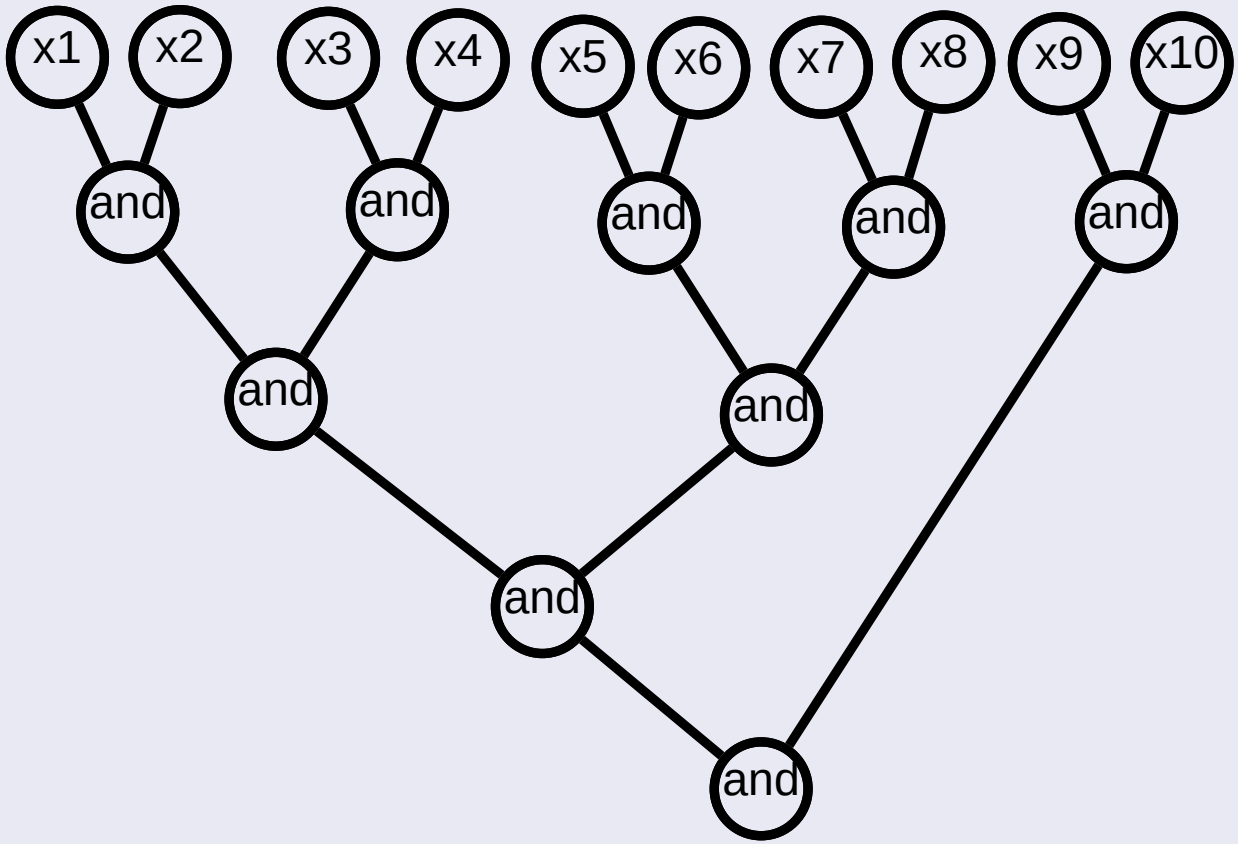
Példa: 1^n

Az 1^* nyelv eldönthető logaritmikus mélységű hálózattal: C_n az input kapukat $\wedge$ kapukkal köti össze, egy majdnem teljes bináris fába szervezett topológiával.

NC^1: logaritmikus mélységű hálózatcsalád

Példa: 1^n

Az 1^* nyelv eldönthető logaritmikus mélységű hálózattal: C_n az input kapukat $\wedge$ kapukkal köti össze, egy majdnem teljes bináris fába szervezett topológiával. Pl. C_{10} :



Függvények kiszámítása

Több output kapu

Hálózatcsaládokkal függvényeket is kiszámíthatunk: ha a $\mathcal{C} = C_0, C_1, \ldots$ családban a C_n tagnak o_n kimeneti kapuval rendelkezik (rögzített sorrendben), akkor a család értelemszerűen kiszámít egy $\{0,1\}^* \to \{0,1\}^*$ függvényt; az n hosszú input szavakhoz o_n hosszú output szót rendel.

