

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Aspect historique

- C'est le mathématicien suisse Leonhard Euler (1707-1783) qui est à l'origine de la création de la théorie des graphes.

Définition

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Aspect historique

- C'est le mathématicien suisse Leonhard Euler (1707-1783) qui est à l'origine de la création de la théorie des graphes.
- Il en pose les bases en résolvant le problème des 7 ponts de Königsberg en 1740.

Définition

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Aspect historique

- C'est le mathématicien suisse Leonhard Euler (1707-1783) qui est à l'origine de la création de la théorie des graphes.
- Il en pose les bases en résolvant le problème des 7 ponts de Königsberg en 1740.
- Les graphes interviennent à présent dans de nombreux problèmes (recherche de chemins, réseau, ...) en informatique comme en mathématiques.

Définition

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Aspect historique

- C'est le mathématicien suisse Leonhard Euler (1707-1783) qui est à l'origine de la création de la théorie des graphes.
- Il en pose les bases en résolvant le problème des 7 ponts de Königsberg en 1740.
- Les graphes interviennent à présent dans de nombreux problèmes (recherche de chemins, réseau, ...) en informatique comme en mathématiques.

Définition

Un **graphe orienté** est la donnée :

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Aspect historique

- C'est le mathématicien suisse Leonhard Euler (1707-1783) qui est à l'origine de la création de la théorie des graphes.
- Il en pose les bases en résolvant le problème des 7 ponts de Königsberg en 1740.
- Les graphes interviennent à présent dans de nombreux problèmes (recherche de chemins, réseau, ...) en informatique comme en mathématiques.

Définition

Un **graphe orienté** est la donnée :

- D'un ensemble de **sommets** S (V pour *vertex* en anglais.).

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Aspect historique

- C'est le mathématicien suisse Leonhard Euler (1707-1783) qui est à l'origine de la création de la théorie des graphes.
- Il en pose les bases en résolvant le problème des 7 ponts de Königsberg en 1740.
- Les graphes interviennent à présent dans de nombreux problèmes (recherche de chemins, réseau, ...) en informatique comme en mathématiques.

Définition

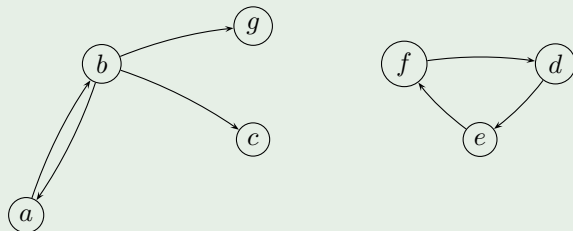
Un **graphe orienté** est la donnée :

- D'un ensemble de **sommets** S (V pour *vertex* en anglais.).
- D'un ensemble de couples de sommets $A \subseteq S \times S$ appelés **arc** (notés $x \rightarrow y$). (E pour *edges* en anglais).

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

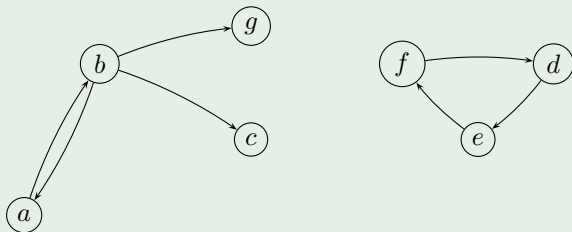
Exemple



C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

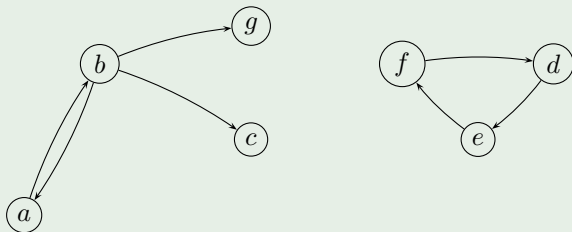


$$S = \{a, b, c, d, e, f, g\}$$

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple



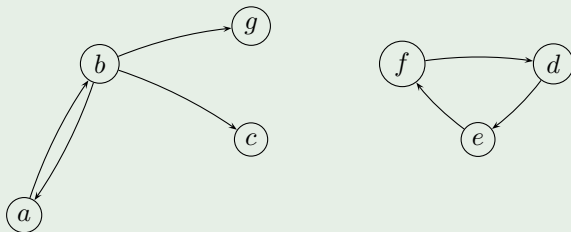
$$S = \{a, b, c, d, e, f, g\}$$

$$A = \{(e, f), (d, e), (f, d), (a, b), (b, a), (b, c), (b, g)\}$$

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple



$$S = \{a, b, c, d, e, f, g\}$$

$$A = \{(e, f), (d, e), (f, d), (a, b), (b, a), (b, c), (b, g)\}$$

⚠ Seule la données de S et A défini le graphe et *pas* les positions des sommets sur le schéma.

