République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ESI ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
Ecole nationale Supérieure d' Informatique
ex. INI (Institut National de formation en Informatique)

Rapport du TP n°= 1

Théorie de programmation et graphes orientés

Thème :

Problème du sac à dos (Programmation dynamique)

Réalisé par :

- BOUTIBA Karim
- REMIL Youcef

✓ **Option :** SIQ

✓ **Groupe :** 2

Proposé par :

- Mr HADIM Boukhalfa

Promotion : 2018/2019

1. Problème du sac à dos :

1.1 Contexte du problème :

le problème du sac à dos, noté également KP (en anglais, Knapsack problem) est un problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

1.2 Enoncé mathématique :

On dispose de n objets de poids positifs $p_1, p_2, \dots, p_n$ et de bénéfices positifs $b_1, b_2, \dots, b_n$. Notre sac à doc à une capacité maximale en poids de P
Le but est de maximiser $\sum_{i=1}^{n} x_i b_i$ tel que $\sum_{i=1}^{n} x_i p_i \leq P$ et $x_i \in \{0,1\}$

1.3 Procédé d'exploration systématique :

Cet examen systématique peut être réalisé à l'aide d'un arbre d'exploration binaire (les triangles représentent des sous-arbres).

L'arbre se décrit en descendant depuis le sommet jusqu'au bas des triangles (les feuilles de l'arbre). Chaque case correspond à un unique parcours possible. En suivant les indications portées le long des arêtes de l'arbre, à chaque parcours correspond une suite de valeurs pour $x_1, x_2, \dots, x_n$ formant un vecteur contenu. Il est alors possible de reporter dans chaque case la valeur totale et le poids total du contenu correspondant. Il ne reste plus qu'à éliminer les cases qui ne satisfont pas la contrainte, et à choisir parmi celles qui restent celle (ou une de celles) qui donne la plus grande valeur à la fonction objectif.

À chaque fois qu'un objet est ajouté à la liste des objets disponibles, un niveau s'ajoute à l'arbre d'exploration binaire, et le nombre de cases est multiplié par 2. L'exploration de l'arbre et le remplissage des cases ont donc un coût qui **croît exponentiellement** avec le nombre n d'objets.

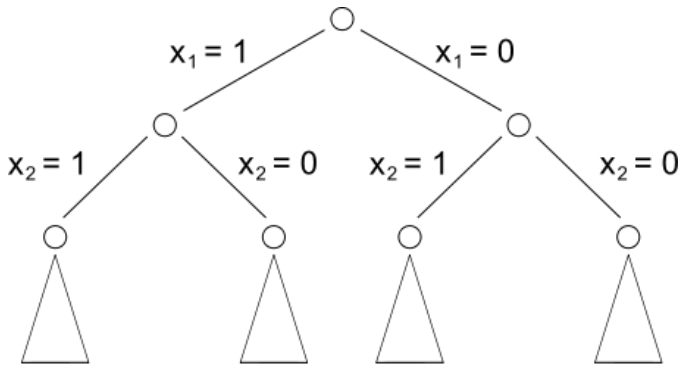


Figure 1 : Arbre d'exploration binaire

1.3.1 Inconvénient :

Cette approche suggère un algorithme avec une complexité o(2^n), ce qui est généralement déconseillé parce qu'il est à la fois lent et consomme beaucoup d'espace mémoire.

1.4 L'objectif :

L'objectif de ce TP dans le cadre du module théorie du programmation et graphes orientés (TPGO) est de trouver un algorithme de résolution du problème de sac à dos de manière à ce que la complexité de l'algorithme doit être polynomiale o(n^k) à l'aide de la programmation dynamique.

2. Programmation dynamique :

La **programmation dynamique** est un paradigme de programmation, c'est-à-dire une façon particulière d'appréhender un problème algorithmique donné.

C'est une méthode utile pour obtenir une solution exacte à un problème algorithmique, là où une solution « classique » se trouve être trop complexe, c'est-à-dire trop peu efficace. On parle alors d'**optimisation combinatoire**.

L'efficacité de cette méthode repose sur le principe d'optimalité énoncé par le mathématicien Richard Bellman : *« toute politique optimale est composée de sous-politiques optimales »*

Autrement dit, elle consiste à résoudre un problème en le décomposant en sous-problèmes puis à résoudre les sous-problèmes, des plus petits aux plus grands en stockant les résultats intermédiaires. Elle a d'emblée connu un grand succès, car de nombreuses fonctions économiques de l'industrie étaient de ce type, comme la conduite et l'optimisation de procédés chimiques, ou la gestion de stocks.

