Programmation `Java` : Algorithmes

Valeur d'accueil et de reconversion en informatique (VARI1)

Daniel Porumbel (dp.cnam@gmail.com)
`http://cedric.cnam.fr/~porumbed/vari1/`

Contributions aux slides :
Cédric Bentz

Plan

1. Méthodes de Tri
2. Sac-à-Dos : heuristique et programmation dynamique

Début du tri par sélection

Que fait ce morceau de code ?

```
int indiceMin = a;
for(int i=a ; i<=b ; i++)
    if(tab[i]<tab[indiceMin])
        indiceMin = i;
```

Début du tri par sélection

Que renvoie cette fonction ?

```
class TriSelect{
   static int[] tab;

   //chercher l'indice de valeur min
   //dans l'intervalle [a, b]
   static int intervIndiceMin (int a, int b) {
         int indiceMin = a;
         for(int i=a ; i<=b ; i++)
              if(tab[i]<tab[indiceMin])
                   indiceMin = i;
         return indiceMin;
   }
```

Que fait ce morceau de code ? ?

```
for(int i=0;i<n;i++){
    int j = intervIndiceMin(i,n-1);
    int tmp = tab[j];
    tab[j] = tab[i];
    tab[i] = tmp;
}
```

Code complèt : Tri par sélection

```
import java.util.*;
class TriSelect{
    static int[] tab;
    static int intervIndiceMin(int a, int b){
        //prendre le code de la diapo avant la précédente
    }
    public static void main(String[] args){
        int n  = 20;
        tab = new int[n];
        for(int i=0;i<n;i++)
            tab[i] = (int)(50*Math.random());
        //Afficher le tableau de départ:
        System.out.println(Arrays.toString(tab));
        //Réaliser le tri avec une boucle:
        for(int i=0;i<n;i++){
            int j = intervIndiceMin(i,n-1);
            int tmp = tab[j];
            tab[j]  = tab[i];
            tab[i]  = tmp;
        }
        System.out.println(Arrays.toString(tab));//tab trié
    }
}
```

```
boolean triFini = false;
while(triFini==false){
    triFini = true;
    for(int i=0;i<n-1;i++)
        if(tab[i]>tab[i+1]){
            triFini = false;
            int tmp = tab[i];
            tab[i] = tab[i+1];
            tab[i+1]=tmp;
        }
}
```

1. Quel est le résultat de ce (pseudo-)code ?
2. Il s'appelle le `tri à bulles` !

```
boolean triFini = false;
while(triFini==false){
    triFini = true;
    for(int i=0;i<n-1;i++)
        if(tab[i]>tab[i+1]){
            triFini = false;
            int tmp = tab[i];
            tab[i] = tab[i+1];
            tab[i+1]=tmp;
        }
}
```

1. Quel est le résultat de ce (pseudo-)code ?
2. Il s'appelle le `tri à bulles`!

```
int[] apparitions = new int[50];
for(int i=0;i<n;i++)
    apparitions[tab[i]]++;
```

1. Quel est le résultat de ce (pseudo-)code ?
2. Comment remplir le tableau trié ?

```
int pos = 0;
for(int val=0;val<50;val++)
    for(int j=0;j<apparitions[val];j++){
        tab[pos] = val;
        pos++;
    }
```

3. Il s'appelle tri par dénombrement !

```
int[] apparitions = new int[50];
for(int i=0;i<n;i++)
    apparitions[tab[i]]++;
```

1. Quel est le résultat de ce (pseudo-)code ?
2. Comment remplir le tableau trié ?

```
int pos = 0;
for(int val=0;val<50;val++)
    for(int j=0;j<apparitions[val];j++){
        tab[pos] = val;
        pos++;
    }
```

: Il s'appelle tri par dénombrement !

```
int n  = 20;
tab = new int[n];
for(int i=0;i<n;i++)
    tab[i] = (int)(50*java.lang.Math.random());
//Affichage tableau de départ
println(java.util.Arrays.toString(tab));
//Tri:
java.util.Arrays.sort(tab);
//Affichage tableau trié
println(java.util.Arrays.toString(tab));
```

- Inconvénient : pour trier en ordre décroissant, il faut
  - inverser les nombres (ex, $5 \rightarrow -5$) ;
  - Appeler `java.util.Arrays.sort(tab)` ;
  - inverser les nombres de nouveau.

