Éléments d'Algorithmique

Séance 3: Algorithmique des graphes, quelques exemples

Jérémie Jakubowicz

Récapitulatif de la précédente séance

La précédente séance avait été l'occasion d'aborder trois types de techniques qu'on avait implémentées sur trois problèmes :

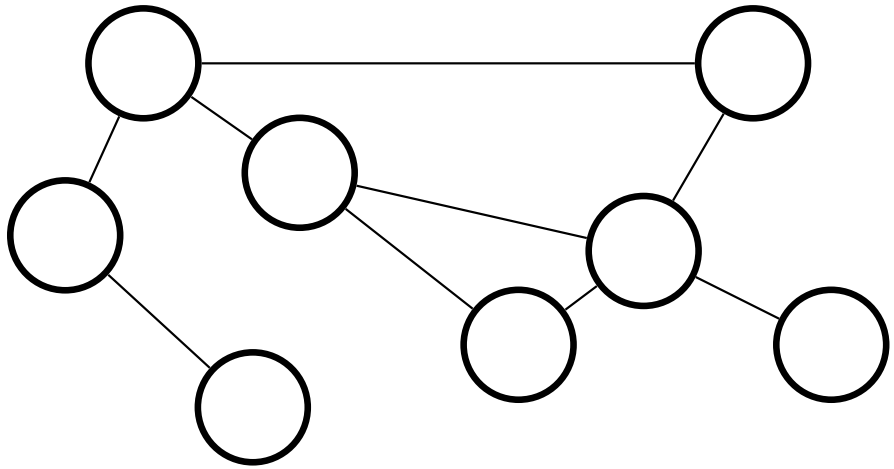
- L'approche par "force brute" qu'on avait implémentée pour le problème des contrexemples
- Les algorithmes gloutons, implémentés pour le problème de la couverture minimale
- La programmation dynamique, implémentée pour le problème "Soudoyer les prisonniers"

Cette séance : les graphes

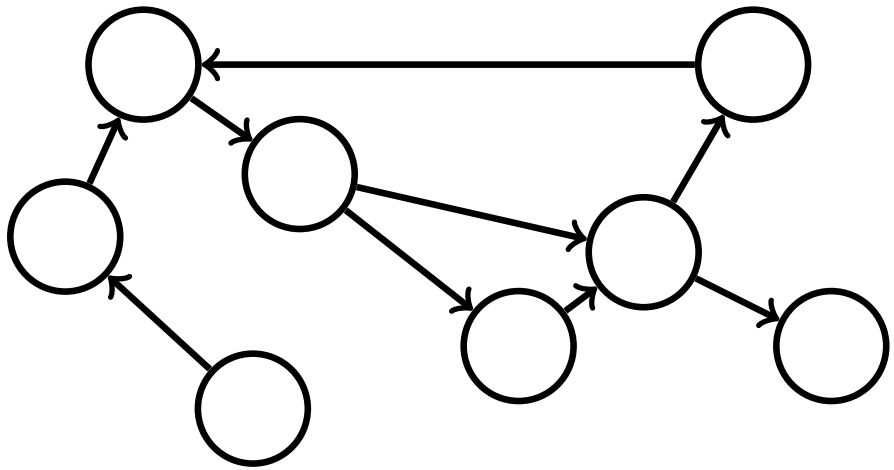
Qu'est-ce qu'un graphe ?
Il existe plusieurs sortes de graphes.

- Les graphes orientés/non-orientés
- Les graphes orientés acycliques (DAG)
- Les graphes avec des poids/labels sur les arêtes/noeuds

Les graphes en images (I)