Összeadó hálózatok

Összeadás

Összeadás: a C_n hálózat input kapui az $x_0, \ldots, x_{n-1}$, $y_0, \ldots, y_{n-1}$ változókkal címkézettek, output kapui $z_0, \ldots, z_n$.

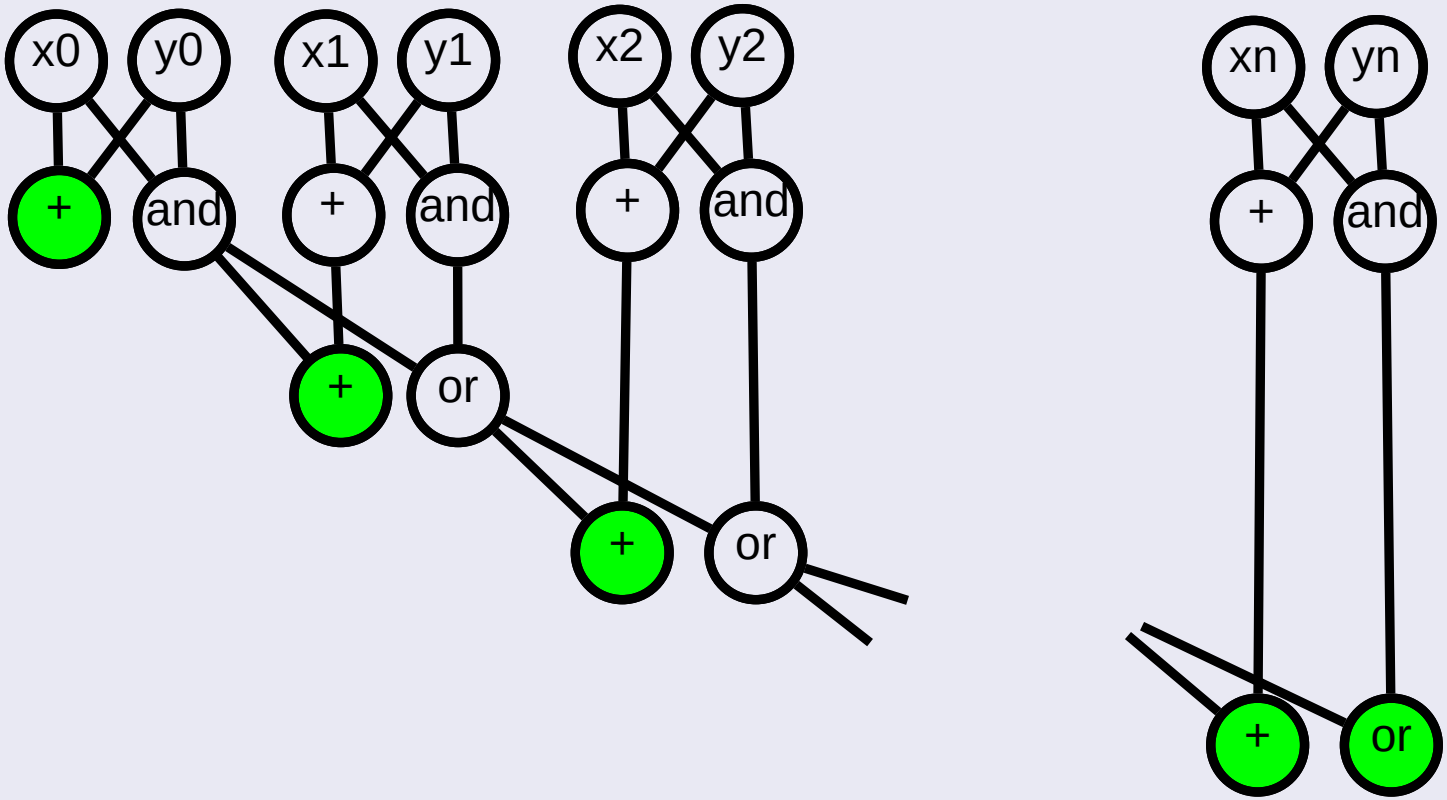
Összeadó hálózatok

Összeadás

Összeadás: a C_n hálózat input kapui az $x_0, \ldots, x_{n-1}$, $y_0, \ldots, y_{n-1}$ változókkal címkézettek, output kapui $z_0, \ldots, z_n$.
A bemeneti két n-bites szám összegét (egy $n+1$-bites számot) kell a kimenetre küldenie.

