

# Graphes - Plus court chemin

Algorithmique et structures de données, 2022-2023

---

P. Albuquerque (B410), P. Künzli et O. Malaspinas (A401), ISC, HEPIA  
2023-05-24

En partie inspirés des supports de cours de P. Albuquerque

## Rappel du dernier cours

- Qu'est-ce qu'un graphe? Un graphe orienté? Un graphe pondéré?

# Rappel du dernier cours

- Qu'est-ce qu'un graphe? Un graphe orienté? Un graphe pondéré?
- Ensemble de sommets et arêtes, avec une direction, possédant une pondération.
- Comment représenter un graphe informatiquement?

# Rappel du dernier cours

- Qu'est-ce qu'un graphe? Un graphe orienté? Un graphe pondéré?
- Ensemble de sommets et arêtes, avec une direction, possédant une pondération.
- Comment représenter un graphe informatiquement?
- Liste ou matrice d'adjacence.
- Quel sont les parcours que nous avons vu?

# Rappel du dernier cours

- Qu'est-ce qu'un graphe? Un graphe orienté? Un graphe pondéré?
- Ensemble de sommets et arêtes, avec une direction, possédant une pondération.
- Comment représenter un graphe informatiquement?
- Liste ou matrice d'adjacence.
- Quel sont les parcours que nous avons vu?
- Le parcours en largeur et le parcours en profondeur.

## Contexte: les réseaux (informatique, transport, etc.)

- Graphe orienté;
- Source: sommet  $s$ ;
- Destination: sommet  $t$ ;
- Les arêtes ont des poids (coût d'utilisation, distance, etc.);
- Le coût d'un chemin est la somme des poids des arêtes d'un chemin.

### Problème à résoudre

- Quel est le plus court chemin entre  $s$  et  $t$ .

# Exemples d'application de plus court chemin

## Devenir riches!

- On part d'un tableau de taux de change entre devises.
- Quelle est la meilleure façon de convertir l'or en dollar?

Currency	£	Euro	¥	Franc	\$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3929	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

**Figure 1:** Taux de change.

# Exemples d'application de plus court chemin

## Devenir riches!

- On part d'un tableau de taux de change entre devises.
- Quelle est la meilleure façon de convertir l'or en dollar?

Currency	£	Euro	¥	Franc	\$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3929	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

**Figure 1:** Taux de change.

- 1kg d'or  $\Rightarrow$  327.25 dollars
- 1kg d'or  $\Rightarrow$  208.1 livres  $\Rightarrow$  327 dollars
- 1kg d'or  $\Rightarrow$  455.2 francs  $\Rightarrow$  304.39 euros  $\Rightarrow$  327.28 dollars



# Exemples d'application de plus court chemin

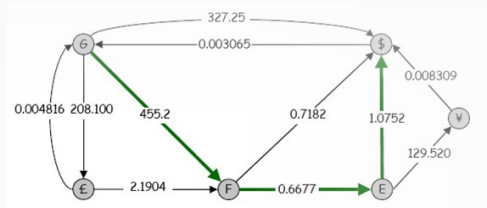
## Formulation sous forme d'un graphe: Comment ?

Currency	£	Euro	¥	Franc	\$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3929	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

Figure 2: Taux de change.

# Exemples d'application de plus court chemin

## Formulation sous forme d'un graphe: Comment ?

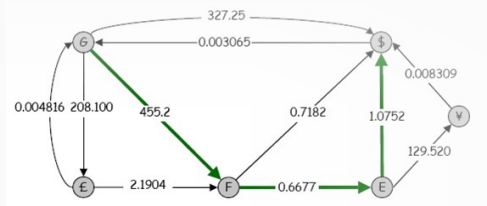


**Figure 3:** Graphes des taux de change.

- Un sommet par devise;
- Une arête orientée par transaction possible avec le poids égal au taux de change;
- Trouver le chemin qui maximise le produit des poids.

# Exemples d'application de plus court chemin

## Formulation sous forme d'un graphe: Comment ?



**Figure 3:** Graphes des taux de change.

- Un sommet par devise;
- Une arête orientée par transaction possible avec le poids égal au taux de change;
- Trouver le chemin qui maximise le produit des poids.