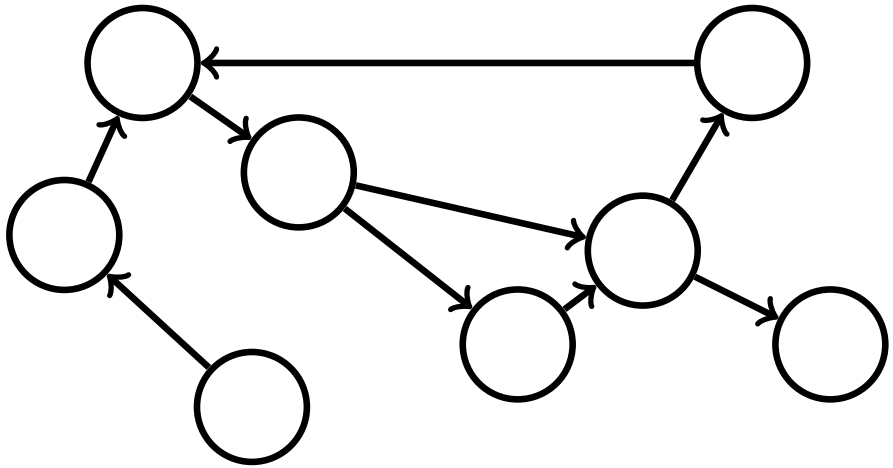


- Non-orienté

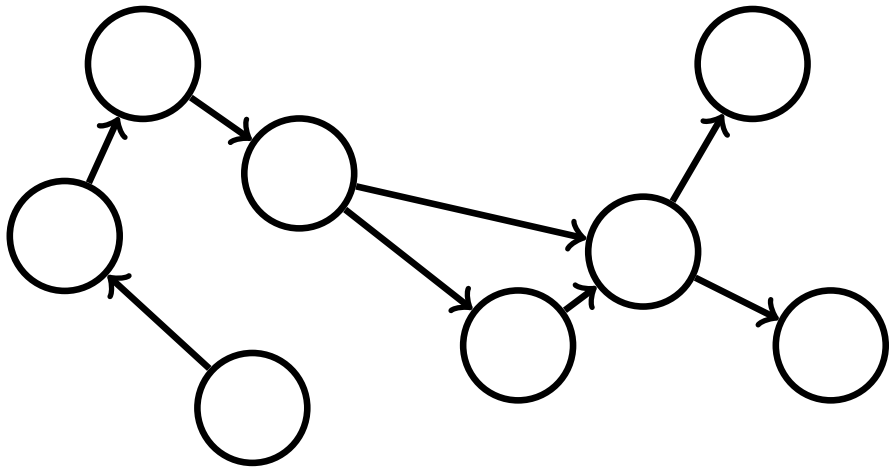


- Orienté

Les graphes en images (II)