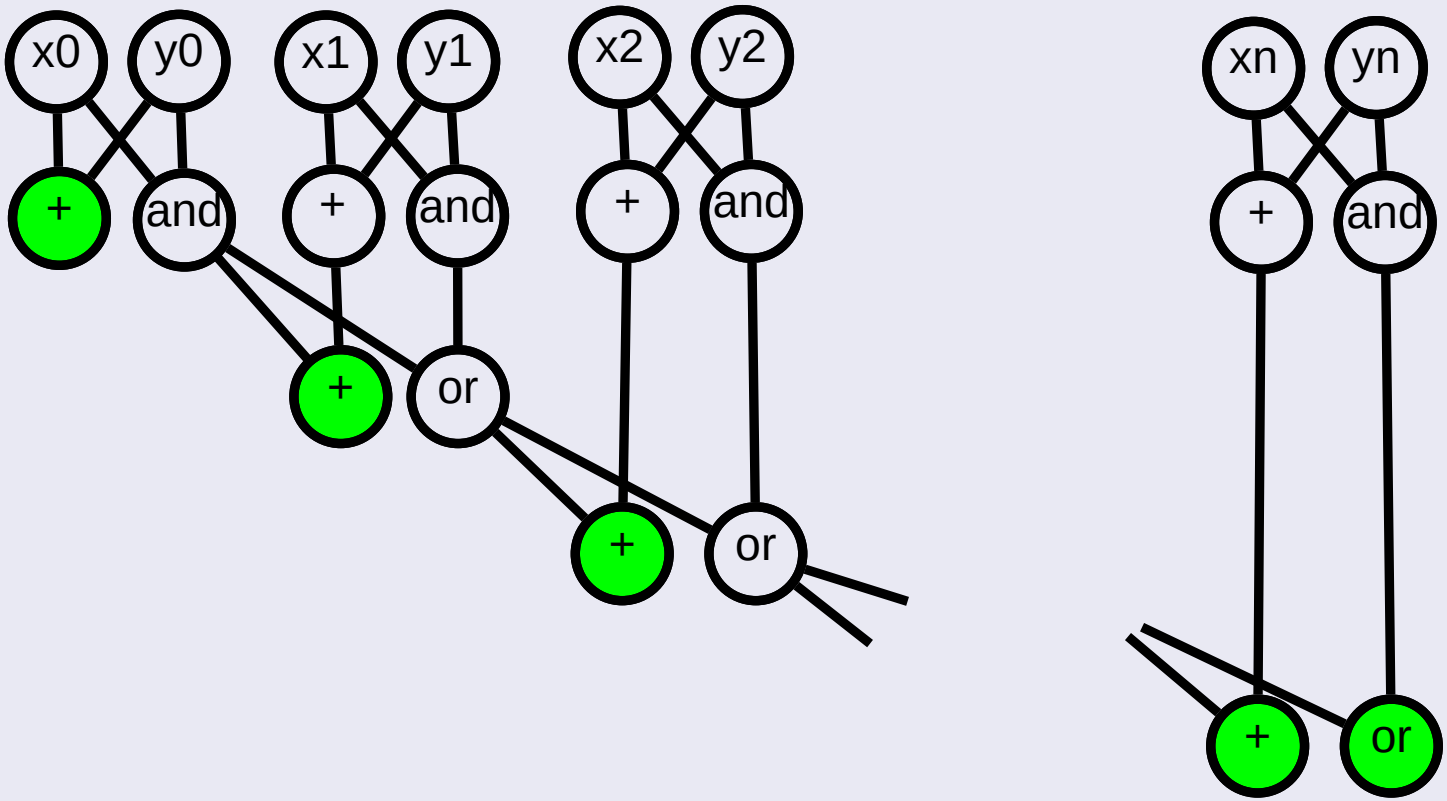
Összeadó hálózatok

Triviális megvalósítás



Összeadó hálózatok

Triviális megvalósítás



Lineáris mélység!