## Problème

- On aimerait plutôt avoir une somme...

# Exemples d'application de plus court chemin

## Conversion du problème en plus court chemin

- Soit  $\text{taux}(u, v)$  le taux de change entre la devise  $u$  et  $v$ .
- On pose  $w(u, v) = -\log(\text{taux}(u, v))$
- Trouver le chemin poids minimal pour les poids  $w$ .

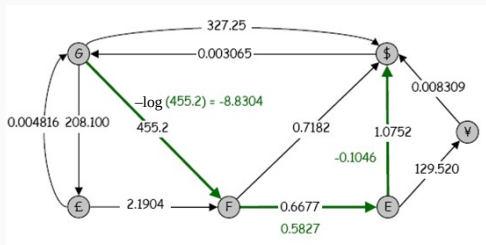


Figure 4: Graphe des taux de change avec logs.

- Cette conversion se base sur l'idée que

$$\log(u \cdot v) = \log(u) + \log(v).$$

# Applications de plus courts chemins

**Quelles applications voyez-vous?**

## Quelles applications voyez-vous?

- Déplacement d'un robot;
- Planification de trajet / trafic urbain;
- Routage de télécommunications;
- Réseau électrique optimal;
- ...

# Plus courts chemins à source unique

- Soit un graphe,  $G = (V, E)$ , une fonction de pondération  $w : E \rightarrow \mathbb{R}$ , et un sommet  $s \in V$ 
  - Trouver pour tout sommet  $v \in V$ , le chemin de poids minimal reliant  $s$  à  $v$ .
- Algorithmes standards:
  - Dijkstra (arêtes de poids positif seulement);
  - Bellman-Ford (arêtes de poids positifs ou négatifs, mais sans cycles).
- Comment résoudre le problème si tous les poids sont les mêmes?

# Plus courts chemins à source unique

- Soit un graphe,  $G = (V, E)$ , une fonction de pondération  $w : E \rightarrow \mathbb{R}$ , et un sommet  $s \in V$ 
  - Trouver pour tout sommet  $v \in V$ , le chemin de poids minimal reliant  $s$  à  $v$ .
- Algorithmes standards:
  - Dijkstra (arêtes de poids positif seulement);
  - Bellman-Ford (arêtes de poids positifs ou négatifs, mais sans cycles).
- Comment résoudre le problème si tous les poids sont les mêmes?
- Un parcours en largeur!



# Algorithme de Dijkstra

**Comment chercher pour un plus court chemin?**

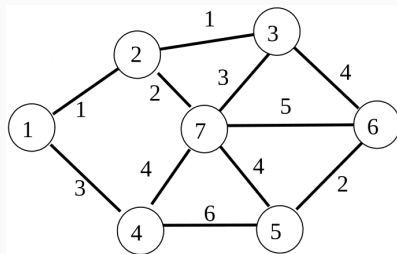
# Algorithme de Dijkstra

**Comment chercher pour un plus court chemin?**

si  $\text{distance}(u,v) > \text{distance}(u,w) + \text{distance}(w,v)$   
on passe par  $w$  plutôt qu'aller directement

# Algorithme de Dijkstra (1 à 5)

- $D$  est le tableau des distances au sommet 1:  $D[7]$  est la distance de 1 à 7.
- Le chemin est pas forcément direct.
- $S$  est le tableau des sommets visités.



**Figure 5:** Initialisation.

# Algorithme de Dijkstra (1 à 5)

- $D$  est le tableau des distances au sommet 1:  $D[7]$  est la distance de 1 à 7.
- Le chemin est pas forcément direct.
- $S$  est le tableau des sommets visités.

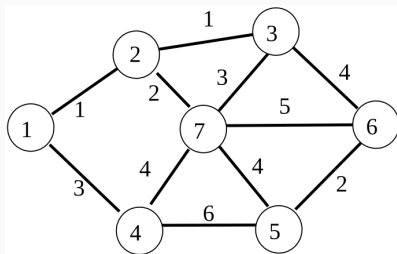


Figure 5: Initialisation.

