Esiee-Paris - cours d'algorithmique - feuille d'exercices numéro 2

Septembre 2023 – R. Natowicz, I. Alamé, A. Çela, X. Hilaire, T. Wu, W. Xu

La solution de chaque exercice doit être programmée en langage Java, voir l'ébauche de programme.

Exercice 4. Deux sacs. Nous avons deux sacs S_0 et S_1 de contenances C_0 et C_1. Il y a n objets, l'objet i est de valeur v_i et de taille t_i. Nous voulons deux sacs de valeur totale maximum construits sur cet ensemble d'objets.

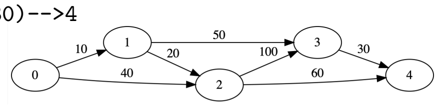
Nous savons déjà construire un sac de valeur maximum. « *Du coup* » chacune et chacun de nous a une première idée : « *Je calcule le premier sac de valeur maximum sur les n objets puis le second sac de valeur maximum sur le sous-ensemble des objets qui restent après avoir construit le premier sac.* » C'est une idée. Mais cette méthode séquentielle – un sac puis l'autre – ne résout pas le problème posé. Donner un exemple sur lequel cette méthode ne retourne pas un couple de sacs de valeur totale maximum.

Comment résoudre le problème ? Supposons le problème résolu ! Et, magie de la programmation dynamique, trois cas se présentent : 1) le n-ème objet n'est dans aucun des deux sacs 2) le n-ème objet est dans le premier sac 3) le n-ème objet est dans le second sac. Il n'y a pas d'autre cas.

Notons $m(n,C_0,C_1)$ la somme maximum des valeurs des objets présents dans les sacs S_0 et S_1 de contenances C_0 et C_1 contenant un sous-ensemble des n objets.
– Quelle est la valeur maximum $m(n,C_0,C_1)$ si le n-ème objet n'est dans aucun des deux sacs ?
– même question s'il est dans le premier sac ;
– même question s'il est dans le second sac.
En déduire la valeur $m(n,C_0,C_1)$.
– Généralisation : 1) pour tous k, c_0 et c_1, $1 \leq k < n+1$, $0 \leq c_0 < C_0+1$, $0 \leq c_1 < C_1+1$, donner l'expression de la valeur maximum $m(k,c_0,c_1)$ des deux sacs de contenance c_0 et c_1 dont les contenus sont un sous-ensemble des k premiers objets.
– Base de la récurrence : donner la valeur $m(0,c_0,c_1)$ pour toutes contenances c_0 et c_1, $0 \leq c_0 < C_0+1$, $0 \leq c_1 < C_1+1$.
– Les valeurs et tailles des objets sont dans deux tableaux, $V[0:n]$ et $T[0:n]$ de termes généraux v_i et t_i.
Écrire une fonction `int[][][] calculerM(int[] V, int[] T, int C0, int C1)` qui calcule et retourne un tableau $M[0:n+1][0:C_0+1][0:C_1+1]$ de terme général $M[k][c_0][c_1] = m(k,c_0,c_1)$.
– Écrire une procédure `acsm(int[][][] M, int[] V, int[] T, int k, int c0, int c1)` qui affiche les contenus des sacs de valeur maximum, de contenances c_0 et c_1, contenant un sous-ensemble des k premiers objets (acsm = afficher les contenus des sacs de valeur maximum). Appel principal : `acsm(M,V,T,n,C0,C1)`.

Exercice 5. Un trajet de coût minimum. Les villes $0,1,\ldots,n-1$ sont desservies par deux lignes de bus. Au départ de la ville i vous pouvez aller à la ville $i+1$ avec un coût $d_1(i)$ ou directement à la ville $i+2$ avec un coût $d_2(i)$. Les coûts $d_1(i)$ et $d_2(i)$ sont dans les tableaux d'entiers $D1[0:n]$ et $D2[0:n]$.
Remarque : dans ces tableaux les valeurs $D1[n-1]$ et $D2[n-2]$ et $D2[n-1]$ sont quelconques, ci-dessous fixées à -1.
On veut connaître le coût minimum $m(n-1)$ d'un trajet allant de la ville 0 à la ville $n-1$ et connaître le trajet.

Exemple avec $n=5$:
– $D1 = [10,20,100,30,-1]$ `// 0--(10)-->1, 1--(20)-->2, 2--(100)-->3, 3--(30)-->4`
– $D2 = [40,50,60,-1,-1]$ `// 0--(40)-->2, 1--(50)-->3, 2--(60)-->4`
– Trajet de coût minimum : `0--(10)-->1--(50)-->3--(30)-->4`
– Coût de ce trajet : 90.



1) Donner l'expression de la valeur $m(n-1)$.
2) Équation de récurrence : donner les valeurs $m(0)$ et $m(1)$ et l'expression de la valeur $m(j)$, $\forall j$, $2 \leq j < n$.
3) Écrire une fonction `int[] calculerM(int[] D1, int[] D2)` qui calcule et retourne le tableau $M[0:n]$ de terme général $M[j] = m(j)$.
4) Écrire une fonction `afficherTrajetMinimum(int[] M, int[] D1, int[] D2, int j)` qui affiche un chemin de coût minimum de la ville 0 à la ville j. L'affichage du chemin de la ville 0 à la ville $n-1$ tel que dans l'exemple ci-dessus s'obtient par l'appel de fonction `afficherTrajetMinimum(M, D1, D2, n-1)`.

Exercice 6. Un trajet de coût minimum (encore plus fort!). Pour tout couple de villes (i,j), $0 \leq i < j < n$, il existe une ligne directe pour aller de la ville i à la ville j. Ce trajet direct coûte $d(i,j)$. Ces coûts directs sont dans un tableau $D[0:n][0:n]$ de terme général $D[i][j] = d(i,j)$ pour tous indices i et j, $0 \leq i < j < n$. Les autres valeurs du tableau sont quelconques. Ainsi :
– $d(0,1)$ est le coût direct pour aller de la ville 0 à la ville 1, $d(0,2)$ de la ville 0 à la ville 2, ..., $d(0,n-1)$ de la ville 0 à la ville $n-1$;
– $d(1,2)$ est le coût direct pour aller de la ville 1 à la ville 2, $d(1,3)$ de la ville 1 à la ville 3, ... $d(1,n-1)$ de la ville 1 à la ville $n-1$;
. . .
– $d(n-3,n-2)$ est le coût direct pour aller de la ville $n-3$ à la ville $n-2$, $d(n-3,n-1)$ de la ville $n-3$ à $n-1$;
– $d(n-2,n-1)$ coût direct de $n-2$ à $n-1$.

1) Donner l'expression de la valeur $m(n-1)$.
2) Équation de récurrence : donner la valeur $m(0)$ et l'expression de la valeur $m(j)$, $\forall j$, $1 \leq j < n$.
3) Écrire une fonction `int[][] calculerM(int[][] D)` qui calcule et retourne le tableau $M[0:n]$ de terme général $M[j] = m(j)$ et le tableau $A[0:n]$ de terme général $A[j] = \arg m(j)$.
4) Écrire une fonction `afficherTrajetMinimum(int[] M, int[][] D, int j)` qui affiche un chemin de coût minimum de la ville 0 à la ville j.
L'appel principal de cette fonction est `afficherTrajetMinimum(M, D, n-1)`.
Sur l'exemple ci-contre : `0--(40)-->2--(40)-->3--(50)-->4`