Chapitre 6 : Programmation dynamique

INF4705 - Analyse et conception d'algorithmes

Gilles Pesant Simon Brockbank

École Polytechnique Montréal
gilles.pesant@polymtl.ca, simon.brockbank@polymtl.ca

Hiver 2017

Plan

Nombre de façons de choisir k objets parmi n

Formulation récursive de ce calcul (relation de récurrence)

Dénotons ce nombre de façons $\binom{n}{k}$
et considérons un objet quelconque :

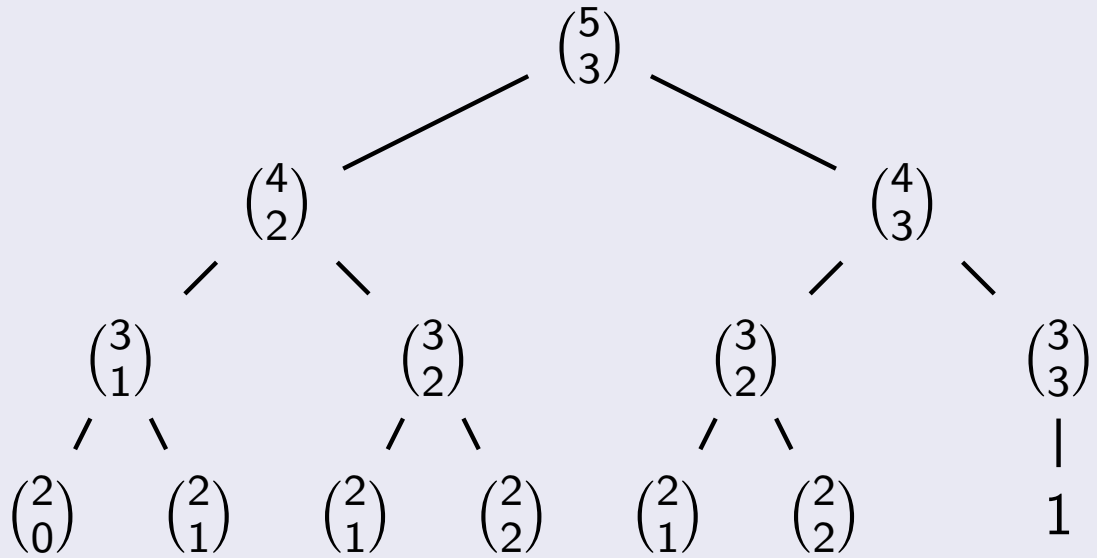
- soit on le choisit, et il reste alors $\binom{n-1}{k-1}$ façons
- soit on ne le choisit pas, et il reste alors $\binom{n-1}{k}$ façons

Ainsi $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

Cas de base : $\binom{n}{0} = \binom{n}{n} = 1$

Nombre de façons de choisir k objets parmi n

Chevauchement des sous-exemplaires



Nombre de façons de choisir k objets parmi n

Conservons nos résultats intermédiaires dans un tableau !

n \ k	0	1	2	3
0				
1				
2				
3		?	?	?
4			?	?
5				?

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Nombre de façons de choisir k objets parmi n

n \ k	0	1	2	3
0	1			
1	1	1		
2	1		1	
3	1			1
4	1			
5	1			

Cas de base

Nombre de façons de choisir k objets parmi n

n \ k	0	1	2	3
0	1			
1	1	1		
2	1		1	
3	1			1
4	1			
5	1			

Cas de base

Remplissons le reste

Nombre de façons de choisir k objets parmi n

n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

Cas de base

Remplissons le reste

Nombre de façons de choisir k objets parmi n

n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

Triangle de Pascal

Cas de base

Remplissons le reste

Nombre de façons de choisir k objets parmi n

Analyse de la consommation de l'espace mémoire

Le tableau contient $\sum_{i=0}^{n}(\min(i,k)+1) \in \Theta(nk)$ cases.

Analyse de la consommation du temps de calcul

On doit remplir chaque case, qui se calcule en $\Theta(1)$:
deux lectures au tableau et une addition.
Donc $\Theta(nk)$ en tout.

Par exemple, si $k = n/2$ on obtient $\Theta(n^2)$.

n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

Nombre de façons de choisir k objets parmi n

Même analyse du temps de calcul pour l'approche DpR

Plusieurs sous-exemplaires sont répétés : on refait donc des calculs inutilement. Si on prend "+1" comme instruction baromètre, on obtient un temps dans $\Theta(\binom{n}{k})$.
Par exemple, si $k = n/2$:

$$\begin{aligned}
\binom{n}{n/2} &= \frac{n!}{((n/2)!)^2} \\
&= \frac{n}{n/2} \times \frac{n-1}{n/2-1} \times \cdots \times \frac{n-(n/2-1)}{1} \\
&\geq (\frac{n}{n/2})^{n/2} \\
&= (\sqrt{2})^n
\end{aligned}$$

Nombre de façons de choisir k objets parmi n

Peut-on économiser de l'espace mémoire ?

n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

Nombre de façons de choisir k objets parmi n

Peut-on économiser de l'espace mémoire ?