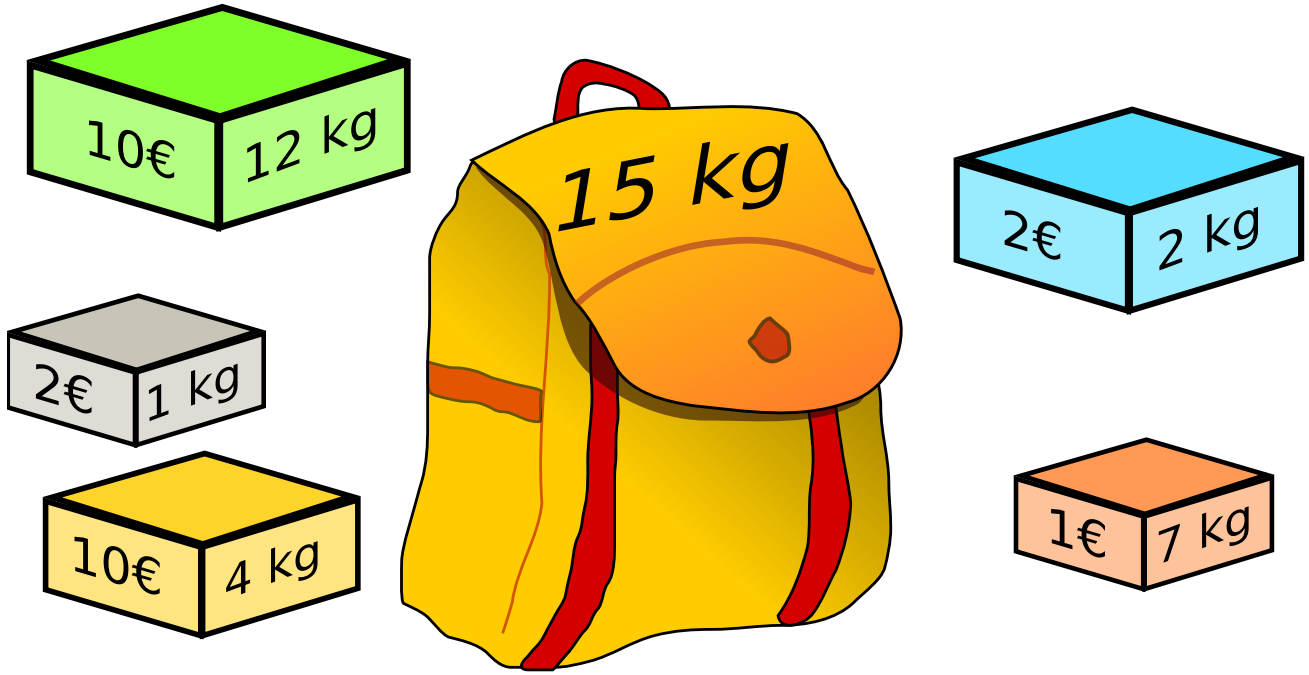
```
int n  = 20;
tab = new int[n];
for(int i=0;i<n;i++)
    tab[i] = (int)(50*java.lang.Math.random());
//Affichage tableau de départ
println(java.util.Arrays.toString(tab));
//Tri:
java.util.Arrays.sort(tab);
//Affichage tableau trié
println(java.util.Arrays.toString(tab));
```

- Inconvénient : pour trier en ordre décroissant, il faut
  - inverser les nombres (ex, $5 \rightarrow -5$) ;
  - Appeler `java.util.Arrays.sort(tab)` ;
  - inverser les nombres de nouveau.

Plan

1 Méthodes de Tri

2 Sac-à-Dos : heuristique et programmation dynamique



Problème du sac-à-dos

- Soit un sac-à-dos de capacité donnée,
- Soit une liste d'articles de poids et valeurs variées
- Quels articles choisir pour maximiser le profit ?

