Programmation sur architectures parallèlles

8INF856 - Devoir 1

Étudiants : **Lapu Matthias, Pellevoizin Jules, Guebel Samuel**

1 Question 1 :

1.1 Combien de commutateurs le réseau contient-il ?

D'après le cours, pour un réseau *Butterfly* avec 2^d processeurs connectés, il y a $n(\log(n)+1)$ commutateurs. Dans notre cas, le réseau étudié est un réseau de Benes de dimension k. Celui-ci correspond à deux réseaux *Butterfly* joints avec l'entrée du deuxième devenant alors la sortie. En reprenant la formule du cours avec $N = 2^n$, nous avons donc :

$$\mathit{Butterfly}(N) = N(\log(N)+1)$$

Ainsi, le nombre de commutateurs va donc doubler. Il sera cependant nécessaire d'enlever la partie jointe, et de les relier entre eux. En représentant cela sous forme de formule, nous avons donc :

$$\begin{aligned}\text{Benes}(N) &= 2\left(N(\log(N)+1)\right) - N \\ &= 2N\log(N) + 2N - N \\ &= 2N\log(N) + N \\ &= N(2\log(N)+1)\end{aligned}$$

Nous pouvons alors essayer d'appliquer notre formule au réseau de Benes de dimension 3 de l'énoncé. Celui-ci peut être représenté par un tableau de longueur 8 et de largeur 7. Ainsi, il y a donc 56 commutateurs. En utilisant la formule plus haut, avec $N = 2^3 = 8$ nous avons donc :

$$\begin{aligned}\text{Benes}(2^3 = 8) &= 2\left(8(\log(2^3)+1)\right) - 8 \\ &= 8(2\log(2^3)+1) \\ &= 8(2*3+1) \\ &= 8(7) \\ &= 56\end{aligned}$$

Nous retrouvons bien le même nombre de commutateurs.

1.2 Quel est le diamètre du réseau ?

D'après le cours, le diamètre correspond à la distance maximale entre 2 nœuds. Pour le réseau *Butterfly*, celui-ci est de $\log(N)$, ou de $\log(N)+1$, cela dépend de si nous prenons en compte la connexion avec le processeur en entrée. Dans le cas du réseau de Benes, car deux réseaux sont inter-connectés et collés entre eux, la distance entre les deux nœuds va par conséquent être doublée. Ainsi, si le diamètre comprend la connexion avec le processeur en entrée, celui-ci sera de : $2 \cdot \log(N) + 2$, si nous comptons uniquement les arêtes sans prendre en compte les processeurs, il sera donc de : $2 \cdot \log(N)$.

1.3 Quel est la largeur de coupe du réseau ?

La largeur de coupe de réseau correspond au nombre minimum d'arêtes à enlever pour diviser le graphe en deux composantes connexes. Ici, on double simplement un réseau *Butterfly* donc la largeur de coupe est elle aussi doublée : elle est donc de 16.

1.4 Quel est le degré du réseau ?

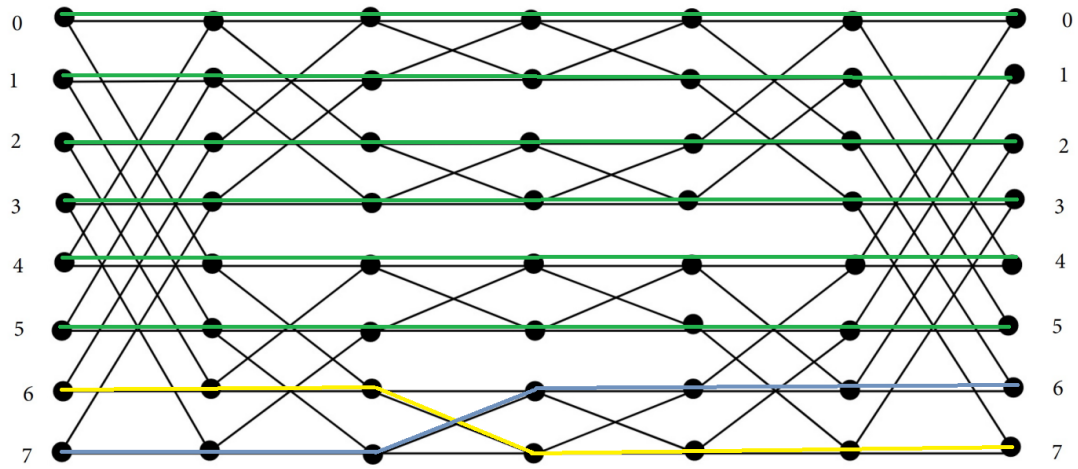
Le degré correspond au nombre d'arêtes adjacentes à un nœud. Dans le cas du réseau *Butterfly*, tous les commutateurs possèdent deux entrées ainsi que deux sorties. Cela ne change pas pour le réseau de Benes, par conséquent le degré reste le même, c'est-à-dire 4.