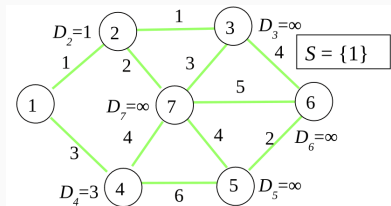
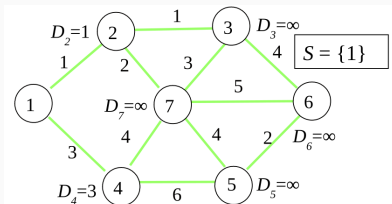


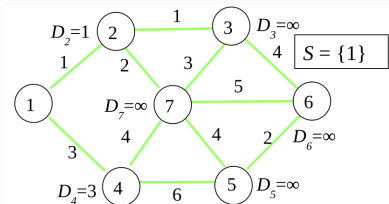
Figure 6: 1 visité,  $D[2]=1$ ,  $D[4]=3$ .

# Algorithme de Dijkstra (1 à 5)

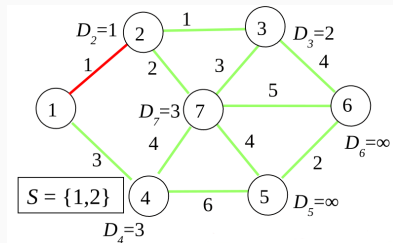


**Figure 7:** Plus court est 2.

# Algorithme de Dijkstra (1 à 5)

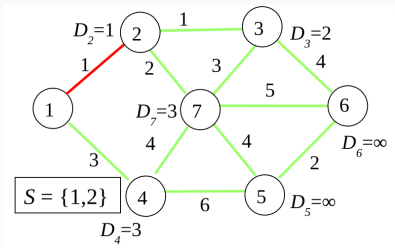


**Figure 7:** Plus court est 2.



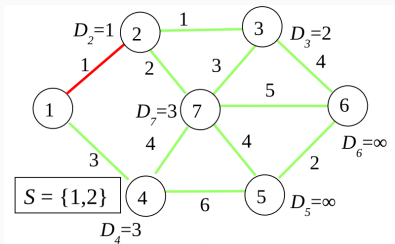
**Figure 8:** 2 visité,  $D[3]=2$ ,  $D[7]=3$ .

# Algorithme de Dijkstra (1 à 5)

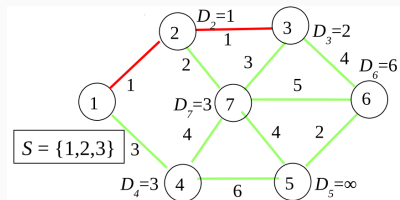


**Figure 9:** Plus court est 3.

# Algorithme de Dijkstra (1 à 5)



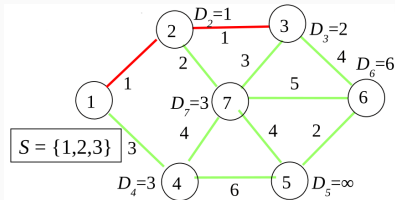
**Figure 9:** Plus court est 3.



**Figure 10:** 3 visité,  $D[7]=3$  inchangé,  $D[6]=6$ .

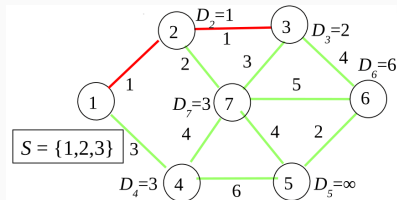


# Algorithme de Dijkstra (1 à 5)

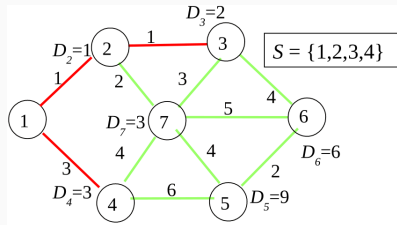


**Figure 11:** Plus court est 4 ou 7.

# Algorithme de Dijkstra (1 à 5)

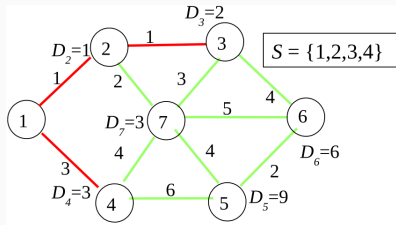


**Figure 11:** Plus court est 4 ou 7.



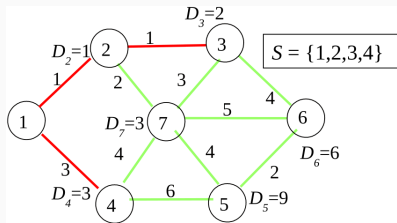
**Figure 12:** 4 visité,  $D[7]=3$  inchangé,  $D[5]=9$ .

