Algorithmique 2 (INFO-F203)
Plus Courts Chemins dans les Graphes Acycliques

Jean Cardinal

Avril 2021

Plus courts chemins : optimalité

Proposition 1

Soit $G = (\mathcal{V}, \mathcal{E})$ un graphe dirigé pondéré positivement sur les arcs, $s \in \mathcal{V}$ un sommet particulier, et une fonction $d : \mathcal{V} \to \mathbb{R}$. La valeur de $d(v)$ est la longueur du plus court chemin de s vers v si et seulement si les trois conditions suivantes sont satisfaites :

1. $d(s) = 0$,
2. pour tout sommet v, $d(v)$ est la longueur d'un chemin de s vers v,
3. pour tout arc $e = (u, v)$, $d(v) \leq d(u) + w(e)$.

Démonstration

Nous devons démontrer les implications dans les deux directions.
($\Rightarrow$) Supposons d'abord que la fonction d représente effectivement les longueurs des plus courts chemins de s vers chacun des sommets. Par contradiction, supposons de plus qu'il existe un arc $e = (u, v)$ telle que $d(v) > d(u) + w(e)$. Il existe donc un chemin de s vers v qui passe par u et dont la longueur est inférieure à $d(v)$, une contradiction avec notre hypothèse.

Démonstration

($\Leftarrow$) Supposons maintenant que les trois conditions sont satisfaites. Considérons un plus court chemin de s vers v de la forme $(s = v_0, v_1, v_2, \ldots, v_k = v)$. On note $e_i = (v_{i-1}, v_i)$, $i = 1, 2, \ldots, k$ les arcs successifs du chemin. Par la troisième condition, on a :

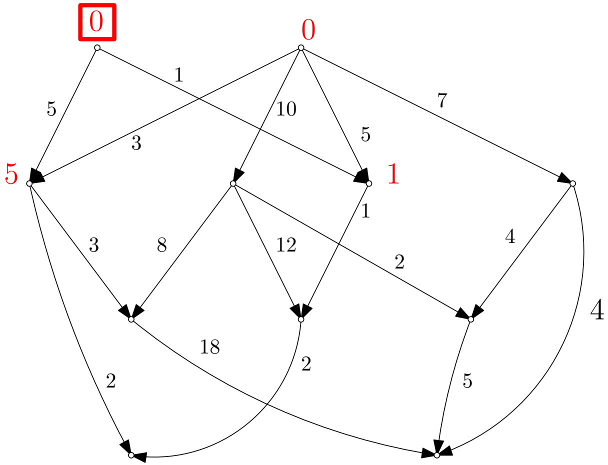
$$\begin{array}{rcl} d(v_1) & \leq & d(s) + w(e_1) \\ d(v_2) & \leq & d(v_1) + w(e_2) \\ & \ldots & \\ d(v_k) & \leq & d(v_{k-1}) + w(e_k). \end{array}$$

Par substitutions successives et en utilisant $d(s) = 0$:

$$d(v) = d(v_k) \leq w(e_1) + w(e_2) + \ldots + w(e_k),$$

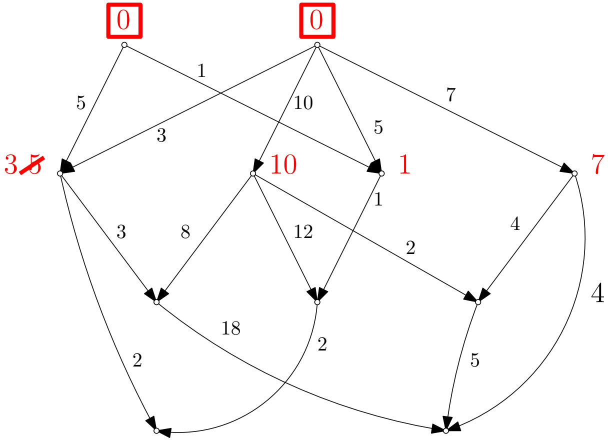
qui est bien la longueur du plus court chemin de s vers v. Par la deuxième condition, cette dernière inégalité doit être une égalité, et donc $d(v)$ est bien la longueur du plus court chemin de s vers v.