Un algorithme glouton

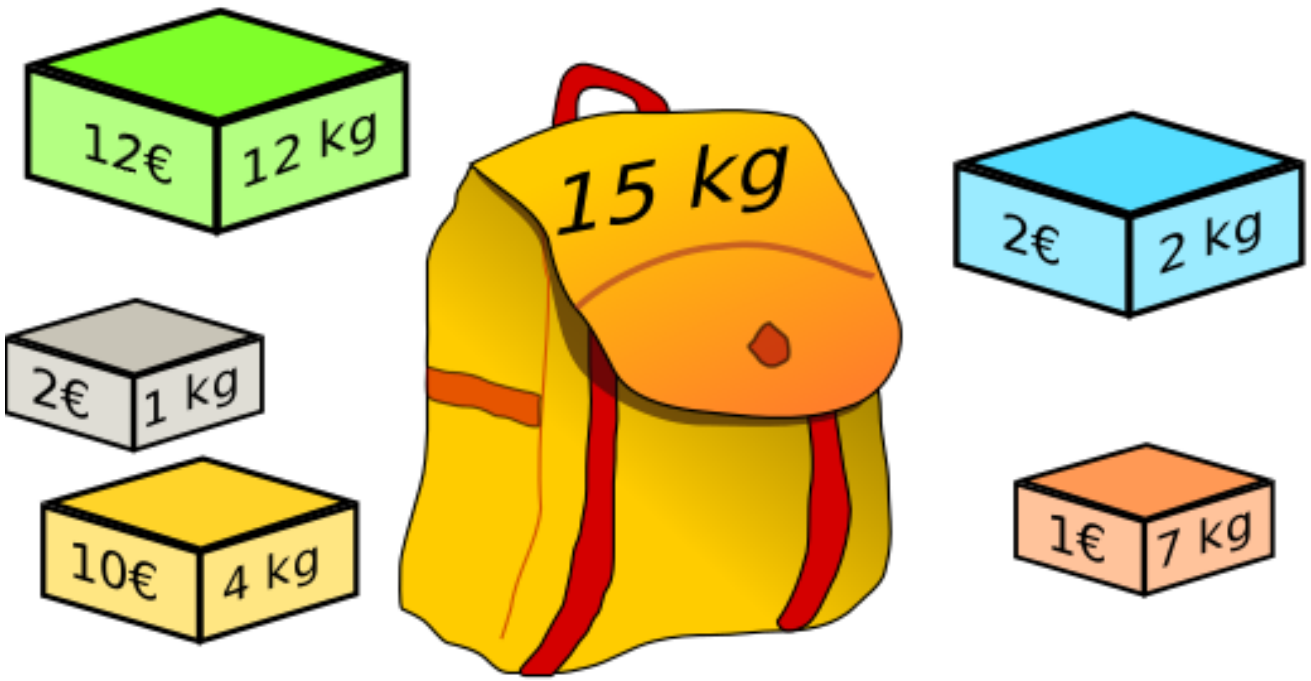
1. Trier les article dans l'ordre décroissant de leur valeur par kg $\frac{v_i}{p_i}$ (rentabilité)
   - Utiliser un `tri à bulles` : à chaque inversion, il faut inverser les valeurs par kg, les poids et les valeurs.
2. Remplir le sac avec les articles un par un dans cet ordre
   - on trouve la bonne solution sur l'exemple précédent

! On ne trouve pas toujours la bonne solution

Un algorithme glouton

1. Trier les article dans l'ordre décroissant de leur valeur par kg $\frac{v_i}{p_i}$ (rentabilité)
   - Utiliser un `tri à bulles` : à chaque inversion, il faut inverser les valeurs par kg, les poids et les valeurs.
2. Remplir le sac avec les articles un par un dans cet ordre
   - on trouve la bonne solution sur l'exemple précédent

! On ne trouve pas toujours la bonne solution



États de programmation dynamique

Soit un sac-à-dos de $Cap = 10$, poids : $5,\ 4,\ 3,\ 2$
tous les valeurs valent 1

prof max	0	0	0	0	0	0	0	0	0	0	0
poids total	0	1	2	3	4	5	6	7	8	9	10

- $p_1 = 5$ génère un profit/valeur de 1
- $p_2 = 4$ génère un profit/valeur de 1
- $p_2 = 3$ génère un profit/valeur de 1
- $p_2 = 2$ génère un profit/valeur de 1

États de programmation dynamique

Soit un sac-à-dos de $Cap = 10$, poids : $5,\ 4,\ 3,\ 2$
tous les valeurs valent 1

