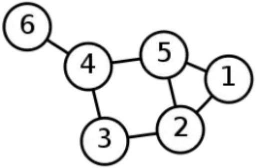
Algorithmique de graphe

Rakotoarimalala Tsinjo Tony

Introduction Graphe

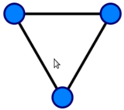
Définition

- Wikipedia: un graphe est une structure composée d'objets dans laquelle certaines paires d'objets sont en relation. Les objets correspondent à des abstractions mathématiques et sont appelés sommets (ou nœuds ou points), et les relations entre sommets sont des arêtes (ou liens ou lignes).

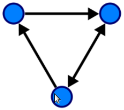


Définitions

- **Un graphe simple non orienté est un couple $G = (E, V))$ avec**
  - V un ensemble de sommets (aussi appelés nœuds ou points)
  - $E \subset \{(x, y)|(x, y) \in V^2, x \neq y\}$ un ensemble d'arêtes



- **Un graphe simple orienté est un couple $G = (E, V))$ avec**
  - V un ensemble de sommets (aussi appelés nœuds ou points)
  - $E \subset \{(x, y)|(x, y) \in V^2, x \neq y\}$ un ensemble de flèches ou arcs.



Graphe non orienté

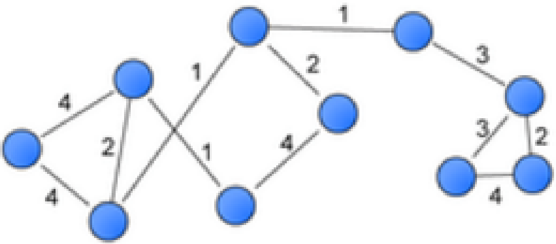
- Une boucle est une arête partant d'un sommet vers lui-même
- Si deux sommets sont liés par une arête alors on dit que les deux sommets sont **adjacents** ou encore **voisins**, on note $x \sim y$
- Le degré d'un sommet est le nombre de voisins de ce sommet

Graphe orienté

- Une boucle est une arête partant d'un sommet vers lui-même
- Dans la flèche (x, y) orientée de x vers y, x est appelé la queue de la flèche et y la tête de la flèche. La flèche (y, x) est appelée la flèche inverse de (x, y).

Graphe pondéré

- Un graphe pondéré ou un réseau est un graphe où chaque arête porte un nombre (son poids)
- Ces poids peuvent représenter par exemple des coûts, des longueurs ou des capacités, en fonction du problème traité.



Exemple d’objets modélisables avec un graphe

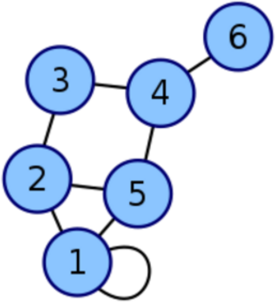
D’après vous ?

Matrice d'adjacence

- Supposons que $G = (V, E)$ est un graphe simple, où $|V| = n$.
- Supposons aussi que les sommets de G sont numérotés arbitrairement $v_1, v_2, \ldots v_n$.
- La matrice d'adjacence A de G se rapportant à cet ensemble de sommets est la matrice $n \times n$ booléenne A avec

$$a_{ij} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \\ 0 & \text{sinon} \end{cases}$$

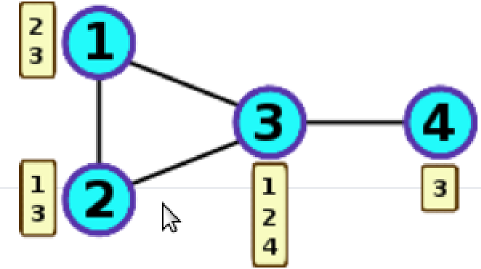
Matrice d'adjacence



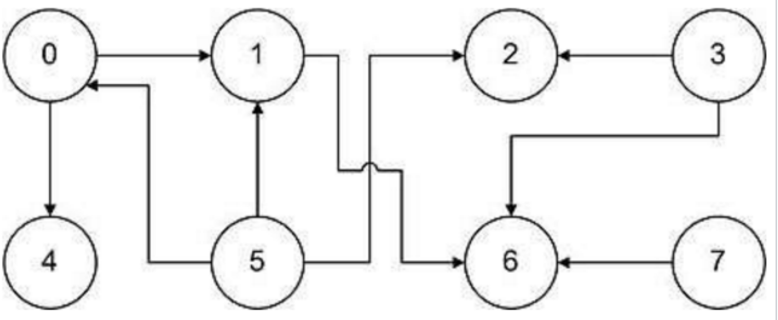
$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Liste d'adjacence

- **La liste d'adjacence d'un graphe non orienté, est la liste des voisins de chaque sommet**



