PRÉPARATION AU SWERC : DIAMÈTRE, CHEMINS & CYCLES EULÉRIENS, PLUS COURTS CHEMINS, APCM, SCC

Mattéo Delabre & Guillaume Pérution-Kihli

Université de Montpellier
14 décembre 2022

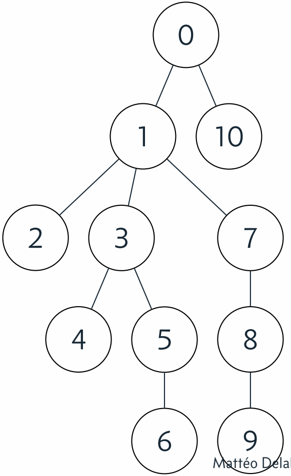
OBJECTIFS

- Certains problèmes **font appel explicitement à des graphes** dans leur énoncé, par exemple :
  - Recherche d'éléments dans un graphe donné.
  - Calcul de propriétés d'un graphe donné.
- D'autres peuvent être résolus en les **modélisant par des graphes** même si ce n'est pas explicite, par exemple :
  - Résolution d'inéquations linéaires.
  - Affectation de ressources.
  - Résolution de 2-SAT.
- Notre programme d'aujourd'hui et de la semaine prochaine :
  - Passer en revue des techniques de graphes classiques.
  - S'entraîner à modéliser des problèmes sous forme de graphes.

PLAN

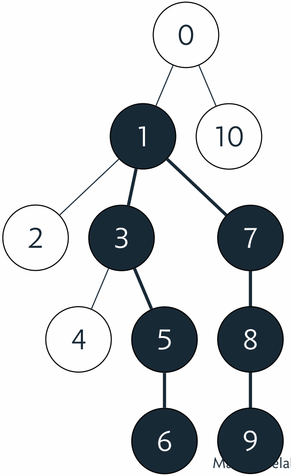
RÉFLÉCHISSONS SUR UN EXEMPLE

Quel est le plus long chemin dans cet arbre ?
Comment l'avez-vous trouvé ?

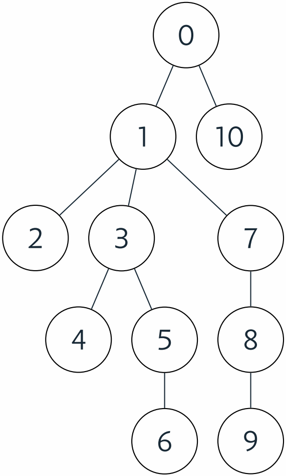


RÉFLÉCHISSONS SUR UN EXEMPLE

Quel est le plus long chemin dans cet arbre ?
Comment l'avez-vous trouvé ?

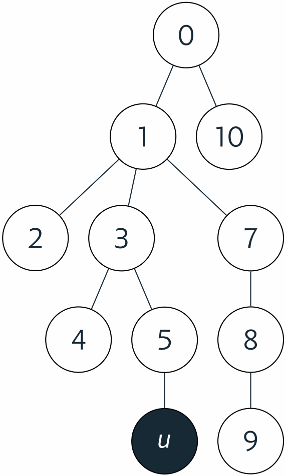


DOUBLE PARCOURS EN PROFONDEUR



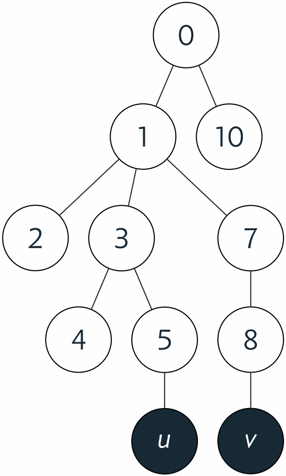
- Algorithme glouton.
- Parcours à partir d'une racine r quelconque et trouver un sommet u de profondeur maximale.
- Parcours à partir de u et trouver un sommet v de profondeur maximale.
- $(u, \ldots, v)$ est un chemin de longueur maximale.

DOUBLE PARCOURS EN PROFONDEUR



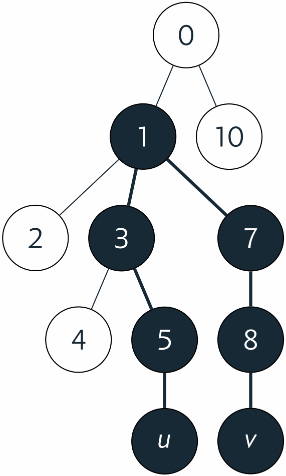
- Algorithme glouton.
- Parcours à partir d'une racine r quelconque et trouver un sommet u de profondeur maximale.
- Parcours à partir de u et trouver un sommet v de profondeur maximale.
- $(u, \ldots, v)$ est un chemin de longueur maximale.