Összeadás logaritmikus mélységben

Elég kiszámítani a carry biteket ($c_i = 1$, ha az i. helyiértéken keletkezik átvitel, vagyis ha az $x_0 x_1 \dots x_{n-1} + y_0 y_1 \dots y_{i-1}$ összeg i. helyiértéke 1-es.

Összeadás logaritmikus mélységben

Elég kiszámítani a carry biteket ($c_i = 1$, ha az i. helyiértéken keletkezik átvitel, vagyis ha az $x_0 x_1 \dots x_{n-1} + y_0 y_1 \dots y_{i-1}$ összeg i. helyiértéke 1-es.
Ezek után $z_i = x_i \oplus y_i \oplus c_{i-1}$, ahol $\oplus$ a kizáró vagy művelet (összeadás $\mathbb{Z}_2$-ben).

Összeadás logaritmikus mélységben

Elég kiszámítani a carry biteket ($c_i = 1$, ha az i. helyiértéken keletkezik átvitel, vagyis ha az $x_0 x_1 \dots x_{n-1} + y_0 y_1 \dots y_{i-1}$ összeg i. helyiértéke 1-es.
Ezek után $z_i = x_i \oplus y_i \oplus c_{i-1}$, ahol $\oplus$ a kizáró vagy művelet (összeadás $\mathbb{Z}_2$-ben).
(pontosabban, $z_n = c_{n-1}$ és $z_0 = x_0 \oplus y_0$.)
Ez párhuzamosan, konstans mélységben elvégezhető.

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összefüggések

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összefüggések

- $c_{i,i} = x_i \ \wedge \ y_i$.

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összefüggések

- $c_{i,i} = x_i \ \wedge \ y_i$.
- $p_{i,i} = x_i \ \vee \ y_i$.

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összefüggések

- $c_{i,i} = x_i \ \wedge \ y_i$.
- $p_{i,i} = x_i \ \vee \ y_i$.
- $c_{i,i+2d} = (c_{i,i+d} \wedge p_{i+d+1,i+2d}) \vee c_{i+d+1,i+2d}$.

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összefüggések

- $c_{i,i} = x_i \ \wedge \ y_i$.
- $p_{i,i} = x_i \ \vee \ y_i$.
- $c_{i,i+2d} = (c_{i,i+d} \wedge p_{i+d+1,i+2d}) \vee c_{i+d+1,i+2d}$.
- $p_{i,i+2d} = p_{i,i+d} \wedge p_{i+d+1,i+2d}$.

Összeadás logaritmikus mélységben

Carry és propagate

Adott $x_0 \dots x_{n-1}$ és $y_0 \dots y_{n-1}$ input egészek, bitsorozat formájában és $0 \leq i \leq j \leq n$. Legyen

- $c_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j$ összegnél keletkezik carry;
- $p_{i,j} = 1$, ha az $x_i x_{i+1} \dots x_j + y_i y_{i+1} \dots y_j + 1$ összegnél keletkezik carry.

Összefüggések

- $c_{i,i} = x_i \ \wedge \ y_i$.
- $p_{i,i} = x_i \ \vee \ y_i$.
- $c_{i,i+2d} = (c_{i,i+d} \wedge p_{i+d+1,i+2d}) \vee c_{i+d+1,i+2d}$.
- $p_{i,i+2d} = p_{i,i+d} \wedge p_{i+d+1,i+2d}$.

Ezekkel az összefüggésekkel $\mathcal{O}(\log n)$ mélységben kiszámítható a carry és a propagate az összes $(0, j)$ párra.

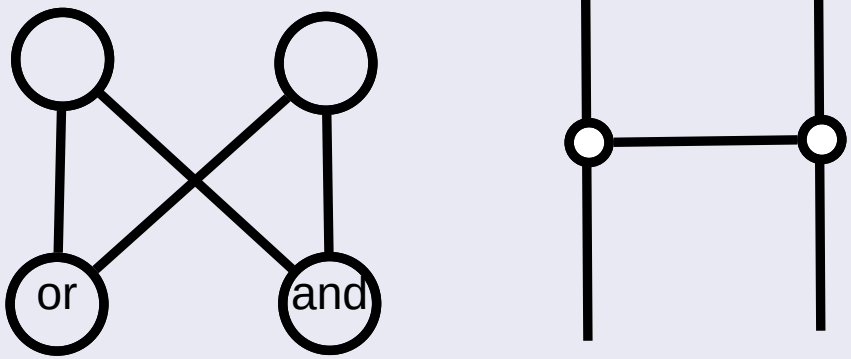
Rendezés $\mathcal{O}(\log^2 n)$ mélységben