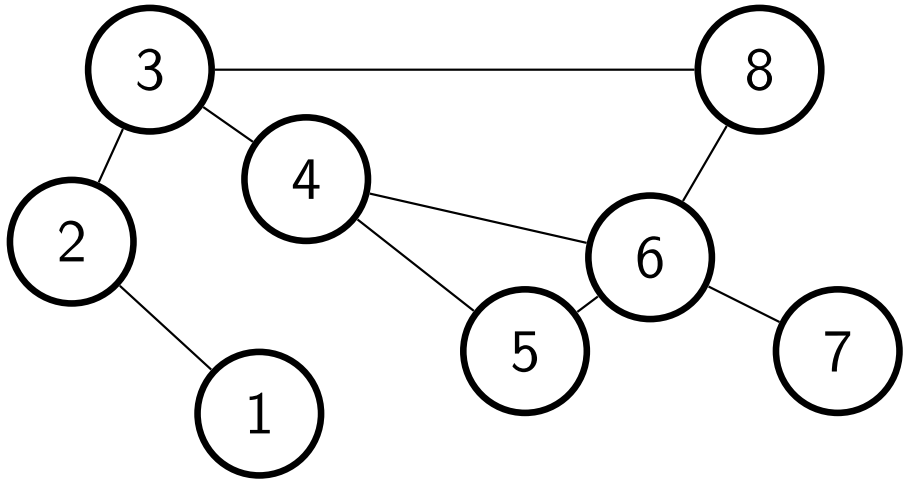
- Cyclique



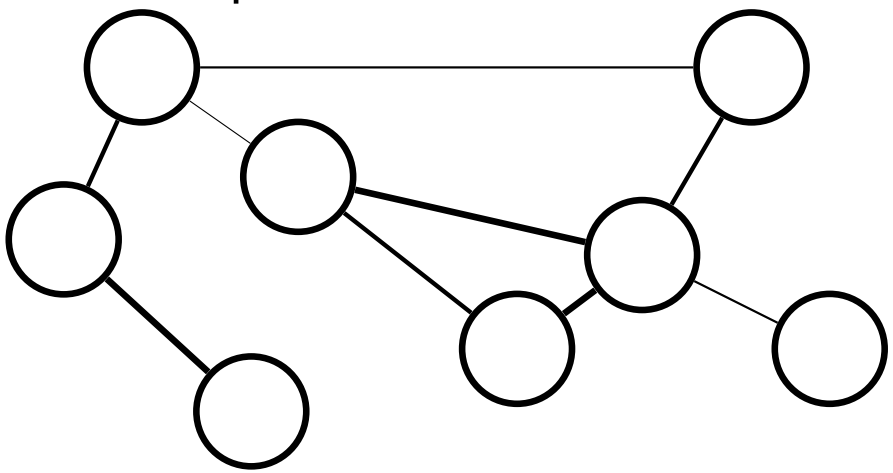
- Acyclique

Les graphes en images (III)

- Avec des labels sur les sommets



- Avec des poids sur les arêtes



Formellement

- Un graphe est défini grâce à un ensemble de sommets V et un ensemble d'arêtes E.
- Lorsque le graphe est orienté une arête est la donnée d'un couple de sommets $(v, w) \in V^2$.
- Lorsque le graphe est non-orienté une arête est la donnée d'un ensemble de deux sommets $\{v, w\} \subset V$.
- Les labels/poids sur les sommets sont donnés par des fonctions $x : V \to X$
- Les labels/poids sur les arêtes sont donnés par des fonctions $x : E \to X$

On pourrait aussi définir des multigraphes à travers une application $E \to V^2$ qui à chaque arête associe les deux sommets à son extremité.

Représentations

Les graphes sont classiquement représentés en mémoire :

- Soit grâce à une matrice d'adjacence $A : V \times V \to \mathbb{R}$
- Soit grâce à la liste des sommets adjacents pour chaque sommet du graphe

Problème 5 : GCJ 2009 QB - Partage des eaux

Les géologues divisent parfois un terrain en différentes régions suivant l'endroit où coulent les eaux de pluie. Ces régions sont appelées bassins de drainage.
Etant donné une carte d'élévation (une matrice d'altitude), étiquetez la carte de telle sorte que les cellules appartenant au même bassin de drainage partagent la même étiquette, en suivants les règles suivantes.
Depuis chaque cellule, l'eau coule vers au plus une de ses quatre cellules voisines. Si l'altitude des cellules voisines est plus élevée, la cellule est appelée un *puits* et l'eau ne coule pas du tout. Sinon l'eau coule vers la cellule voisine de plus basse altitude. En cas d'égalité, l'eau coule d'abord vers le Nord, puis vers l'Ouest, l'Est et enfin le Sud.
Chaque cellule qui draine directement ou indirectement vers le même puits fait partie du même bassin de drainage. Chaque bassin doit être étiqueté avec une lettre minuscule de telle sorte que deux bassins distincts possèdent deux étiquettes distinctes et que la chaîne de caractères résultante, lue ligne par ligne, soit la plus petite possible suivant l'ordre lexicographique.

Entrées/Sorties

Entrées

La première ligne du fichier d'entrées contient le nombre de cas à traiter, T. Ensuite chacune des T cartes est formatée de la manière suivante : une première ligne contient deux entiers H et W séparés par un espace qui correspondent à la hauteur et à la largeur de la matrice d'élévation ; enfin suivent H lignes qui correspondent à la matrice proprement dite.

Sorties

Dans chaque cas, $H + 1$ lignes sont affichées. La première ligne comporte le numéro du cas sous la forme "Cas #X :" et les H suivantes correspondent aux étiquettes des bassins en suivant le même ordre que pour les entrées.

Ordres de grandeurs

$T \leq 100$, $1 \leq H, W \leq 100$, $0 \leq$ altitudes $\leq 10^4$. Au plus 26 bassins.

