

1D m° 1. Sorting networks

Hugo Sacou

Q1. Let α be a sorting network.

" \Rightarrow " Suppose α sorts $\langle n, \dots, 1 \rangle$. For all $i \in [1, n]$, we define $f_i : j \mapsto \begin{cases} 1 & \text{if } i \leq j \\ 0 & \text{else.} \end{cases}$

We have that $\alpha(\langle n, \dots, 1 \rangle) = \langle 1, \dots, n \rangle$

Then, for all $i \in [1, n]$,

$$\begin{aligned} \alpha(f_i(\langle n, \dots, 1 \rangle)) &= \alpha(\underbrace{\langle 1, \dots, 1}_{\overset{i}{\circ}}, 0, \dots, 0) \\ &= f_i(\langle 1, \dots, n \rangle) \\ &= \langle 0, \dots, 0, 1, \dots, 1 \rangle. \end{aligned}$$

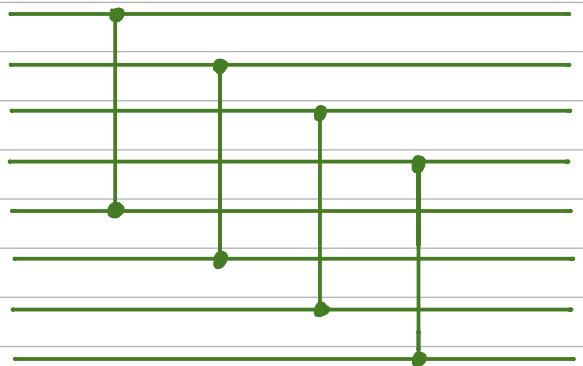
" \Leftarrow " Suppose $\alpha := \alpha(\langle n, \dots, 1 \rangle)$ is unsorted : there exists $i \in [1, n-1]$ such that $a_i < a_{i+1}$.

Define $b := \alpha(f_i(\langle n, \dots, 1 \rangle))$. Thus $b_i = 1$ and $b_{i+1} = 0$ with the lemma. Therefore, $\alpha(f_i(\langle n, \dots, 1 \rangle))$ is unsorted.

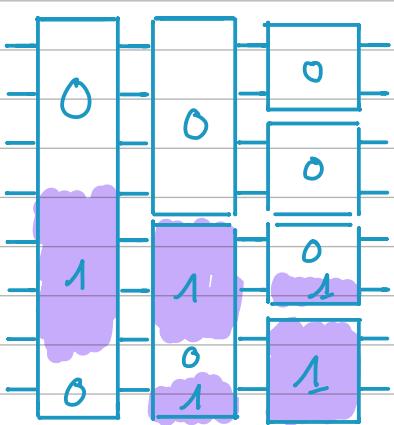
Example of a separator with 8 inputs

Q2. Yes it does !

Intuition:



Q3(a) We use the following network



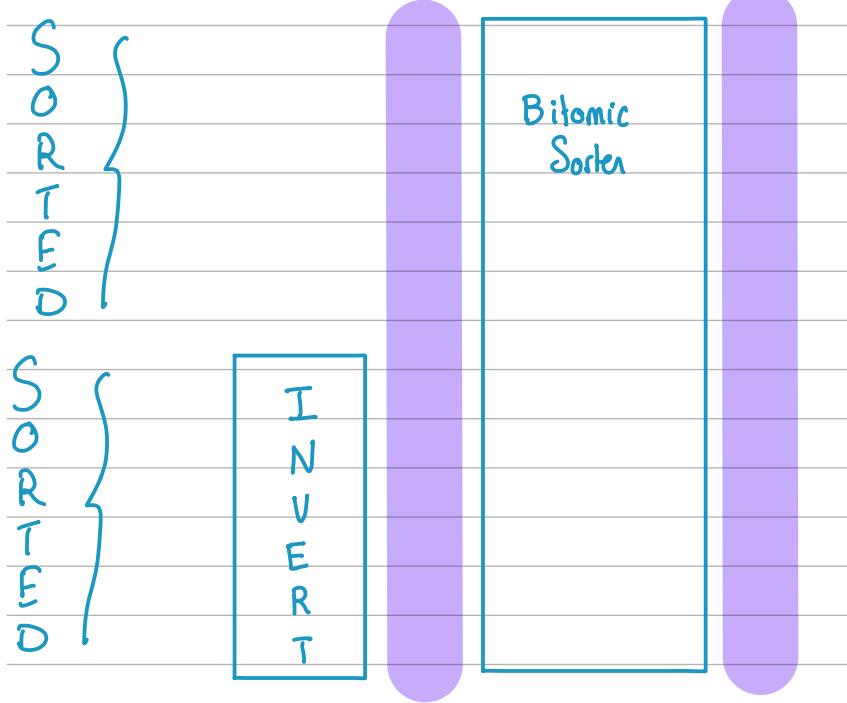
To see why it is correct, we show by induction that, for all binary bitonic sequences, they are correctly sorted.

