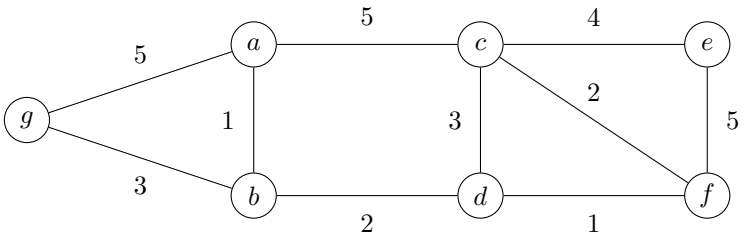
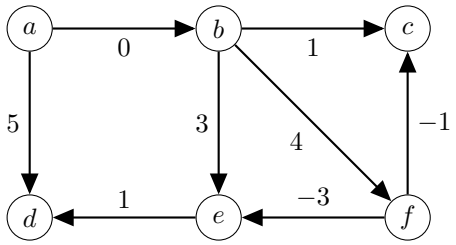
Exercice 1. Application des algorithmes

1. Appliquer l'algorithme de Dijkstra à la main sur le graphe ci-dessous pour déterminer les distances de a aux autres sommets.



2. Appliquer l'algorithme de Bellman-Ford pour trouver les distances depuis a vers n'importe quel autre sommet :



3. Appliquer l'algorithme de Floyd-Warshall au graphe représenté par la matrice d'adjacence pondérée ci-dessous, pour trouver la matrice des distances.

$$\begin{pmatrix} 0 & \infty & 2 & 4 \\ 1 & 0 & 5 & \infty \\ 2 & -1 & 0 & 7 \\ \infty & \infty & -3 & 0 \end{pmatrix}$$

On pourra remplir les matrices intermédiaires suivantes (pour chaque d_k):

$$\begin{pmatrix} & & & \end{pmatrix} \quad \begin{pmatrix} & & & \end{pmatrix} \quad \begin{pmatrix} & & & \end{pmatrix} \quad \begin{pmatrix} & & & \end{pmatrix}$$

Exercice 2. Petites questions

1. Trouver un exemple de graphe pour lequel Dijkstra ne fonctionne pas (ne donne pas les plus courts chemins).
2. Change t-on les plus courts chemins si on additionne/multiplie les poids de toutes les arêtes d'un graphe par une constante?
3. On suppose avoir un graphe orienté $\vec{G}$ dont les **sommets** sont pondérés par w, au lieu des arêtes. Le poids d'un chemin est alors la somme des poids de ses sommets. Comment modifier $\vec{G}$ pour pouvoir trouver des chemins de plus petit poids de $\vec{G}$, avec les algorithmes du cours?
4. Le graphe orienté $\vec{G} = (V, \vec{E})$ d'une chaîne de Markov possède une probabilité sur chaque arc. La probabilité d'un chemin est le produit des probabilités de ses arcs. Comment trouver un chemin le plus probable d'un sommet à un autre?
5. Comment pourrait-on déterminer le poids minimum d'un cycle dans un graphe orienté? En quelle complexité?
6. On suppose donné un graphe pondéré où un sommet est une ville et un arc est une route dont le poids est sa longueur. On part d'une ville s avec une voiture consommant 5L/100km et avec un réservoir rempli de 50L d'essence. On suppose connu le prix de l'essence dans chaque ville. Comment trouver un chemin de coût minimum jusqu'à une autre ville t? Quelle serait la complexité?
7. Soit $\vec{G}$ un graphe sans arc de poids négatif et s, t deux sommets. Montrer que l'on peut trouver tous les arcs utilisés par au moins un plus court chemin de s à t, en utilisant 2 fois l'algorithme de Dijkstra.

Exercice 3. Plus courts chemins dans un graphe sans cycle
On veut trouver les distances d'un sommet à tous les autres dans un graphe orienté sans cycle $\vec{G} = (V, \vec{E})$ pondéré par w. On pourra utiliser la fonction `tri_topo : int list array -> int list` du TD 2, renvoyant une liste des sommets d'un graphe sans cycle « compatible » avec les arcs ($(u, v) \in \vec{E} \Longrightarrow u$ est avant v dans la liste).

1. Expliquer comment trouver les distances d'un sommet r à tous les autres dans $\vec{G}$ représenté par liste d'adjacence, en complexité $O(|\vec{E}| + |V|)$.
2. Écrire une fonction correspondant à la question précédente.
3. Comment modifier l'algorithme précédent pour obtenir des chemins de poids **maximum**? Une modification similaire fonctionnerait-elle avec l'algorithme de Dijkstra?

Exercice 4. Algorithme de Johnson
Soit $\vec{G} = (V, \vec{E})$ un graphe orienté pondéré par $w : \vec{E} \longrightarrow \mathbb{R}$ (des poids peuvent être négatifs). On a vu en cours l'algorithme de Floyd-Warshall pour trouver les plus courts chemins de n'importe quel sommet u à n'importe quel autre v. L'algorithme de Johnson résout le même problème, mais possiblement avec une meilleure complexité. L'idée de l'algorithme de Johnson est de modifier les poids de $\vec{G}$ pour les rendre positifs, sans modifier les plus courts chemins. On peut alors appliquer $|V|$ fois l'algorithme de Dijkstra (une fois depuis chaque sommet) pour obtenir tous les plus courts chemins.

1. Soit $h : V \longrightarrow \mathbb{R}$. On définit $w_h : (u, v) \longmapsto w(u, v) + h(u) - h(v)$. Montrer que, dans $\vec{G}$, les plus courts chemins pour w et w_h sont les mêmes et qu'il existe un cycle de poids négatif pour w ssi il en existe un pour w_h.
2. Trouver $h : V \longrightarrow \mathbb{R}$ telle que $w_h \geq 0$. On pourra supposer dans un premier temps que tous les sommets de $\vec{G}$ sont atteignables depuis un sommet r.
3. En utilisant plusieurs fois l'algorithme de Dijkstra, écrire une fonction `johnson` renvoyant la matrice des distances entre toute paire de sommets, dans $\vec{G}$ pondéré par w.
4. Comparer la complexité de `johnson` avec celle de Floyd-Warshall