IFT 2125 Automne 2017 **Pierre McKenzie**
Introduction à l'algorithmique

Devoir 4 (complet)

Remise : le mercredi 6 décembre (au *début* de la démo)

1. Faites le problème 8.32 de B&B (le canoteur économe).

   Do the problem 8.32 in the Brassard and Bratley book.

2. Une application $\rho : A \times A \to A$ est fixée une fois pour toutes, où A est l'ensemble $\{a, b, c, \ldots, y, z\}$ des 26 lettres de l'alphabet. Donnez un algorithme utilisant la technique de la programmation dynamique et résolvant le problème suivant :

   <u>ÉVALUATION</u>
   DONNÉE: $\sigma_1, \ldots, \sigma_n \in A$
   DÉCIDER: s'il existe un parenthésage complet de $\sigma_1 {*} \sigma_2 {*} \cdots {*} \sigma_n$ qui permet, en remplaçant à répétition $(\alpha {*} \beta)$ où $\alpha \in A$ et $\beta \in A$ par $\rho(\alpha, \beta)$ dans un ordre prescrit par le parenthésage, d'obtenir tout simplement à la fin la lettre a.

   Indice. Pensez à un tableau dont chaque entrée est un ensemble de lettres de A.

   A total function $\rho : A \times A \to A$ is specified once and for all, where $A = \{a, b, c, \ldots, y, z\}$. Give a dynamic programming algorithm that solves the following problem :

   <u>ÉVALUATION</u>
   DONNÉE: $\sigma_1, \ldots, \sigma_n \in A$
   DÉCIDER: if there exists a complete bracketing of $\sigma_1 {*} \sigma_2 {*} \cdots {*} \sigma_n$ that leaves a in the end when $(\alpha {*} \beta)$ for $\alpha, \beta \in A$ is systematically replaced with $\rho(\alpha, \beta)$ according to the bracketing.

   Hint. Think of a table with subsets of A as entries.

3. Donnez en Python (et imprimez papier et placez sur Studium) un algorithme de retour arrière qui résout le problème suivant :

   <u>MÊMEGRAPHE</u>
   DONNÉE: matrices d'adjacence symétriques $A, B \in \{0, 1\}^{m \times m}$
   DÉCIDER: si A et B représentent le même graphe non orienté.

   Indice. A et B représentent le même graphe si et seulement si une permutation σ de l'ensemble $\{1, 2, \ldots, m\}$ existe telle que pour tout $i, j \in \{1, 2, \ldots, m\}$, $A(i, j) = B(\sigma(i), \sigma(j))$.

   Give (and print and upload on Studium) a Python backtracking algorithm to solve :

   <u>SAMEGRAPH</u>
   DONNÉE: symmetric adjacency matrices $A, B \in \{0, 1\}^{m \times m}$
   DÉCIDER: if A and B represent the same undirected graph.

   Hint. A and B represent the same graph iff a permutation σ of the set $\{1, 2, \ldots, m\}$ exists such that for all $i, j \in \{1, 2, \ldots, m\}$, $A(i, j) = B(\sigma(i), \sigma(j))$.

4. Deux ensembles $A, B \subseteq \mathbb{N}$ de taille k vous sont donnés. On vous promet que, ou bien, $A \cap B = \emptyset$, ou bien, $A \cap B$ est le singleton $\{s\}$ et $s = \max A = \max B$. Voici un algorithme de Monte Carlo qui cherche à déterminer si $\max A = \max B$:

   tirer au hasard $a_1 \in A, a_2 \in A, b_1 \in B, b_2 \in B$
   `si` $\max\{a_1, a_2\} = \max\{b_1, b_2\}$ `alors retourner` VRAI
   `sinon retourner` FAUX

   (a) Cet algorithme est-il faux-biaisé ? vrai-biaisé ? ni l'un ni l'autre ?

   (b) Calculez en fonction de k la probabilité d'erreur de cet algorithme.

(c) En répétant l'algorithme, est-il possible de réduire la probabilité d'erreur à moins de 1% lorsque $k = 2$? $k = 10$? $k = 100$? (Si oui, combien faut-il de répétitions dans chacun des cas ?)

Two sets $A, B \subseteq \mathbb{N}$ of size k are given. You are promised that either $A \cap B = \emptyset$, or $A \cap B$ is the singleton $\{s\}$ and $s = \max A = \max B$. Here is a Monte Carlo algorithm that tries to determine whether $\max A = \max B$ *:*

draw $a_1 \in A, a_2 \in A, b_1 \in B, b_2 \in B$ at random
`if` $\max\{a_1, a_2\} = \max\{b_1, b_2\}$ `then return` TRUE
`else return` FALSE

(a) Is this algorithm false-biased ? true-biaised ? neither ?

(b) Compute as a function of k the error probability of this algorithm.

(c) By repeating the algorithm, is it possible to reduce the error probability to less than 1% when $k = 2$? $k = 10$? $k = 100$? (If so, give in each case the number of repetitions needed.)