A páros-páratlan mergesort (Batcher)

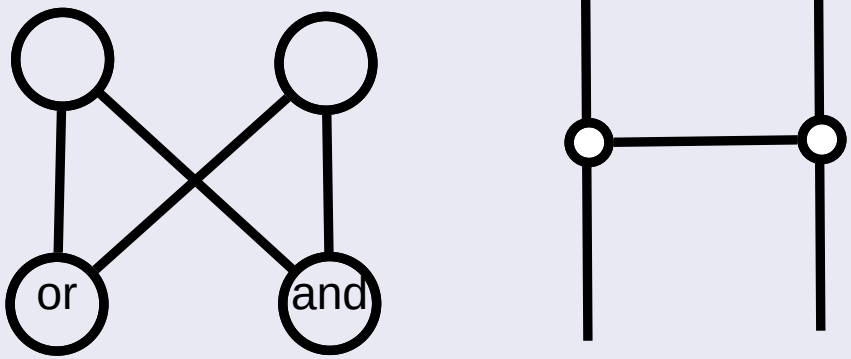
function MERGE($T[1..2N]$)
    if $N == 1$ **then** compareAndSwap($T[1], T[2]$)
    else
        MERGE($T[1, 3, 5, \ldots, 2N-1]$)
        MERGE($T[2, 4, 6, \ldots, 2N]$)
        for $i = 2, 4, 6, \ldots, 2N-2$ **do**
compareAndSwap($T[i], T[i+1]$)

function SORT($T[1..2N]$)
    if $N == 1$ **then** compareAndSwap($T[1], T[2]$)
    else
        SORT($T[1..N]$)
        SORT($T[N+1..2N]$)
        MERGE($T[1..2N]$)