Plutôt que tout le tableau, on ne garde que deux lignes à la fois :

1	3	3	1
1			

n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

Nombre de façons de choisir k objets parmi n

Peut-on économiser de l'espace mémoire ?

Encore mieux ! Une seule ligne qu'on met à jour de droite à gauche :

1	3	3	1

Nous n'aurons ainsi plus besoin de $\Theta(nk)$ espace
mais plutôt de $\Theta(k)$ espace.

n \ k	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

Principe général

Principe

On définit un tableau devant contenir la solution de chacun des sous-exemplaires potentiels de l'exemplaire originel. On remplit le tableau dans un certain ordre en utilisant les valeurs déjà présentes, puis on y lit notre réponse.

Remarques

- Un algorithme de *programmation dynamique* procède *de bas en haut* ("bottom-up") alors qu'un algorithme *diviser-pour-régner* procède *de haut en bas* ("top-down").
- Définir un tableau qu'on doit remplir est avant tout une image qui permet de bien saisir la méthode. Fondamentalement il s'agit d'une méthode de calcul d'une récurrence.

Patron de conception de la programmation dynamique

fonction programmation_dynamique(x {exemplaire}) : {solution}
définir un tableau T à d dimensions ;
initialiser les valeurs frontières dans T ;
pour $i_1 = deb_1$ **à** fin_1 **faire**
⋮
pour $i_d = deb_d$ **à** fin_d **faire**
$T[i_1, \ldots, i_d] \leftarrow$ expression utilisant des cases déjà calculées et x ;
retourner expression utilisant des cases de T ;

Comment utiliser la programmation dynamique ?

Cinq étapes

1. Définition du tableau : quels sont les sous-exemplaires d'intérêt ?
2. Définition de la relation de récurrence.
3. Établissement des valeurs frontières.
4. Remplissage du tableau, dans un ordre établi d'après la récurrence.
5. Récupération de la solution, à partir de certaines cases du tableau.

Quand utiliser la programmation dynamique ?

Chevauchement des sous-exemplaires

Une approche récursive résolverait alors de nombreuses fois le même sous-exemplaire. La programmation dynamique, elle, ne le résoudra qu'une seule fois et le stockera dans un tableau, ce qui est beaucoup plus efficace.

C'est une condition souhaitable afin de gagner en efficacité.

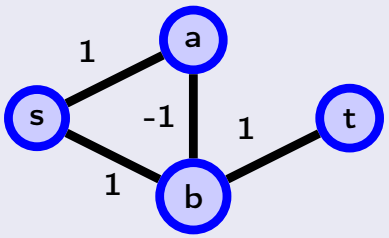
Sous-structure optimale ("Principe d'optimalité")

La solution optimale à un exemplaire est la combinaison des solutions optimales de certains de ses sous-exemplaires. Autrement dit, dans une suite optimale de décisions ou de choix, chaque sous-suite doit aussi être optimale.

C'est une condition nécessaire.

Ce Principe d'optimalité ne s'applique pas toujours ?

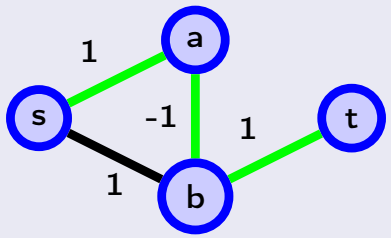
Ex. Plus court chemin *simple* avec arêtes de longueur possiblement négative



Ce Principe d'optimalité ne s'applique pas toujours ?

Ex. Plus court chemin *simple* avec arêtes de longueur possiblement négative

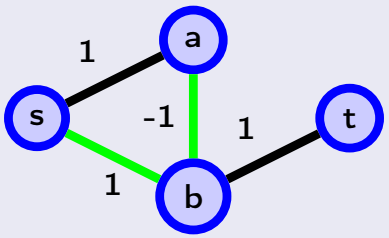
$pccs(s, t) = 1$

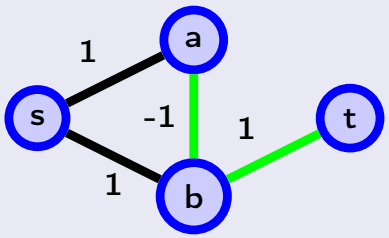


Ce Principe d'optimalité ne s'applique pas toujours ?

Ex. Plus court chemin *simple* avec arêtes de longueur possiblement négative

$pccs(s, a) = 0$
$+$
$pccs(a, t) = 0$





Le plus court chemin simple de s à t ne peut pas être la combinaison des plus courts chemins simples de s à a et de a à t.

Le Principe d'optimalité ne s'applique donc pas ici.

Alternative : Fonction à mémoire (mémoïzation)

Patron

fonction $f(x_1, \ldots, x_d)$
  si $T[x_1, \ldots, x_d] \neq$ la valeur d'initialisation
    alors retourner $T[x_1, \ldots, x_d]$;
  $s \leftarrow$ calcul récursif de $f(x_1, \ldots, x_d)$;
  $T[x_1, \ldots, x_d] \leftarrow s$; {on note la valeur de la solution}
  retourner s ;

Pour :

- On ne calcule pas inutilement certaines cases du tableau.