# Algorithme de Dijkstra (1 à 5)

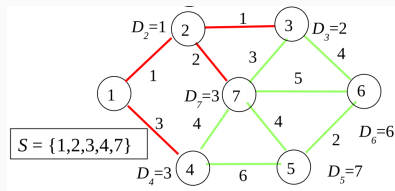


**Figure 13:** Plus court est 7.

# Algorithme de Dijkstra (1 à 5)

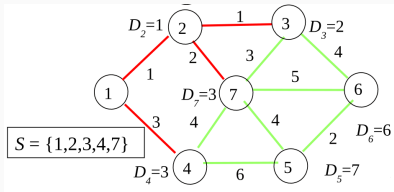


**Figure 13:** Plus court est 7.



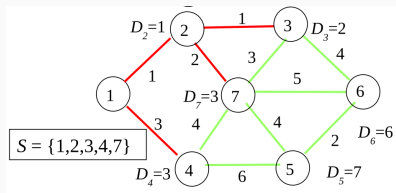
**Figure 14:** 7 visité,  $D[5]=7$ ,  $D[6]=6$  inchangé.

# Algorithme de Dijkstra (1 à 5)

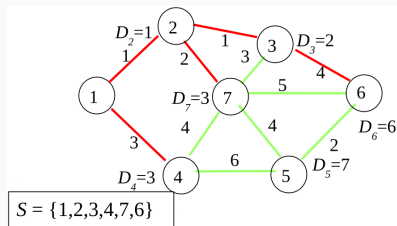


**Figure 15:** Plus court est 6.

# Algorithme de Dijkstra (1 à 5)

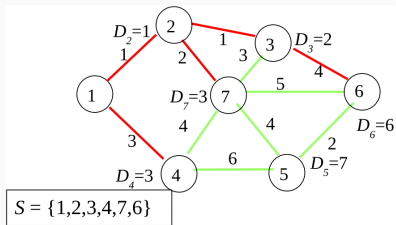


**Figure 15:** Plus court est 6.



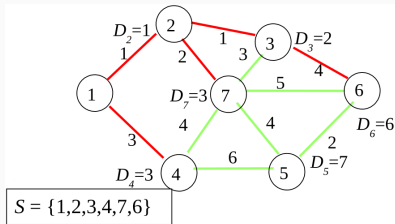
**Figure 16:** 6 visité,  $D[5]=7$  inchangé.

# Algorithme de Dijkstra (1 à 5)

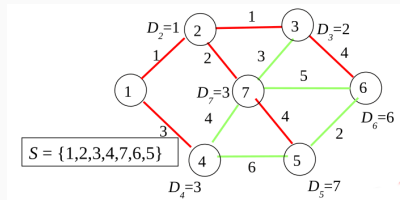


**Figure 17:** Plus court est 5 et c'est la cible.

# Algorithme de Dijkstra (1 à 5)



**Figure 17:** Plus court est 5 et c'est la cible.



**Figure 18:** The end, tous les sommets ont été visités.



# Algorithme de Dijkstra

## Idée générale

- On assigne à chaque noeud une distance 0 pour  $s$ ,  $\infty$  pour les autres.
- Tous les noeuds sont marqués non-visités.
- Depuis le noeud courant, on suit chaque arête du noeud vers un sommet non visité et on calcule le poids du chemin à chaque voisin et on met à jour sa distance si elle est plus petite que la distance du noeud.
- Quand tous les voisins du noeud courant ont été évalués, le noeud est mis à visité (il ne sera plus jamais visité).
- Continuer avec le noeud à la distance la plus faible.
- L'algorithme est terminé lorsque le noeud de destination est marqué comme visité, ou qu'on a plus de noeuds qu'on peut visiter et que leur distance est infinie.