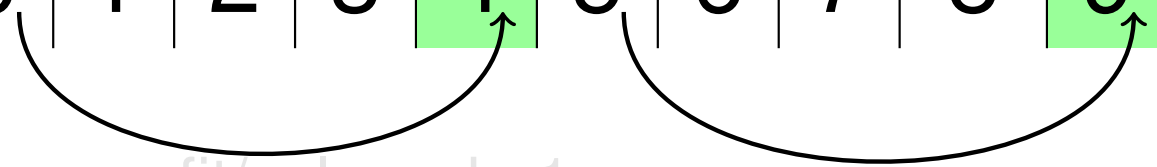
prof max	0	0	0	0	0	1	0	0	0	0	0
poids total	0	1	2	3	4	5	6	7	8	9	10

- $p_1 = 5$ génère un profit/~~valeur~~ de 1
- $p_2 = 4$ génère un profit/valeur de 1
- $p_2 = 3$ génère un profit/valeur de 1
- $p_2 = 2$ génère un profit/valeur de 1

États de programmation dynamique

Soit un sac-à-dos de $Cap = 10$, poids : $5,\ 4,\ 3,\ 2$
tous les valeurs valent 1

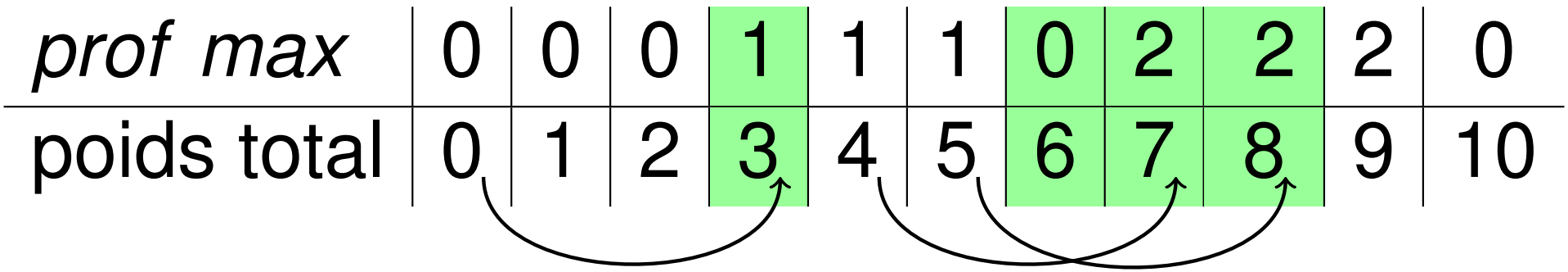
prof max	0	0	0	0	1	1	0	0	0	2	0
poids total	0	1	2	3	4	5	6	7	8	9	10



- $p_1 = 5$ génère un profit/valeur de 1
- $p_2 = 4$ génère un profit/valeur de 1
- $p_2 = 3$ génère un profit/valeur de 1
- $p_2 = 2$ génère un profit/valeur de 1

États de programmation dynamique

Soit un sac-à-dos de $Cap = 10$, poids : $5,\ 4,\ 3,\ 2$
tous les valeurs valent 1



- $p_1 = 5$ génère un profit/valeur de 1
- $p_2 = 4$ génère un profit/valeur de 1
- $p_2 = 3$ génère un profit/valeur de 1
- $p_2 = 2$ génère un profit/valeur de 1

États de programmation dynamique

Soit un sac-à-dos de $Cap = 10$, poids : $5,\ 4,\ 3,\ 2$
tous les valeurs valent 1

prof max	0	0	1	1	1	2	2	2	2	3	3
poids total	0	1	2	3	4	5	6	7	8	9	10

- $p_1 = 5$ génère un profit/valeur de 1
- $p_2 = 4$ génère un profit/valeur de 1
- $p_2 = 3$ génère un profit/valeur de 1
- $p_2 = 2$ génère un profit/valeur de 1

Program. dynamique :méthode tabulaire

On utilise une structure tabulaire f à deux dimensions/indices

$f(i, P)$ optimum d'un sac-à-dos de capacité P en utilisant que les i premiers articles

- C'est un sac-à-dos plus petit

objectif calculer $f(n, P_{max})$

recursion Toute valeur $f(i, P)$ peut être calculée à partir des valeurs $f(i-1, 0)$, $f(i-1, 1)$, $f(i-1, 2)$, $\ldots f(i-1, P)$

La récursion pour le sac-à-dos

Comment calculer $f(i, P)$ à partir d'un sac-à-dos avec $i-1$ articles : $f(i-1,0),\ f(i-1,1),\ f(i-1,2),\ \ldots f(i-1,P)$?