1.5 Est-ce que la longueur d'arêtes du réseau demeure constante lorsque l'on augmente le nombre de processeurs ?

Dans le cas du réseau *Butterfly*, la longueur d'arête n'est pas constante. Par conséquent, un réseau de Benes qui est essentiellement 2 réseaux *Butterfly* gardera cette propriété.

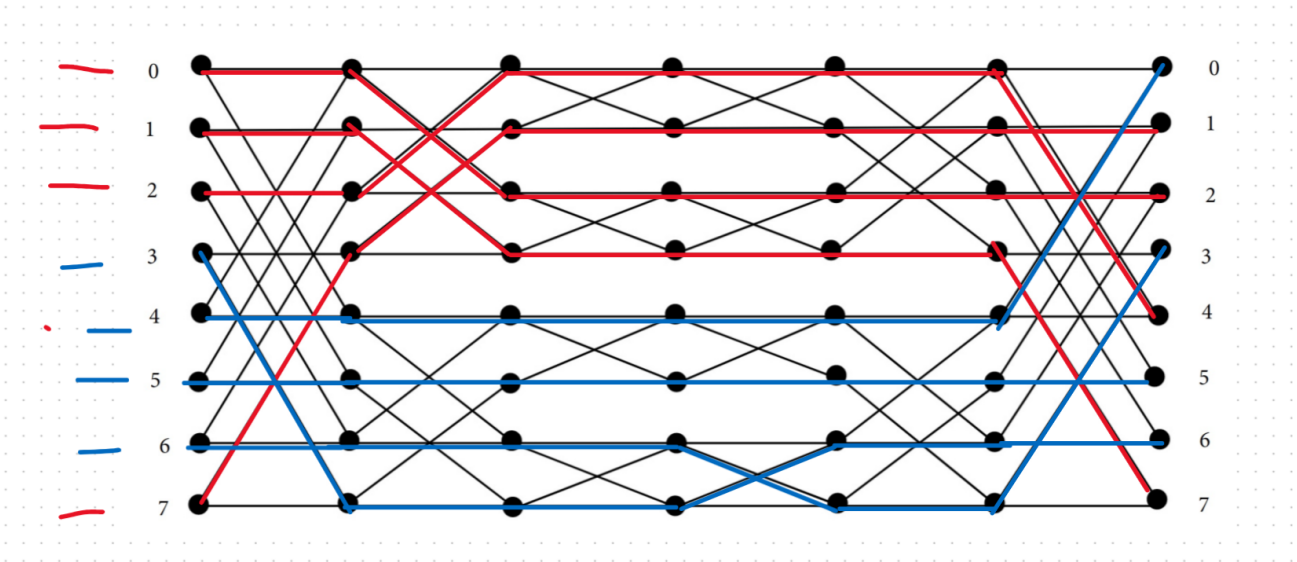
1.6 En utilisant le réseau de dimension 3 précédent, donnez l'état des commutateurs permettant la communication selon la permutation suivante :

$$a = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 7 & 6 \end{pmatrix}$$



1.7 Même question avec la permutation suivante :

$$b = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 7 & 4 & 6 & 0 & 5 & 3 & 1 \end{pmatrix}$$