Entrées/Sorties, Exemples

```
5
3 3
9 6 3
5 9 6
3 5 9
1 10
0 1 2 3 4 5 6 7 8 7
2 3
7 6 7
7 6 7
5 5
1 2 3 4 5
2 9 3 9 6
3 3 0 8 7
4 9 8 9 8
5 6 7 8 9
2 13
8 8 8 8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8
```

```
Cas #1 :
a b b
a a b
a a a
Cas #2 :
a a a a a a a a a a b
Cas #3 :
a a a
b b b
Cas #4 :
a a a a a
a a b b a
a b b b a
a b b b a
a a a a a
Cas #5 :
a b c d e f g h i j k l m
n o p q r s t u v w x y z
```

Des idées ?

Le parcours en profondeur Depth-First-Search (DFS)

On construit le graphe associé à la carte d'élévation et on parcours ce graphe à l'aide de l'algorithme suivant :

Algorithm 1 `solve`

Input: Graphe G, sommet initial v

Ouput: Composante connexe de v dans G

```
1: if card(N(v)) == 0 then
2:     return v
3: else
4:     w ← voisin(v)
5:     return solve(G,w)
```

Problème 6 : GCJ 2013 - Bad Horse

En tant que leader de la Ligue Maléfique du Mal, Bad Horse a beaucoup de problèmes à gérer. Ces derniers temps, il y a eu tellement de disputes et de perfidies que Bad Horse a décidé de couper la Ligue en deux ; de sorte à séparer les fauteurs de trouble. Il compte sur vous pour l'aider.

Entrées/Sorties

Entrées

La première ligne donne le nombre de cas, T. Ensuite pour chaque cas, la première ligne est un nombre entier M qui correspond au nombre de couples de fauteurs de troubles. Chacune des M lignes suivantes est composée de deux noms séparés par un espace.

Sorties

Pour chaque cas, la sortie consiste en une ligne unique "Cas #X : y" où X est le numéro du cas et y un booléen suivant que la Ligue peut être ou non séparée en deux parties qui ne contiennent aucun couple de fauteurs de trouble.

Ordres de grandeur

$1 \leq T \leq 100$, $1 \leq M \leq 100$

Entrées/Sorties, Exemples

2
1
Dead_Bowie Fake_Thomas_Jefferson
3
Dead_Bowie Fake_Thomas_Jefferson
Fake_Thomas_Jefferson Fury_Leika
Fury_Leika Dead_Bowie

Cas #1 : Yes
Cas #2 : No

Des idées ?

Le parcours en largeur Breadth-First-Search (BFS)

On construit le graphe de fauteurs de troubles, puis on parcourt le graphe en largeur en étiquettant le premier sommet 0, puis ses voisins 1, puis les voisins de ses voisins 0, jusqu'à étiqueter tout le graphe, auquel cas on répond "Vrai", ou rencontrer une inconsistence, auquel cas on répond "Faux".

Pour aller plus loin

Nous n'avons abordé qu'une toute petite partie du design d'algorithmes en laissant de côté un grand nombre de domaines. On n'a, par exemple, pas du tout parlé de :

1. De structures de données (*e.g.* deque, hashmap, suffix tree, ...)
2. D'algorithmes spécifiques aux chaînes de caractères (*e.g.* Knuth-Morris-Pratt)
3. D'optimisation combinatoire (branch & bound, maxflow/mincut)
4. De la majorité des algorithmes sur les graphes (coloriage, parcours, matchings, ...)
5. ...

Problème 7 : UVa 10911 - Former des équipes de deux

Soit (x, y) les coordonnées d'une maison d'étudiant dans un plan. Il y a $2N$ étudiants et on veut les apparier en N équipe de 2 de telle sorte que la somme des distances entre les maisons de paires d'étudiants soit minimale. Contraintes : $0 \leq x, y \leq 1000$.

Des idées ?

Pointeurs

- Livres
  - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to algorithms. MIT Press
  - Steven S. Skiena. The Algorithm Design Manual. Springer
  - Steven Halim. Competitive Programming.
- Problèmes en ligne
  - `https://code.google.com/codejam`
  - `http://www.topcoder.com/`
  - `http://uva.onlinejudge.org/`
- MOOCs
  - `https://www.coursera.org/course/algo`
  - `https://www.youtube.com/playlist?list=PL5F43156F3F22C349`