On a deux cas :

si l'article i est sélectionné : $f(i,P) = v_i + f(i-1, P-p_i)$

- si $p_i \leq P$

si l'article i n'est **pas** sélectionné : $f(i,P) = f(i-1,P)$

$$\Longrightarrow$$

$$f(i,P) = \max(v_i + f(i-1, P-p_i), f(i-1,P))$$

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2								
i=1								

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2								
i=1	0	0	0	0	0			

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2								
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0							
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0					
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55				
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55			
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55	100		
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55	100	100	
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0							
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18						
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18					
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55				
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73			
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73	100		
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73	100	118	
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0							
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55				
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73			
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100		
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100	118	
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100	118	125
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100	118	125
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

==> **VALEUR OPTIMALE = 125**

Implémentation à l'aide d'une matrice *mat*

- La valeur $f(i, p)$ est stockée dans la case $mat[i][p]$
- Initialiser $mat[0][0] = mat[0][1] = \cdots = mat[0][Pmax] = 0$

```
for(int i=0;i<n;i++)
  for(int p=0;p<=PMax;p++)
    if(poids[i]<=p)
      mat[i+1][p]=Math.max(
                          mat[i][p],
                          mat[i][p-poids[i]]+val[i]
                          );
```

Ce code calcule $mat[i+1][p]$ à partir de $mat[i][p]$ et $max[i][p - poids[i]]$ avec la formule de récurrence

- Attention : le premier article se trouve à la case 0 dans les tableaux `val` et `poids`

Le reste des diapos sont uniquement pour votre culture générale, ils ne seront pas utilisées pour l'examen ou les TPs.

Rappels matrices sous Java

```
int[][] mat = new int[9][10];//matrice 9 X 10
for(int i=0;i<9;i++)          //i est la ligne
    for(int j=0;j<10;j++)
        mat[i][j] = i*j;
```

1. Le code plus haut initialise une matrice (tableau de tableaux)

```
void afficherMatrice(int[][] m){
    for(int i=0;i<m.length;i++)
        System.out.println(
                java.util.Arrays.toString(m[i]));
}
```

2. Voici une méthode qui affiche une matrice
   - `m.length` est le nombre de lignes
   - `java.util.Arrays.toString(m[i])` affiche la ligne *i*

Rappels matrices sous Java

```
int[][] mat = new int[9][10];//matrice 9 X 10
for(int i=0;i<9;i++)          //i est la ligne
    for(int j=0;j<10;j++)
        mat[i][j] = i*j;
```

1. Le code plus haut initialise une matrice (tableau de tableaux)

```
void afficherMatrice(int[][] m){
   for(int i=0;i<m.length;i++)
      System.out.println(
            java.util.Arrays.toString(m[i]));
}
```

