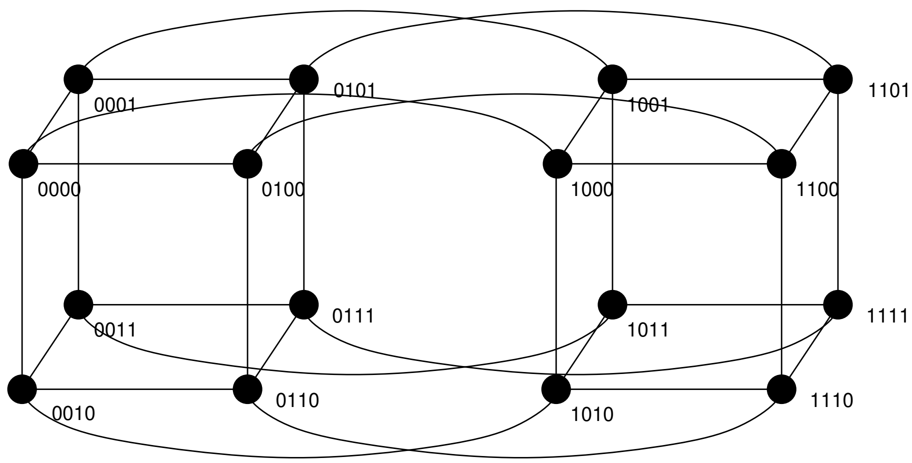
Algorithmique parallèle
TD – Enoncé

Plan

1 Etude de cas : l'hypercube

Définition et numérotation des sommets

– Les hypercubes se construisent par duplication
– Etant donnés deux m-cubes identiques (i.e. avec 2^m sommets), on construit un $m+1$-cube en reliant deux à deux les sommets correspondants



– Autre définition (équivalente) : un m-cube est composé de 2^m sommets numérotés de 0 à 2^m-1, et il existe un lien entre deux sommets $\Longleftrightarrow$ leur écriture binaire ne diffère que par un seul bit.
– Construction récursive $\rightarrow$ étiquetage des sommets :
étant donnés deux m-cubes, on renumérote les sommets du premier en $0a_i$ et ceux du second en $1a_i$, où a_i est l'écriture binaire représentant les deux sommets correspondants du m-cube.
– Un m-cube est de diamètre $D = m$ et de degré $\Delta = m$

Chemins dans l'hypercube

- A et B deux sommets du m-cube
- $H(A, B)$ distance de Hamming entre A et B : nombre de bits qui diffèrent dans l'écriture binaire de A et B.
- *Router* un message de A à B = trouver un chemin dans le m-cube qui mène de A à B, de longueur minimale.
- Questions :
  1. Montrer que H est bien une distance
  2. Construire un chemin de A à B
  3. Combien y-a-t-il de chemins indépendants de longueur minimum entre A et B ? Deux chemins sont dits indépendants s'ils n'ont aucun sommet en commun à l'exception de A et de B.

Routage

1. Comment implanter le routage d'un message de A à B
2. Peut-on trouver un algorithme non statique, i.e. capable de choisir entre plusieurs routes ?

Plongements d'anneaux et de grilles

- G_m code de Gray de dimension $m \geq 2$ défini récursivement comme la concaténation
$$G_m = 0G_{m-1}|1G_{m-1}^{rev}$$
  - xG énumère les éléments de G en rajoutant un x en tête de leur écriture binaire.
  - G^{rev} énumère les éléments de G dans l'ordre renversé (en partant de la fin).
  - On initie la construction avec $G_1 = \{0, 1\}$
- Questions
  1. Construire G_2, G_3 et G_4
  2. Définir un anneau de 2^m processeurs dans le m-cube
  3. Définir une grille torique de taille $2^r \times 2^s$ dans un m-cube, avec $r + s = m$.