Vocabulaire

- Le **degré** d'un graphe est son nombre de sommets.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Vocabulaire

- Le **degré** d'un graphe est son nombre de sommets.
- Un arc de la forme (x, x) est une **boucle**.
- Le **degré sortant** d'un sommet s noté $d_+(s)$ est le nombre d'arcs de la forme (s, x) .

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Vocabulaire

- Le **degré** d'un graphe est son nombre de sommets.
- Un arc de la forme (x, x) est une **boucle**.
- Le **degré sortant** d'un sommet s noté $d_+(s)$ est le nombre d'arcs de la forme (s, x) .
- Le **degré entrant** d'un sommet s noté $d_-(s)$ est le nombre d'arcs de la forme (x, s) .

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Vocabulaire

- Le **degré** d'un graphe est son nombre de sommets.
- Un arc de la forme (x, x) est une **boucle**.
- Le **degré sortant** d'un sommet s noté $d_+(s)$ est le nombre d'arcs de la forme (s, x) .
- Le **degré entrant** d'un sommet s noté $d_-(s)$ est le nombre d'arcs de la forme (x, s) .
- Un **chemin** de longueur n du sommet s au sommet t dans un graphe (S, A) est une séquence x_0, \dots, x_n de sommets tels que $x_0 = s$, $x_n = t$ et $(x_i, x_{i+1}) \in A$ pour $i \in \llbracket 0; n-1 \rrbracket$.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Vocabulaire

- Le **degré** d'un graphe est son nombre de sommets.
- Un arc de la forme (x, x) est une **boucle**.
- Le **degré sortant** d'un sommet s noté $d_+(s)$ est le nombre d'arcs de la forme (s, x) .
- Le **degré entrant** d'un sommet s noté $d_-(s)$ est le nombre d'arcs de la forme (x, s) .
- Un **chemin** de longueur n du sommet s au sommet t dans un graphe (S, A) est une séquence x_0, \dots, x_n de sommets tels que $x_0 = s$, $x_n = t$ et $(x_i, x_{i+1}) \in A$ pour $i \in \llbracket 0; n-1 \rrbracket$.
- Un chemin est **simple** lorsqu'il est sans répétition d'arcs.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

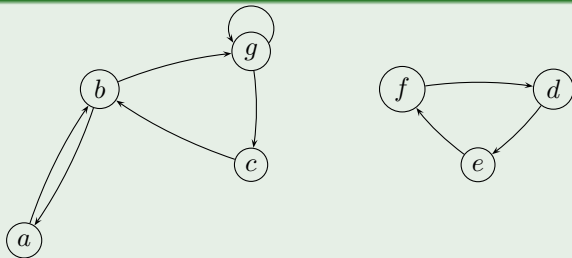
Vocabulaire

- Le **degré** d'un graphe est son nombre de sommets.
- Un arc de la forme (x, x) est une **boucle**.
- Le **degré sortant** d'un sommet s noté $d_+(s)$ est le nombre d'arcs de la forme (s, x) .
- Le **degré entrant** d'un sommet s noté $d_-(s)$ est le nombre d'arcs de la forme (x, s) .
- Un **chemin** de longueur n du sommet s au sommet t dans un graphe (S, A) est une séquence x_0, \dots, x_n de sommets tels que $x_0 = s$, $x_n = t$ et $(x_i, x_{i+1}) \in A$ pour $i \in \llbracket 0; n-1 \rrbracket$.
- Un chemin est **simple** lorsqu'il est sans répétition d'arcs.
- Un **cycle** est un chemin simple d'un sommet à lui même.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

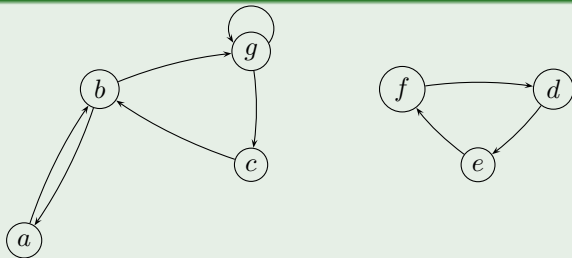


- Donner $d_+(b)$.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

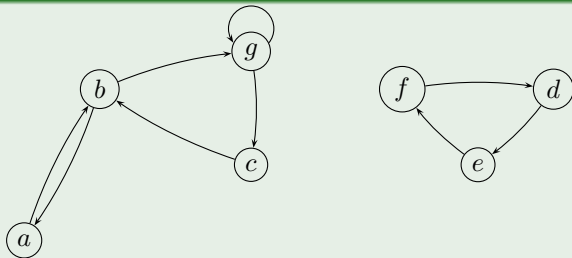


- Donner $d_+(b)$.
- Donner $d_-(g)$.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple



- Donner $d_+(b)$.
- Donner $d_-(g)$.
- Donner un chemin sans cycle de q à a .