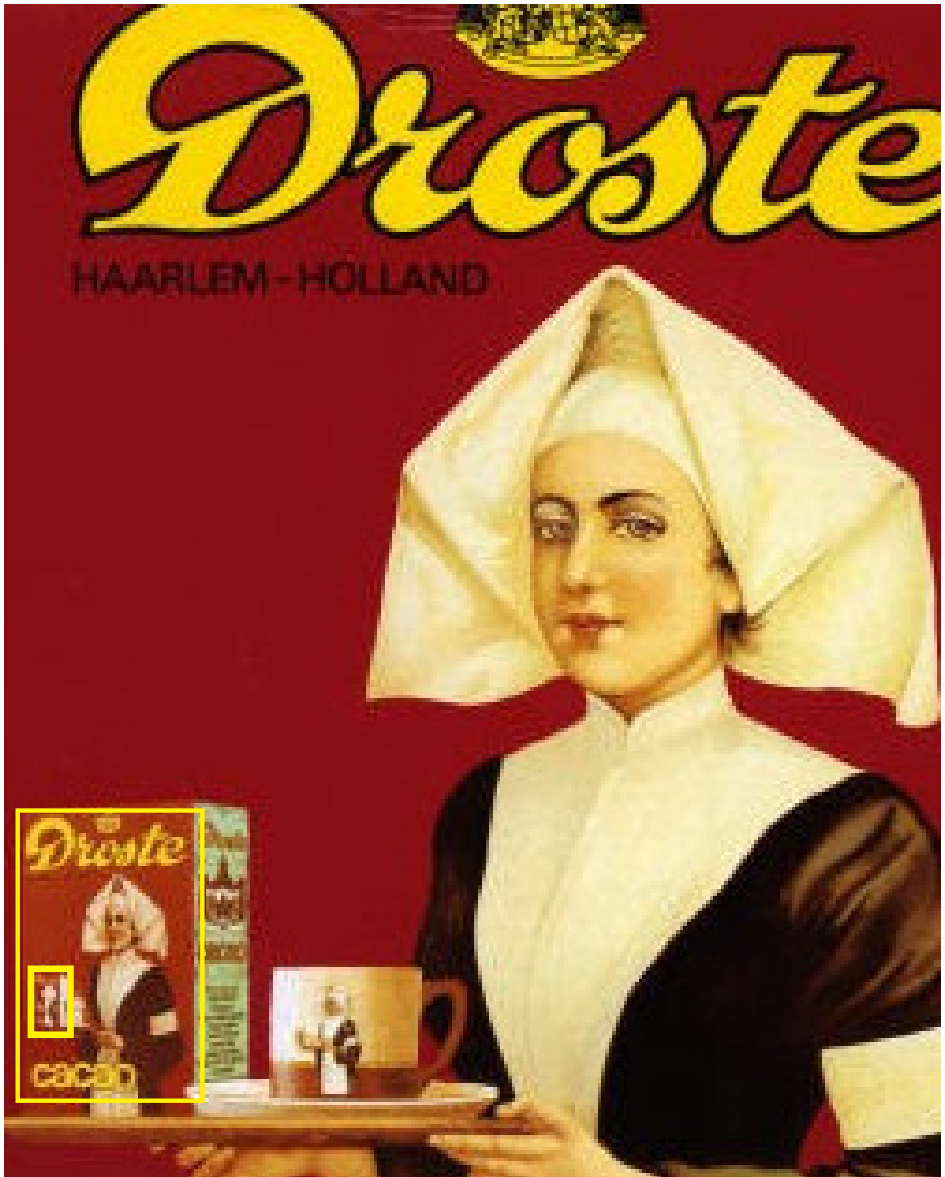
2. Voici une méthode qui affiche une matrice
   - `m.length` est le nombre de lignes
   - `java.util.Arrays.toString(m[i])` affiche la ligne *i*

D’autres opérations sur des tableaux

- Calculer la sommes des éléments d’un tableau
- Calculer cette somme de manière récursive !

D'autres opérations sur des tableaux

- Calculer la sommes des éléments d'un tableau
- Calculer cette somme de manière récursive !




```
f(0,9):
    return tab[0]+f(1,9)
                 |
  <--------------+
f(1,9):
    return tab[1]+f(2,9)
                 |
  <--------------+
f(2,9):
    return tab[2]+f(2,9)
                 |
  <--------------+
   ...  ...  .  ...  ...
f(9,9):
    return tab[9]
```

Récursivité et passage des paramètres

```
int f(int x, int y){
    if(x==y)
        return tab[x];
    return tab[x]+f(x+1,y);
}
public static void main(){
    int somme;
    somme = f(0,9); //tab[0]+tab[1]+...tab[9]
    System.out.println(somme);
}
```

les valeurs 0 et 9 sont passées/affectées à x et y

Récursivité et passage des paramètres

```
int f(int x, int y){
    if(x==y)
        return tab[x];
    return tab[x]+f(x+1,y);
}
public static void main(){
    int somme;
    somme = f(0,9);//tab[0]+tab[1]+...tab[9]
    System.out.println(somme);
}
```

pour calculer `f(0,9)`, on appelle `f(1,9)`, qui appelle `f(2,9)`, etc.

Récursivité et passage des paramètres

```
int f(int x, int y){
    if(x==y)
        return tab[x];
    return tab[x]+f(x+1,y);
}
public static void main(){
    int somme;
    somme = f(0,9);//tab[0]+tab[1]+...tab[9]
    System.out.println(somme);
}
```

D'autres calculs récursif :

- Calculer le factoriel de manière récursive !
- Calculer la suite de Fibonacci de manière récursive !