Afin de trouver cette solution, il a été nécessaire de représenter les contraintes que chaque entrée possédait. En effet, nous pouvons remarquer que chaque entrée (ou sortie) possède une partie allant vers un sous-réseau inférieur ou

supérieur, cela veut donc dire que deux entrées possèdent un même commutateur s'il décide de se diriger vers le même sous-réseau. Nous avons donc différentes contraintes.

$$R(0) \longleftrightarrow R(4)$$
$$R(1) \longleftrightarrow R(5)$$
$$R(2) \longleftrightarrow R(6)$$
$$R(3) \longleftrightarrow R(7)$$

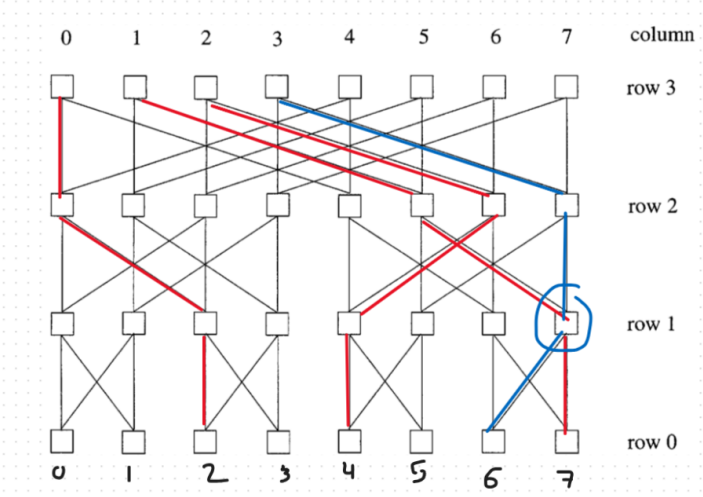
Il a été ensuite nécessaire de représenter les entrées en lien avec ces contraintes. Dans notre cas, l'entrée 0 ainsi que l'entrée 3 doivent communiquer avec 2 et 6, qui possèdent une relation de contraintes. Il sera donc nécessaire que ceux-ci utilisent 2 différents sous-réseaux. En trouvant les paires d'entrées communiquant avec nos paires de sorties sous contraintes, nous avons finalement un ensemble de contraintes dépendant des contraintes en sortie énoncées plus haut :

$$R(0) \longleftrightarrow R(3)$$
$$R(7) \longleftrightarrow R(5)$$
$$R(1) \longleftrightarrow R(6)$$
$$R(2) \longleftrightarrow R(4)$$

Cela peut donc se lire : "0 ne doit pas être dans le même sous-réseau que 3, etc.". Nous obtenons donc finalement notre séparation en 2 sous-réseaux. À partir de là, il est simple d'effectuer le routage afin d'avoir la solution demandée.

1.8 Démontrez par un exemple que les réseaux *Butterfly* ne sont pas non bloquants.

En essayant de résoudre le problème énoncé plus haut avec le réseau *Butterfly*, il semble évident que celui-ci sera bloquant. En effet, c'est uniquement grâce à ces deux sous-réseaux que nous avons réussi à résoudre ce problème. Or, ne possédant pas de sous-réseau supérieur et inférieur, il y aura donc une congestion au niveau d'un commutateur, car il n'y a qu'un seul chemin permettant de changer de sous-réseau.



Dans notre cas, la première communication bloquante est lorsque nous souhaitons envoyer le message de 3 vers 6. Soit celle-ci :

$$b = \begin{pmatrix} 0 & 1 & 2 & \mathbf{3} & 4 & 5 & 6 & 7 \\ 2 & 7 & 4 & \mathbf{6} & 0 & 5 & 3 & 1 \end{pmatrix}$$

2 Question 2 : A partir du théorème sur les récurrences, démontrez formellement le troisième cas de la version simplifiée.