Diffusion dans l'hypercube

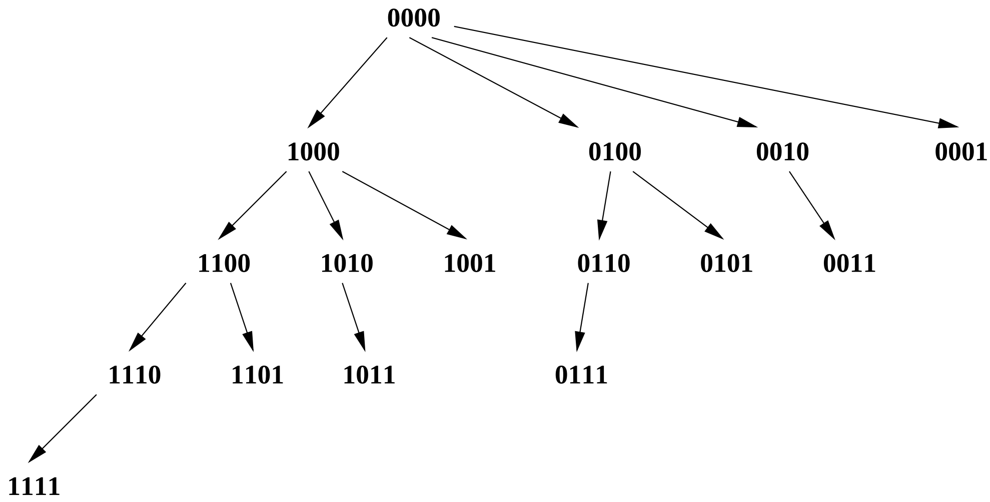
(A) Diffusion simple

– Primitives de communication

send(cube-link, send-adr, L)

receive(cube-link, recv-adr, L)

– Diffusion selon l'arbre couvrant ($m = 4$) :



– Ecrire la procédure de diffusion à partir du processeur k :

broadcast(k, adr, L)

– Quel est le coût en modèle *store and forward*?

(B) Diffusion m-ports

– Construction de m arbres couvrants à arêtes indépendantes :

1. Prendre l'arbre couvrant précédent, enraciné en 0, noté A_0
2. Effectuer une rotation (à gauche) des écritures binaires des sommets de A_0 pour construire les autres arbres : A_i est l'image de A_0 par l'opérateur de rotation $R^{(i)}$.
   Si $s = s_{m-1}s_{m-2}\ldots s_1 s_0$, on définit $R(s) = s_{m-2}s_{m-3}\ldots s_0 s_{m-1}$.
   On pose $R^{(1)} = R$ et $R^{(i)} = R \circ R^{(i-1)}$ pour $2 \leq i \leq m-1$
3. On inverse le i-ème bit (modulo m) des sommets de A_i
4. Après cette opération XOR, le noeud 0 est une feuille dans chacun des m arbres A_i. On fusionne ceux-ci en un seul arbre de racine 0, et de hauteur m

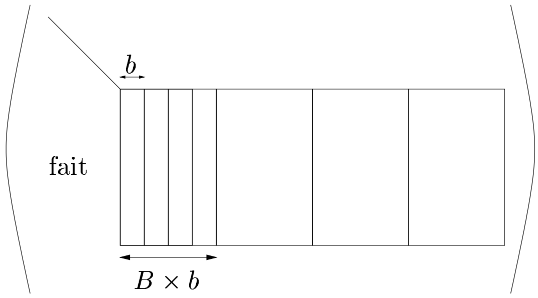
– Faire la construction pour $m = 3$

– Quel est le coût en modèle *store and forward*?

2 Algorithmique hétérogène

LU hétérogène

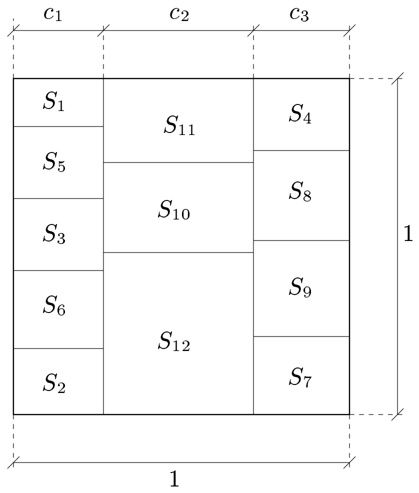
- A chaque étape, le processeur qui possède le bloc pivot le factorise et le diffuse
- Les autres processeurs mettent à jour les colonnes restantes
- A l'étape suivante le bloc des b colonnes suivantes devient le pivot



Question : Trouver une allocation statique de période B qui équilibre au mieux les tâches lors de chaque étape de mise à jour.

Partitionnement en colonnes

– n réels positifs $s_1, s_2, \ldots, s_n$ de somme 1
– Partitionner le carré unité en n rectangles de surfaces $s_1, s_2, \ldots, s_n$
– On cherche un partitionnement en colonnes



– Fonction objective à minimiser : somme des traits dessinés, i.e. (à une constante près) la somme des demi-périmètres des n rectangles.

Questions

1. Résoudre si $n = 4$ et $s_1 = s_2 = s_3 = s_4 = 0.25$.
2. Résoudre si $n = 6$, $s_1 = 0.5$ et $s_2 = s_3 = s_4 = s_5 = s_6 = 0.1$
3. Dans le cas général, on ne connaît pas le nombre de colonnes qu'on va utiliser. Mais :
   (a) Montrer qu'on peut se ramener à placer les s_i dans l'ordre croissant
   (b) Résoudre le problème par programmation dynamique