# Algorithme de Dijkstra

## Pseudo-code

# Algorithme de Dijkstra

## Pseudo-code

```
tab dijkstra(graph, s, t)
  pour chaque v dans graphe
    distance[v] = infini
    q = ajouter(q, v)
  distance[s] = 0
  tant que non_vide(q)
    // sélection de u t.q. la distance dans q est min
    u = min(q, distance)
    si u == t // on a atteint la cible
      retourne distance
    q = remove(q, u)
    // voisin de u encore dans q
    pour chaque v dans voisinage(u, q)
      // on met à jour la distance du voisin en passant par u
      n_distance = distance[u] + w(u, v)
      si n_distance < distance[v]
        distance[v] = n_distance
  retourne distance
```

# Algorithme de Dijkstra

- Cet algorithme, nous donne le plus court chemin mais...
- ne nous donne pas le chemin!

**Comment modifier l'algorithme pour avoir le chemin?**

# Algorithme de Dijkstra

- Cet algorithme, nous donne le plus court chemin mais...
- ne nous donne pas le chemin!

## **Comment modifier l'algorithme pour avoir le chemin?**

- Pour chaque nouveau noeud à visiter, on enregistre d'où on est venu!
- On a besoin d'un tableau précédent.

**Modifier le pseudo-code ci-dessus pour ce faire**

# Algorithme de Dijkstra

```
tab, tab dijkstra(graph, s, t)
  pour chaque v dans graphe
    distance[v] = infini
    précédent[v] = indéfini
    q = ajouter(q, v)
  distance[s] = 0
  tant que non_vide(q)
    // sélection de u t.q. la distance dans q est min
    u = min(q, distance)
    si u == t
      retourne distance
    q = remove(q, u)
    // voisin de u encore dans q
    pour chaque v dans voisinage(u, q)
      n_distance = distance[u] + w(u, v)
      si n_distance < distance[v]
        distance[v] = n_distance
        précédent[v] = u
  retourne distance, précédent
```

Comment reconstruire un chemin ?

# Algorithme de Dijkstra

## Comment reconstruire un chemin ?

```
pile parcours(précédent, s, t)
    sommets = vide
    u = t
    // on a atteint t ou on ne connaît pas de chemin
    si u != s && précédent[u] != indéfini
        tant que vrai
            sommets = empiler(sommets, u)
            u = précédent[u]
        si u == s // la source est atteinte
            retourne sommets
    retourne sommets
```



# Algorithme de Dijkstra (exercice)

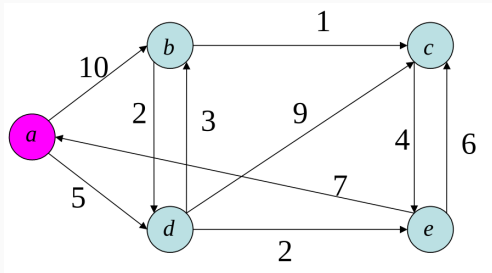


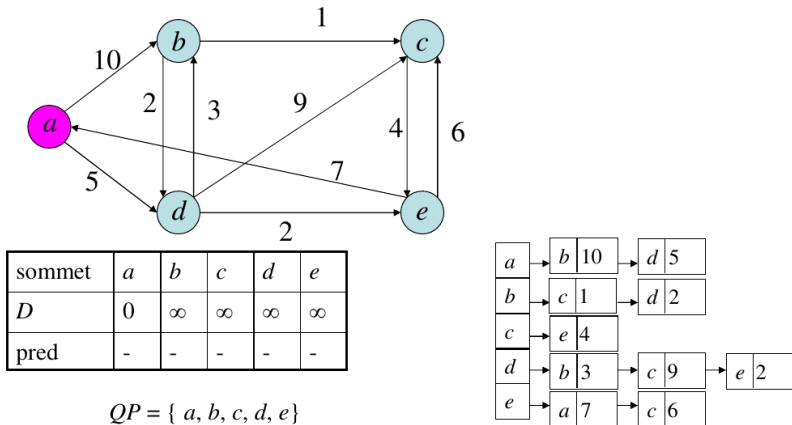
Figure 19: L'exercice.