Soient $a \geq 1$ et $b > 1$ deux constantes. De plus, considérons c et k deux constantes réelles positives. On suppose que $a < b^k$. On a $T : \mathbb{N} \to \mathbb{R}^+$ définie par :

$$T(n) = \begin{cases} \Theta(1) & \text{si} \quad n < b \\ aT(n/b) + cn^k & \text{sinon} \end{cases}$$

Nous allons utiliser le troisième cas du *Théorème sur les récurrences* pour démontrer le troisième cas de la version simplifiée du théorème.
Nous allons commencer par montrer qu'il existe : $\epsilon > 0$ tel que $cn^k = \Omega(n^{\log_b a + \epsilon})$,

On a $b^k > a$, ce qui implique que : $k > \log_b a$
Puis, étant donné que n est positif et $c > 0$, nous obtenons donc : $cn^k \geq n^{\log_b a}$
Comme les deux expressions sont positives, nous avons donc : $|cn^k| \geq |n^{\log_b a}|$
On pose maintenant, $\epsilon > 0$, tel que $|cn^k| = |n^{\log_b a + \epsilon}|$
Pour conclure, par définition, il existe bien un $\epsilon > 0$ tel que $cn^k = \Omega(n^{\log_b a + \epsilon})$

Pour finir, nous allons maintenant montrer qu'il existe : $c' < 1$ tel que $af(\frac{n}{b}) \leq c'f(n)$.

On a $a < b^k$, donc $\frac{a}{2} + \frac{a}{2} < \frac{a}{2} + \frac{b^k}{2}$
Et ainsi $a < \frac{a+b^k}{2}$
On pose maintenant $c' = \frac{a+b^k}{2b^k}$ et on a $a < c'b^k$
On a ainsi $ac\frac{n^k}{b^k} < c'b^k c\frac{n^k}{b^k}$
Et donc $af(\frac{n}{b}) < c'f(n)$
En conclusion il existe bien $c' < 1$ tel que $af(\frac{n}{b}) \leq c'f(n)$.

Ainsi d'après le troisième cas du *Théorème sur les récurrences*, nous avons bien $T(n) = \Theta(cn^k)$ et c étant une constante réelle positive, on a $T(n) = \Theta(n^k)$. Ce qui termine de démontrer le troisième cas de la version simplifiée du *Théorème sur les récurrences.*

3 Question 3 :

On a :

$$n^2 + \log(n)^2 = \Theta(n^2).$$

Puis, posons $f(n) = n^2$ et $g(n) = n^2 \log(n)^2$.
Calculons la limite :

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = \lim_{n \to +\infty} \frac{n^2}{n^2 \log(n)^2} = \lim_{n \to +\infty} \frac{1}{\log(n)^2} = 0.$$

D'où $n^2 = o(n^2 \log(n)^2)$.
De plus, $(n+1)^2 \log(n)^2 \sim n^2 \log(n)^2$ lorsque $n \to +\infty$.

Posons à présent $f(n) = n^2 \log(n)^2$ et $g(n) = n^{\log(n)}$.
Calculons la limite :

$$\lim_{n \to +\infty} \frac{f(n)}{g(n)} = \lim_{n \to +\infty} \frac{n^2 \log(n)^2}{n^{\log(n)}} = \lim_{n \to +\infty} \frac{\log(n)^2}{n^{\log(n)-2}} = 0.$$

En effet, $\forall n \geq 100$, $log(n) - 2 \geq 0$ donc $n^{\log(n)-2}$ croît plus rapidement que $\log(n)^2$.

Posons désormais $f(n) = n^{\log(n)}$ et $g(n) = 2^n$.

Calculons la limite :

$$\lim_{n\to+\infty}\frac{f(n)}{g(n)}=\lim_{n\to+\infty}\frac{n^{\log(n)}}{2^n}=\lim_{n\to+\infty}\frac{e^{\log(n)^2}}{e^{n\log(2)}}=\lim_{n\to+\infty}e^{\log(n)^2-n\log(2)}.$$

Démontrons que $\lim_{n\to+\infty}\log(n)^2-n\log(2)=-\infty$.
Pour cela, posons $x(n)=\log(n)^2$ et $y(n)=n\log(2)$ et calculons la limite :

$$\lim_{n\to+\infty}\frac{x(n)}{y(n)}=\lim_{n\to+\infty}\frac{\log(n)^2}{n\log(2)}.$$