Contre :

- On perd la possibilité de parfois réduire la taille du tableau.
- En pratique, la récursivité est plus lourde.

Plan

Faire de la monnaie

Rappel

Minimiser le nombre de pièces utilisées pour totaliser un montant donné.

- Un algorithme glouton peut le faire pour certaines combinaisons de valeurs de pièces disponibles seulement.
- Ici nous y arriverons peu importe la valeur des pièces.

Principe d'optimalité (sous-structure optimale) ?

Rappel ("Prouver l'optimalité d'un algorithme glouton")

Soit une solution optimale $\langle p_1, \ldots, p_k, p_{k+1}, \ldots, p_n \rangle$.

Considérons $\langle p_1, \ldots, p_k \rangle$ et $\langle p_{k+1}, \ldots, p_n \rangle$: chacune doit être une solution optimale pour la sous-somme correspondante, sinon on obtiendrait une meilleure solution en leur substituant une (sous-) solution utilisant moins de pièces. □

Étape 1 : Définition du tableau

Étant donné n types de pièces $1, \ldots, n$ de valeur $d_1, \ldots, d_n$ et une somme N à rendre...

Étape 1 : Définition du tableau

Étant donné n types de pièces $1, \ldots, n$ de valeur $d_1, \ldots, d_n$ et une somme N à rendre...
Définissons un tableau $c[1 \ldots N]$ où $c[j]$ est le nombre minimum de pièces requises pour totaliser un montant j.

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :								

Étape 2 : Définition de la récurrence

Étape 2 : Définition de la récurrence

Soit on sépare en deux sous-montants 1 et $j - 1$

Étape 2 : Définition de la récurrence

Soit on sépare en deux sous-montants 1 et $j-1$
ou encore en deux sous-montants 2 et $j-2$

Étape 2 : Définition de la récurrence

Soit on sépare en deux sous-montants 1 et $j-1$
ou encore en deux sous-montants 2 et $j-2$
ou encore 3 et $j-3$
$\vdots$

Étape 2 : Définition de la récurrence

Relation de récurrence

$$c[j] = \begin{cases} 1 & \text{, si } j = d_i \text{ pour un certain } i \\ \min_{i=1}^{\lfloor j/2 \rfloor}(c[i] + c[j-i]) & \text{, sinon} \end{cases}$$

Soit on sépare en deux sous-montants 1 et $j-1$
ou encore en deux sous-montants 2 et $j-2$
ou encore 3 et $j-3$
$\vdots$

Étape 3 : Les valeurs frontières

Valeurs frontières ? $c[d_i] = 1, \quad \forall\, 1 \leq i \leq n$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1			1		1		

Étape 4 : Remplissage du tableau

Dans quel ordre ?
Selon la récurrence, on consulte des cases à gauche de la case courante.
Donc, on remplira le tableau de gauche à droite.

$$c[j] = \min_{i=1}^{\lfloor j/2 \rfloor} (c[i] + c[j-i])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	

Étape 4 : Remplissage du tableau

Dans quel ordre ?
Selon la récurrence, on consulte des cases à gauche de la case courante.
Donc, on remplira le tableau de gauche à droite.

$$c[j] = \min_{i=1}^{\lfloor j/2 \rfloor} (c[i] + c[j-i])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	

Étape 4 : Remplissage du tableau

Dans quel ordre ?
Selon la récurrence, on consulte des cases à gauche de la case courante.
Donc, on remplira le tableau de gauche à droite.

$$c[j] = \min_{i=1}^{\lfloor j/2 \rfloor} (c[i] + c[j-i])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	

Étape 4 : Remplissage du tableau

Dans quel ordre ?
Selon la récurrence, on consulte des cases à gauche de la case courante.
Donc, on remplira le tableau de gauche à droite.

$$c[j] = \min_{i=1}^{\lfloor j/2 \rfloor} (c[i] + c[j-i])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	

Étape 4 : Remplissage du tableau

Dans quel ordre ?
Selon la récurrence, on consulte des cases à gauche de la case courante.
Donc, on remplira le tableau de gauche à droite.

$$c[j] = \min_{i=1}^{\lfloor j/2 \rfloor} (c[i] + c[j-i])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	

Étape 4 : Remplissage du tableau

Dans quel ordre ?
Selon la récurrence, on consulte des cases à gauche de la case courante.
Donc, on remplira le tableau de gauche à droite.

$$c[j] = \min_{i=1}^{\lfloor j/2 \rfloor} (c[i] + c[j-i])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	2

Étape 5 : Récupération de la réponse

Où est notre réponse ?

À la case $c[N]$.

Et pour retrouver les pièces qu'il faut donner ?

- On retrace nos pas à partir de la case réponse, identifiant laquelle des alternatives nous avons utilisée à chaque étape.
- On crée un autre tableau qui mémorise l'index i qui minimise l'expression (négatif si c'est une valeur frontière).