- Donner la liste de priorité, puis...

## A chaque étape donner:

- Le tableau des distances à a;
- Le tableau des prédécesseurs;
- L'état de la file de sommets.

# Algorithme de Dijkstra (corrigé)



**Figure 20:** Le corrigé partie 1.

# Algorithme de Dijkstra (corrigé)

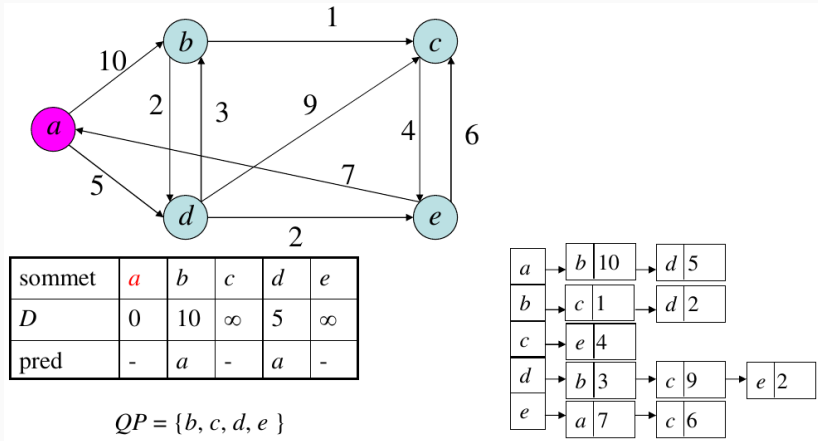


Figure 21: Le corrigé partie 2.

# Algorithme de Dijkstra (corrigé)

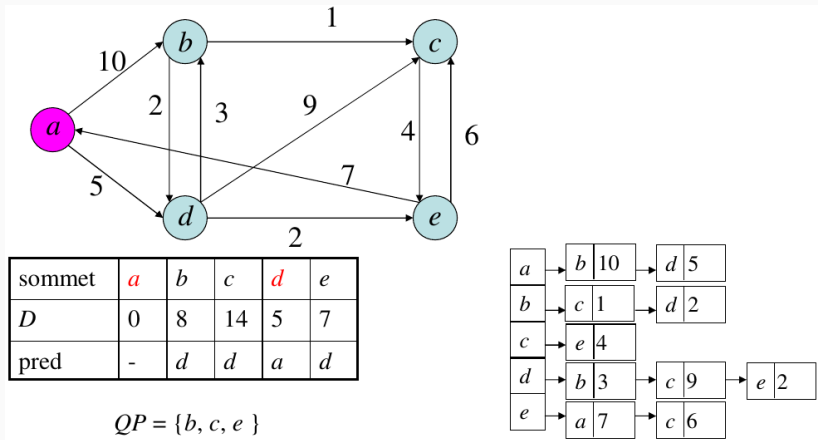


Figure 22: Le corrigé partie 3.

# Algorithme de Dijkstra (corrigé)

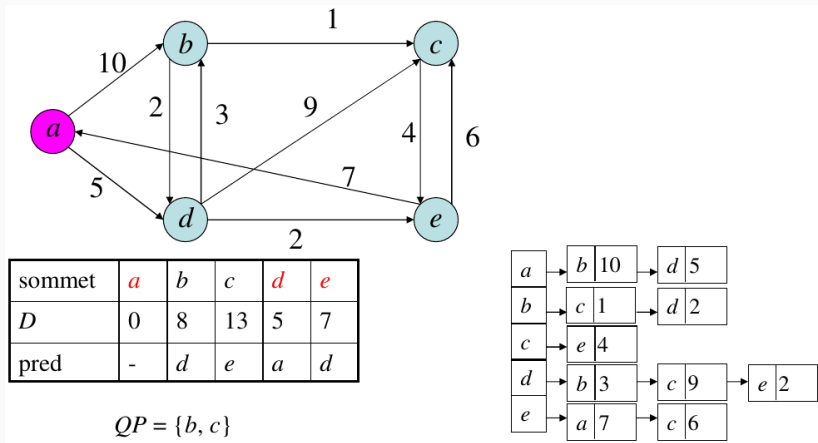


Figure 23: Le corrigé partie 4.