Or le numérateur et le dénominateur de cette fraction tendent vers $+\infty$ lorsque $n\to+\infty$. La règle de l'Hôpital donne alors :

$$\lim_{n\to+\infty}\frac{x(n)}{y(n)}=\lim_{n\to+\infty}\frac{x'(n)}{y'(n)}=\lim_{n\to+\infty}\frac{\frac{2\log(n)}{n}}{log(2)}=\lim_{n\to+\infty}\frac{2\log(n)}{n\log(2)}.$$

À nouveau, le numérateur et le dénominateur de cette fraction tendent vers $+\infty$ lorsque $n\to+\infty$. En utilisant à nouveau la règle de l'Hôpital, il vient :

$$\lim_{n\to+\infty}\frac{x'(n)}{y'(n)}=\lim_{n\to+\infty}\frac{x''(n)}{y''(n)}=\lim_{n\to+\infty}\frac{\frac{2}{n}}{1}=\lim_{n\to+\infty}\frac{2}{n}=0.$$

Ainsi, on a $\lim_{n\to+\infty}\frac{\log(n)^2}{n\log(2)}=0$ et on en déduit $\lim_{n\to+\infty}\log(n)^2-n\log(2)=-\infty$.
D'où :

$$\lim_{n\to+\infty}\frac{f(n)}{g(n)}=\lim_{n\to+\infty}e^{\log(n)^2-n\log(2)}=0.$$

Ainsi $n^{\log(n)}=o(2^n)$.

Posons maintenant $f(n)=2^n$ et $g(n)=\log(n)^n$.
Calculons la limite :

$$\lim_{n\to+\infty}\frac{f(n)}{g(n)}=\lim_{n\to+\infty}\frac{2^n}{\log(n)^n}.$$

Afin de simplifier l'expression, appliquons le logarithme :

$$\lim_{n\to+\infty}\log\left(\frac{2^n}{\log(n)^n}\right)=\lim_{n\to+\infty}n\log(2)-n\log(\log(n))=-\infty.$$

En effet, on voit facilement :

$$\lim_{n\to+\infty}\frac{n\log(2)}{n\log(\log(n)}=\lim_{n\to+\infty}\frac{\log(2)}{\log(\log(n)}=0.$$

Puisque $\lim_{x\to-\infty}e^x=0$, on obtient :

$$\lim_{n\to+\infty}\frac{f(n)}{g(n)}=\lim_{n\to+\infty}\frac{2^n}{\log(n)^n}=0.$$

D'où $2^n=o(\log(n)^n)$.

Posons enfin $f(n)=\log(n)^n$ et $g(n)=n!$.
Calculons la limite :

$$\lim_{n\to+\infty}\frac{f(n)}{g(n)}=\lim_{n\to+\infty}\frac{\log(n)^n}{n!}.$$

D'après la formule de Stirling, on a :

$$n!\sim_{n\to+\infty}\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

On a alors :

$$\lim_{n\to+\infty} \frac{\log(n)^n}{n!} \sim \lim_{n\to+\infty} \frac{\log(n)^n}{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n} = \frac{1}{\sqrt{2\pi}} \lim_{n\to+\infty} \frac{(e\log(n))^n}{\sqrt{n}n^n}.$$

Puisque e est une constante et $\sqrt{n} = n^{\frac{1}{2}}$ est négligeable devant n^n pour de grandes valeurs de n, on peut se limiter à étudier la limite suivante :

$$\lim_{n\to+\infty} \left(\frac{\log(n)}{n}\right)^n = \left(\lim_{n\to+\infty} \frac{\log(n)}{n}\right)^n.$$

On a démontré dans un cas précédent en appliquant la règle de l'Hôpital :

$$\lim_{n\to+\infty} \frac{\log(n)}{n} = 0.$$

D'où :

$$\left(\lim_{n\to+\infty} \frac{\log(n)}{n}\right)^n = 0.$$

Ainsi :

$$\lim_{n\to+\infty} \frac{\log(n)^n}{n!} = 0.$$

Il vient alors $\log(n)^n = o(n!)$.

