

# Circuits. PC 3.

## Correction

### 1 Parité

La fonction  $\text{PARITY}_n$  est la fonction  $\{0, 1\}^n \rightarrow \{0, 1\}$  qui vaut 1 si et seulement si un nombre impair de ses arguments valent 1. Montrer que  $\text{PARITY}_n$  se calcule par un circuit booléen de profondeur  $\mathcal{O}(\log(n))$  et de taille  $\mathcal{O}(n)$ .

Correction: On écrit récursivement  $\text{PARITY}_n(x_1, \dots, x_n)$  comme

$$\text{PARITY}_{n/2}(x_1, \dots, \lfloor n/2 \rfloor) \oplus \text{PARITY}_{n/2}(\lfloor n/2 \rfloor + 1, \dots, x_n),$$

où  $\oplus$  désigne  $\text{PARITY}_2$ , facile à écrire comme un circuit booléen, et on déroule la récurrence. (les transparents du cours montre  $\text{PARITY}_4$  écrit comme  $(x_1 \oplus x_2) \oplus (x_3 \oplus x_4)$ ).

### 2 Majorité

1. Soit  $\text{ADD}_n : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$  la fonction qui prend en entrée deux nombres binaires et produit le résultat de leur somme. Montrer qu'elle peut se calculer par un circuit booléen de de taille  $\mathcal{O}(n)$ .

Correction: On utilise la méthode apprise à l'école pour réaliser une addition en binaire, en ajoutant les chiffres de droite à gauche, et en propageant la retenue.

Formellement, on crée une brique de base  $\text{ADDER}$  à 3 entrées  $x, y$  et  $z$  et 2 sorties : la seconde est  $x \oplus y \oplus z$  ; la première est  $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ .

Soit  $x$  écrit en binaire  $x = x_n \dots x_2 x_1$  et  $y$  écrit en binaire  $y = y_n \dots y_2 y_1$ . Notons  $z = x + y$ , qui s'écrit en binaire  $z = z_{n+1} z_n \dots z_1$ .  $z_1$  est donné par la deuxième sortie de  $\text{ADDER}(x_1, y_1, \mathbf{0})$ . Pour l'indice  $i = 2, 3, \dots, n$ , on ajoute un circuit  $\text{ADDER}(x_i, y_i, c_i)$  où  $c_i$  est la première sortie du  $\text{ADDER}$  que l'on vient d'ajouter pour l'indice  $i - 1$ .  $z_i$  se lit alors sur la deuxième sortie de celui ajouté à l'indice  $i$ .  $z_{n+1}$  est donné par la première sortie du  $\text{ADDER}$  pour l'indice  $i = n$ .

2. On définit  $\text{MAJORITY}_n : \{0, 1\}^n \rightarrow \{0, 1\}$  telle que  $\text{MAJORITY}_n(x_1, \dots, x_n)$  vaut 0 si  $\sum x_i < n/2$ , 1 sinon. Montrer que cette fonction se calcule

- (a) par un circuit de taille  $\mathcal{O}(n^2)$ .  
 Correction: On utilise  $n$  fois l'additionneur  $\text{ADD}_n$ , et on branche le résultat sur un circuit qui compare le résultat (avec la méthode apprise à l'école) à  $n/2$ .
- (b) par un circuit de taille  $\mathcal{O}(n \log(n))$ .  
 Correction: Plutôt que d'additionner  $n$ -fois successivement, on calcule récursivement la somme de la première moitié, et la somme de la seconde moitié. Cela donne  $\mathcal{O}(n \log(n))$  au lieu de  $\mathcal{O}(n^2)$ . On branche toujours le résultat sur un comparateur à  $n/2$ .
- (c) par un circuit de taille  $\mathcal{O}(n)$  (on pourra utiliser le fait que l'addition de 3 entiers se ramène à calculer la somme de deux entiers en profondeur constante).  
 Correction:  
 En fait, on va calculer  $\sum_i x_i$  avec un circuit de taille  $\mathcal{O}(n)$ . En le branchant sur un circuit de comparaison à  $n/2$ , cela sera terminé.  
 L'astuce est la suivante : pour additionner  $a = a_n \cdots a_1$ ,  $b = b_n \cdots b_1$ ,  $c = c_n \cdots c_1$ , on peut calculer  $a_i + b_i + c_i$  pour chaque  $i$ , qui s'écrit en binaire  $p_i q_i$ . On définit  $p$  qui s'écrit en binaire  $p_n \cdots p_1$  et  $q$  qui s'écrit en binaire  $q_n \cdots q_1$ . On a alors  $a + b + c = 2p + q$ . En observant que  $2p$  est un décalage de  $p$ , cela donne le moyen de ramener l'addition de 3 nombres  $a, b, c$  sur  $n$  bits à l'addition de 2 nombres  $p$  et  $q$  sur  $n$  bits.  
 Partant de  $n$  nombres sur 1 bits, en les groupant par 3, cela donne  $2 * \lceil n/3 \rceil$  nombres sur 1 bits à ajouter : les  $\lceil n/3 \rceil$   $p$  correspondants, et les  $\lceil n/3 \rceil$   $q$  correspondant. On utilise récursivement ce principe pour faire les additions des  $p$  et  $q$ .  
 On va obtenir un circuit de profondeur de l'ordre de  $\lceil \log_3(n) \rceil = \mathcal{O}(\log(n))$ .  
 Evaluons sa taille  $T(n)$ , pour  $n$  nombres. Chaque construction récursive ramène le problème sur  $n$  nombres à celui sur  $2 * \lceil n/3 \rceil$  nombres, avec  $\mathcal{O}(n)$  portes répétées pour produire les bits des  $p$  et des  $q$ . On a donc  $T(n) = \mathcal{O}(n) + T(2 \lceil n/3 \rceil)$ . En déroulant cette récurrence, on obtient que  $T(n) \leq \mathcal{O}(n)(1 + 2/3 + 4/9 + \cdots + (2/3)^k + \cdots) \leq \mathcal{O}(n)$ .