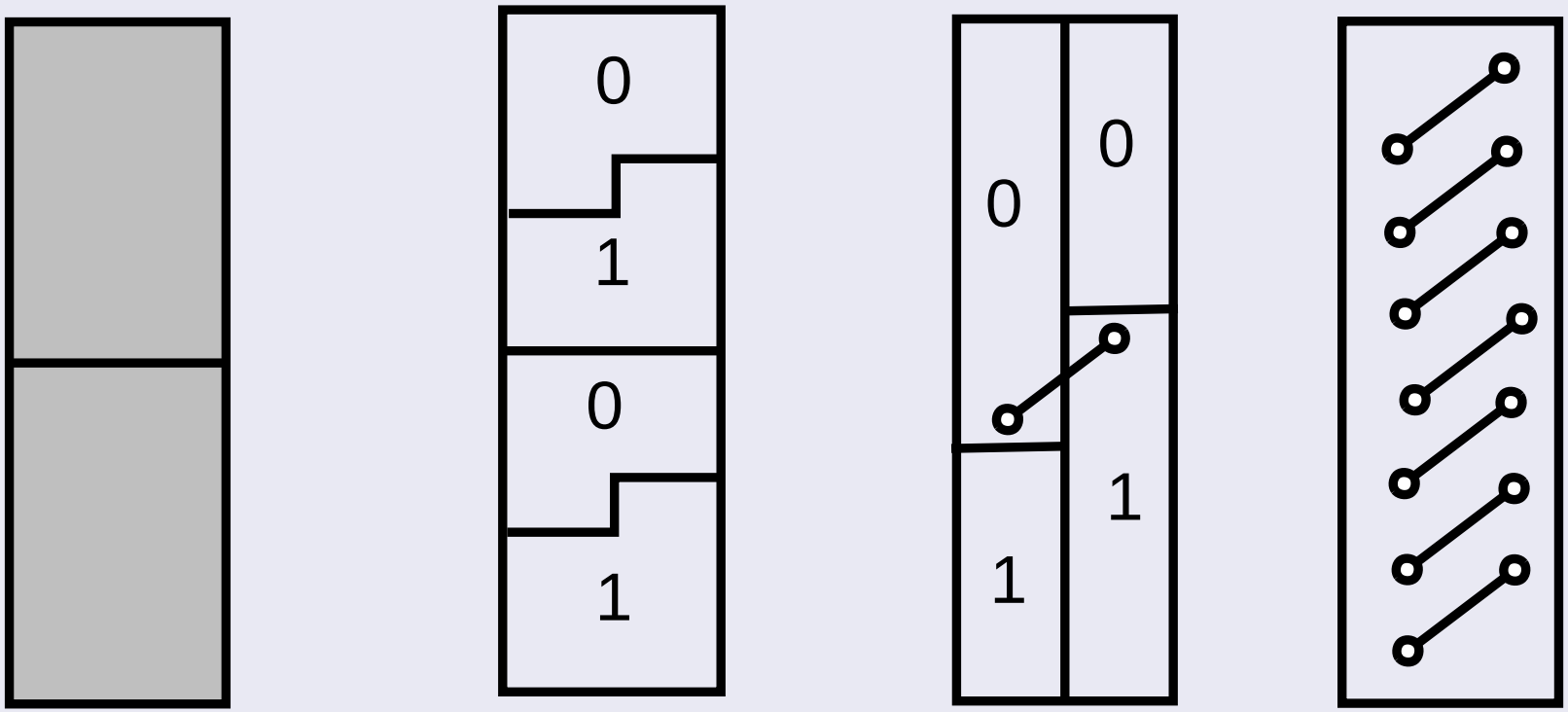
compareAndSwap



compareAndSwap



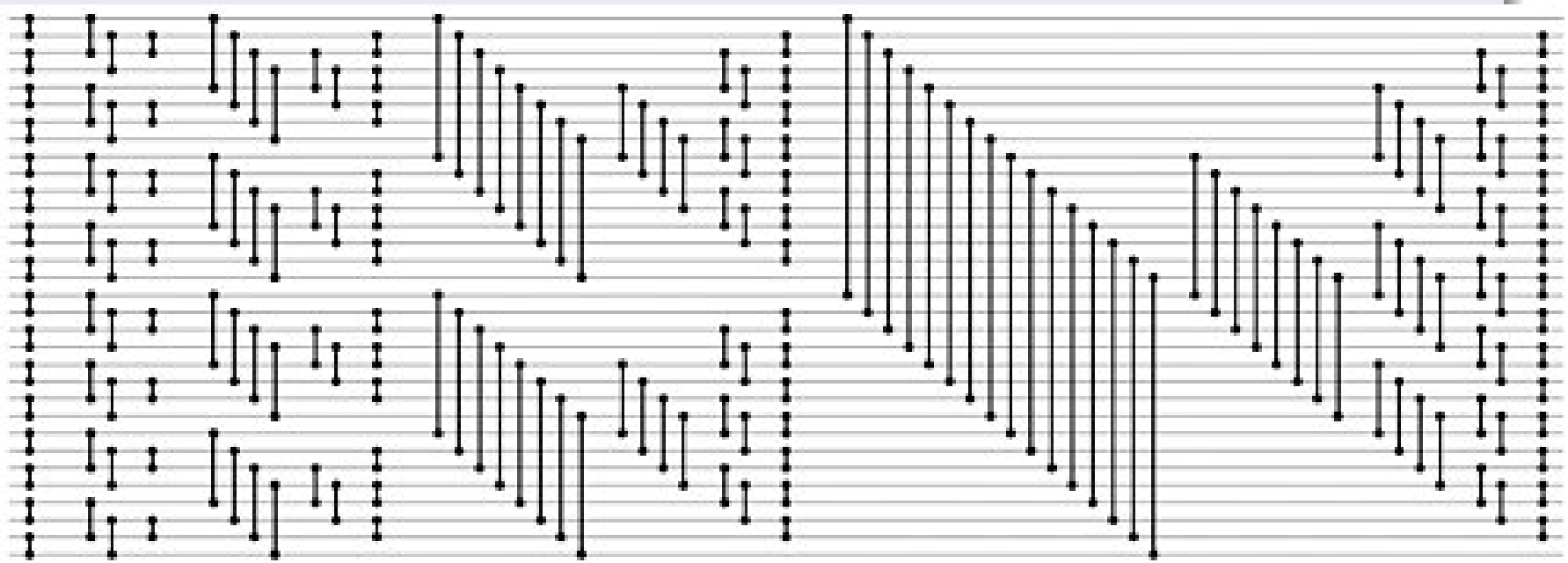
A páros-páratlan mergesort vizualizálva



Rendezés $\mathcal{O}(\log^2 n)$ mélységben

A teljes rendező hálózat mélysége: $\mathcal{O}(\log^2 n)$.