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	1	2	3	4	5	6	7	8
nb de pièces :	1	2	3	1	2	1	2	2
index :	-1	1	1	-2	1	-3	1	4

Algorithme

fonction faire_monnaie($N, n, \langle d_1, \ldots, d_n \rangle$)
    définir $c[1 \ldots N]$ et $I[1 \ldots N]$;
    pour $j = 1$ **à** N **faire**
        $c[j] \leftarrow \infty$;
    pour $i = 1$ **à** n **faire**
        $c[d_i] \leftarrow 1$;
        $I[d_i] \leftarrow -i$;
    pour $j = 1$ **à** N **faire**
        si $c[j] \neq 1$ **alors**
            pour $i = 1$ **à** $\lfloor j/2 \rfloor$ **faire**
                si $c[j] > c[i] + c[j-i]$ **alors**
                    $c[j] \leftarrow c[i] + c[j-i]$;
                    $I[j] \leftarrow i$;
    retourner $c[N]$ et I ;

Analyse de la consommation de ressources

fonction faire_monnaie($N, n, \langle d_1, \ldots, d_n \rangle$)
    définir $c[1 \ldots N]$ et $I[1 \ldots N]$;
    pour $j = 1$ **à** N **faire**
        $c[j] \leftarrow \infty$;
    pour $i = 1$ **à** n **faire**
        $c[d_i] \leftarrow 1$;
        $I[d_i] \leftarrow -i$;
    pour $j = 1$ **à** N **faire**
        si $c[j] \neq 1$ **alors**
            pour $i = 1$ **à** $\lfloor j/2 \rfloor$ **faire**
                si $c[j] > c[i] + c[j-i]$ **alors**
                    $c[j] \leftarrow c[i] + c[j-i]$;
                    $I[j] \leftarrow i$;
    retourner $c[N]$ et I ;

Polynomial ?

Non, car la consommation de l'algorithme dépend de la *magnitude* des nombres en entrée (ici, N dans la consommation $\Theta(n + N^2)$) :
elle est polynomiale dans la *valeur* de l'exemplaire,
mais exponentielle dans la *taille* de l'exemplaire
(nombre de bits ou de chiffres décimaux pour le représenter).

On dit qu'il s'agit d'un algorithme pseudo-polynomial.

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

	1	2	3	4	5	6	7	8

Étape 1 : Définition du tableau

Étant donné n types de pièces $1, \ldots, n$ de valeur $d_1, \ldots, d_n$ et une somme N à rendre...

Étape 1 : Définition du tableau

Étant donné n types de pièces $1, \ldots, n$ de valeur $d_1, \ldots, d_n$ et une somme N à rendre...
Définissons un tableau $c[1 \ldots n, 0 \ldots N]$ où $c[i, j]$ est le nombre minimum de pièces requises pour totaliser un montant j en n'utilisant que des pièces parmi les i premières.

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$									
$d_2 = 4$									
$d_3 = 6$									

Étape 2 : Définition de la récurrence

Étape 2 : Définition de la récurrence

Soit on utilise une pièce de type i : $1 + c[i, j - d_i]$

Étape 2 : Définition de la récurrence

Soit on utilise une pièce de type i : $1 + c[i, j - d_i]$

Soit on n'en utilise pas : $c[i - 1, j]$

Étape 2 : Définition de la récurrence

Relation de récurrence

$$c[i,j] = \min(1 + c[i, j - d_i],\ c[i-1, j])$$

Soit on utilise une pièce de type i : $1 + c[i, j - d_i]$

Soit on n'en utilise pas : $c[i-1, j]$

Étape 3 : Les valeurs frontières

Valeurs frontières ? $c[i, 0] = 0$

Mais aussi...

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0								
$d_3 = 6$	0								

Étape 3 : Les valeurs frontières

Valeurs frontières ? $c[i, 0] = 0$

Mais aussi...

- aucune pièce disponible : $c[0, j] = \infty$
- somme négative : $c[i, j] = \infty, \quad j < 0$

$$c[i, j] = \min(1 + c[i, j - d_i],\ c[i - 1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0								
$d_3 = 6$	0								

Étape 3 : Les valeurs frontières

Valeurs frontières ? $c[i, 0] = 0$

Mais aussi...

- aucune pièce disponible : $c[0, j] = \infty$
- somme négative : $c[i, j] = \infty, \quad j < 0$

$$c[i, j] = \min(1 + c[i, j - d_i],\ c[i - 1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0								
$d_3 = 6$	0								

Étape 4 : Remplissage du tableau

Dans quel ordre ? Consultons la récurrence :

$$c[i,j] = \min(1 + c[i, j - d_i],\ c[i-1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0					?			
$d_3 = 6$	0								

Étape 4 : Remplissage du tableau

Dans quel ordre ? Consultons la récurrence :

$$c[i,j] = \min(1 + c[i, j - d_i],\ c[i-1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0					?			
$d_3 = 6$	0								

Donc : ligne par ligne, ou encore colonne par colonne, ou encore...

Pouvons-nous réduire la taille du tableau ?

$$c[i,j] = \min(1 + c[i, j - d_i],\ c[i-1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0					?			
$d_3 = 6$	0								

Pouvons-nous réduire la taille du tableau ?

$$c[i,j] = \min(1 + c[i, j - d_i],\ c[i-1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0					?			
$d_3 = 6$	0								

- On pourrait ne garder qu'une seule ligne, qu'on met à jour de gauche à droite.