### 3 Réseaux de tri

Considérons une liste  $L = x_0, x_1, \dots, x_{n-1}$  de  $n$  éléments de  $E$ , un ensemble ordonné, avec  $n$  puissance de 2.

La liste  $L$  est dite *bitonique* si  $L$  est dans l'un des quatre cas suivants :

- Cas 1 :  $L$  est croissante.
- Cas 2 :  $L$  est décroissante.
- Cas 3 :  $L$  est croissante puis décroissante.
- ou un décalage circulaire de la liste est dans l'un de ces cas.

Voici un exemple de liste bitonique : 10 20 30 40 50 60 70 80 75 65 55 45 35 25 15 5. Un autre exemple 6 9 4 2 3 5.

Un *circuit comparateur* est un circuit  $C(x, y)$  à deux entrées  $x$  et  $y$  et deux sorties : la première donne le minimum de  $x$  et de  $y$ , et la seconde le maximum de  $x$  et de  $y$ .

On appelle *réseau de tri* un circuit constitué de circuits comparateurs.

On appelle un *réseau séparateur* un réseau de tri, constitué d'une ligne de circuits comparateurs tel que si la liste  $L = x_0, x_1, \dots, x_{n-1}$  est donnée en entrée (avec  $n = 2p$ ), alors seuls les éléments  $x_i$  et  $x_{i+p}$  sont comparés ensemble.

1. Montrer qu'un réseau séparateur appliqué à une liste bitonique produit une liste telle que :
  - (a) tout élément de la première moitié de la liste de sortie est inférieur à tous les éléments de la seconde moitié de la liste de sortie.
  - (b) la première moitié et la seconde moitié de la liste de sortie sont des listes bitoniques de  $p = n/2$  éléments.

Correction: *Laborieux : on épluche chacun des cas, et on vérifie que la propriété est bien vérifiée.*

2. Construire un circuit constitué de circuits comparateurs qui trie les éléments d'une liste bitonique en utilisant des réseaux séparateur. Évaluer la taille et la profondeur du circuit.

Correction: *Un réseau séparateur sur  $n$  entrées décompose une liste  $L$  bitonique à  $n = 2p$  éléments en deux sous-listes  $L_1$  et  $L_2$  à  $p$  éléments bitoniques, avec chaque élément de  $L_2$  supérieur à celui de  $L_1$ . On applique récursivement un réseau séparateur sur  $L_1$  et un réseau séparateur sur  $L_2$ , puis sur les 4 sous-listes obtenues, puis sur les 8 sous-listes obtenues, et ainsi de suite. Par récurrence sur la taille de la liste, le résultat sera trié, puisque concaténer deux listes triées avec la propriété que tout élément de la seconde est supérieur à la première garantit un tri.*

*La profondeur du circuit obtenu est  $\mathcal{O}(\log(n))$ . La taille est  $\mathcal{O}(n \log(n))$ .*

3. Expliquez comment adapter le réseau séparateur pour trier en ordre décroissant une liste bitonique.

Correction: *Inverser les sorties de chaque circuit comparateur.*

4. Décrire un réseau *fusion* qui transforme une liste en une liste bitonique en utilisant les réseaux tri croissant et tri décroissant.

Correction: *Etant donnée une liste à  $n = 2p$  entrées, on trie de façon croissante la première moitié de la liste, de façon décroissante la deuxième moitié, de la liste : en concaténant les résultats, on a une liste bitonique.*

5. En déduire un circuit qui trie les éléments d'une liste quelconque constitué uniquement de circuits comparateurs avec une profondeur  $\mathcal{O}(\log(n)^2)$ .