# Algorithme de Dijkstra (corrigé)

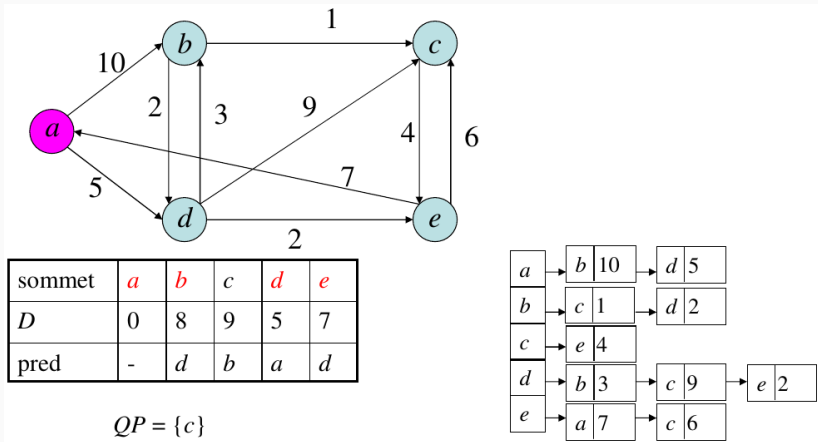


Figure 24: Le corrigé partie 5.

# Algorithme de Dijkstra (corrigé)

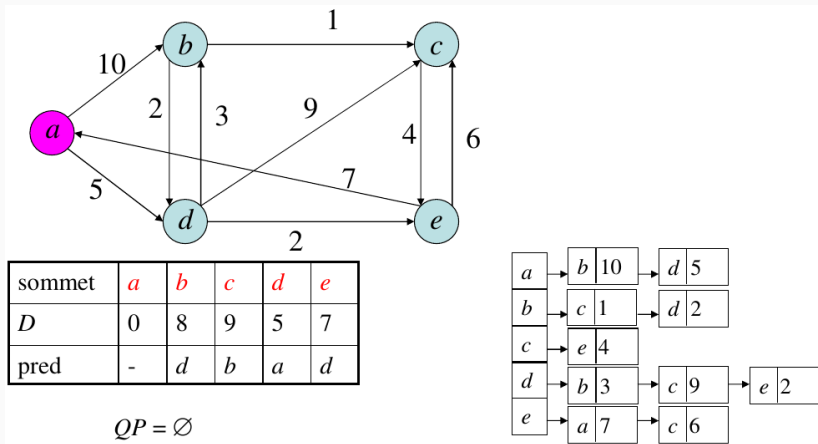


Figure 25: Le corrigé partie 6.

# Limitation de l'algorithme de Dijkstra

Que se passe-t-il pour?

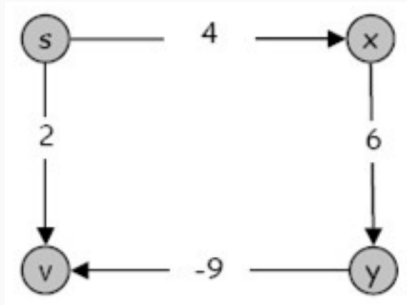


Figure 26: Exemple.

Quel est le problème?



# Limitation de l'algorithme de Dijkstra

Que se passe-t-il pour?

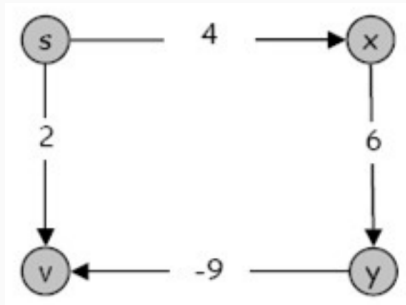


Figure 26: Exemple.

Quel est le problème?

- L'algorithme n'essaiera jamais le chemin  $s \rightarrow x \rightarrow y \rightarrow v$  et prendra direct  $s \rightarrow v$ .
- Ce problème n'apparaît que s'il y a des poids négatifs.