Pouvons-nous réduire la taille du tableau ?

$$c[i,j] = \min(1 + c[i, j - d_i],\ c[i-1, j])$$

Exemple ($N = 8,\ n = 3,\ d_1 = 1, d_2 = 4, d_3 = 6$)

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0								
$d_2 = 4$	0					?			
$d_3 = 6$	0								

- On pourrait ne garder qu'une seule ligne, qu'on met à jour de gauche à droite.
- Mais pour pouvoir retrouver les pièces, on conservera tout le tableau.

Étape 5 : Récupération de la réponse

Où est notre réponse ?

À la case $c[n, N]$.

Et pour retrouver les pièces qu'il faut donner ?

On retrace nos pas à partir de la case réponse, identifiant à chaque étape laquelle des deux alternatives nous avons utilisée.

montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0	1	2	3	4	5	6	7	8
$d_2 = 4$	0	1	2	3	1	2	3	4	2
$d_3 = 6$	0	1	2	3	1	2	1	2	2

Analyse de la consommation de ressources

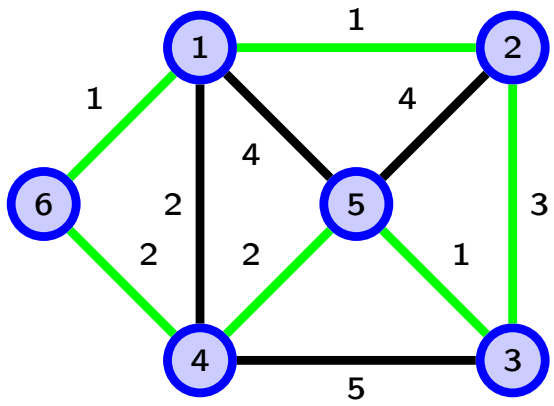
montant :	0	1	2	3	4	5	6	7	8
$d_1 = 1$	0	1	2	3	4	5	6	7	8
$d_2 = 4$	0	1	2	3	1	2	3	4	2
$d_3 = 6$	0	1	2	3	1	2	1	2	2

On a un tableau de taille $n \times (N + 1)$ à remplir
et deux alternatives à considérer pour chaque case.

Plan

Voyageur de commerce

Rappel : Tournée hamiltonienne la plus courte



- Identifions les sommets par les entiers $\{1, 2, \ldots, n\}$.
- Sans perte de généralité, notre tournée débutera à 1.
- Dénotons la distance du sommet i au sommet j par d_{ij}.

Étape 1 : Définition du tableau

Étape 1 : Définition du tableau

Définissons un tableau D où $D[i, S]$ représente la longueur d'un plus court chemin partant de i, passant par tous les sommets de S et se terminant à 1.

Exemple ($n = 4$)

i	S	$\{\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
2								
3								
4								

Ce tableau a...

Étape 1 : Définition du tableau

Définissons un tableau D où $D[i, S]$ représente la longueur d'un plus court chemin partant de i, passant par tous les sommets de S et se terminant à 1.

Exemple ($n = 4$)

i	S	$\{\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
2								
3								
4								

Ce tableau a... $n-1$ rangées et...

Étape 1 : Définition du tableau

Définissons un tableau D où $D[i, S]$ représente la longueur d'un plus court chemin partant de i, passant par tous les sommets de S et se terminant à 1.

Exemple ($n = 4$)

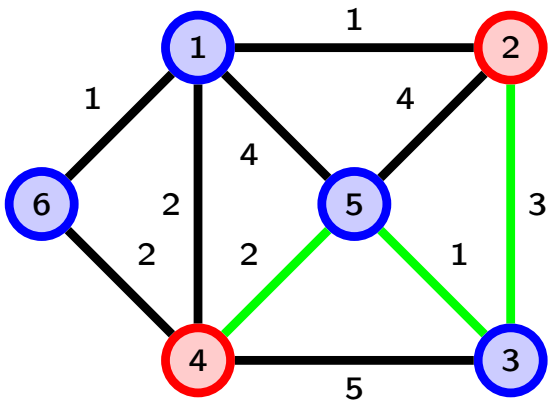
i	S	$\{\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
2								
3								
4								

Ce tableau a... $n-1$ rangées et... $2^{n-1}-1$ colonnes

Voyageur de commerce

Principe d'optimalité ?

Il s'applique :
une tournée la plus courte sera composée de chemins les plus courts entre une origine et une destination donnée et passant par un sous-ensemble donné des sommets.
S'il existait une façon plus courte d'enchaîner ces sommets, on l'utiliserait dans notre tournée.