Dijkstra : Exemple sur un graphe acyclique



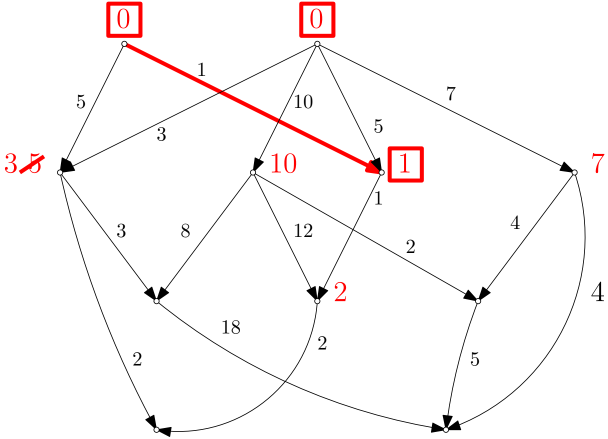
•

Dijkstra : Exemple sur un graphe acyclique



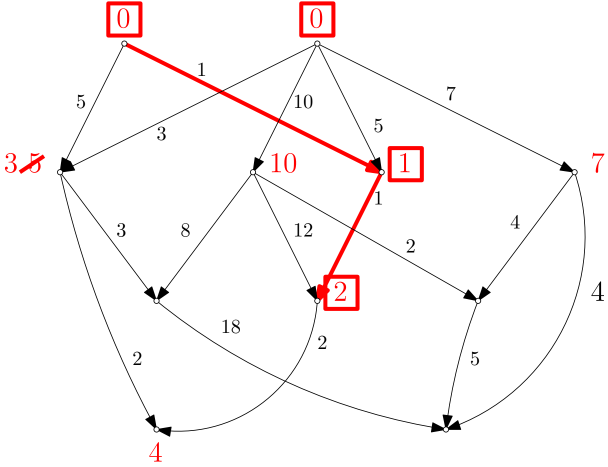
•

Dijkstra : Exemple sur un graphe acyclique



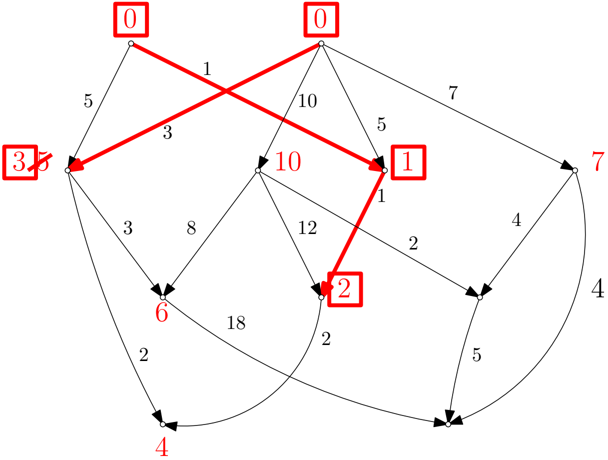
•

Dijkstra : Exemple sur un graphe acyclique



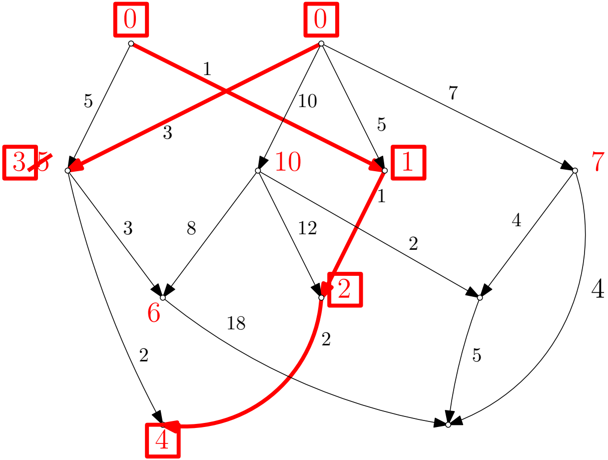
•

