

Graphes : Représentations

Quentin Fortier

February 2, 2022

Structure abstraite graphe

On souhaite implémenter une structure de graphe possédant les opérations :

- ➊ ajouter / supprimer une arête
- ➋ (ajouter / supprimer un sommet)
- ➌ savoir s'il existe une arête entre 2 sommets
- ➍ connaître la liste des voisins d'un sommet
- ➎ ...

Avec, si possible, une faible complexité en temps et espace.

Type abstrait graphe

Exemple de type abstrait de graphe :

```
type 'v graph = { (* 'v est le type des sommets *)  
  add_edge : 'v -> 'v -> unit;  
  del_edge : 'v -> 'v -> unit;  
  edge : 'v -> 'v -> bool;  
  n : int; (* nombre de sommets *)  
  adj : 'v -> 'v list (* liste des sommets adjacents *)  
}
```

Les sommets seront des entiers consécutifs à partir de 0.

Matrice d'adjacence

On peut représenter un graphe non orienté (V, E) , où $V = \{0, \dots, n-1\}$ par une **matrice d'adjacence** M de taille $n \times n$ définie par :

- $M_{i,j} = 1 \iff \{i, j\} \in E$
- $M_{i,j} = 0 \iff \{i, j\} \notin E$

Remarque : M est symétrique.

Matrice d'adjacence

On peut représenter un graphe non orienté (V, E) , où $V = \{0, \dots, n-1\}$ par une **matrice d'adjacence** M de taille $n \times n$ définie par :

- $M_{i,j} = 1 \iff \{i, j\} \in E$
- $M_{i,j} = 0 \iff \{i, j\} \notin E$

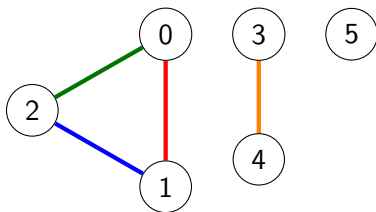
Remarque : M est symétrique.

Pour un graphe orienté (V, \vec{E}) :

- $M_{i,j} = 1 \iff (i, j) \in \vec{E}$
- $M_{i,j} = 0 \iff (i, j) \notin \vec{E}$

M n'est pas symétrique (a priori).

Exemple de matrice d'adjacence (non orienté)



$$\begin{pmatrix} 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Quitte à permuter lignes et colonnes (i.e renuméroter les sommets), les composantes connexes apparaissent par bloc.

Matrice d'adjacence

Si on représente un graphe orienté à n sommets et m arêtes par matrice d'adjacence m :

- complexité en espace : $\Theta(n^2)$
- ajouter arête $\{u, v\}$: $m.(u).(v) \leftarrow 1$, $O(1)$
- supprimer arête $\{u, v\}$: $m.(u).(v) \leftarrow 0$, $O(1)$
- ajouter / supprimer sommet : impossible de modifier un array
- test d'existence d'arête : $m.(i).(j) = 1$, $O(1)$
- parcourir les voisins d'un sommet : parcourir $m.(i)$, $\Theta(n)$

Pour un graphe non orienté, il faut modifier $m.(u).(v)$ et $m.(v).(u)$.

Matrice d'adjacence

`create_adj_mat` n renvoie un graphe orienté à n sommets (et 0 arête) représenté par matrice d'adjacence :

```
let create_adj_graph n =  
  let m = Array.make_matrix n n 0 in {  
    add_edge = (fun u v -> m.(u).(v) <- 1);  
    del_edge = (fun u v -> m.(u).(v) <- 0);  
    edge = (fun u v -> m.(u).(v) = 1);  
    n = n;  
    adj = (fun u ->  
      let rec aux v =  
        if v = n then []  
        else if m.(u).(v) = 1 then v::aux (v + 1)  
        else aux (v + 1) in  
      aux 0  
    )  
  }
```

Puissance de la matrice d'adjacence

Question

Si $A = (a_{u,v})$ est une matrice d'adjacence d'un graphe à n sommets, que représente les coefficients de $A^k = (a_{u,v}^{(k)})$?

