Introduction à l'informatique – graphe I

Table des matières

Introduction

Graphes -> Euler en 1736 avec les ponts de Königsberg

Est-il possible de partir d'un endroit x, de traverser tous les ponts une unique fois et de revenir en x ? -> Non. (Modélisation abstraite, chaque région est un sommet de graphe avec des droites pour les ponts qui les relient. p4 du PowerPoint)

Contexte et exemples :

- Domaine de l'informatique et des mathématiques (théorie des graphes)
- Objet interdisciplinaire (physique, chimie, biologie, sciences humaines)
- Développement théoriques et appliqués
- Représentation (ou modélisation) de la structure et des connexions d'un ensemble complexe.

➔ Besoin d'abstraction : enlever un certain nombre d'informations inutiles.

Exemple : plans ferroviaires, métro, modélisation d'internet en graphes, biologie, réseaux de scientifiques, etc.

Utilisation de graphes dans des objets a énigme (ex : théorème des 4 couleurs, le dodécaèdre d'Hamilton (problème des chemins/cycles Hamiltonien))

Graphe planaire = représentation des graphes où aucune droite ne se croise.

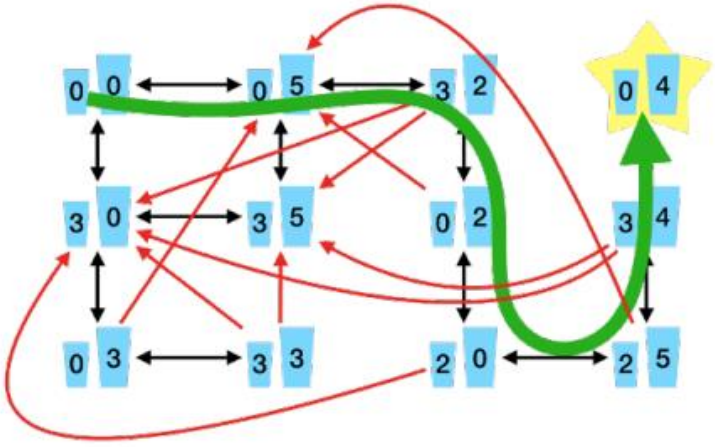
Formalisation

Graphes orientés :

- Un graphe orienté est un 2-uple G = (S, A) où :
  - S est un ensemble de sommets,
  - A c SxS est un ensemble d'arcs
  - sommets (nœuds) / arcs (arêtes)
- Exemple : Plan d'une ville
  - superposé un graphe des routes sur le plan d'un ville (avec sens unique ou double sens)
  - quel est le chemin le plus court entre un point A et B -> graphe et on trouve
- Représentation algébrique
  - représenter le graphe sous forme de matrice d'adjacence (quand un lien entre 2 sommet : 1, sinon 0 dans la matrice) ex de notation : $B(G)_{u,v} = 1\ sur\ (u,v) \in A \mid 0\ sinon$

Graphes non orientés (pas de direction sur les arcs) :

- Idée : partant d'un graphe orienté : ajoute des arcs symétriques (abstraire en enlevant les flèches)
  - depuis n'importe quelle graphe non orientés on peut prendre retrouver un graphe orientés mais un graphe orientés n'est pas tout le temps non orientés
  - ex : réseau social (symétrique ou non): graphes non orienté)
- Ex : expérience de Milgram
  - règle : passer les lettres à la main a des connaissances : observations : 232 des 296 lettres n'arrivent jamais. Moyenne d'envois par lettre pour arriver à destination : 6
  - fb : chaque personne est à distance en moyenne de 3.5 personnes de toutes les personnes.
- Notion de Diamètre d'un graphe
  - le plus long chemin entre 2 sommets tels qu'il ne contient pas de cycle (chemin qui revient sur le point initial en utilisant des arêtes différents)
  - boucle sur le même sommet : cycle de taille 1
- Objets a énigme pour les résoudre
  - ex : énigme des récipients (de {0, 0} à {0, 4}



Algorithmes

- Algorithme pour trouver un chemin
  - il faut parcourir le graphe (abstraction)
- Parcours en largeur
  - on veut partir d'un point A et passer par tous les sommets du graphe.
  - représentation en file (le premier rentrer dois être le premier sortis -> liste)
  - premier élément, puis voisins directs du premier élément, sortir le premier élément et voir les différents voisins (arcs sortant) du nouveau premier élément. (si un élément est déjà dans la file on ne le remet pas)
  - deux opérations : (on utilise plutôt des listes chainées pour représenter une file/pile
    - enfiler(F, x) (.append)
    - x := Défiler(F) (.pop)
  - algorithme du parcours en largeur : (voir cours d'erik demaine du MIT)

```
Fonction parcours_largeur(d G[1..n,1..n] : tab_entier, d s : entier) : tab_entier
    i : entier ;      H : tab_entier ;
    F : file_entier ;      couleur[1..n] : tab_couleur ;
Début
    Pour i de 1 à n faire      couleur[i] := blanc ;      Fin faire
    couleur[s] := vert ;      F := ∅ ;
    enfiler(F, s) ;
    Tant que (F ≠ ∅) faire
        u := défiler(F) ;      couleur[u] := rouge ;
        Pour v de 1 à n faire
            Si (G[u,v] = 1 et couleur[v] = blanc) alors
                couleur[v] := vert ;
                H[u,v] := 1 ;
                enfiler(F, v) ;
            Fin si
        Fin faire
        couleur[u] := doré ;
    Fin faire
    retour H ;
Fin
```

- Parcours en profondeur
  - Je prends un des voisins du premier terme et je réitère le processus sur lui.
  - une fois un chemin validé on revient un cran en arrière et on refait
  - une pile au lieu d'une file.

- Graphes pondérés
  - on va mettre des poids sur des arcs
  - L'algorithme de parcours en largeur ne garantit pas d'obtenir un chemin de poids minimal dans un graphe pondéré
  - algo trouvé par dijkstra
  - file de priorité : je fais un parcours dans ma file pour ordonner par ordre croissant)