Étape 2 : Définition de la récurrence

Relation de récurrence

$$D[i, S] =$$

Étape 2 : Définition de la récurrence

Relation de récurrence

$$D[i,S] = \begin{cases} \min_{j \in S}(d_{ij} + D[j, S \setminus \{j\}]) & \text{, si } S \neq \emptyset \\ d_{i1} & \text{, si } S = \emptyset \end{cases}$$

Étape 3 : Les valeurs frontières

Valeurs frontières ? $D[i, \{\}] = d_{i1}, \quad \forall\, 2 \leq i \leq n$

Exemple ($n = 4$)

i	S	$\{\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
2		d_{21}						
3		d_{31}						
4		d_{41}						

Étape 4 : Remplissage du tableau

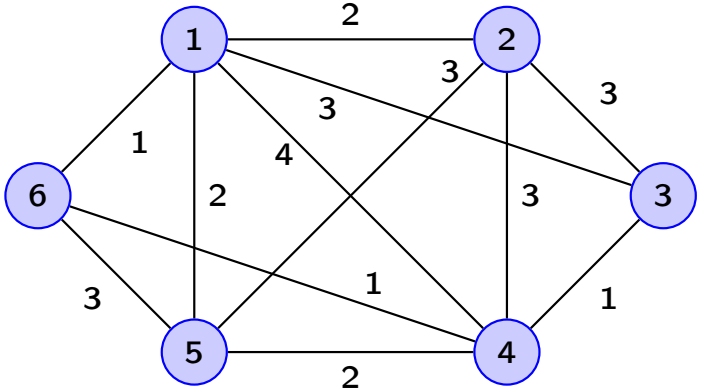
Dans quel ordre ?

$$D[i, S] = \min_{j \in S}(d_{ij} + D[j, S \setminus \{j\}])$$

Exemple ($n = 4$)

i	S	$\{\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
2		d_{21}						
3		d_{31}						
4		d_{41}						

Étape 4 : Remplissage du tableau



$$D[i, S] = \min_{j \in S}(d_{ij} + D[j, S \setminus \{j\}])$$

Exemple

i	S	$\{\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$	$\{6\}$
2		2	-	6	7	5	∞
3		3	5	-	5	∞	∞
4		4	5	4	-	4	2
5		2	5	∞	6	-	4
6		1	∞	∞	5	5	-

Étape 4 : Remplissage du tableau

Exemple

i	S	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$	$\{6\}$
2		-	6	7	5	∞
3		5	-	5	∞	∞
4		5	4	-	4	2
5		5	∞	6	-	4
6		∞	∞	5	5	-

i	S	$\{2,3\}$	$\{2,4\}$	$\{2,5\}$	$\{2,6\}$	$\{3,4\}$	$\{3,5\}$	$\{3,6\}$	$\{4,5\}$	$\{4,6\}$	$\{5,6\}$
2		-	-	-	-	7	∞	∞	7	5	7
3		-	6	8	∞	-	-	-	5	3	∞
4		6	-	7	∞	-	∞	∞	-	-	6
5		9	7	-	∞	6	-	-	-	4	-
6		∞	6	8	-	5	∞	-	5	-	-

Étape 4 : Remplissage du tableau

Exemple

i	S	$\{2,3\}$	$\{2,4\}$	$\{2,5\}$	$\{2,6\}$	$\{3,4\}$	$\{3,5\}$	$\{3,6\}$	$\{4,5\}$	$\{4,6\}$	$\{5,6\}$
2		-	-	-	-	7	∞	∞	7	5	7
3		-	6	8	∞	-	-	-	5	3	∞
4		6	-	7	∞	-	∞	∞	-	-	6
5		9	7	-	∞	6	-	-	-	4	-
6		∞	6	8	-	5	∞	-	5	-	-

i	S	$\{2,3,4\}$	$\{2,3,5\}$	$\{2,3,6\}$	$\{2,4,5\}$	$\{2,4,6\}$	$\{2,5,6\}$	$\{3,4,5\}$	$\{3,4,6\}$	$\{3,5,6\}$	$\{4,5,6\}$
2		-	-	-	-	-	-	8	6	∞	9
3		-	-	-	8	8	10	-	-	-	7
4		-	9	∞	-	-	10	-	-	∞	-
5		8	-	∞	-	8	-	-	8	-	-
6		7	12	-	8	-	-	9	-	-	-

Étape 4 : Remplissage du tableau

Exemple

i	S	$\{2,3,4\}$	$\{2,3,5\}$	$\{2,3,6\}$	$\{2,4,5\}$	$\{2,4,6\}$	$\{2,5,6\}$	$\{3,4,5\}$	$\{3,4,6\}$	$\{3,5,6\}$	$\{4,5,6\}$
2		-	-	-	-	-	-	*8*	*6*	∞	*9*
3		-	-	-	*8*	*8*	*10*	-	-	-	*7*
4		-	*9*	∞	-	-	*10*	-	-	∞	-
5		*8*	-	∞	-	*8*	-	-	*8*	-	-
6		*7*	*12*	-	*8*	-	-	*9*	-	-	-

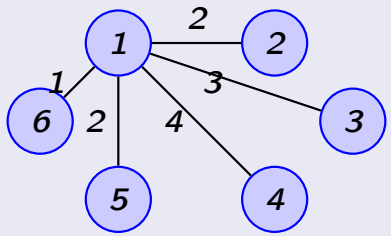
i	S	$\{2,3,4,5\}$	$\{2,3,4,6\}$	$\{2,3,5,6\}$	$\{2,4,5,6\}$	$\{3,4,5,6\}$
2		-	-	-	-	*10*
3		-	-	-	*11*	-
4		-	-	*11*	-	-
5		-	*9*	-	-	-
6		*10*	-	-	-	-

Étape 5 : Récupération de la réponse

Où est notre réponse ? $D[1, \{2, 3, \ldots, n\}]$.

