SÉANCE DE SOUTIEN

1 Différents types d'algos

Dans les exercices qui suivent, il faudra trouver

- 2 algos de programmation dynamique,
- 1 algo de type diviser pour régner,
- 1 algo glouton.

Pour calculer les complexités, on supposera dans tous les exercices sauf le 1.3, que les opérations sur les entiers ou sur les réels (addition, multiplication...) se font en $O(1)$.

1.1 Rendre la monnaie

On suppose que l'on a accès à des pièces de valeur $0 < s_1 < \cdots < s_k \in \mathbb{N}$ (on a une infinité de pièces de chaque valeur). Soit $x \in \mathbb{N}$ une somme à atteindre, on veut calculer le nombre de façons différentes d'obtenir x en choisissant des pièces de valeur dans $\{s_1, \cdots, s_k\}$.

1. En prenant $s_1 = 1, s_2 = 3$ et $s_3 = 5$, combien y a-t-il de façon d'obtenir $x = 10$?
2. Donner un algorithme pour résoudre ce problème (attention, (s_1, s_2) et (s_2, s_1) donnent la même façon de réaliser $s_1 + s_2$).
3. Refaire la question 1 en utilisant votre algorithme.

1.2 Arbre couvrant de poids minimal

Soit $G = (V, E)$ un graphe connexe non orienté et $w : E \to]0, +\infty[$ une fonction de poids sur les arêtes. l'objectif est de calculer un arbre couvrant de poids minimal. Un arbre couvrant pour le graphe G est un sous ensemble d'arêtes $E' \subset E$ tel que tous les sommets de V appartiennent au moins à une arête de E' (c'est le côté "couvrant") et tel que le graphe (V, E') soit connexe et ne contiennent pas de cycle (c'est le côté "arbre"). On veut donc trouver, parmi tous les arbres couvrants de G, celui dont le poids, c'est à dire $\sum_{e' \in E'} w(e')$, est minimal.

1. Proposer un algorithme pour résoudre ce problème (au choix parmi programmation dynamique, diviser pour régner ou glouton). Quelle est sa complexité ?

1.3 Calcul de factorielle

L'objectif de cet exercice est d'estimer le temps de calcul de $n! = n(n-1)\cdots 2$ en fonction du coût de la multiplication de deux entiers (dans cet exercice, on suppose que le coût de la multiplication de deux entiers dépend de la taille des entiers, et n'est pas en $O(1)$). Dans la suite de l'exercice, on suppose que multiplier deux entiers $a, b \leq N$ prend un temps $O(M(N))$. L'algorithme naïf de multiplication que l'on apprend en primaire donne $M(N) = \log(N)^2$, mais on peut faire mieux. Par exemple, en utilisant la transformée de Fourrier rapide, on peut avoir $M(N) = \log(N) \cdot \log\log(N) \cdot \log\log\log(N)$ (ce que l'on arrondira en $\log(N)$). Dans la suite de l'exercice, on utilisera $M(N)$ et on discutera dans la dernière question des complexité que l'on obtient selon le choix de la multiplication choisie.

Plutôt que de donner un algorithme pour calculer la factorielle, on va donner un algorithme un peu plus général, qui étant donné k entiers $a_1, \cdots, a_k \leq n$, calcule le produit de ces k éléments.

1. Donner un algorithme naïf pour calculer le produit des k éléments $a_1, \cdots, a_k$. Quelle est sa complexité ?
2. Donner un algorithme plus malin pour calculer le produit des k éléments (à choisir parmi programmation dynamique, diviser pour régner ou glouton).
3. Calculer la complexité de l'algorithme précédent dans le cas où $M(N) = \log(N)^2$ puis dans le cas où $M(N) = \log(N)$.

1.4 Coefficients binomiaux

L'objectif de cet exercice est de calculer tous le coefficient binomial $\binom{n}{k}$.

1. On rappelle le triangle de Pascal : $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. Donner un algorithme pour calculer $\binom{n}{k}$ en utilisant le triangle de Pascal. Quelle est sa complexité (en terme d'opérations sur les entiers) ?
2. En utilisant l'algorithme naïf de l'exercice précédent pour calculer le produit de k éléments inférieurs à n et en supposant que la multiplication de deux entiers coûte $O(1)$ quelque soit leur taille, la formule $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ permet d'obtenir un algorithme en $O(k)$ pour calculer le coefficient binomial $\binom{n}{k}$. Est-ce mieux que votre algorithme précédent ? Qu'en est-il si vous voulez calculer tous les coefficients binomiaux $\binom{n'}{k'}$ pour $k' \leq k$ et $n' \leq n$?

2 Problèmes NP-complets

Montrer que les problèmes suivants sont NP-complets.

1. 3-SAT-NAE (not all equal) : Étant donné une conjonction de clauses de taille 3, déterminer s'il existe un assignement des variables tel qu'il n'existe aucune clause où tous les littéraux sont égaux.
2. 3-SAT-OIT (one in three) : Étant donné une conjonction de clauses de taille 3, déterminer s'il existe un assignement des variables tel que chaque clause possède exactement un littéral à `vrai` et les deux autres à `faux`.
3. Max-2-SAT : Étant donné une conjonction de clauses de taille 2 et un entier k, déterminer s'il existe un assignement des variables tel que au moins k clauses soient satisfaites.