Példa: 32-bites rendező hálózat



Többségi Függvény $\in NC^2$

A Többségi Függvény akkor vesz fel 1 értéket, ha argumentumainak legalább a fele 1-es.

Többségi Függvény $\in NC^2$

A Többségi Függvény akkor vesz fel 1 értéket, ha argumentumainak legalább a fele 1-es.

A függvény az előzőek szerint NC^2-beli:

Többségi Függvény $\in NC^2$

A Többségi Függvény akkor vesz fel 1 értéket, ha argumentumainak legalább a fele 1-es.

A függvény az előzőek szerint NC^2-beli:

- rendezzük a biteket;

Többségi Függvény $\in NC^2$

A Többségi Függvény akkor vesz fel 1 értéket, ha argumentumainak legalább a fele 1-es.

A függvény az előzőek szerint NC^2-beli:

- rendezzük a biteket;
- visszaadjuk az $\frac{n}{2}$. elemet a rendezett sorozatból.

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke
Az output kapuk: $z_{i,j}$, a két mátrix szorzatának i. sorának j. oszlopának értéke

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke
Az output kapuk: $z_{i,j}$, a két mátrix szorzatának i. sorának j. oszlopának értéke

$$z_{i,j} = \bigvee_{k=1}^{n} (x_{i,k} \wedge y_{k,j}).$$

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke
Az output kapuk: $z_{i,j}$, a két mátrix szorzatának i. sorának j. oszlopának értéke

$$z_{i,j} = \bigvee_{k=1}^{n} (x_{i,k} \wedge y_{k,j}).$$

$\log n$ mélység

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke
Az output kapuk: $z_{i,j}$, a két mátrix szorzatának i. sorának j. oszlopának értéke

$$z_{i,j} = \bigvee_{k=1}^{n} (x_{i,k} \wedge y_{k,j}).$$

$\log n$ mélység

- Minden (i, k, j) hármasra kiszámítjuk az $x_{i,k} \wedge y_{k,j}$ értéket párhuzamosan: 1 mélység, n^3 kapu.

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke
Az output kapuk: $z_{i,j}$, a két mátrix szorzatának i. sorának j. oszlopának értéke

$$z_{i,j} = \bigvee_{k=1}^{n} (x_{i,k} \wedge y_{k,j}).$$

$\log n$ mélység

- Minden (i, k, j) hármasra kiszámítjuk az $x_{i,k} \wedge y_{k,j}$ értéket párhuzamosan: 1 mélység, n^3 kapu.
- Minden (i, j) párosra a fenti $z_{i,j}$ értéket párhuzamosan: $\log n$ mélység (az 1^n-nél látott majdnem teljes bináris fa topológiával), összesen n^3 kapu.

Mátrixszorzás: $\mathcal{O}(\log n)$ mélység

Az input kapuk: két $n \times n$-es Boole-mátrix elemei, $x_{i,j}$ ill. $y_{i,j}$ a megfelelő mátrix i. sorának, j. oszlopának értéke
Az output kapuk: $z_{i,j}$, a két mátrix szorzatának i. sorának j. oszlopának értéke

$$z_{i,j} = \bigvee_{k=1}^{n} (x_{i,k} \wedge y_{k,j}).$$

$\log n$ mélység

- Minden (i, k, j) hármasra kiszámítjuk az $x_{i,k} \wedge y_{k,j}$ értéket párhuzamosan: 1 mélység, n^3 kapu.
- Minden (i, j) párosra a fenti $z_{i,j}$ értéket párhuzamosan: $\log n$ mélység (az 1^n-nél látott majdnem teljes bináris fa topológiával), összesen n^3 kapu.