Dijkstra : Exemple sur un graphe acyclique



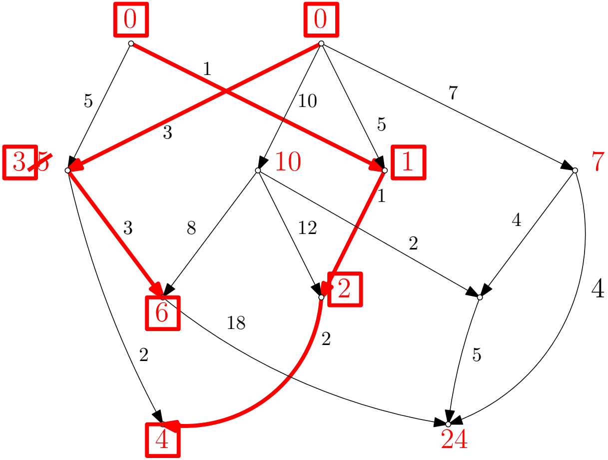
-

Dijkstra : Exemple sur un graphe acyclique



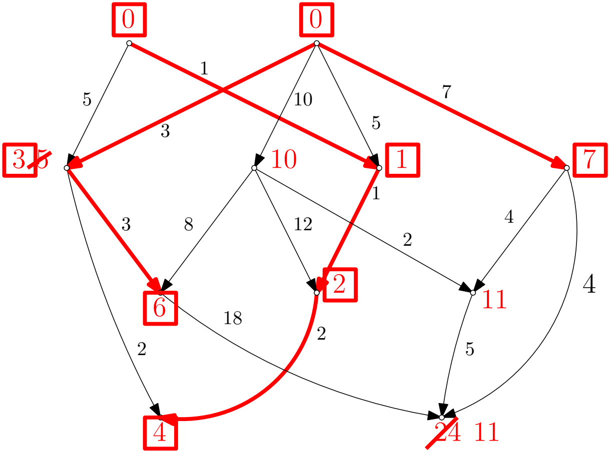
-

Dijkstra : Exemple sur un graphe acyclique



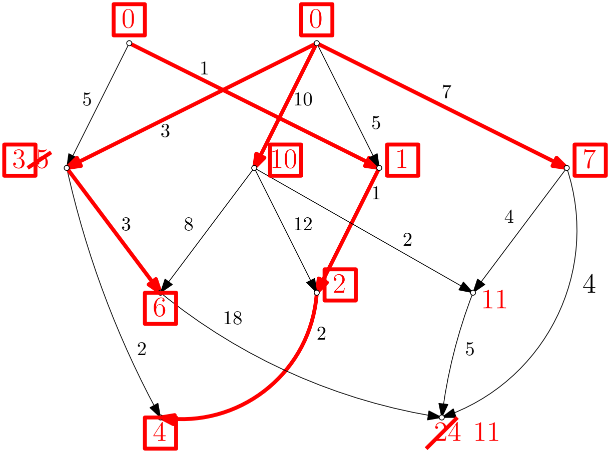
-

Dijkstra : Exemple sur un graphe acyclique



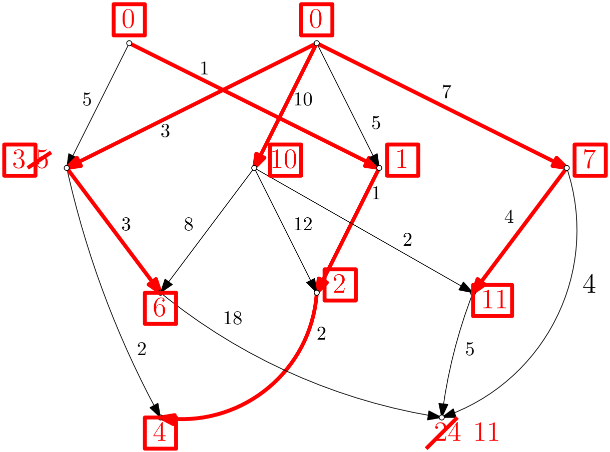
-

Dijkstra : Exemple sur un graphe acyclique



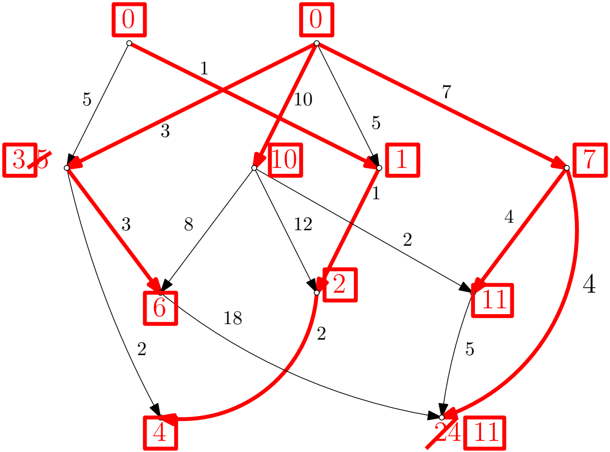
-

Dijkstra : Exemple sur un graphe acyclique



-

Dijkstra : Exemple sur un graphe acyclique



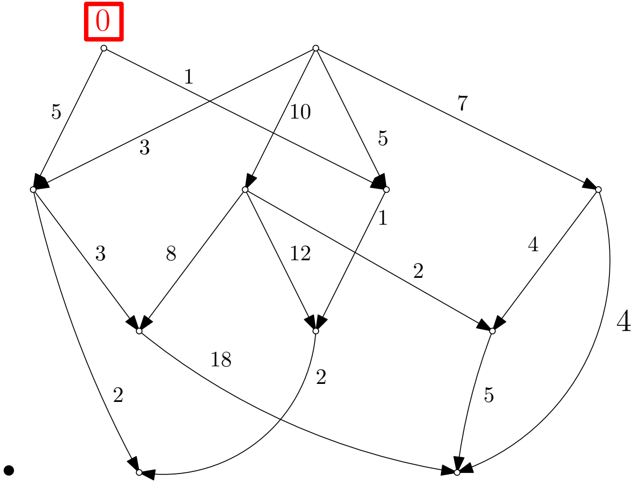
-

Algorithme pour les graphes acycliques

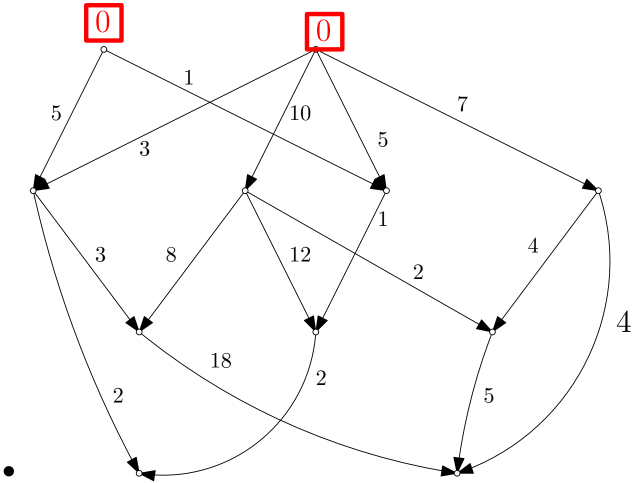
- Examiner les sommets dans un ordre topologique,
- pour chaque sommet v, on assigne la valeur

$$d(v) \leftarrow \min_{e=(u,v)\in\mathcal{E}} \{d(u) + w(e)\}$$

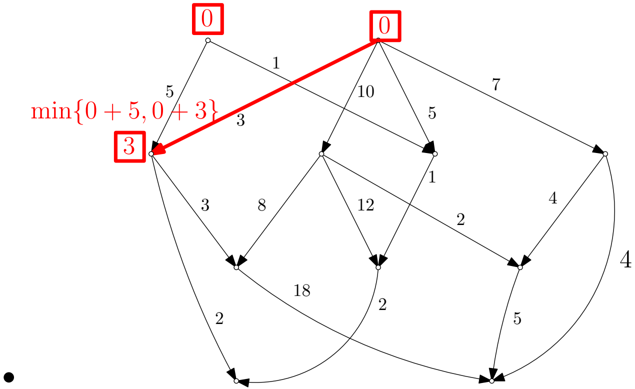
Exemple



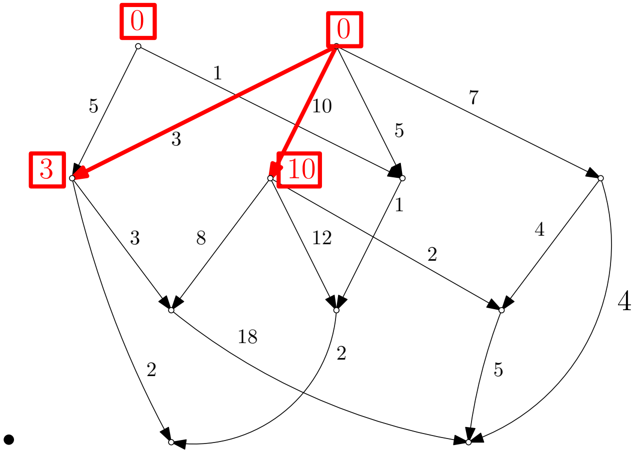
Exemple



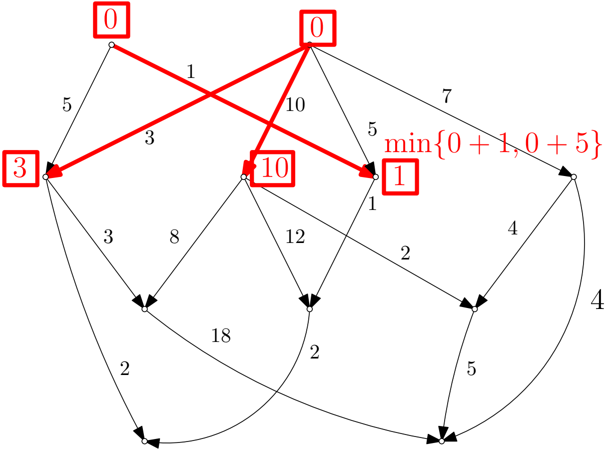
Exemple



Exemple

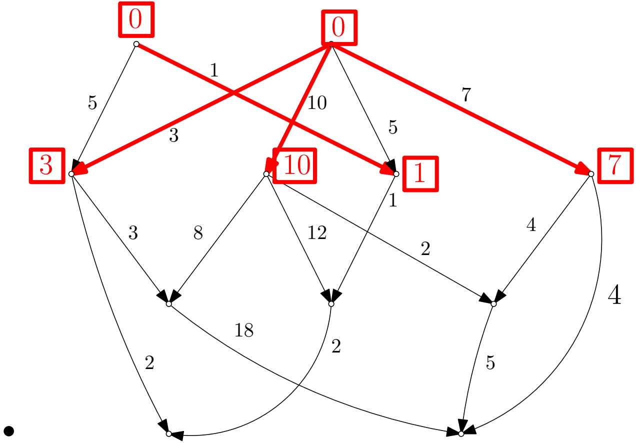


Exemple

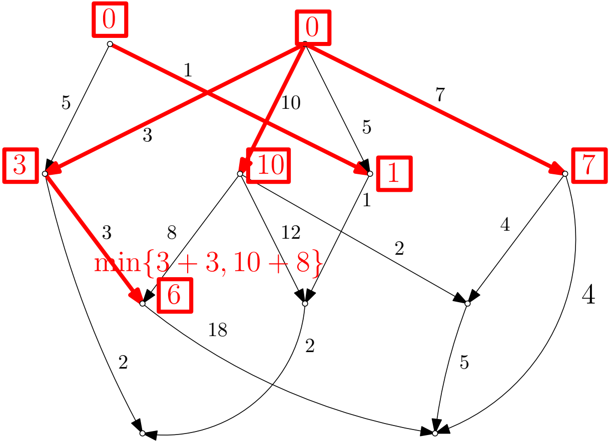


-

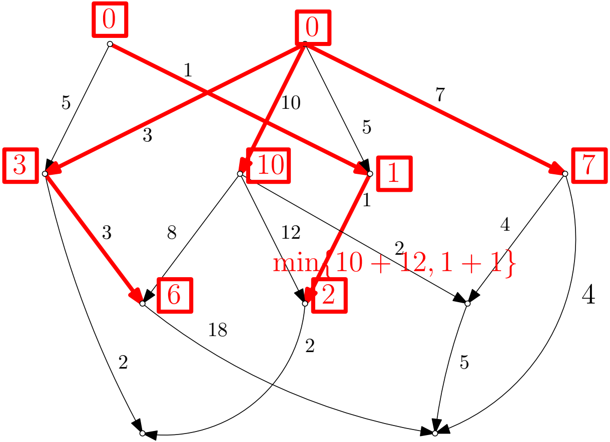
Exemple



Exemple

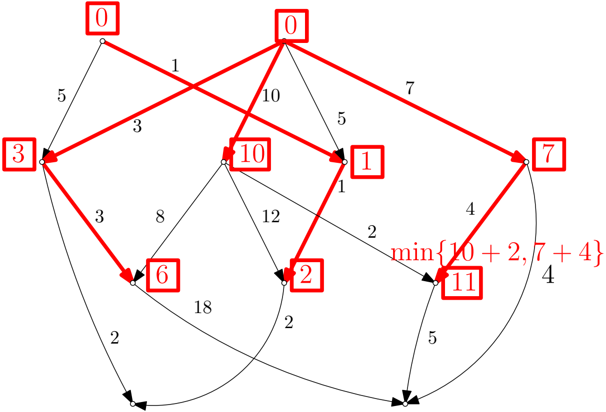


Exemple

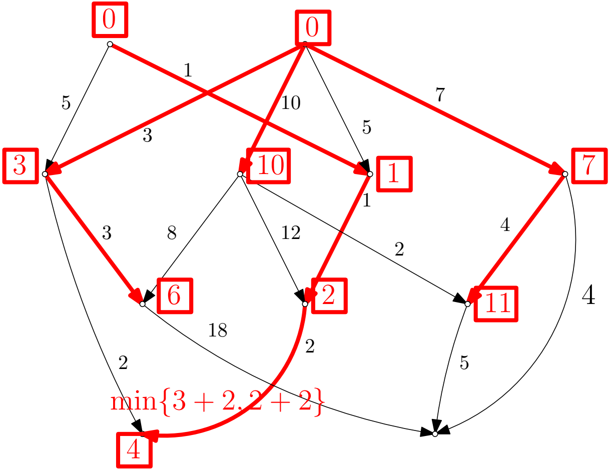


-

Exemple

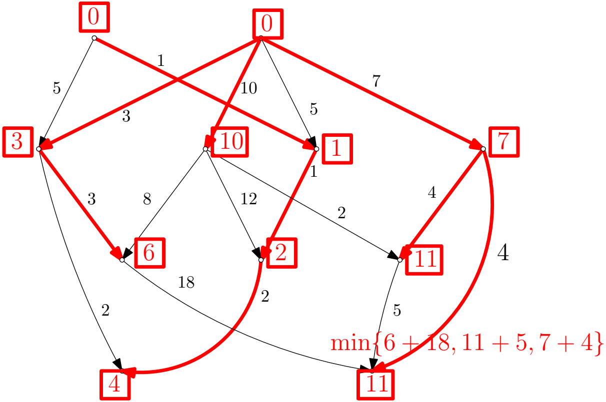


Exemple



•

Exemple



-

Problème de la plus longue sous-séquence commune

On appelle ici *sous-séquence* d'une chaîne de symboles X une chaîne obtenue en supprimant certains symboles de X. On se donne deux chaînes A et B de symboles dans un alphabet donné, et on souhaite trouver une sous-séquence commune aux deux chaînes de longueur maximum.

Exemple 1

Considérons par exemple les deux chaînes A = houseboat et B = computer. Les plus longues sous-séquences communes sont de longueur trois: il s'agit par exemple de la chaîne **out** :

- h**ou**seboa**t**
- c**o**mp**ut**er

(ou encore **oue**.)

Complexité

Quel est le nombre de sous-séquences possibles, que nous devrions examiner dans une recherche exhaustive ? Il est clairement exponentiel. En effet, le nombre de sous-séquences d'une chaîne de n symboles est égal au nombre de sous-ensembles d'un ensemble de n éléments, soit 2^n.

Programmation dynamique

Le *préfixe* de longueur i d'une chaîne est la chaîne obtenue en ne conservant que les i premiers symboles. On définit $L_{i,j}$ comme la longueur de la plus longue sous-séquence commune aux deux préfixes de longueurs i et j de A et B.

Proposition 2

$L_{i,0} = 0$ pour tout $i \geq 0$, $L_{0,j} = 0$ pour tout $j \geq 0$, et

$$L_{i,j} = \begin{cases} L_{i-1,j-1} + 1 & \text{si } A_i = B_j \\ \max\{L_{i-1,j}, L_{i,j-1}\} & \text{sinon.} \end{cases}$$

Exemple

	h	o	u	s	e	b	o	a	t
c	0	0	0	0	0	0	0	0	0
o	0	1	1	1	1	1	1	1	1
m	0	1	1	1	1	1	1	1	1
p	0	1	1	1	1	1	1	1	1
u	0	1	2	2	2	2	2	2	2
t	0	1	2	2	2	2	2	2	3
e	0	1	2	2	3	3	3	3	3
r	0	1	2	2	3	3	3	3	3

Complexité

Proposition 3

Le problème de calcul d'une plus longue sous-séquence commune à deux chaînes de symboles de longueurs m et n peut être résolu en temps $O(m \times n)$.

Code

```
int plss(char[] A, char[] B, int m, int n)
  {
    int L[][] = new int[m + 1][n + 1];

    for (int i = 0; i <= m; i++) {
      for (int j = 0; j <= n; j++) {
        if (i == 0 || j == 0)
          L[i][j] = 0;
        else if (A[i - 1] == B[j - 1])
          L[i][j] = L[i - 1][j - 1] + 1;
        else
          L[i][j] = max(L[i - 1][j],
            L[i][j - 1]);
      }
    }
    return L[m][n];
  }
}
```

Plus courts chemins

- Le calcul de la plus longue sous-séquence commune par programmation dynamique est un cas particulier de recherche de plus courts chemins dans un graphe dirigé acyclique !

Exemple :