Correction: On construit récursivement la solution  $Tri(n)$  à  $n$  entrées. Etant donnée une liste à  $n = 2p$  entrées,  $Tri(n)$  consiste à appeler  $Tri(n/2)$  sur la première moitié de la liste dans l'ordre croissant, et  $Tri(n/2)$  sur la deuxième moitié de la liste en inversant les sorties de chaque comparateur, et donc dans l'ordre décroissant. Le résultat obtenu est bitonique. On le branche ensuite sur le circuit de tri de la question 2.

La profondeur  $p(n)$  vérifie  $p(n) = p(n/2) + \log n$ . En déroulant, cela donne une profondeur  $\mathcal{O}(\log(n)^2)$ .

6. Montrer que tout réseau de tri à  $n$  entrée possède  $\Omega(n \log n)$  circuits comparateurs.

Correction: C'est l'incarnation du fait que l'on ne peut pas trier en moins de  $n \log(n)$  comparaisons.

7. Montrer que tout réseau de tri à  $n$  entrée possède une profondeur au moins  $\log(n)$  (voir votre cours préféré).

Correction: C'est l'incarnation du fait que l'on ne peut pas calculer le min ou le max de  $n$  entrées en moins de  $\log(n)$  opérations (exercice classique).

## 4 Circuits sur $\mathbb{R}$

On considère des circuits et algorithmes sur la structure  $(\mathbb{R}, +, -, *, =, <)$ .

On cherche, étant donnés  $n$  nombres  $x_1, x_2, \dots, x_n$  à déterminer s'ils sont tous distincts deux à deux (problème DISTINCT)

1. Proposer un algorithme en temps  $\mathcal{O}(n \log n)$ .

Correction: On tri les réels  $x_1, x_2, \dots, x_n$ , et on vérifie qu'il n'y a pas deux indices consécutifs égaux.

2. Lorsque  $W \subset \mathbb{R}^n$ , on note  $\#(W)$  pour le nombre de ses composantes connexes.

Soit  $W = \{(x_1, \dots, x_n) \mid \prod_{i \neq j} (x_i - x_j) \neq 0\}$ . Montrer que  $\#(W) \geq n!$ .

Correction: Pour  $\sigma$  une permutation de  $\{1, 2, \dots, n\}$ , on note  $W_\sigma$  pour l'ensemble des  $(x_1, \dots, x_n)$  avec  $x_{\sigma(1)} < x_{\sigma(2)} < \dots < x_{\sigma(n)}$ . Il suffit de montrer que pour deux permutations distinctes  $\sigma, \sigma'$ , les  $W_\sigma$  et  $W_{\sigma'}$  ne sont pas connectés. Il doit exister des indices  $i$  et  $j$  tels que dans l'un  $x_i < x_j$  et dans l'autre  $x_i > x_j$ . Le théorème des valeurs intermédiaire dit que l'on ne peut pas passer de l'un à l'autre sans annuler  $x_j - x_i$ .

3. Montrer que si  $C$  est un arbre de décision sur  $(\mathbb{R}, +, -, *, =, <)$  à  $n$  entrées de profondeur  $d$ , alors l'ensemble des entrées acceptés est une union d'au plus  $2^d$  ensembles  $C_1, C_2, \dots, \subset \mathbb{R}^n$ , où chaque  $C_i$  se définit par un système de  $\leq d$  équations ou inéquations polynomiales de degré 2.

Correction: *Il suffit en fait d'associer un tel ensemble à chacune des au plus  $2^d$ . On ajoute une variable par sortie des portes sur chaque chemin de la racine vers l'une des feuilles. On écrit les contraintes qui correspondent à ce chemin.*

4. On admettra que si  $S \subset \mathbb{R}^n$  est défini par des contraintes de degré  $d$  avec  $m$  égalités et  $h$  inégalités, alors  $\#(S) \leq d(2d-1)^{n+h-1}$  (ne dépend pas de  $m$ ).

En déduire qu'il faut au moins un temps  $\mathcal{O}(n \log n)$  pour résoudre le problème DISTINCT.

Correction:

*Il n'y a pas plus que  $2^d$  fois le nombre  $n_\ell$  de composante connexes des contraintes correspondante à chaque branche  $\ell$  d'un arbre de décision dans un ensemble décidé avec un arbre de décision de profondeur  $d$ . On observe que  $n_\ell \leq 3^{n+d}$  par le résultat admis, et en lien avec la question précédente pour l'arbre de décision de  $W$ . Le nombre total de composante connexes de  $W$  doit donc être majoré par  $2^d 3^{n+d}$ , ce qui donne  $d \geq \Omega(\#(W)) - \mathcal{O}(n)$ .*

*Le résultat découle alors de la formule de Stirling, qui donne  $\log(n!) = \Omega(n \log(n))$ .*