DOUBLE PARCOURS EN PROFONDEUR



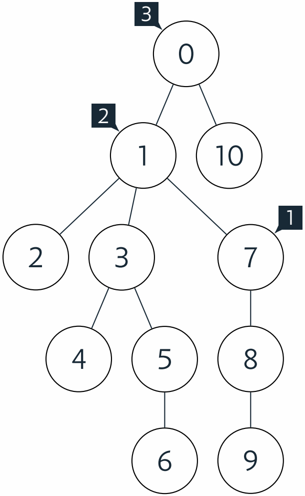
- Algorithme glouton.
- Parcours à partir d'une racine r quelconque et trouver un sommet u de profondeur maximale.
- Parcours à partir de u et trouver un sommet v de profondeur maximale.
- $(u, \ldots, v)$ est un chemin de longueur maximale.

DOUBLE PARCOURS EN PROFONDEUR



- Algorithme glouton.
- Parcours à partir d'une racine r quelconque et trouver un sommet u de profondeur maximale.
- Parcours à partir de u et trouver un sommet v de profondeur maximale.
- $(u, \ldots, v)$ est un chemin de longueur maximale.

- Un seul parcours (grands arbres).
- À quoi ressemble le chemin le plus long dans le sous-arbre issu de *v* ?
  1. Commence en *v*.
  2. Commence dans un sous-arbre de *v*, passe par *v*, puis se termine dans un autre sous-arbre.
  3. Ne passe pas par *v*.



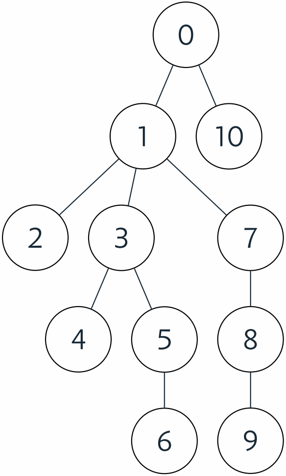
PROGRAMMATION DYNAMIQUE — RÉCURRENCE

- Deux tables, calcul en postfixe.
- b[*v*] (**b**egins) : longueur max. des chemins qui commencent en *v* dans le sous-arbre issu de *v*.

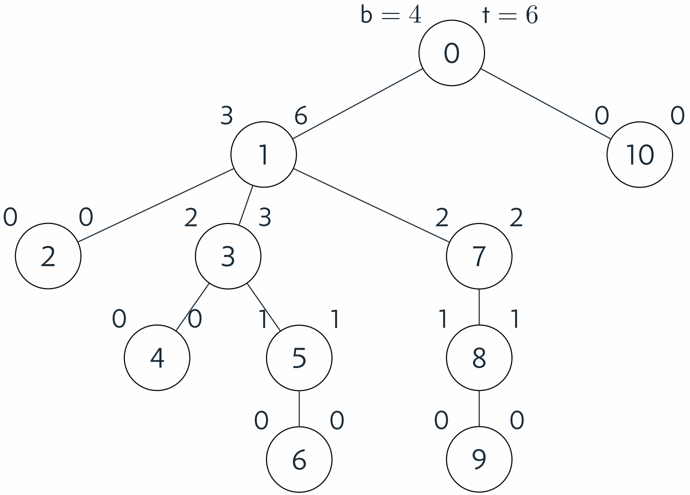
$$\mathsf{b}[v] = 1 + \operatorname*{MAX}_{u \text{ enfant de } v} \mathsf{b}[u]$$

- t[*v*] (**t**otal) : longueur max. des chemins dans le sous-arbre sous *v*.

$$\mathsf{t}[v] = \mathrm{MAX} \begin{cases} \operatorname*{MAX}_{u \text{ enfant de } v} \mathsf{t}[u] \\ \operatorname*{MAX}_{u_1, u_2 \text{ enfants de } v} \mathsf{b}[u_1] + 2 + \mathsf{b}[u_2] \end{cases}$$



PROGRAMMATION DYNAMIQUE — EXEMPLE



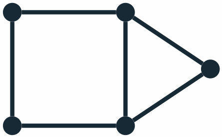
EXERCICES ET RÉFÉRENCES

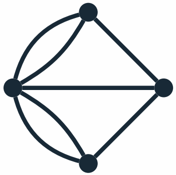
- Exercices :
  - CEPC 1999 (Prague), “Labyrinth”
  - **UVa 10308, “Roads in the North”**
- Dans les livres de référence :
  - Dürr et Vie, §10.3.
  - Laaksonen, §10.1.2.
  - Halim, §4.7.2.

PLAN

QUELQUES EXEMPLES

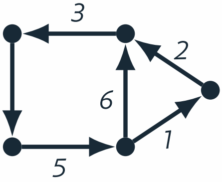
Pouvez-vous tracer les figures suivantes d’un seul coup de crayon en partant d’un des points ? Si non, pourquoi ?

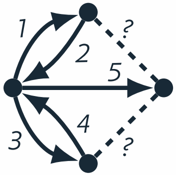




QUELQUES EXEMPLES

Pouvez-vous tracer les figures suivantes d'un seul coup de crayon en partant d'un des points ? Si non, pourquoi ?

