TD n° 1. Sorting networks

Hugo SACAU

Q1. Let α be a sorting network.

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

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

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

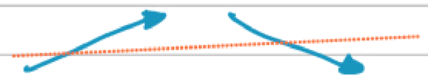
$$\alpha(f_i(\langle n, \dots, 1\rangle)) = \alpha(\langle \overbrace{1, \dots, 1}^{i}, 0, \dots, 0\rangle)$$
$$= f_i(\langle 1, \dots, n\rangle)$$
$$= \langle 0, \dots, 0, 1, \dots, 1\rangle.$$

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

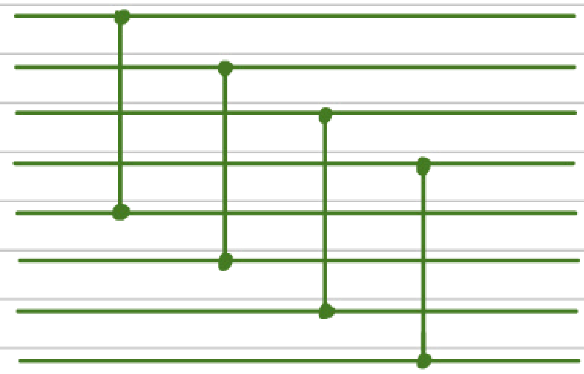
Define $b := \alpha(f_i(\langle n, \dots, 1\rangle))$. Thus $b_i = 1$ and $b_{i+1} = 0$ with the lemma. Therefore, $\alpha(f_i(\langle n, \dots, 1\rangle))$ is unsorted.

Q2. Yes it does!

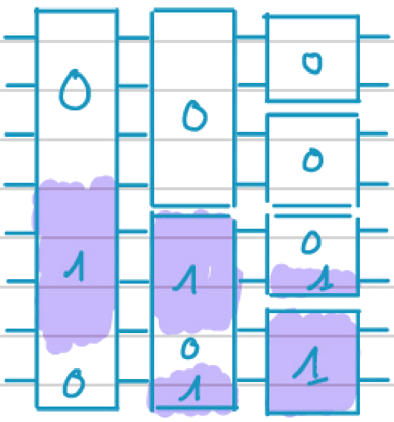
Intuition:



Example of a separator with 8 inputs



Q3 (a) We use the following network



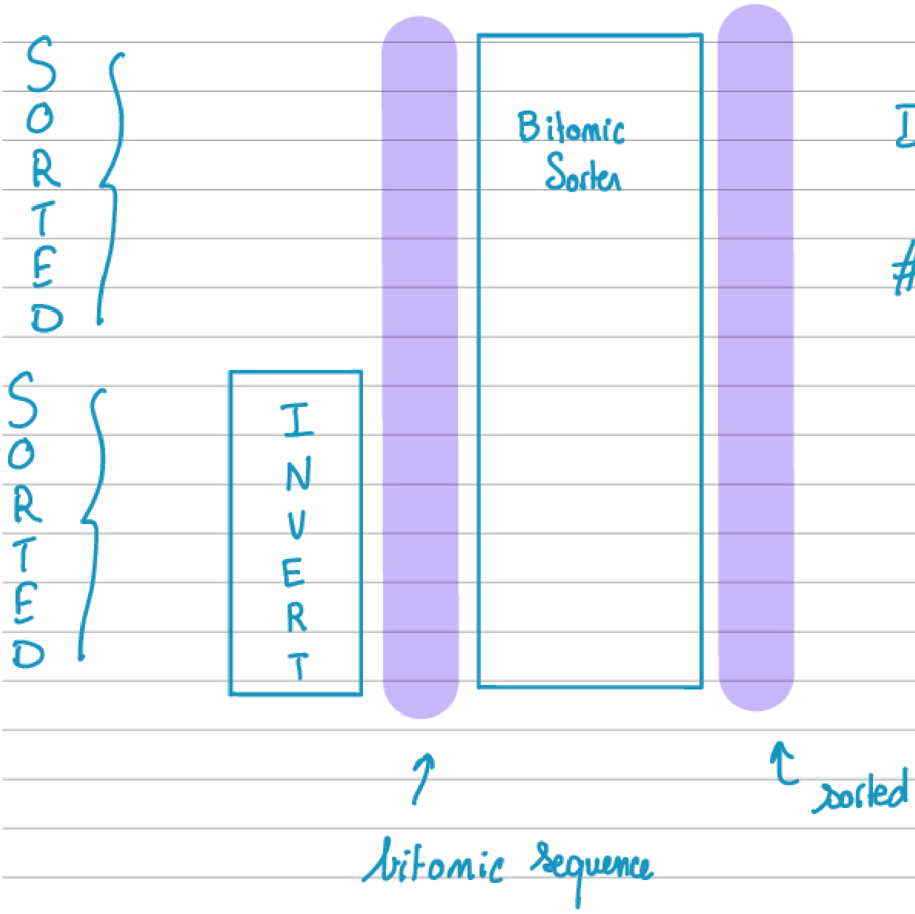
To see why it is correct, we show by induction that, for all binary bitonic sequences, they are correctly sorted.

