

Algorithmique et Programmation avancée

3: Programmation dynamique

Principe

La programmation dynamique est un paradigme simple et puissant de conception d'algorithmes que l'on peut appliquer pour résoudre un problème si :

- La solution (optimale) d'un problème de taille n s'exprime en fonction de la solution (optimale) de sous-problèmes de taille inférieure à n , c'est le principe d'optimalité
- Une implémentation récursive naïve conduit à calculer de nombreuses fois la solution de mêmes sous-problèmes.

L'idée est alors d'éviter de calculer plusieurs fois la solution du même sous-problème. On définit une table pour mémoriser les calculs déjà effectués : à chaque élément correspondra la solution d'un et d'un seul problème intermédiaire, un élément correspondant au problème final. Il faut donc qu'on puisse déterminer les sous-problèmes qui seront traités au cours du calcul. Ensuite il faut remplir cette table.

Plus schématiquement, on suivra les étapes:

- Caractériser la structure d'une solution optimale,
- Définir récursivement la valeur d'une solution optimale,
- Calculer la valeur d'une solution optimale,
- Construire une solution optimale à partir des informations calculées.

Problème des pièces de monnaie

On dispose de pièces de monnaie en nombre illimité, de valeur $(a_1, a_2, \dots, a_n)$. On veut arriver à une somme S avec le minimum de pièces.

Pour cela on introduit la fonction z (sous forme de tableau) telle que $z(T, i)$ = nombre minimum de pièces choisies parmi les i premières $(a_1, a_2, \dots, a_i)$ pour faire T

La solution de notre problème est donnée par $z(S, n)$.

Illustration $((1, 2, 5, 10), S = 12)$:

On remarque que

$$z(T, i) = \min \begin{cases} z(T, i-1) & \text{i-eme pièce non utilisée} \\ z(T - a_i, i) + 1 & \text{sinon} \end{cases}$$

On complète le tableau z pour obtenir la solution optimale.

Exercices 1. Écrire un pseudo-code de l'algorithme décrit précédemment.

2. Coder cet algorithme sous Python3
3. Adapter cet algorithme pour qu'il renvoie aussi les pièces utilisées.
4. Adapter cet algorithme pour qu'il fonctionne même sans pièce de 1. (possibilité d'absence de solution)
5. Considérer (pseudo-code) un algorithme qui, partant d'un ensemble de pièces triées par ordre décroissant, donne une solution "efficace" en faisant le meilleur choix local. Comparer les résultats selon les pièces utilisées (ex: $(6, 4, 1)$ pour faire 8). Conclure
6. Le coder