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Définition

Un **graphe non orienté** est la donnée :

Vocabulaire

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Définition

Un **graphe non orienté** est la donnée :

- D'un ensemble de **sommets** ou **noeuds** S .

Vocabulaire

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Définition

Un **graphe non orienté** est la donnée :

- D'un ensemble de **sommets ou noeuds** S .
- D'un ensemble de **paires** de sommets A appelés **arcs ou arêtes** notés $x - y$.

Vocabulaire

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Définition

Un **graphe non orienté** est la donnée :

- D'un ensemble de **sommets ou noeuds** S .
- D'un ensemble de **paires** de sommets A appelés **arcs ou arêtes** notés $x - y$.

Vocabulaire

- Dans le contexte des graphes orientés cela revient à $(x, y) \in A$ ssi $(y, x) \in A$.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Définition

Un **graphe non orienté** est la donnée :

- D'un ensemble de **sommets ou noeuds** S .
- D'un ensemble de **paires** de sommets A appelés **arcs ou arêtes** notés $x - y$.

Vocabulaire

- Dans le contexte des graphes orientés cela revient à $(x, y) \in A$ ssi $(y, x) \in A$.
- Les définitions de chemin, degrés, ... des graphes orientés s'étendent naturellement aux graphes non orientés.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Définition

Un **graphe non orienté** est la donnée :

- D'un ensemble de **sommets ou noeuds** S .
- D'un ensemble de **paires** de sommets A appelés **arcs ou arêtes** notés $x - y$.

Vocabulaire

- Dans le contexte des graphes orientés cela revient à $(x, y) \in A$ ssi $(y, x) \in A$.
- Les définitions de chemin, degrés, ... des graphes orientés s'étendent naturellement aux graphes non orientés.
- On dit qu'un graphe non orienté (S, A) est **connexe** lorsqu'il existe un chemin entre toute paire de sommets.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Graphes pondérés

- Dans de nombreuses situations, on est amené à attacher une information aux arcs d'un graphe (ex : distance entre deux villes, coût d'une liaison dans un réseau informatique, ...), on parle alors de **graphe pondéré**.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Graphes pondérés

- Dans de nombreuses situations, on est amené à attacher une information aux arcs d'un graphe (ex : distance entre deux villes, coût d'une liaison dans un réseau informatique, ...), on parle alors de **graphe pondéré**.
- L'information, ou **étiquette** attaché à un noeud est souvent de nature numérique, on parle alors de **poids** ou **longueur** d'un arc.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Graphes pondérés

- Dans de nombreuses situations, on est amené à attacher une information aux arcs d'un graphe (ex : distance entre deux villes, coût d'une liaison dans un réseau informatique, ...), on parle alors de **graphe pondéré**.
- L'information, ou **étiquette** attaché à un noeud est souvent de nature numérique, on parle alors de **poids** ou **longueur** d'un arc.
- Le coût d'un chemin est alors la somme des poids des arcs qui le compose.

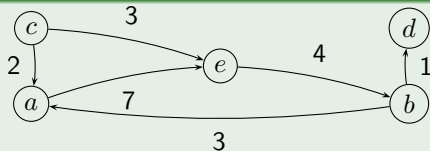
C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Graphes pondérés

- Dans de nombreuses situations, on est amené à attacher une information aux arcs d'un graphe (ex : distance entre deux villes, coût d'une liaison dans un réseau informatique, ...), on parle alors de **graphe pondéré**.
- L'information, ou **étiquette** attaché à un noeud est souvent de nature numérique, on parle alors de **poids** ou **longueur** d'un arc.
- Le coût d'un chemin est alors la somme des poids des arcs qui le compose.

