Algorithmes exponentiels et gloutons

Djaouida Zaouche-Dahmani, EISTI
Tahar Gherbi, EISTI
Stefan Bornhofen, EISTI
Bartholomew George, EISTI

13 juin 2019

0.1 Le problème du rendu de monnaie

On considère un système de pièces de monnaie. La question est la suivante : quel est le nombre minimal de pièces à utiliser pour rendre un montant donné ? Le système de pièces de monnaie peut être modélisé par un n-uplet d'entiers naturels $S = (c_1, c_2, \ldots, c_n)$, où c_i représente la valeur de la pièce i. On suppose que $c_1 = 1$ et que $c_1 < c_2 < ... < c_n$. Un montant à rendre est un entier naturel X. Une répartition de pièces est un n-uplet d'entiers naturels $(x_1, x_2, ..., x_n)$, où x_i représente le nombre de pièces c_i. Notre question peut alors être reformulée comme suit :

Pour tout entier naturel X , on cherche un n-uplet d'entiers naturels $(x_1, x_2, ..., x_n)$ qui minimise $\Sigma_{i=1}^{n} x_i$ avec la contrainte $\Sigma_{i=1}^{n} c_i * x_i = X$.

0.1.1 Solution naive

La solution à laquelle on pense immédiatement est d'énumérer toutes les combinaisons possibles, de sélectionner celles qui impliquent un minimum de pièces et de choisir la meilleure.

1. Que garantit la propriété $c_1 = 1$?
2. Ecrire un programme python pour implémenter la solution naive.
3. Dérouler votre programme pour $S = (1, 3, 4)$ et $X = 6$, ensuite pour $X = 100$.
4. Donner la complexité de votre programme. Conclure.

0.1.2 Méthode gloutonne

La méthode gloutonne vise à optimiser la résolution d'un problème en partant du principe suivant : des choix locaux optimaux, étape après étape, devraient produire un résultat global optimal. Dans notre cas, on va répéter le choix de la pièce de plus grande valeur qui ne dépasse pas la somme restante.

1. Ecrire un programme python pour implémenter la méthode gloutonne.
2. Dérouler votre programme pour $S = (1, 3, 4)$ et $X = 6$, ensuite pour $X = 100$.
3. Pour $X = 6$, a t-on atteint une solution optimale ? Discuter.
4. Donner un jeu d'essai pour lequel une solution optimale est trouvée par la méthode gloutonne.
5. Donner la complexité de votre programme. Conclure.

0.2 Problème du sac à dos

On dispose d'un sac à dos S contenant n objets. Chaque objet i possède une valeur v_i et un poids w_i. On souhaiterait prendre une partie T de ces objets dans notre sac à dos. Il est à noter que ce dernier dispose d'une capacité limitée W. On va cependant chercher à maximiser la somme des valeurs des objets à mettre dans le sac à dos en respectant sa capacité maximale. Notre question peut alors être reformulée comme suit :

Etant donnée une capacité W, on cherche un sous-ensemble T d'objets qui maximise $\Sigma_{i \in T}\ v_i$ avec la contrainte $\Sigma_{i \in T}\ w_i \leq W$.

0.2.1 Solution naïve

On pourrait être tenté d'énumérer toutes les combinaisons d'objets possibles qui satisfont la capacité maximale du sac ou qui s'en rapprochent (le sac ne doit pas être obligatoirement rempli à fond).

1. Ecrire un programme qui implémente la solution naive pour le problème du sac à dos.
2. Donner des jeux de tests réalisables et d'autres non.
3. Donner la complexité de votre programme. Conclure.

0.2.2 Méthode gloutonne

Pour le problème du sac à dos, on va répéter le choix de l'objet de plus grand poids tant que le sac ne soit pas plein.

1. Ecrire un programme python pour implémenter la méthode gloutonne.
2. Reprendre le jeu d'essai précédent pour l'algorithme glouton.
3. Donner un jeu d'essai pour lequel une solution optimale est trouvée par la méthode gloutonne, un second pour lequel aucune solution optimale n'est trouvée.
4. Donner la complexité de votre programme. Conclure.

0.3 Bipartition

Étant donné un ensemble de n nombres, on veut le répartir en 2 sous-ensembles tels que la somme des éléments du premier soit égal (ou très proche) à la somme des éléments du second. Proposez un algorithme glouton pour résoudre ce problème. Appliquez le aux ensembles suivants :

1. $E_1 = \{2, 10, 3, 8, 5, 7, 9, 5, 3, 2\}$ peut être décomposé en :
   — $PE1 = \{10, 7, 5, 3, 2\}$,
   — $SE2 = \{9, 8, 5, 3, 2\}$. Les sommes sont les mêmes : 27.
2. $E_2 = \{771, 121, 281, 854, 885, 734, 486, 1003, 83, 62\}$ peut être décomposé en :
   — $PE_2 = \{1003, 771, 486, 281, 83\}$
   — $SE2 = \{885, 854, 734, 121, 62\}$
   Les sommes ne sont pas les mêmes mais sont proches : 2624 et 2656.