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 : $(\log n) \times \underbrace{(\log n)}_{\text{bitonic sorter}}$

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

Q4(a)

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

Step 2

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

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.

TD n°2 PRAM 1

I. Selection in a list.

Question #1

For all i in parallel do
  If $next[i] \neq Nil$ or $color[next[i]] = blue$ then
    $end[i] \leftarrow true$; $blue[i] \leftarrow next[i]$

While $\exists i$ such that $end[i] = false$ do
  For all i in parallel do
    if $end(i) = false$ then
      $end(i) \leftarrow end(next(i))$
      if $end(i) = true$ then
        $blue(i) \leftarrow blue(next(i))$
      $next(i) \leftarrow next(next(i))$

II. Mystery Procedure

Question #2.

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

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

$Scan(Rev(Flags)) = [0, 1, 1, 1, 2, 3, 4, 5]$

$Index = [3, 4, 5, 6, 0, 1, 7, 2]$

$Result = [4, 2, 2, 5, 7, 3, 1, 7]$

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

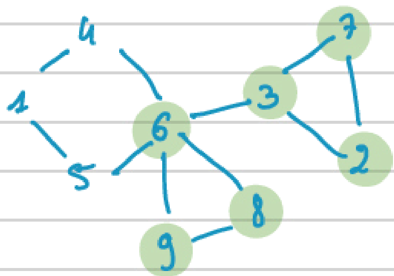
II Connected components.

a) 1st step of GATHER

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

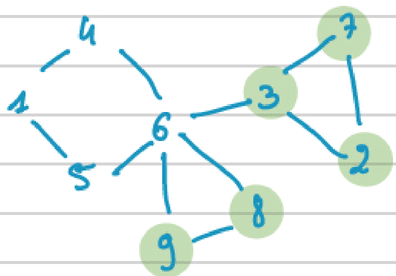
1st step of Jump

* $B = \{\overset{1}{4}, \overset{2}{3}, \overset{3}{2}, \overset{4}{1}, \overset{5}{1}, \overset{6}{3}, \overset{7}{2}, \overset{8}{6}, \overset{9}{6}\}$
* $T = \{\overset{1}{1}, \overset{2}{2}, \overset{3}{3}, \overset{4}{4}, \overset{5}{4}, \overset{6}{2}, \overset{7}{3}, \overset{8}{3}, \overset{9}{3}\}$
* $T = \{1, 2, 3, 4, 4, 2, 3, 3, 3\}$
* $T = \{1, 2, 3, 4, 4, 2, 3, 3, 3\}$
* $C = \{1, 2, 2, 1, 1, 2, 2, 2, 2\}$



2nd step of Gather

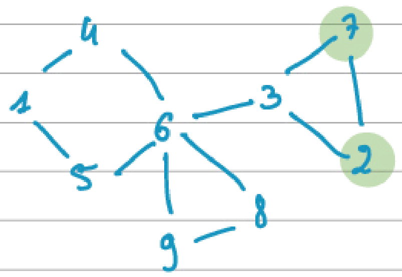
* $T = \{1, 2, 2, 1, 1, 1, 2, 2, 2\}$
* $T = \{1, 2, 2, 1, 1, 2, 2, 2, 2\}$



2nd step of Jump

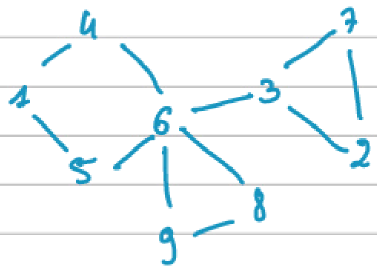
* $C = \{1, 2, 2, 1, 1, 2, 2, 2, 2\}$

3rd step of Gather



3rd step of Jump

4th step of Gather



(d)

TD n° 4

Scheduling

Question #1.