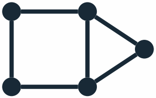


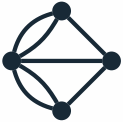


Sept ponts de Königsberg

CHEMINS ET CYCLES EULÉRIENS

- **Chemin *[cycle]* eulérien :** marche qui passe exactement une fois par chaque arête du graphe *[et qui se termine sur le sommet où elle a commencé]*.
- **Cycle** existe ssi le degré de chaque sommet est pair.
- **Chemin** existe ssi le degré de chaque sommet sauf deux est pair.





- *Note :* Différent de chemin *[cycle]* hamiltonien !

ALGORITHME DE HIERHOLZER

- Algorithme glouton en temps linéaire.
  1. Tant qu'on n'a pas visité toutes les arêtes.
  2. Choisir un sommet v dont l'une des arêtes adjacentes n'a pas encore été visitée.
  3. Faire une marche partant de v jusqu'à revenir sur v, sans utiliser les arêtes déjà visitées.
  4. Répéter.
- En concaténant tous les cycles générés, on obtient un **cycle** eulérien.
- Pour trouver un **chemin**, il suffit de commencer sur un sommet impair et d'ajouter virtuellement l'arête (u, v).

EXERCICES ET RÉFÉRENCES

- Exercices :
  - SPOJ, “Free tour”
  - **UVa 10054, “The Necklace”**
- Dans les livres de référence :
  - Dürr et Vie, §7.1.
  - Laaksonen, §12.2.1.
  - Halim, §4.7.3.

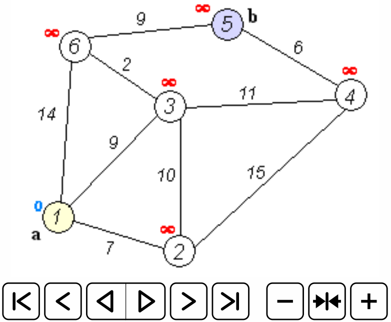
PLAN

PLUS COURT CHEMIN

- Plus court chemin : trouver un chemin d'un sommet à un autre minimisant poids des arcs
- Si graphe non pondéré : parcours en largeur (complexité linéaire)
- Pour les graphe avec arcs pondérés positivement : Dijkstra généralise parcours en largeur
- Si arcs pondérés négativement : Floyd-Warshall
- Pour les plus courts chemins entre toute paire de sommets : Bellmand-Ford

ALGORITHME DE DIJSKTRA

- Plus court chemin entre un sommet et tous les autres sommets
- Complexité : $O(m + n \operatorname{LOG} n)$
- Attention à l'implémentation : la meilleure complexité s'atteint en utilisant un tas



ALGORITHME DE BELLMAN-FORD

- Généralise Dijkstra : gère les poids négatifs
- Détecte la présence de cycle de poids négatif
- Complexité : $O(nm)$

Premier exercice : French dinner, swerc practice 2017

ALGORITHME DE BELLMAN-FORD

- Généralise Dijkstra : gère les poids négatifs
- Détecte la présence de cycle de poids négatif
- Complexité : $O(nm)$

Premier exercice : French dinner, swerc practice 2017

Permet de résoudre des systèmes d'inéquations linéaires de la forme $a - b \leq w$ en les modélisant par le graphe :

$$a \xleftarrow{w} b$$

FLOYD-WARSHALL

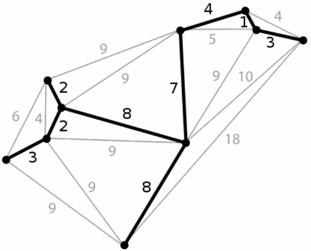
- Calcule le plus court chemin entre toute paire de sommets
- Complexité : $O(n^3)$
- Attention : gère les poids négatifs mais pas les circuits de poids négatifs !

Exercice utilisant une variante d'un de ces 3 algorithmes :
Environment-Friendly Travel SWERC 2019-2020

PLAN

ARBRE COUVRANT DE POIDS MINIMUM

- Arbre couvrant de G : sous-graphe de G qui est un arbre couvrant toutes ses arêtes
- De poids minimum : on choisit les arêtes de telles que leur poids total soit minimum
- Algorithme de Kruskal - complexité $O(m \log m)$



Exercice : Dark roads

PLAN

COMPOSANTES FORTEMENT CONNEXES

- Composantes fortement connexes (scc) : dans un graphe orienté, 2 sommets sont dans la même scc s'ils font parti d'un même circuit
- Peut se rencontrer dans plein de types de problèmes, à des endroits où on ne l'attend pas forcément (par exemple, test de satisfiabilité d'une formule 2-SAT)
- Algorithme classique : Tarjan (complexité linéaire)

Exercice : The Door Problem

2-SAT

- On a un ensemble de clauses de 2 littéraux : $x_i \vee x_j$
- Peut se ré-écrire : $\neg x_i \rightarrow x_j$ et $\neg x_j \rightarrow x_i$
- On construit un graphe des implications : les littéraux positifs et négatifs sont les sommets, les arcs sont les implications
- Une variable et sa négation sont dans la même composante fortement connexe si et seulement si la formule est insatisfiable

Exercice : The Door Problem