Each step has the following property :

The current block is divided in two blocks such that one block is constant and the other is bitonic.

With $n = 2^m$ inputs, we have a network with depth $m = \log_2 n$ and $(n \cdot m)/2$ comparators.

Q 3(b)



Depth : $(\log n) \times (\log n)$


$$\# \text{ operators} : \frac{m}{2} (\log n) (\log n)$$

bitomic sequence

Q4(a)

$$\begin{bmatrix} 1 & 3 & 5 & 6 \\ 11 & 8 & 16 & 10 \\ 4 & 7 & 2 & 9 \\ 14 & 13 & 15 & 12 \end{bmatrix}$$

Step 1

$$\begin{bmatrix} 1 & 5 & 3 & 6 \\ 11 & 16 & 8 & 10 \\ 4 & 2 & 7 & 9 \\ 14 & 15 & 13 & 12 \end{bmatrix}$$

Step 2

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 14 & 13 \end{bmatrix}$$

Step 3

$$\begin{bmatrix} 1 & 2 & 3 & 6 \\ 5 & 4 & 8 & 7 \\ 11 & 14 & 9 & 10 \\ 16 & 15 & 13 & 12 \end{bmatrix}$$

Q4(b)

This can be done with an odd-even swap (like odd-even sort), thus we get a complexity of $n/2 - 1$ neighbor-swapping steps.

1D m^og PRAM 1

I. Selection in a list.

Question #1

For all i in parallel do

If $\text{next}[i] \neq \text{Nil}$ or $\text{color}[\text{next}[i]] = \text{blue}$ then
 $\text{end}[i] \leftarrow \text{true}$; $\text{blue}[i] \leftarrow \text{next}[i]$

While $\exists i$ such that $\text{end}[i] = \text{false}$ do

 For all i in parallel do
 if $\text{end}[i] = \text{false}$ then
 $\text{end}[i] \leftarrow \text{end}[\text{next}[i]]$
 if $\text{end}[i] = \text{true}$ then
 $\text{blue}[i] \leftarrow \text{blue}[\text{next}[i]]$
 $\text{next}[i] \leftarrow \text{next}[\text{next}[i]]$

II. Mystery Procedure

Question #2.

$$a) I_{\text{down}} = [0, 0, 0, 0, 0, 1, 2, 2]$$

$$I_{\text{up}} = [3, 4, 5, 6, 7, 7, 7, 8]$$

$$\text{Scan}(\text{Rew}(\text{Flags})) = [0, 1, 1, 1, 2, 3, 4, 5]$$

$$\text{Index} = [3, 4, 5, 6, 0, 1, 7, 2]$$

$$\text{Result} = [4, 2, 2, 5, 7, 3, 1, 7]$$

(a) It splits the array as "elts w/ flag=1" and "elts w/ flag=0"

(c) It can be done in $O(\log n)$ as PERMUTE and REVERSE can be done in $O(1)$ given enough PUs.

Question # 3 .