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

$$f_P(\mathcal{A}(c)) \leq \rho \cdot f_P(OPT(c)) \iff f_P(\mathcal{A}(c)) \leq \rho c \iff f_P(\mathcal{A}(c)) \leq \lceil \rho c \rceil$$

thus, assuming $\rho < (c+1)/c$ $\to \lceil \rho c \rceil < \lceil c+1 \rceil = c+1$ and the existence of $\mathcal{A}$, then we can decide P in poly-time, thus $P = NP$.

Question #2

a)

Assume $D_{OPT} < 3\,w(T_i)\ \forall i$. Then, by considering the shortest task, then $D_{OPT} < 3\,w(T_{shortest})$ thus each processor can execute at most two tasks.

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

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

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

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

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

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

D'après (*), on a alors

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

donc $3d > D_{OPT}$.

Si $d = \min_i w(T_i)$ alors on applique (a) alors σ est optimal en contradiction avec (**) car $D(\sigma) = D_{OPT}$.

Sinon, on considère $\sigma' = \sigma|_{\{i \mid w(T_i) \geq d\}}$ et on a : $D(\sigma') = D(\sigma)$
$D_{OPT'} \leq D_{OPT}$

et on applique (a) à σ'.

Question #3.

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

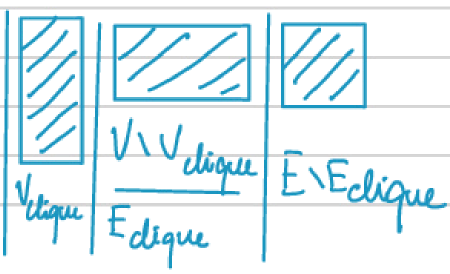
On considère les tâches :

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

Posons $p := \max\Big(\underbrace{k}_{\text{sommets de la clique}}, \underbrace{|V| - k}_{\text{sommets restants}} + \underbrace{k\frac{(k-1)}{2}}_{\text{arêtes de la clique}}, \underbrace{m - k\frac{(k-1)}{2}}_{\text{arêtes restantes}}\Big)$.

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,

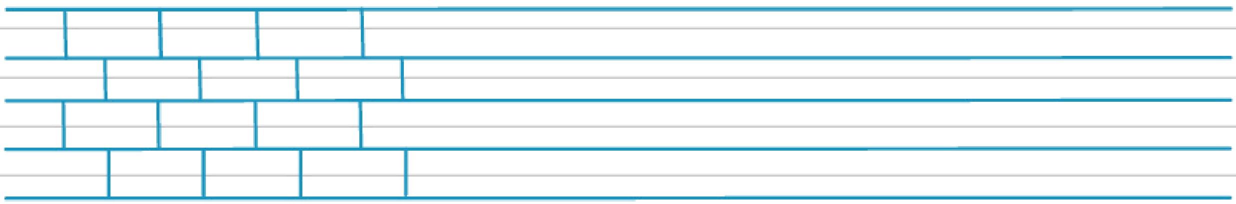


b) Il n'existe pas de $(\rho < \frac{4}{3})$-approximation sauf si $P = NP$.

Final Exam

I Sorting network

Q1. 

Q2. 

II Algorithms for PRAMs

1. Well of a graph

1) For all i in parallel
   └ well[i] ← true

   For all i, j in parallel
   | if edge[i, j] exists then
   | | well[i] ← false
   | else
   └ └ well[j] ← false

2) Prefix computation w/ n
   (set well'[i, j] ← (i/j, false)

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

3) $O(n)$ with the algorithm of Q1.

4) $O(n)$

2. A walk in the forest

for all i in parallel

feuille[i] ← true

feuille[parent[i]] ← false

taille[i] ← feuille[i] ? 1 : 0

tant que parent[i] ≠ NULL faire

    taille[parent[i]] ← taille[parent[i]] + taille[i]
    parent[i] ← parent[parent[i]]

si parent[i] ≠ NULL

III Message-passing algorithms for rings and grids of processors

1. Gossip