Puissance de la matrice d'adjacence

Question

Si $A = (a_{u,v})$ est une matrice d'adjacence d'un graphe à n sommets, que représente les coefficients de $A^k = (a_{u,v}^{(k)})$?

Pour $k = 2$:

$$a_{u,v}^2 = \sum_{w=0}^{n-1} a_{u,w} a_{w,v}$$

Puissance de la matrice d'adjacence

Question

Si $A = (a_{u,v})$ est une matrice d'adjacence d'un graphe à n sommets, que représente les coefficients de $A^k = (a_{u,v}^{(k)})$?

Pour $k = 2$:

$$a_{u,v}^2 = \sum_{w=0}^{n-1} a_{u,w} a_{w,v}$$

$$a_{u,w} a_{w,v} = 1 \iff u \rightarrow w \rightarrow v \text{ est un chemin}$$

$a_{u,v}^2$ est le nombre de chemins de longueur 2 de u à v !

Puissance de la matrice d'adjacence

Question

Si $A = (a_{u,v})$ est une matrice d'adjacence d'un graphe à n sommets, que représente les coefficients de $A^k = (a_{u,v}^{(k)})$?

$$a_{u,v}^{(k)} = \sum_{w=0}^{n-1} a_{u,w}^{(k-1)} a_{w,v}$$

Puissance de la matrice d'adjacence

Question

Si $A = (a_{u,v})$ est une matrice d'adjacence d'un graphe à n sommets, que représente les coefficients de $A^k = (a_{u,v}^{(k)})$?

$$a_{u,v}^{(k)} = \sum_{w=0}^{n-1} a_{u,w}^{(k-1)} a_{w,v}$$

Par récurrence sur k :

$$a_{u,v}^{(k)} = \text{nombre de chemins de longueur } k \text{ de } u \text{ à } v$$

Remarque : c'est vrai aussi bien pour les graphes orientés que non-orientés.

Puissance de la matrice d'adjacence

Soit $M(n)$ la complexité pour multiplier 2 matrices $n \times n$

Puissance de la matrice d'adjacence

Soit $M(n)$ la complexité pour multiplier 2 matrices $n \times n$
($M(n) = \Theta(n^3)$ en naïf, $O(n^{2,8})$ avec la méthode de Strassen).

On peut calculer $A^k = A \times \dots \times A$ en $O(kM(n))$.

Puissance de la matrice d'adjacence

Soit $M(n)$ la complexité pour multiplier 2 matrices $n \times n$
($M(n) = \Theta(n^3)$ en naïf, $O(n^{2,8})$ avec la méthode de Strassen).

On peut calculer $A^k = A \times \dots \times A$ en $O(kM(n))$.

Ou, mieux, par exponentiation rapide en utilisant :

$$\begin{cases} A^k = (A^{\frac{k}{2}})^2 & \text{si } k \text{ est pair} \\ A^k = A(A^{\frac{k-1}{2}})^2 & \text{sinon} \end{cases}$$

Complexité :

Puissance de la matrice d'adjacence

Soit $M(n)$ la complexité pour multiplier 2 matrices $n \times n$
($M(n) = \Theta(n^3)$ en naïf, $O(n^{2,8})$ avec la méthode de Strassen).

On peut calculer $A^k = A \times \dots \times A$ en $O(kM(n))$.

Ou, mieux, par exponentiation rapide en utilisant :

$$\begin{cases} A^k = (A^{\frac{k}{2}})^2 & \text{si } k \text{ est pair} \\ A^k = A(A^{\frac{k-1}{2}})^2 & \text{sinon} \end{cases}$$

Complexité : $O(\log(k)M(n))$.

Liste d'adjacence

La représentation par **liste d'adjacence** consiste à stocker, pour chaque sommet, la liste de ses voisins.

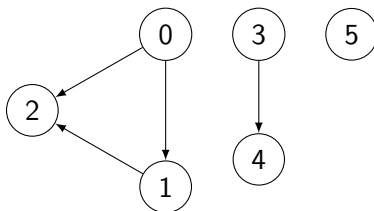
Liste d'adjacence

La représentation par **liste d'adjacence** consiste à stocker, pour chaque sommet, la liste de ses voisins.