(a) $A = [101, 111, 011, 001, 100, 010, 111, 010]$

On trie la liste. C'est un Radix Sort.

(b)

(c) $O(\log^2 n)$ $O\left(\left(\frac{n}{p} + \log p\right) \log n\right)$.

III Connected components.

a) 1st step of GATHER

* $T = \{4, 3, 2, 1, 1, 3, 2, 6, 6\}$

1st step of Jump

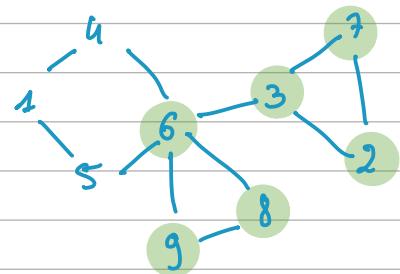
* $B = \{1, 2, 3, 4, 1, 1, 3, 2, 6, 6\}$

* $T = \{1, 2, 3, 4, 4, 1, 2, 3, 3, 3\}$

* $T = \{1, 2, 3, 4, 4, 1, 2, 3, 3, 3\}$

* $T = \{1, 2, 3, 4, 4, 1, 2, 3, 3, 3\}$

* $C = \{1, 2, 2, 1, 1, 1, 2, 2, 2, 2\}$



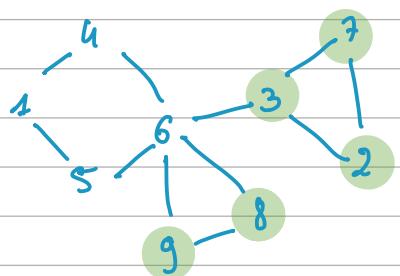
2nd step of Gather

* $T = \{1, 2, 2, 1, 1, 1, 2, 2, 2\}$

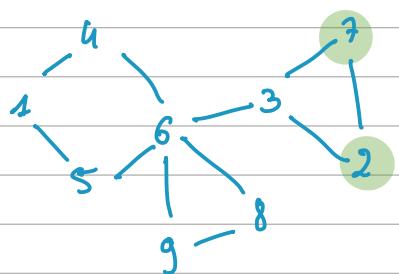
* $T = \{1, 2, 2, 1, 1, 1, 2, 2, 2, 2\}$

2nd step of Jump

* $C = \{1, 2, 2, 1, 1, 1, 2, 2, 2, 2\}$

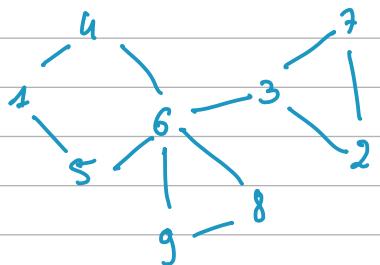


3rd step of Gather



3rd step of Jump

4th step of Gather



(d)

Scheduling

Question #1.

Let P be an optimization problem. Let A be a ρ -approximation of P . Then, take an input c of P and apply A to input $\lceil \rho c \rceil$.

$$f_p(A(c)) \leq \rho \cdot f_p(OPT(c)) \Leftrightarrow f_p(A(c)) \leq \rho c \Leftrightarrow f_p(A(c)) \leq \lceil \rho c \rceil$$

thus, assuming $\rho < (c+1)/c$ $\Rightarrow \lceil \rho c \rceil < \lceil c+1 \rceil = c+1$ and the existence of A , then we can decide

P in poly-time, thus $P = NP$.

Question #2

a)

Assume $D_{OPT} < 3w(T_i) \forall i$. Then, by considering the shortest task, then

$D_{OPT} < 3w(T_{\text{shortest}})$ thus each processor can execute at most two tasks.