Donc finalement :

$$n^2 + \log(n)^2 = \Theta(n^2) \quad o \quad n^2\log(n)^2 = (n+1)^2\log(n)^2 \quad o \quad n^{\log(n)} \quad o \quad 2^n \quad o \quad \log(n)^n \quad o \quad n!$$

4 Question 4 :

4.1 $T(n) = 2T\left(\frac{n}{2}\right) + 1$

On a $a = 2$, $b = 2$ et $f(n) = 1 = cn^k$ avec $c = 1$ et $k = 0$.
Comme $b^k = 2^0 = 1$, $a > b^k$ et $T(n) = \Theta(n^{\log_b(a)}) = \Theta(n^{\log_2(2)}) = \Theta(n)$.

4.2 $T(n) = T\left(\frac{9n}{10}\right) + n$

On a $a = 1$, $b = \frac{10}{9}$ et $f(n) = n = cn^k$ avec $c = 1$ et $k = 1$.
Comme $b^k = \frac{10}{9}^1 = \frac{10}{9}$, $a < b^k$ et $T(n) = \Theta(n^k) = \Theta(n^1) = \Theta(n)$.

4.3 $T(n) = 16T\left(\frac{n}{4}\right) + n^2$

On a $a = 16$, $b = 4$ et $f(n) = n^2 = cn^k$ avec $c = 1$ et $k = 2$.
Comme $b^k = 4^2 = 16$, $a = b^k$ et $T(n) = \Theta(n^k\log(n)) = \Theta(n^2\log(n))$.

4.4 $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

On a $a = 7$, $b = 3$ et $f(n) = n^2 = cn^k$ avec $c = 1$ et $k = 2$.
Comme $b^k = 3^2 = 9$, $a < b^k$ et $T(n) = \Theta(n^k) = \Theta(n^2)$.

4.5 $T(n) = 7T\left(\frac{n}{2}\right) + n^2$

On a $a = 7$, $b = 2$ et $f(n) = n^2 = cn^k$ avec $c = 1$ et $k = 2$.
Comme $b^k = 2^2 = 4$, $a > b^k$ et $T(n) = \Theta(n^{\log_b(a)}) = \Theta(n^{\log_2(7)}) \approx \Theta(n^{2.8})$.

4.6 $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

On a $a = 2$, $b = 4$ et $f(n) = \sqrt{n} = cn^k$ avec $c = 1$ et $k = \frac{1}{2}$.
Comme $b^k = \sqrt{4} = 2$, $a = b^k$ et $T(n) = \Theta(n^k \log(n)) = \Theta(\sqrt{n}\log(n))$.

4.7 $T(n) = 2T\left(\sqrt{n}\right) + \log(n)$

On pose $n = 2^m$, et on a alors :

$$T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m.$$

En définissant $S(m) = T(2^m)$, on obtient ainsi :

$$S(m) = 2S\left(\frac{m}{2}\right) + m.$$

On peut ainsi appliquer le théorème sur les récurrences, et on a ici $a = 2$, $b = 2$ et $f(m) = m = cm^k$ avec $c = 1$ et $k = 1$. Comme $b^k = 2^1 = 2$, $a = b^k$ et $S(m) = \Theta(m^k \log(m)) = \Theta(m\log(m))$. Puis, comme $m = \log(n))$, on a finalement :

$$T(n) = \Theta(\log(n)\log(\log(n))).$$

4.8 $T(n) = T\left(\frac{n}{2} + 5\right) + 1$

Pour cette partie, il est nécessaire de "dérouler" la récurrence. Nous avons donc subséquemment :

$$\begin{aligned} T(n) &= T\left(\frac{n}{2} + 5\right) + 1 \\ &= T\left(\frac{n+10}{2}\right) + 1 \end{aligned}$$

$$\begin{aligned} T\left(\frac{n+10}{2}\right) &= T\left(\frac{n+10}{4} + 5\right) + 1 \\ &= T\left(\frac{n+10+20}{4}\right) + 1 \end{aligned}$$

$$\begin{aligned} T\left(\frac{n+10+20}{4}\right) &= T\left(\frac{n+10+20}{8} + 5\right) + 1 \\ &= T\left(\frac{n+10+20+40}{8}\right) + 1 \end{aligned}$$

$$\begin{aligned} T\left(\frac{n+10+20+40}{8}\right) &= T\left(\frac{n+10+20+40}{16} + 5\right) + 1 \\ &= T\left(\frac{n+10+20+40+80}{16}\right) + 1 \\ &= T\left(\frac{n}{2^4} + \frac{5}{2^3} + \frac{5}{2^2} + \frac{5}{2^1} + \frac{5}{2^0}\right) + 1 \end{aligned}$$