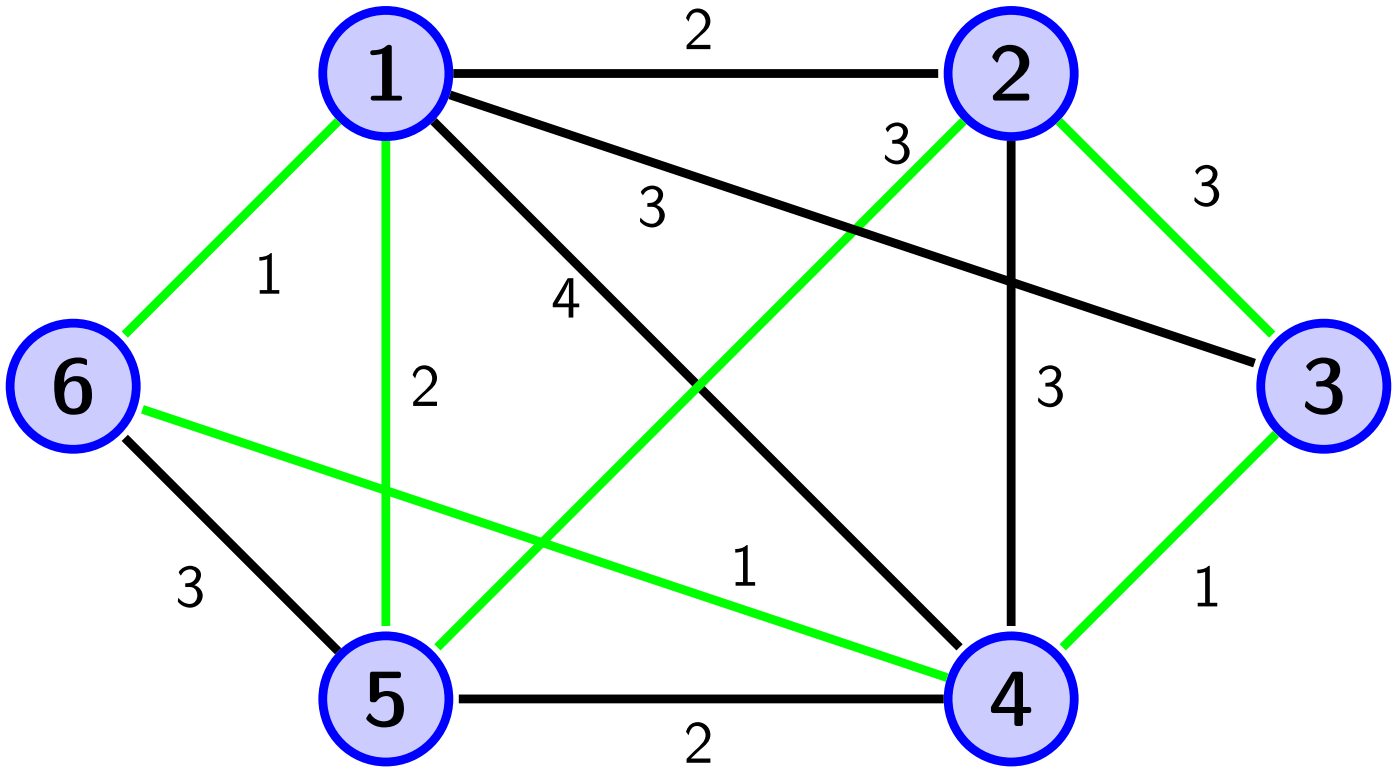
Exemple

i S	$\{2,3,4,5\}$	$\{2,3,4,6\}$	$\{2,3,5,6\}$	$\{2,4,5,6\}$	$\{3,4,5,6\}$
2	-	-	-	-	*10*
3	-	-	-	*11*	-
4	-	-	*11*	-	-
5	-	*9*	-	-	-
6	*10*	-	-	-	-



D[1,{2,3,4,5,6}] =
min{ d_{12} *+ D[2,{3,4,5,6}],* d_{13} *+ D[3,{2,4,5,6}],* d_{14} *+ D[4,{2,3,5,6}],* d_{15} *+ D[5,{2,3,4,6}],* d_{16} *+ D[6,{2,3,4,5}]},*
= min {12, 14, 15, 11, 11} = 11

Étape 5 : Récupération de la réponse



Analyse de la consommation de ressource

Relation de récurrence

$$D[i,S] = \begin{cases} \min_{j \in S}(d_{ij} + D[j, S \setminus \{j\}]) & \text{, si } S \neq \emptyset \\ d_{i1} & \text{, si } S = \emptyset \end{cases}$$

Plan

Plus courts chemins

Problème

Soit $G = (N, A)$ un graphe orienté avec $N = \{1, \ldots, n\}$ et dont les arcs ont une longueur non négative.
Quelle est la longueur du plus court chemin entre chaque paire de sommets ?