Deux possibilités :

- 1 une liste de listes (**int list list**) $l_0::l_1::l_2::\dots$ où l_0 est la liste des voisins du sommet 0, l_1 est la liste des voisins du sommet 1...
- 2 un **tableau de listes** (**int list array**) t où $t.(i)$ est la liste des voisins du sommet i .

Exemple de liste d'adjacence int list array (orienté)



```
#let g = [| [1; 2]; [2]; []; [4]; []; [] |];;  
g : int list vect = [| [1; 2]; [2]; []; [4]; []; [] |]
```

Liste d'adjacence

`create_adj_list n` renvoie un graphe **orienté** à n sommets (et aucune arête) représenté par liste d'adjacence :

```
let create_adj_list n =  
  let g = Array.make n [] in {  
    add_edge = (fun u v -> g.(u) <- v::g.(u));  
    del_edge = (fun u v -> List.filter ((<>) u) g.(v));  
    edge = (fun u v -> List.mem v g.(u));  
    n = n;  
    adj = (fun u -> g.(u))  
  }
```

Comparaison

Pour un graphe orienté à n sommets et m arêtes :

	array array	list array	list list
espace	$\Theta(n^2)$		
ajouter (u, v)	$O(1)$		
supprimer (u, v)	$O(1)$		
existence (u, v)	$O(1)$		
voisins de u	$\Theta(n)$		
ajouter sommet	X		
supprimer sommet	X		

Comparaison

Pour un graphe orienté à n sommets et m arêtes :

	array array	list array	list list
espace	$\Theta(n^2)$	$\Theta(n + m)$	$\Theta(n + m)$
ajouter (u, v)	$O(1)$	$O(1)$	$O(n)$
supprimer (u, v)	$O(1)$	$O(\deg^+(u))$	$O(n)$
existence (u, v)	$O(1)$	$O(\deg^+(u))$	$O(n)$
voisins de u	$\Theta(n)$	$\Theta(\deg^+(u))$	$O(n)$
ajouter sommet	X	X	$O(n)$
supprimer sommet	X	X	$O(n)$

Comparaison

Pour un graphe orienté à n sommets et m arêtes :

	array array	list array	list list
espace	$\Theta(n^2)$	$\Theta(n + m)$	$\Theta(n + m)$
ajouter (u, v)	$O(1)$	$O(1)$	$O(n)$
supprimer (u, v)	$O(1)$	$O(\deg^+(u))$	$O(n)$
existence (u, v)	$O(1)$	$O(\deg^+(u))$	$O(n)$
voisins de u	$\Theta(n)$	$\Theta(\deg^+(u))$	$O(n)$
ajouter sommet	X	X	$O(n)$
supprimer sommet	X	X	$O(n)$

Si $m = \Theta(n^2)$ (graphe dense) : matrice d'adjacence conseillée.

Si $m = O(n)$ (graphe creux, ex : arbre) : liste d'adjacence conseillée.

Exercice

Ecrire deux fonctions pour convertir une matrice d'adjacence en liste d'adjacence et vice-versa.

Dictionnaire d'adjacence

Représentation plus générale : un dictionnaire qui à chaque sommet associe l'ensemble de ses voisins, de type `('v, 'v set) dico`.

On peut utiliser n'importe quelle implémentation de `dico` et `set`, et n'importe quel type de sommet compatible avec ces implémentations.

Dictionnaire d'adjacence

Graphe orienté à n sommets et m arêtes représenté par un ('a, 'a set) dico avec la même implémentation de set et dico :

	hashtbl	AVL
type sommet	hachable	ordonné
espace	$\Theta(n + m)$	$\Theta(n + m)$
ajouter (u, v)	$O(1)$ moyenne	$O(\log(n))$
supprimer (u, v)	$O(1)$ moyenne	$O(\log(n))$
existence (u, v)	$O(1)$ moyenne	$O(\log(n))$
voisins de u	$\Theta(\deg^+(u))$ moyenne	$\Theta(\deg^+(u))$
ajouter sommet	$O(1)$ moyenne	$O(\log(n))$
supprimer sommet	$O(n)$ moyenne	$O(n)$