2.1 Principe d'optimalité de Bellman :

La programmation dynamique est fondée sur le principe d'optimalité de Bellman :

Soit f une fonction réelle de x et $y = (y_1, y_2, \dots, y_n)$
Si f est décomposable sous la forme : $f(x) = f_1\big(x, f_2(y)\big)$

Alors, on a :

$$Opt_{xy}\, f(x,y) = Opt_x\{\, f_1\big(x, Opt_y\{\, f_2(y)\,\}\big)$$

Ou Opt représente Min ou Max .

2.2 Equation de la programmation dynamique :

Pour une séquence de décisions $u_1, u_2, \dots, u_T$, une séquence d'états $x_1, x_2, \dots, x_T$, une fonction de transition :

$$x_{k+1} = \theta_k(x_k, u_k)$$

On définit pour chaque étape une fonction de coût optimal (cas additif) :

$$F_k(x_k) = Opt_{u_k}\{f_k(x_k, u_k) + F_{k+1}(\theta_k(x_k, u_k))\}$$

3. Solution du problème du sac à dos :

3.1 Idée de résolution du problème dynamiquement :

- On note $B(k, p)$ le bénéfice maximal réalisable avec des objets $1,2, \dots, k$ et le poids maximal P

- On a pour $k = 1$:

$$B(1,p) = \begin{cases} 0 & si \quad P < p_1 \\ b_1 & si \quad P \geq p_1 \end{cases}$$

- Pour $k > 1$:

$$B(k,p) = \begin{cases} B(k-1,p) & si \quad P < p_k \\ \max(B(k-1,p), B(k-1,p-p_k) + b_k) & si \quad P \geq p_k \end{cases}$$

3.2 Algorithme de résolution du problème :

- On définit une structure d'un objet :

```
Structure
{
    poids : entier ;
    bénéfice : entier ;
    pris : booléen ;
} Objet ;
```

- On définit les variables globales

```
poidsMax : entier ;
nbObjs : entier ;
tabObjs : tableau [nbObjs] d'Objet ;
matBenif :  tableau [nbObjs] [poidsMax+1] d'entier ;
```

- La fonction qui permet de remplir la matrice et par la suite retourner le gain maximum :

```
Fonction sac_a_dos () : entier
{
    i, j, k : entier ;
    pour (i = 0 ; i <= poidsMax ; i++ )
        matBenif [0] [i] = ( i < tabObjs[0].poids() ) ? 0 : tabObjs[0].
    benifice() ;
    pour (k = 1 ; k < nbObjs ; k++ )
    {
        pour (i = 0 ; i <= poidsMax ; i++)
        {
          matBenif [k] [i]  = ( i < tabObjs[k].poids())? matBenif[k-1][i] :
          max(matBenif [k-1] [i] ,matBenif [k-1] [i-tabObjs[k].poids()]
            +tabObjs[k].benifice());
        }

    }
    sac_a_dos = MatBenif [nbObjs-1] [poidsMax] ;

}
```

- La fonction qui permet de déterminer les éléments à prendre :

```
Procedure retournerElementsPris ()
{
    k = nbObjs - 1 : entier ;
    l = poidsMax : entier ;
    valeurA = 0 , valeurB = 0  : entier ;
    Tant que (k > 0)       {

    valeurA  =  matBenif[k][l];
    valeurB  =  matBenif[k-1][l];
    si (valeurA  <> valeurB)

    {
            tabObjs[k].pris = vrai ;
            l = l - tabObjs[k].poids;

    }
```

```
            sinon     tabObjs[k].pris = faux;
            k = k-1 ;
        }
        si  (valeurB <> 0)  tabObjs[0].pris = vrai ;
        sinon  tabObjs[0].pris = faux ;
    }
```

3.3 Jeu d'essai :

On va dérouler l'exemple suivant :

Objets	1	2	3	4	5	6	7	8
Poids	2	3	5	2	4	6	3	1
Bénéfices	5	8	14	6	13	17	10	4

On prend le poids maximum = 12

$B(k,p)$	0	1	2	3	4	5	6	7	8	9	10	11	12
$k=1$	0	0	5	5	5	5	5	5	5	5	5	5	5

$B(k,p)$	0	1	2	3	4	5	6	7	8	9	10	11	12
$k=1$	0	0	5	5	5	5	5	5	5	5	5	5	5
$k=2$	0	0	5	8	8	13	13	13	13	13	13	13	13

$B(k,p)$	0	1	2	3	4	5	6	7	8	9	10	11	12
$k=1$	0	0	5	5	5	5	5	5	5	5	5	5	5
$k=2$	0	0	5	8	8	13	13	13	13	13	13	13	13
$k=3$	0	0	5	8	8	14	14	19	22	22	27	27	27