Ainsi, pour le cas k, il vient :

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2^k} + 5\sum_{i=0}^{k-1}\frac{1}{2^i}\right) + \sum_{i=0}^{k-1} 1 \\
&= T\left(\frac{n}{2^k} + 5\frac{1-(\frac{1}{2})^k}{1-\frac{1}{2}}\right) + k \\
&= T\left(\frac{n}{2^k} + 5\cdot 2\left(1-\frac{1}{2^k}\right)\right) + k \\
&= T\left(\frac{n}{2^k} + 5\cdot 2\cdot\frac{1}{2^k}(2^k-1)\right) + k \\
&= T\left(\frac{n+10(2^k-1)}{2^k}\right) + k
\end{aligned}
$$

La condition d'arrêt a lieu lorsque $\frac{n+10(2^k-1)}{2^k}$ devient suffisamment petit, i.e. lorsqu'il atteint une constante n_0 telle que $n_0 = 1$.

Posons alors $\frac{n+10(2^k-1)}{2^k} = 1$ et déterminons k en fonction de n :

$$
\begin{aligned}
\frac{n+10(2^k-1)}{2^k} &= 1 \\
n+10(2^k-1) &= 2^k \\
n &= -9\cdot 2^k + 10
\end{aligned}
$$

Puis ensuite :

$$
\begin{aligned}
9\cdot 2^k &= 10-n \\
2^k &= \frac{10-n}{p} \\
k &= \log\left(\frac{10-n}{p}\right)
\end{aligned}
$$

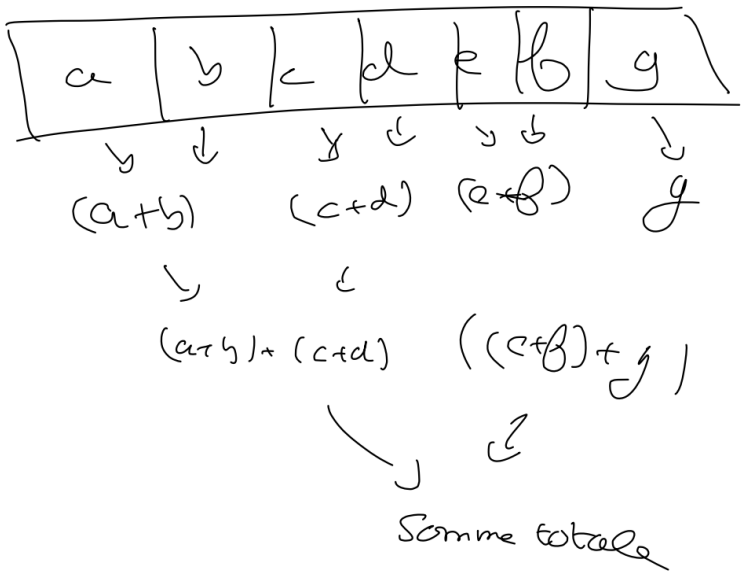
Il faut donc nécessairement que $10-n \geq 0$ pour que l'expression soit correcte, soit $n \leq 10$.

On obtient donc finalement :

$$T(n) = \Theta(\log(n))$$

5 Donnez un algorithme parallèle de durée $\Theta(\log(n))$ pour résoudre ce problème. Analysez aussi le travail de votre algorithme.

Nous pouvons remarquer que ce problème se rapproche du calcul en parallèle d'un tableau, vu en cours, afin d'obtenir la somme de celui-ci.