Étape 1 : Définition du tableau

Définissons un tableau $D[1 \ldots n, 1 \ldots n, 0 \ldots n]$ où $D[i, j, k]$ est la longueur du plus court chemin de i à j dont les sommets intermédiaires sont parmi $1, \ldots, k$.

Étape 2 : Définition de la récurrence

Étape 2 : Définition de la récurrence

Soit on passe par k : $D[i, j, k] = D[i, k, k-1] + D[k, j, k-1]$

Étape 2 : Définition de la récurrence

Soit on passe par k : $D[i,j,k] = D[i,k,k-1] + D[k,j,k-1]$

Soit on n'y passe pas : $D[i,j,k] = D[i,j,k-1]$

Étape 2 : Définition de la récurrence

Relation de récurrence

$$D[i,j,k] = \min(D[i,k,k-1] + D[k,j,k-1],\ D[i,j,k-1])$$

Soit on passe par k : $D[i,j,k] = D[i,k,k-1] + D[k,j,k-1]$

Soit on n'y passe pas : $D[i,j,k] = D[i,j,k-1]$

Plus courts chemins

Principe d'optimalité ?

Il s'applique :
si $\alpha \cdot \beta$ (où $\alpha = \langle i, \ldots, k \rangle$ et $\beta = \langle k, \ldots, j \rangle$) est un plus court chemin de i à j alors α et β doivent aussi être les plus courts.

Étape 3 : Les valeurs frontières

Valeurs frontières ? $D[i, j, 0] =$ longueur de l'arc (i, j)

Étape 4 : Remplissage du tableau

Dans quel ordre ?

$$D[i,j,k] = \min(D[i,k,k-1] + D[k,j,k-1],\ D[i,j,k-1])$$

Étape 4 : Remplissage du tableau

Dans quel ordre ?

$$D[i,j,k] = \min(D[i,k,k-1] + D[k,j,k-1],\ D[i,j,k-1])$$

Selon k croissant.

Pouvons-nous réduire la taille du tableau ?

$$D[i,j,k] = \min(D[i,k,k-1] + D[k,j,k-1],\ D[i,j,k-1])$$

Puisque le calcul d'une case $D[\cdot,\cdot,k]$ ne sollicite que des cases $D[\cdot,\cdot,k-1]$, on pourrait utiliser un tableau de dimensions $n \times n \times 2$.

On pourrait même éliminer la 3^e dimension (k) au complet à condition de garantir que $D[i,k,k]$ et $D[k,j,k]$ sont mis à jour après $D[i,j,k]$.

Mais les $k^{\text{ième}}$ ligne et colonne ne changeront pas durant cette itération : $D[i,k,k] = D[i,k,k-1]$ et $D[k,j,k] = D[k,j,k-1]$.

On peut donc passer à un tableau $n \times n$.

Étape 5 : Récupération de la réponse

Où est notre réponse ?

Aux cases $D[\star, \star]$ à la fin de l'algorithme
($D[\star, \star, n]$ dans l'ancien tableau)

Et pour retrouver ces plus courts chemins ?

On maintient un autre tableau I de mêmes dimensions qui contient, pour chaque paire de sommets $\langle i, j \rangle$, la dernière itération à laquelle $D[i, j]$ a été modifiée :

- si $I[i, j] = \ell > 0$ alors ce chemin passe par ℓ et on consultera les cases $I[i, \ell]$ et $I[\ell, j]$ pour la suite ;
- si $I[i, j] = 0$ alors c'est l'arc (i, j).

Algorithme de Floyd

fonction Floyd($L[1..n, 1..n]$:longueurs) : deux tableaux $[1..n, 1..n]$
    tableaux $D[1..n, 1..n]$ et $I[1..n, 1..n]$;
    $D \leftarrow L$;
    $I \leftarrow 0$;
    pour $k = 1$ **à** n **faire**
        pour $i = 1$ **à** n **faire**
            pour $j = 1$ **à** n **faire**
                si $D[i,j] > D[i,k] + D[k,j])$ **alors**
                $D[i,j] \leftarrow D[i,k] + D[k,j]$;
                $I[i,j] \leftarrow k$;
    retourner D et I ;

Analyse de la consommation de ressources

fonction Floyd($L[1..n, 1..n]$:longueurs) : deux tableaux $[1..n, 1..n]$
    tableaux $D[1..n, 1..n]$ et $I[1..n, 1..n]$;
    $D \leftarrow L$;
    $I \leftarrow 0$;
    pour $k = 1$ **à** n **faire**
        pour $i = 1$ **à** n **faire**
            pour $j = 1$ **à** n **faire**
                si $D[i,j] > D[i,k] + D[k,j])$ **alors**
                $D[i,j] \leftarrow D[i,k] + D[k,j]$;
                $I[i,j] \leftarrow k$;
    retourner D et I ;