Vagyis: $\mathcal{O}(\log n)$ mélység, polinom sok kapu.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.
- $A + I$: a főátlót 1-re állítjuk.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.
- $A + I$: a főátlót 1-re állítjuk.
- $(A + I)^k$: az (i, j) cella értéke pontosan akkor 1, ha van legfeljebb k hosszú séta i-ből j-be.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.
- $A + I$: a főátlót 1-re állítjuk.
- $(A + I)^k$: az (i, j) cella értéke pontosan akkor 1, ha van legfeljebb k hosszú séta i-ből j-be.
- Iterált négyzetreemeléssel meghatározzuk $(A + I)^{2^{\lceil \log n \rceil}}$-t, majd ennek visszaadjuk az $(1, n)$ celláját.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.
- $A + I$: a főátlót 1-re állítjuk.
- $(A + I)^k$: az (i, j) cella értéke pontosan akkor 1, ha van legfeljebb k hosszú séta i-ből j-be.
- Iterált négyzetreemeléssel meghatározzuk $(A + I)^{2^{\lceil \log n \rceil}}$-t, majd ennek visszaadjuk az $(1, n)$ celláját.
- Egy négyzetreemelés: $\mathcal{O}(\log n)$ mélység.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.
- $A + I$: a főátlót 1-re állítjuk.
- $(A + I)^k$: az (i, j) cella értéke pontosan akkor 1, ha van legfeljebb k hosszú séta i-ből j-be.
- Iterált négyzetreemeléssel meghatározzuk $(A + I)^{2^{\lceil \log n \rceil}}$-t, majd ennek visszaadjuk az $(1, n)$ celláját.
- Egy négyzetreemelés: $\mathcal{O}(\log n)$ mélység.
- $\log n$ négyzetreemelés szekvenciálisan.

Elérhetőség $\in NC^2$

- Legyen $A \in \{0,1\}^{n \times n}$ a G gráf szomszédsági mátrixa.
- $A + I$: a főátlót 1-re állítjuk.
- $(A + I)^k$: az (i, j) cella értéke pontosan akkor 1, ha van legfeljebb k hosszú séta i-ből j-be.
- Iterált négyzetreemeléssel meghatározzuk $(A + I)^{2^{\lceil \log n \rceil}}$-t, majd ennek visszaadjuk az $(1, n)$ celláját.
- Egy négyzetreemelés: $\mathcal{O}(\log n)$ mélység.
- $\log n$ négyzetreemelés szekvenciálisan.
- Összesen $\mathcal{O}(\log^2 n)$ mélység.

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Nem ismert, hogy

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Nem ismert, hogy

- igaz-e, hogy $NC^i = NC^{i+1}$ valamely i-re?

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Nem ismert, hogy

- igaz-e, hogy $NC^i = NC^{i+1}$ valamely i-re?
- NC^1 vs. L?

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Nem ismert, hogy

- igaz-e, hogy $NC^i = NC^{i+1}$ valamely i-re?
- NC^1 vs. L?
- NC^1 vs. NL?

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Nem ismert, hogy

- igaz-e, hogy $NC^i = NC^{i+1}$ valamely i-re?
- NC^1 vs. L?
- NC^1 vs. NL?
- NC^2 vs. NL?

Az NC osztályok a klasszikusak közt

Tétel

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC^3 \subseteq \ldots \subseteq NC \subseteq \mathbf{P}.$$

Nem ismert, hogy

- igaz-e, hogy $NC^i = NC^{i+1}$ valamely i-re?
- NC^1 vs. L?
- NC^1 vs. NL?
- NC^2 vs. NL?
- NC vs. $\mathbf{P}$?

Az utolsó kérdés másképp: igaz-e, hogy minden hatékonyan megoldható ($\mathbf{P}$-beli) probléma hatékonyan párhuzamosítható is, vagy vannak ,,inherensen szekvenciális" problémák?

Összefoglalás

- Megismertük a hálózatcsaláddal való eldöntés fogalmát.
- Megismertük az $\boldsymbol{NC}^k$ osztályok hierarchiáját és az $\boldsymbol{NC}$ osztályt.
- Láttuk, hogy bináris számokat össze lehet adni logaritmikus mélységben.
- Megismertük a Batcher-féle odd-even mergesortot.
- Láttuk, hogy a Többségi Függvény $\boldsymbol{NC}^2$-beli.
- Láttuk, hogy mátrixokat lehet szorozni $O(\log n)$ mélységű hálózattal.
- Láttuk, hogy Elérhetőség $\in \boldsymbol{NC}^2$.
- Láttuk, hogy $\boldsymbol{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} \subseteq \boldsymbol{NC}^2 \subseteq \boldsymbol{NC}^3 \subseteq \ldots \subseteq \boldsymbol{NC} \subseteq \mathbf{P}$.