Devoir maison numéro 1
A rendre pour le TD du 28 ou 29 mars.

Exercice 1 Recherche de collisions

On considère une fonction de hachage qui, étant donné un fichier, produit une empreinte sous la forme d'une chaîne de 128 bits. On souhaite trouver une collisions (c'est-à-dire deux fichiers différents ayant même empreinte) en hachant des fichiers générés aléatoirement. Combien de fichiers faut-il générer pour que la probabilité d'observer une collision soit $\geqslant 1/2$?

Exercice 2 Randomized Quicksort

On étudie plus en détail l'algorithme Randomized Quicksort vu en cours pour montrer que le temps d'exécution pour trier un ensemble X de cardinal n est $O(n \log n)$ avec grande probabilité.

Quand un pivot est choisi aléatoirement au sein d'un ensemble $Y \subset X$, on dit que le choix est *chanceux* si le pivot p partage $Y \setminus \{p\}$ en deux sous-ensembles de cardinal $\leqslant \frac{2}{3}|Y|$, et *malchanceux* sinon.

1. Montrer que la probabilité qu'un choix de pivot soit chanceux est $\geqslant 2/3$.
2. Dans une suite de N choix de pivots indépendants, soit X_N le nombre de choix chanceux. Montrer avec une inégalité de Chernoff que
$$\mathbf{P}(X_N \leqslant N/2) \leqslant \exp(-cN)$$
pour une constante $c > 0$ à déterminer.
3. Soit $x \in X$. Montrer qu'au cours de toute exécution de l'algorithme Quicksort (aléatoire ou déterministe), parmi les choix de pivots effectués dans un ensemble contenant x, au plus $C \log(n)$ sont chanceux (pour une constante C à déterminer).
4. Conclure que le temps d'exécution de l'algorithme Randomized Quicksort est $O(n \log n)$ avec grande probabilité.

Exercice 3 Routage de paquets par un algorithme probabiliste

On considère le graphe orienté de l'hypercube défini comme suit : l'ensemble des sommets est $V = \{0,1\}^n$, et l'ensemble E des arêtes est défini par

$$(x,y) \in E \iff \text{les sommets } x \text{ et } y \text{ différent en une unique cordonnée.}$$

Ce graphe possède donc 2^n sommets et $n2^n$ arêtes orientées.

Soit $\sigma \in \mathfrak{S}_V$ une permutation de V. On étudie le problème de routage suivant : à l'instant initial, un paquet P_x se trouve en chaque sommet $x \in V$, et doit être acheminé jusqu'à la destination $\sigma(x)$. Chaque arête orientée a une capacité de transmettre un paquet par unité de temps.

Le problème est de déterminer pour chaque paquet P_x un itinéraire, c'est-à-dire un chemin allant de x à $\sigma(x)$ dans le graphe. Une fois tous les itinéraires déterminés, l'algorithme s'exécute en répétant l'opération suivante :

— Tous les paquets demandent à se déplacer vers la prochaine étape de leur itinéraire.
— Pour toute arête orientée (x,y), l'algorithme sélectionne arbitrairement un paquet parmi ceux qui demandent à se déplacer de x vers y, et le déplace.

Il n'y a pas de limite sur le nombre de paquets qui se trouvent à un sommet donné. L'algorithme se termine lorsque tous les paquets ont atteint leur destination.

On admet le lemme suivant (question bonus : le démontrer) : le retard subi par un paquet P_x est majoré par le nombre de paquets P_y ($y \neq x$) tels que les itinéraires I_x et I_y ont une arête en commun.

L'*itinéraire naïf* pour aller de $x = (x_1, \ldots, x_n)$ vers $y = (y_1, \ldots, y_n)$ est donné en ajustant les bits un à un dans l'ordre croissant des coordonnées. Par exemple le chemin de $(0, 0, 1, 0, 0, 1)$ vers $(1, 0, 0, 1, 1, 1)$ est

$$(0, 0, 1, 0, 0, 1) \to (1, 0, 1, 0, 0, 1) \to (1, 0, 0, 0, 0, 1) \to (1, 0, 0, 1, 0, 1) \to (1, 0, 0, 1, 1, 1)$$

1. Expliciter une permutation σ pour laquelle l'algorithme s'exécute en temps $\Omega(2^{n/2})$ si tous les paquets suivent l'itinéraire naïf.

Dans les questions 2. à 4., on étudie un itinéraire randomisé défini comme suit. On choisit pour tout $x \in V$ un sommet $\tau(x)$ au hasard selon la loi uniforme sur V, indépendamment. L'itinéraire I_x suivi par le paquet P_x est alors la concaténation de l'itinéraire naïf allant de x à $\tau(x)$ puis de l'itinéraire naïf allant de $\tau(x)$ à $\sigma(x)$.

2. Soit e une arête orientée et soit H_e l'ensemble des $y \in V \setminus \{x\}$ tels que l'itinéraire naïf de y vers $\tau(y)$ traverse l'arête e. En utilisant le principe de linéarité de l'espérance, montrer que $\mathbf{E}[|H_e|] \leqslant 1$.
3. Soit $x \in V$ et soit K_x l'ensemble des $y \in V \setminus \{x\}$ tels que l'itinéraire naïf de y vers $\tau(y)$ a une arête en commun avec l'itinéraire naïf de x vers $\tau(x)$. En utilisant une inégalité de Chernoff, montrer que pour une constante C à déterminer

$$\mathbf{P}\left(|K_x| \geqslant Cn\right) \leqslant 2^{-2n}.$$

4. Montrer que lorsqu'on utilise l'itinéraire randomisé, le temps d'exécution de l'algorithme est $O(n)$ avec grande probabilité.
5. On étudie maintenant une variante de l'itinéraire naïf où l'itinéraire de chaque paquet est obtenu en ajustant un à un les bits dans un ordre aléatoire des coordonnées. Les choix pour les différents paquets sont faits indépendemment et uniformément dans $\mathfrak{S}_n$. Montrer qu'il existe une permutation $\sigma \in \mathfrak{S}_V$ telle que cet algorithme s'exécute en temps $\Omega(2^{cn})$ avec grande probabilité, pour une constante $c > 0$.