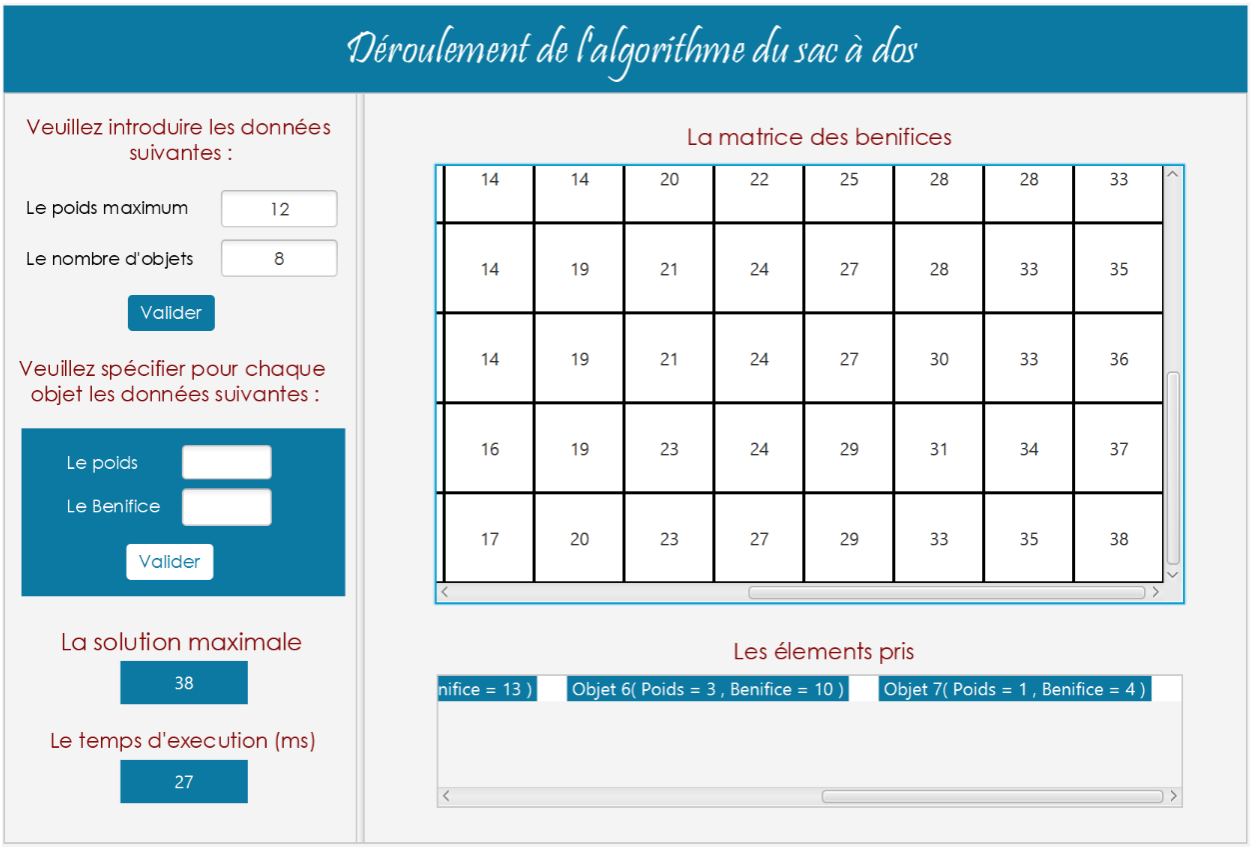
$B(k,p)$	0	1	2	3	4	5	6	7	8	9	10	11	12
$k=1$	0	0	5	5	5	5	5	5	5	5	5	5	5
$k=2$	0	0	5	8	8	13	13	13	13	13	13	13	13
$k=3$	0	0	5	8	8	14	14	19	22	22	27	27	27
$k=4$	0	0	6	8	11	14	14	20	22	25	28	28	30
$k=5$	0	0	6	8	13	14	19	21	24	27	28	33	35
$k=6$	0	0	6	8	13	14	19	21	24	27	30	33	36
$k=7$	0	0	6	10	13	16	19	23	24	29	31	34	37
$k=8$	0	4	6	10	14	17	20	23	27	29	33	35	38

3.4 Présentation de l'application :

L'appli est conçue avec le langage (Java / Java Fx) :

- **Le Menu principal :**

- On va dérouler l'exemple précédent :



4. Webographie :

[1]

https://fr.wikipedia.org/wiki/Programmation_dynamique#Principe

[2]

https://openclassrooms.com/fr/courses/1164481-introduction-a-la-programmation-dynamique

[3]

http://www.lsis.org/master/ancien_site/documents/Supports%20de%20cours_134.pdf

[4]

https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos