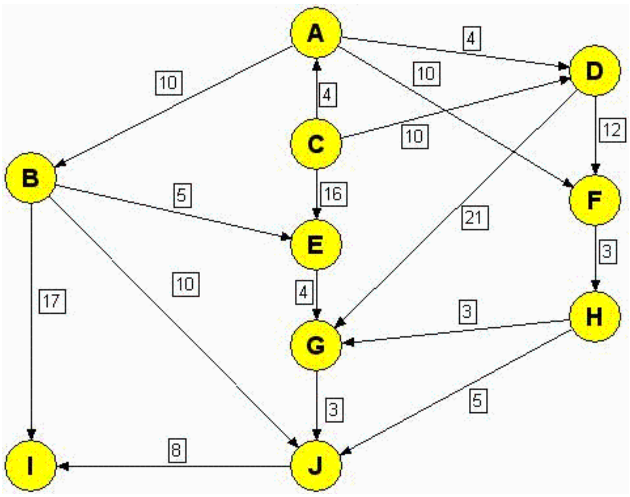
TD RO : Recherche de chemins dans les graphes

Algorithme de Dijkstra

On convient de représenter un graphe simple orienté pondéré ayant n sommets par une matrice M carrée n x n définie par: $M[x,y] = 0$ si $x = y$, $M[x,y] = \text{Poids}(x,y)$ si $x \neq y$ et il existe un arc d'origine le sommet de numéro x et d'extrémité le sommet de numéro y, et $M[x,y] = \infty$ sinon.

Exemple: un graphe et sa représentation matricielle



$$M := \begin{bmatrix} 0 & 10 & \infty & 4 & \infty & 10 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & 5 & \infty & \infty & \infty & 17 & 10 \\ 4 & \infty & 0 & 10 & 16 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 12 & 21 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & 4 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty & 3 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & 3 \\ \infty & \infty & \infty & \infty & \infty & \infty & 3 & 0 & \infty & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 8 & 0 \end{bmatrix}$$

L'*algorithme de Dijkstra* permet de trouver le plus court chemin pour relier le sommet de numéro *dep* au sommet de numéro *arv*.

Un sommet de départ *dep* étant fixé, cet algorithme construit progressivement un ensemble de sommets pour lesquels on connaît un plus court chemin depuis *dep*.

À chaque étape, on choisit un sommet dont la distance à *dep* est minimale parmi ceux qui n'ont pas encore été choisis. On utilisera un ensemble *c* des sommets à choisir successivement et deux tableaux *d* et *pred* des distances et des prédécesseurs.

Commencer par analyser l'algorithme suivant :

Procédure *Dijkstra*(*M* (matrice) , *dep* , *arv*)

Initialisation :

$c \leftarrow \{dep\}$; $d[dep] \leftarrow 0$
pour tout sommet *k* **faire**
   si $k \neq dep$ **alors** $d[k] \leftarrow M[dep,k]$; $pred[k] \leftarrow dep$ **fin si**
fin pour

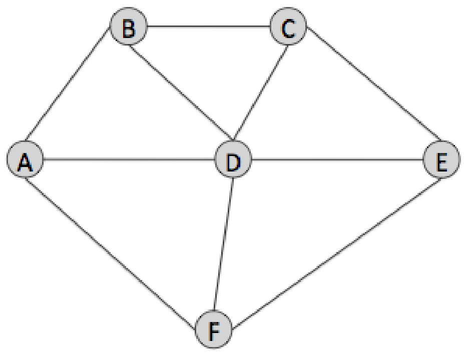
Construction de l'ensemble c :

tant que cardinal(c)< n **faire**
  trouver y tel que $d[y] = \min\{d[k];\ k$ sommet tel que k n'appartient pas à c}
  si ce minimum est ∞ **alors** sortir de la boucle **fin si**
  $c \leftarrow c$ *union* $\{y\}$
  pour chaque sommet k n'appartient pas à c **faire**
    si $d[k] > d[y] + M[y,k]$ **alors** $d[k] \leftarrow d[y] + M[y,k]$; $pred[k] \leftarrow y$ **fin si**
  fin pour
fin tant que

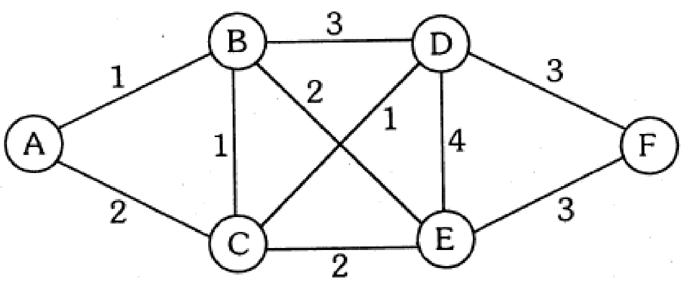
Construction du chemin le plus court :

$ch \leftarrow []$; $k \leftarrow arv$;
tant que $k \neq dep$ **faire**
  $ch \leftarrow [k, ch]$; $k \leftarrow pred[k]$
fin tant que
$ch \leftarrow [dep, ch]$
si $d[arv]=\infty$ alors *pas de chemin possible*
sinon
  afficher *ch* (*chemin le plus court*) et $d[arv]$ (*la distance correspondante*)
fin si
fin procédure

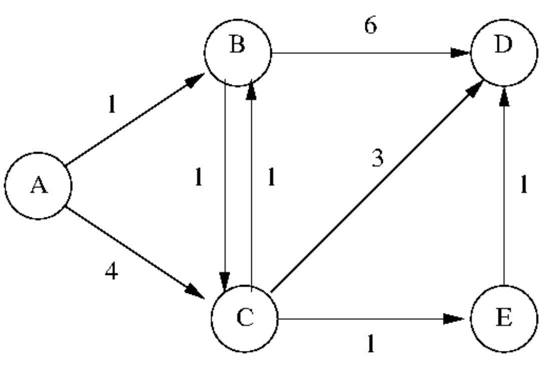
Appliquer l'algorithme sur chacun des cas suivants :



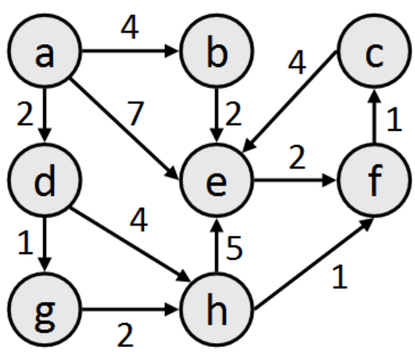
Chemin entre A et E



Chemin entre A et F



Chemin entre A et D



Chemin entre A et F

Algorithme de Floyd-Warshall

Analyser l'algorithme suivant :

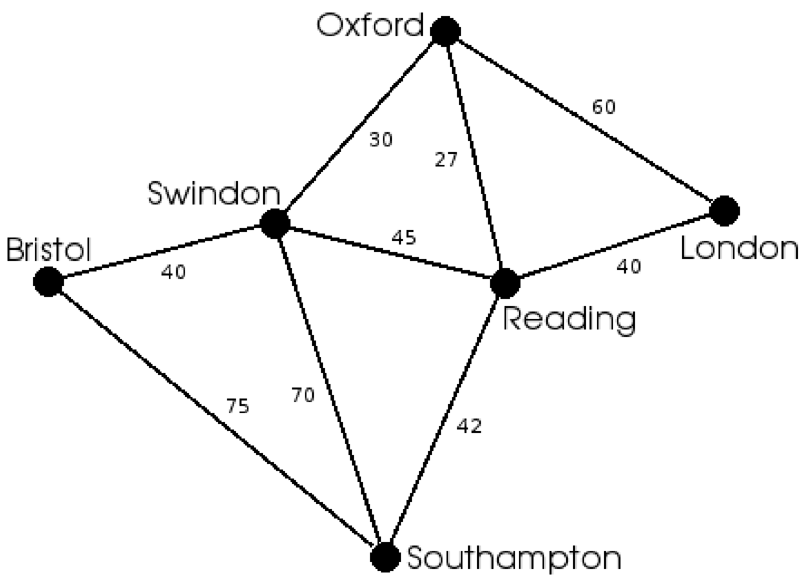
Procédure *FloydWarshall* (G)

$\mathcal{W}$:=matrice d'adjacence de G (matrice $n \times n$)
for $k := 1$ **to** n
  for $i := 1$ **to** n
    for $j := 1$ **to** n
      $\mathcal{W}_{ij} := \min(\mathcal{W}_{ij}, \mathcal{W}_{ik} + \mathcal{W}_{kj})$

Afficher $\mathcal{W}$

Que fait l'algorithme ?

Appliquer le sur le graphe suivant (les distances sont en miles) :



Quelle est la complexité de l'algorithme Floyd-Warshall ? Même question pour celui de Dijkstra.