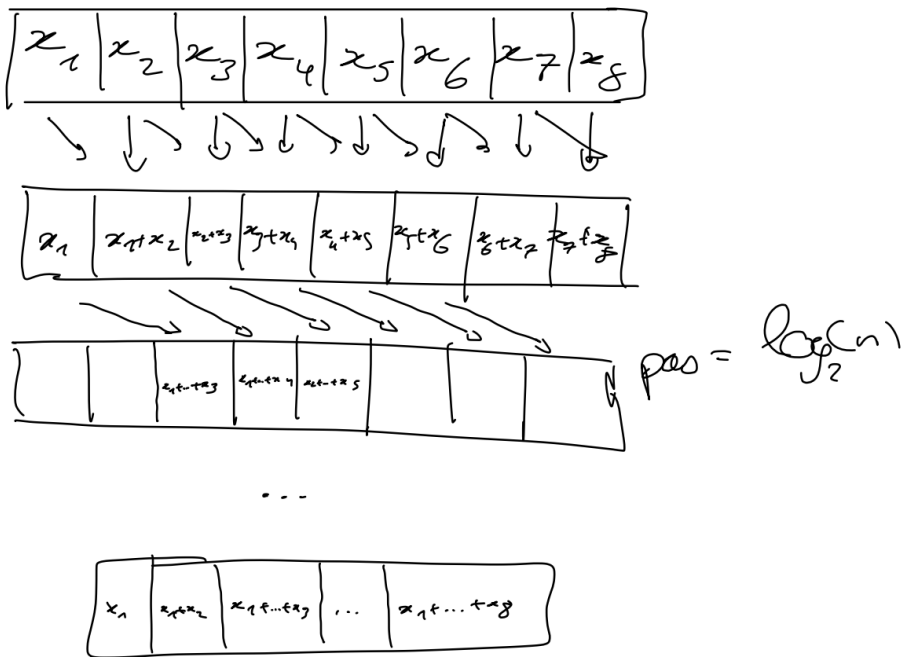
5.1 Algorithme

En utilisant la méthode du cours, nous pouvons utiliser l'algorithme permettant de faire la somme 2 à 2, il est cependant nécessaire de l'adapter afin de garder la somme de chaque élément dans leurs cases respectives. Rappelons l'algorithme vu en cours :

Algorithm 1: Algorithme parallèle de somme

```
Data: Tableau T de taille n
Result: Somme du tableau T
for t ← 1 ; t ≤ n ; t++ ; en parallèle do
    for k ← 2 ; k ≤ n ; k = δk do
        if t mod k == 0 then
            S[t] = S[t] + S[t − k/2]
            sync
        end
    end
end
```

En adaptant cette algorithme afin de sauvegarder la valeur de la case i, nous avons donc :



Algorithm 2: Algorithme parallèle du somme des i premiers éléments

```
Data: Tableau T de taille n
Result: Tableau S étant la somme des 1 ≤ i ≤ n
for i = 1 to log(n) do
    for j = 0 to n ; en parallèle do
        if j ≤ 2^d then
            x[k] += x[k − 2^(d−1)]
        end
    end
end
```

5.2 Analyse de l'algorithme

Dans cet algorithme :

— La première boucle for effectue $\Theta(\log(n))$ itérations sur le tableau T.

— La seconde boucle effectue n opérations en parallèles. Dans cette boucle, une comparaison avec un entier est faite, et celle-ci est de temps constante. De plus, une somme est également effectuée, qui est en temps constant, soit $\Theta(1)$ également.

Ainsi, la complexité est de $\Theta(\log(n) * n)$. Cependant, dans le meilleur cas, s'il y a n processeurs, il semble logique de supposer qu'en les faisant tous en parallèle, il serait possible d'avoir une complexité de $\Theta(\log(n))$. Nous pouvons donc appliquer le calcul du travail afin de vérifier notre hypothèse, car nous avons notre meilleur cas ainsi que notre cas général.

Ainsi :

$$T_1 = n * \log(n)$$
$$T_\infty = \log(n)$$

Nous pouvons donc faire l'analyse de cette solution :

$$\frac{T_1(n)}{T_\infty} = \frac{n * \log(n)}{\log(n)} = n$$

En effet, dans le meilleur cas, où on applique n threads, les opérations se feront donc en parallèle, et ainsi le $\Theta(n * \log(n))$ pourra se transformer en $\Theta(\log(n))$.