
Notation $O()$, notion de complexité

Exercice 1

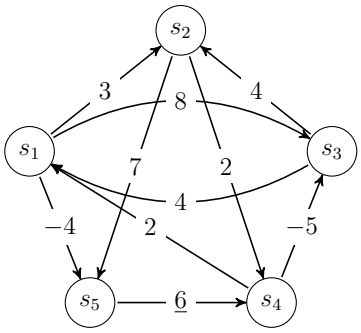
Soit $f(n) = \frac{1}{3}n\left(n + \frac{1}{2}\right)(n+1)$. Montrer que $f(n) = \mathcal{O}(n^3)$

Les relations suivantes sont elles vraies ?

1. $f(n) = \mathcal{O}(n^{10})$
2. $f(n) = \frac{1}{3}n^3 + \mathcal{O}(n^2)$
3. $\mathcal{O}(n^2) = \mathcal{O}(n^3)$
4. $\mathcal{O}(n^3) = \mathcal{O}(n^2)$
5. $\frac{1}{3}n^3 + \mathcal{O}(n^2) = \mathcal{O}(n^3)$

Exercice 2

Soit $G = (X, A)$ le graphe orienté avec une fonction de pondération $w : A \to \mathbb{R}$ représenté ci-dessous.



L'éxécution de l'algorithme de *Floyd-Warshall* a produit le résulat final suivant :

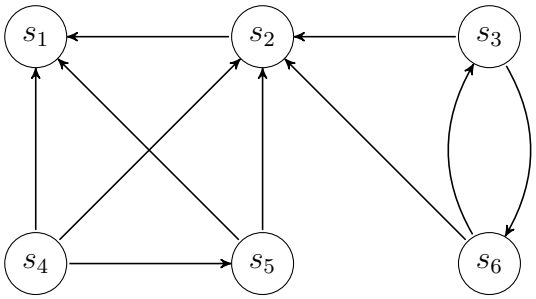
$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 1 & 0 & -3 & 2 & -3 \\ 4 & 4 & 0 & 6 & 0 \\ -1 & -1 & -5 & 0 & -5 \\ 5 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{Nil} & s_3 & s_4 & s_5 & s_1 \\ s_3 & \text{Nil} & s_4 & s_2 & s_1 \\ s_3 & s_3 & \text{Nil} & s_2 & s_1 \\ s_3 & s_3 & s_4 & \text{Nil} & s_1 \\ s_3 & s_3 & s_4 & s_5 & \text{Nil} \end{pmatrix}$$

1. Expliquer comment connaître à partir de la matrice $D^{(5)}$ la distance du plus court chemin entre les sommets s_5 et s_4 puis entre les sommets s_2 et s_5.
2. Expliquer comment reconstruire à partir de la matrice $\Pi^{(5)}$ le plus court chemin entre les sommets s_5 et s_4 puis entre les sommets s_2 et s_5.
3. Calculer le plus court chemin entre chaque couple de sommets, c'est-à-dire de recalculer les matrices $D^{(5)}$ et $\Pi^{(5)}$ en appliquant l'algorithme de *Floyd-Warshall*.

FLOYD-WARSHALL(W)
1 $n \leftarrow$ nombre_de_lignes(W)
2 $D^{(0)} \leftarrow W$
3 **pour** $k \leftarrow 1$ à n
4 **pour** $i \leftarrow 1$ à n
5 **pour** $j \leftarrow 1$ à n
6 $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
7 **calcul**$\left(\pi_{ij}^{(k)}\right)$
8 **retourner** $D^{(n)}$ et $\Pi^{(n)}$

Exercice 3

Soit $G = (X, A)$ le graphe orienté représenté ci-dessous.



1. Calculer la fermeture transitive du graphe G.

Parcours en largeur

Exercice 4

Appliquer l'algorithme du ***parcours en largeur*** `PL(G, s)` au graphe $G1$ à partir du sommet s_1 et au graphe $G2$ à partir du sommet s_2. Chaque fois donner l'ordre d'entrée des sommets dans la file, et le contenu final des tableaux `d[]` et `pere[]`.

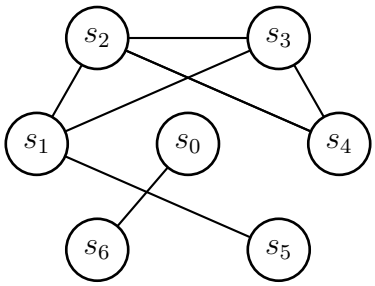


FIGURE 1 – $G1$

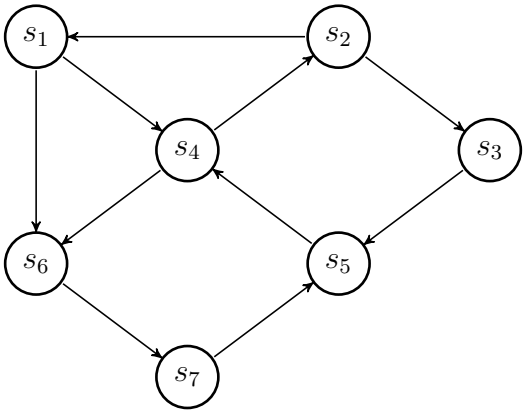


FIGURE 2 – $G2$

```
PL(G, s)
 1  pour chaque sommet u de X[G] - {s}
 2      faire couleur[u] <- BLANC
 3            d[u] <- infini
 4            pere[u] <- nil
 5  couleur[s] <- GRIS
 6  d[s] <- 0
 7  pere[s] <- nil
 8  Enfiler(F, s)
 9  tant que non vide(F)
10    faire u <- tete(F)
11          pour chaque v de Adj(u)
12              faire si couleur[v] = BLANC
13                 alors couleur[v] <- GRIS
14                       d[v] <- d[u] + 1
15                       pere[v] <- u
16                       Enfiler(F, v)
17          Defiler(F)
18          couleur[u] <- NOIR
```

Exercice 5

Proposer une modification de l'algorithme `PL(G,s)` permettant de détecter si un graphe (non-orienté) donné est connexe ou pas. Spécifier les lignes du code-dessus à modifier / supprimer / ajouter. Étudier la complexité de votre algorithme.

Exercice 6

Proposer une modification de l'algorithme `PL(G,s)` permettant de détecter si un graphe (non-orienté) donné est un arbre ou pas. Spécifier les lignes du code-dessus à modifier / supprimer / ajouter. Étudier la complexité de votre algorithme.