Représenter le graphe suivant avec une matrice et une liste

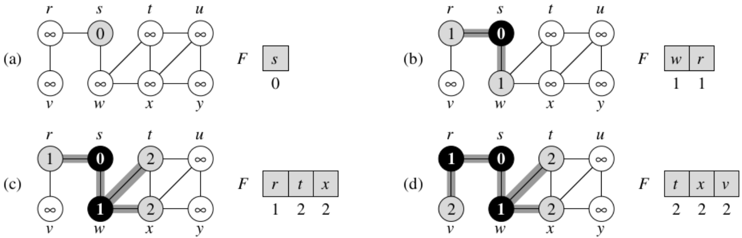


Parcours en largeur

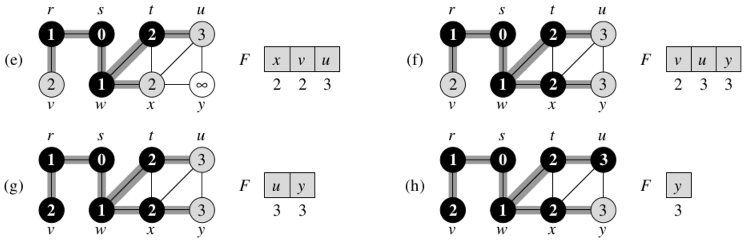
Algorithm Parcours en largeur (BFS)

```
procedure BFS(G, s)          ▷ G: graphe, s: sommet de depart
    visited[s] ← true
    Q ← file vide
    Enfiler(Q, s)
    while Q n'est pas vide do
        v ← Défiler(Q)
        for chaque voisin u de v dans G do
            if u n'est pas visite then
                Marquer u comme visite
                Enfiler(Q, u)
            end if
        end for
    end while
end procedure
```

BFS



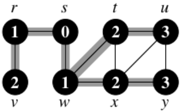
BFS



(e)
r
s
t
u
1
0
2
3
2
1
2
∞
v
w
x
y
F
x
v
u
2
2
3
(f)
r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
F
v
u
y
2
3
3
(g)
r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
F
u
y
3
3
(h)
r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
F
y
3

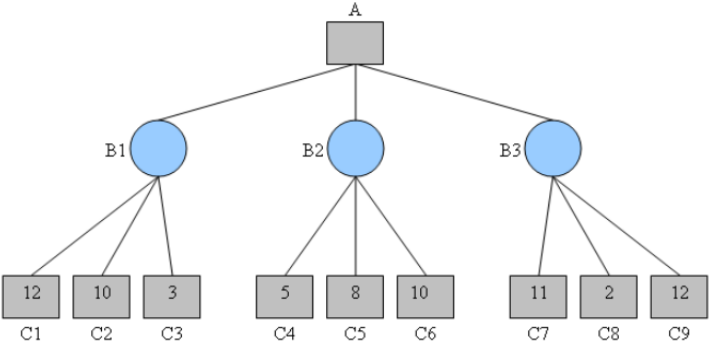
BFS

(i)

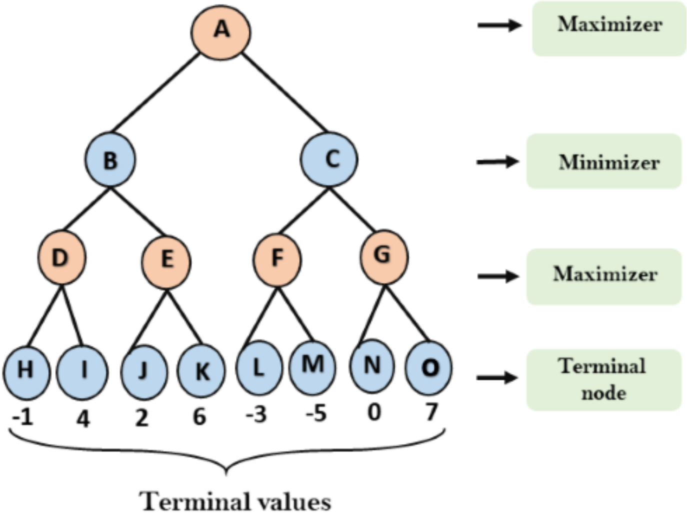


F Ze

Arbre de jeu simple



Arbre de jeu plus compliqué



Minimax

Algorithm Minimax

```
function MINIMAX(état, profondeur, estMax)
    if profondeur = 0 ou état est terminal then
        return la valeur heuristique de état
    end if
    if estMax then
        v ← −∞
        for chaque mouvement valide m depuis état do
            v ← max(v, Minimax(m, profondeur − 1, Faux))
        end for
        return v
    else
        v ← +∞
        for chaque mouvement valide m depuis état do
            v ← min(v, Minimax(m, profondeur − 1, Vrai))
        end for
        return v
    end if
end function
```