CentraleSupélec

Algorithmique et complexité

TD 7/7 – Résolution de problèmes NP-difficiles

L'objectif de ce TD est de traiter un problème NP-difficile.

Exercice 1 : Problème du Sac à dos

Considérons trois différents problèmes pratiques.

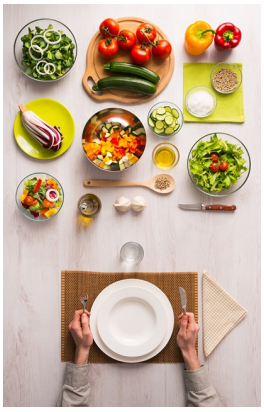
Bagage de cabine

Vous vivez une vie nomade, voyageant tout le temps autour du monde. Vous avez un ensemble d'éléments auxquels vous tenez plus ou moins selon vos besoins, par exemple votre ordinateur portable est quelque chose de très important, il a donc une grande valeur. Au contraire, vous aimez votre t-shirt SF, mais vous pouvez toujours en acheter un à destination, alors votre t-shirt SF a une petite valeur. Vous ne transportez que des bagages de cabine. Différentes compagnies aériennes imposent des limites différentes au poids des bagages à main, par exemple 7kg ou 10kg. Comment maximiser la valeur des articles que vous pouvez embarquer avec vous, tout en restant en dessous du poids maximum autorisé ?

Cambriolage

Un voleur parvient à pénétrer chez un bijoutier. Il y a beaucoup d'objets qu'il peut voler : montres, bagues, colliers ... Tous ces éléments ont des valeurs et des poids différents. Le voleur ne peut transporter qu'un poids limité. Supposons que les articles peuvent être divisés en catégories, chaque catégorie ayant un poids, une valeur et un certain nombre d'éléments disponibles. Comment le voleur devrait-il choisir quoi prendre ?

Repas équilibré



Vous recevez une liste de valeurs nutritionnelles pour une portion de chaque élément servi dans le menu quotidien de votre restaurant universitaire préféré. Pour chaque élément, vous obtenez le nombre de calories (ou poids) et la valeur protéique d'une partie de l'article. Qu'est-ce que vous avez choisi de manger pour maximiser la prise de protéines sans dépasser une certaine quantité de calories ?

Complexité

Question 1

Les trois problèmes peuvent être formalisés en utilisant le même format. Donnez une description formelle de ce problème : quelles sont les entrées, quelle est la question du problème de décision correspondant, quelle est la question du problème d'optimisation ?

Question 2

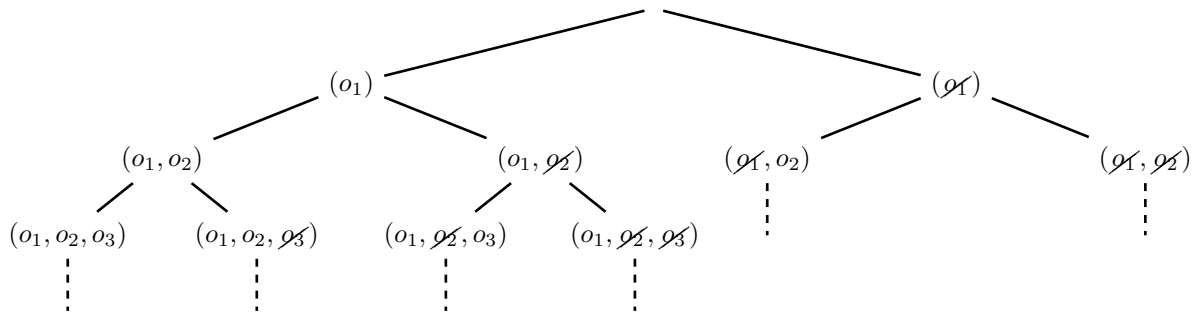
Montrez que ce problème (connu comme **KNAPSACK problem**) est NP-complet. Vous pouvez utiliser un problème NP-complet classique SUBSET-SUM :

SUBSET SUM
Entrée :
- un ensemble A d'entiers positifs $a_1, \ldots, a_n$
- une valeur $t \in \mathbb{N}$

Question : Existe-il un sous-ensemble $S \subseteq [1, n]$ dont la somme satisfait $\sum_{i \in S} a_i = t$?

Backtracking

On considère l'algorithme de **backtracking** qui décide à chaque étape la présence ou l'absence d'un objet dans la solution partielle courante. On construit ainsi l'espace de solutions qui prend la forme d'un arbre binaire de hauteur n :



L'algorithme de *backtracking* va énumérer toutes les solutions possibles pour trouver la meilleure (ici ayant la plus grande valeur).

L'algorithme de *backtracking* évitera d'explorer une branche si elle dépasse déjà la limite de poids.

Question 3

Écrivez en Python le code de cet algorithme. Pour vous aider, rendez-vous sur la page de pratique du TD :

https://wdi.centralesupelec.fr/1CC2000/TD7ProgEn

Glouton

Un algorithme **glouton** construira une solution pas à pas sans remettre en question les choix effectués. La solution obtenue n'aura pas de garantie d'optimalité.

Question 4

Proposez un algorithme **glouton** le plus efficace possible pour le problème du sac à dos.

Question 5

Quelle est la complexité en temps de votre algorithme ?

Question 6

Quelle serait la pire instance pour cet algorithme ?

Question 7

Retournez sur la page de pratique du TD. Complétez le code concernant l'algorithme glouton. Un *benchmark* est fourni pour comparer le temps d'exécution de *backtracking* et de *glouton* sur des instances aléatoires. Un autre *benchmark* est fourni pour comparer la qualité des solutions obtenues par ces algorithmes.

Programmation dynamique

Nous voulons proposer un algorithme de programmation dynamique pour résoudre le problème du sac à dos. On note $V(i, j)$ la valeur totale maximale qu'on peut mettre dans un sac à dos de taille j en embarquant que des objets parmi $o_1, \ldots, o_i$.

Question 8

Quelle est la formule de récursion qui calcule $V(i, j)$ pour tout i et j dans $\mathbb{N}$?

Question 9

Quelle est la complexité en temps et en espace d'un algorithme de programmation dynamique implémentant la formule précédente ?

Question 10

Cela signifie-t-il que $P = NP$?

Question 11

Retournez sur la page de pratique du TD. Écrivez en Python puis tester l'algorithme de résolution basé sur la programmation dynamique. Un *benchmark* est fourni pour comparer le temps d'exécution de tous les algorithmes.