IFT 2125 Hiver 2018 **Pierre McKenzie**
Introduction à l'algorithmique

Devoir 2

Remise : le vendredi 16 février (*English translation on page 2*)

1. Soit une fonction $T(n)$ décrite come ceci :

$$T(n) = \begin{cases} 2 & \text{si } 0 \leq n \leq 1, \\ 3T(\lceil n/3 \rceil) + 2T(\lfloor n/3 \rfloor) + 16 \log_3 n & \text{si } n > 1. \end{cases}$$

   (a) Résolvez *exactement* la récurrence lorsque n est puissance de 3, en utilisant la méthode de l'équation caractéristique. (Commencez par poser $n = 3^k$.)

   (b) Ici vous pouvez faire appel aux résultats vus en cours (lemme des puissances de b, théorème qui a suivi) ou en démo (expéditif pour déduire l'éventuelle non décroissance) lorsque cela vous est utile.

      i. Expliquez comment déduire de (a) l'ordre exact de $T(n)$, i.e., $\Theta(T(n))$, exprimé simplement.

      ii. Était-il nécessaire de faire (a) pour obtenir $\Theta(T(n))$? (Expliquez.)

2. Problème 6.6 de Brassard et Bratley ("money changing with $1, p, p^2, \cdots$ denominations").

3. Vous organisez une compétition culinaire. Chaque chef propose un plat. Au moment de poser sa candidature, le/la chef $i, 1 \leq i \leq n$, doit vous informer des temps p_i de préparation, c_i de cuisson et d_i de décoration finale requis à la fabrication de son plat.

   La compétition disposera de deux salles. La salle A servira à la préparation des plats. Aucun chef ne voulant risquer de divulguer sa recette, chaque chef exige d'accéder seul à la salle A pour la durée p_i de la partie préparation de son plat.

   La salle B sera équipée de n fours et de n comptoirs de travail. Les chefs acceptent de partager sans restriction la salle B pour les opérations de cuisson et de décoration.

   Vous devez confectionner l'horaire de la compétition, c'est à dire, annoncer les temps (débutant à zéro) auxquels les n chefs accéderont à la salle A. (Un chef ayant terminé en A se dirige immédiatement en B.)

   Donnez un algorithme efficace produisant un horaire qui minimise le temps de fin de la fabrication complète de tous les plats par tous les chefs. Ensuite, en quelques lignes, donnez l'intuition qui supporte votre algorithme (une preuve formelle n'est pas demandée).

4. Considérez le problème suivant, où $[m] = \{1, 2, \ldots, m\}$:

   <u>INTERSECTION</u>

   DONNÉE: Ensembles $S_1, \cdots, S_n \subseteq [m]$.

   CALCULER: Ensemble S de cardinalité minimum tel que pour tout i, $S \cap S_i \neq \emptyset$.

   (a) Est-ce que $([m], \{S \subseteq [m] : \forall i, S \cap S_i = \emptyset\})$ forme un matroïde ? (Justifiez.)

   (b) Est-ce que $([m], \{S \subseteq [m] : \forall i, S \cap S_i \neq \emptyset\})$ forme un matroïde ? (Justifiez.)

   (c) Implantez en Python une heuristique vorace pour ce problème, en supposant que les S_i vont sont fournis sous forme d'une matrice booléenne $n \times m$ dont la ligne i est la fonction caractéristique de S_i. (Vous pouvez faire appel à une méthode pour faire le tri sans le programmer vous-même.)

   (d) Fixons $n = 20$ et $m = 100$. À l'aide du module random (voir démo 5), générez 50 exemplaires de INTERSECTION, soumettez ces exemplaires à votre méthode et imprimez la moyenne et l'écart-type (avec numpy.mean et numpy.std, démo 5) des tailles des ensembles S obtenus. (Déposez votre Python sur Studium en vous servant du fichier fourni.)

1. T is defined as follows :

$$T(n) = \begin{cases} 2 & \text{if } 0 \leq n \leq 1, \\ 3T(\lceil n/3 \rceil) + 2T(\lfloor n/3 \rfloor) + 16 \log_3 n & \text{if } n > 1. \end{cases}$$

   (a) Solve the recurrence *exactly* when n is a power of 3, using the characteristic equation method seen in class. (Start be setting $n = 3^k$.)

   (b) Here you may use the results seen in class, (lemma on powers of b, theorem that followed) or in the démo (e.g. to deduce the eventually non decreasing property) if/when this is useful.

      i. Explain how to deduce from (a) the exact order of $T(n)$, i.e., $\Theta(T(n))$, expressed simply.

      ii. Did you have to answer (a) in order to obtain $\Theta(T(n))$? (Explain.)

2. Problem 6.6 from Brassard et Bratley ("money changing with $1, p, p^2, \cdots$ denominations").

3. You organize a culinary competition. Each chef proposes a dish and registers ahead of time. When registering, the ith chef, $1 \leq i \leq n$, declares the respective amounts of time p_i, c_i and d_i that he/she requires for the purposes of preparing, cooking and decorating his/her dish.

   The competition has access to two rooms. Room A is the preparation room. No chef wants to risk divulging his recipe, so every chef requires private access to room A for the duration p_i of the preparation part of his/her dish.

   Room B is equipped with n stoves and n working areas. The chefs don't mind sharing room B for their cooking and decorating activities.

   Your task is to set up the schedule, which means, to announce the times (starting at time zero) at which the n chefs will enter room A in turn. (A chef finished with room A immediately moves on to room B.)

   Describe an efficient algorithm producing a schedule that minimizes the time at which all the chefs are completely done with their dish. Then in a few lines, give the intuition behind the correctness of your algorithm (a formal proof is not requested).

4. Consider the following problem, where $[m] = \{1, 2, \ldots, m\}$:

   <u>INTERSECTION</u>
   DONNÉE: Sets $S_1, \cdots, S_n \subseteq [m]$.
   CALCULER: Set S of minimum cardinality such that for every i, $S \cap S_i \neq \emptyset$.

   (a) Is $([m], \{S \subseteq [m] : \forall i, S \cap S_i = \emptyset\})$ a matroid ? (Justify.)

   (b) Is $([m], \{S \subseteq [m] : \forall i, S \cap S_i \neq \emptyset\})$ a matroid ? (Justify.)

   (c) Implement in Python a greedy heuristic for this problem, assuming that the S_i are given to you in the form of an $n \times m$ boolean matrix whose ith line is the characteristic function of S_i. (You may use a sorting method without programming it yourself.)

   (d) Fix $n = 20$ and $m = 100$. Using the random module (see démo 5), generate 50 instances of INTERSECTION, run those with your method and print the average and standard deviation (using numpy.mean and numpy.std, see démo 5) of the sizes of the the sets S obtained. (Upload your Python on Studium using the file available there.)