Assign the longest unassigned task $\stackrel{\tau}{\sim}$ to the least "occupied" processor such that $D_{processor} + w(\tau) \leq 3 \cdot w(\text{shortest task})$.

b) Soit T_j une tâche finissant au temps $D(\sigma)$ et $w(T_j) = d$ et on pose $s := \frac{m}{D(\sigma) - d}$. Alors, au temps s , si tous les processeurs sont occupés

$$\sum_{i \neq j} w(T_i) \geq p \times s = p(D(\sigma) - d)$$

$$\text{autrement dit, } D(\sigma) \leq \frac{\sum_{i \neq j} w(T_i)}{p} - d \leq \frac{\sum_i w(T_i) - d}{p} - d$$

$$\text{Or, } D_{OPT} \geq \frac{\sum_i w(T_i)}{p} \text{ donc } D(\sigma) \leq D_{OPT} + d \times \frac{p-1}{p}. \quad (\ast)$$

Ensuite, supposons $D(\sigma) > \left(\frac{4}{3} - \frac{1}{3p}\right) D_{OPT}$. $(\ast\ast)$

D'après (*), on a alors

$$D_{\text{OPT}} + \frac{p-1}{p} d > \left(\frac{4}{3} - \frac{1}{3p}\right) D_{\text{OPT}}$$

donc $3d > D_{\text{OPT}}$.

Si $d = \min_{i \in V} w(T_i)$ alors on applique (a) alors τ est optimal en contradiction avec (**) car $D(\tau) = D_{\text{OPT}}$.

Simen, on considère $\tau' = \tau \mid \{i \mid w(T_i) \geq d\}$ et on a: $D(\tau') = D(\tau)$
 $D_{\text{OPT}} \leq D_{\text{OPT}'}$

et on applique (a) à τ' .

Question #3.

a) Réduction depuis CLIQUE. Soit $(G = (V, E), k)$ une instance de CLIQUE.

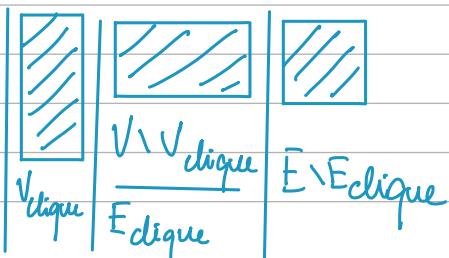
On considère les tâches :

- T_v pour chaque sommet $v \in V$,
- K_e pour chaque arête $e = (u, v) \in E$ et $T_u, T_v \in K_e$,
- trois ensembles $X \cup Y \cup Z$ de tâches.

Posons $p := \max \left(k, |V| - k + k \frac{(k-1)}{2}, m - k \frac{(k-1)}{2} \right)$.
Sommes de la clique Sommets restants Arêtes de la clique Arêtes restantes

On définit $|X| = p - k$, $|Y| = p - \left(|V| - k + \frac{k(k-1)}{2}\right)$ et $|Z| = p - (m - \frac{1}{2}p(p-1))$.

À la première étape, on place les sommets de la clique,



b) Il n'existe pas de $(p < \frac{4}{3})$ -approximation sauf si $P = NP$.

Final Exam

I Sorting network

Q1.



Q2.



II Algorithms for PRAMs

1. Well of a graph

1) For all i^o in parallel
 [well [i^o] \leftarrow true]

2) Prefix computation w/ n
 [set well [i, j^o] \leftarrow (i^o / j^o , false)]

For all i^o, j^o in parallel

if edge [i^o, j^o] exists then
 [well [i^o] \leftarrow false]

else

[well [j^o] \leftarrow false]

$$O(\log n^2) = O(\log n).$$

3) $G(n)$ with ^{the} algorithm of Q1.

4) $G(n)$

2. A walk on the forest

for all i in parallel

feuille [i] \leftarrow true

feuille [parent [i]] \leftarrow false

taille [i] \leftarrow feuille [i] ? 1 : 0

tant que parent [i] \neq NULL faire

| taille [parent [i]] \leftarrow taille [parent [i]] + taille [i]
| parent [i'] \leftarrow parent [parent [i]]

si parent [i] \neq NULL

III Message-passing algorithms for rings and grids of processes

1. Gossip