Exemple



C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par matrice d'adjacence

On peut représenter un graphe à n sommets par sa **matrice d'adjacence** M , c'est à dire un tableau de n lignes et n colonnes :

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par matrice d'adjacence

On peut représenter un graphe à n sommets par sa **matrice d'adjacence** M , c'est à dire un tableau de n lignes et n colonnes :

- On numérote les sommets du graphe

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par matrice d'adjacence

On peut représenter un graphe à n sommets par sa **matrice d'adjacence** M , c'est à dire un tableau de n lignes et n colonnes :

- On numérote les sommets du graphe
- S'il y a une arête du sommet i vers le sommet j alors on place un 1 à la ligne i et à la colonne j de M

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par matrice d'adjacence

On peut représenter un graphe à n sommets par sa **matrice d'adjacence** M , c'est à dire un tableau de n lignes et n colonnes :

- On numérote les sommets du graphe
- S'il y a une arête du sommet i vers le sommet j alors on place un 1 à la ligne i et à la colonne j de M
- Sinon on place un 0

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par matrice d'adjacence

On peut représenter un graphe à n sommets par sa **matrice d'adjacence** M , c'est à dire un tableau de n lignes et n colonnes :

- On numérote les sommets du graphe
- S'il y a une arête du sommet i vers le sommet j alors on place un 1 à la ligne i et à la colonne j de M
- Sinon on place un 0

Remarques

- Si le graphe n'est pas orienté alors la matrice est symétrique par rapport à sa première diagonale.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par matrice d'adjacence

On peut représenter un graphe à n sommets par sa **matrice d'adjacence** M , c'est à dire un tableau de n lignes et n colonnes :

- On numérote les sommets du graphe
- S'il y a une arête du sommet i vers le sommet j alors on place un 1 à la ligne i et à la colonne j de M
- Sinon on place un 0

Remarques

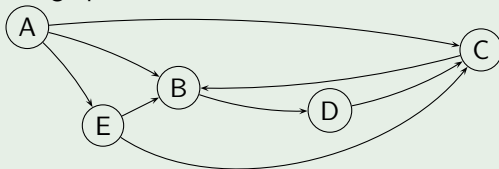
- Si le graphe n'est pas orienté alors la matrice est symétrique par rapport à sa première diagonale.
- On peut représenter les graphes pondérés en écrivant le poids à la place du 1 pour chaque arête.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

- ① En supposant les sommets numérotés dans l'ordre alphabétique, écrire la matrice d'adjacence du graphe suivant :

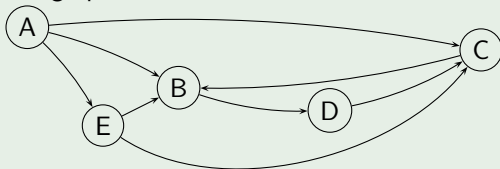


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

- ① En supposant les sommets numérotés dans l'ordre alphabétique, écrire la matrice d'adjacence du graphe suivant :



- ② Dessiner le graphe ayant la matrice d'adjacence suivante (on appellera les sommets S_1, S_2, \dots) :

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

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par listes d'adjacence

On peut représenter un graphe à l'aide de listes d'adjacences, c'est à dire en mémorisant pour chaque sommet du graphe la liste de ses voisins.

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par listes d'adjacence

On peut représenter un graphe à l'aide de listes d'adjacences, c'est à dire en mémorisant pour chaque sommet du graphe la liste de ses voisins.

- On crée pour chaque sommet du graphe une liste

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par listes d'adjacence

On peut représenter un graphe à l'aide de listes d'adjacences, c'est à dire en mémorisant pour chaque sommet du graphe la liste de ses voisins.

- On crée pour chaque sommet du graphe une liste
- S'il y a une arête du sommet S_i vers le sommet S_j alors S_j est dans la liste de S_i

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par listes d'adjacence

On peut représenter un graphe à l'aide de listes d'adjacences, c'est à dire en mémorisant pour chaque sommet du graphe la liste de ses voisins.

- On crée pour chaque sommet du graphe une liste
- S'il y a une arête du sommet S_i vers le sommet S_j alors S_j est dans la liste de S_i

Remarques

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par listes d'adjacence

On peut représenter un graphe à l'aide de listes d'adjacences, c'est à dire en mémorisant pour chaque sommet du graphe la liste de ses voisins.

- On crée pour chaque sommet du graphe une liste
- S'il y a une arête du sommet S_i vers le sommet S_j alors S_j est dans la liste de S_i

Remarques

- Lorsqu'un graphe a "peu" d'arête cette implémentation est plus intéressante en terme d'occupation mémoire que celle par matrice d'adjacence.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Représentation par listes d'adjacence

On peut représenter un graphe à l'aide de listes d'adjacences, c'est à dire en mémorisant pour chaque sommet du graphe la liste de ses voisins.

- On crée pour chaque sommet du graphe une liste
- S'il y a une arête du sommet S_i vers le sommet S_j alors S_j est dans la liste de S_i

Remarques

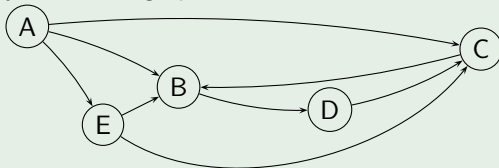
- Lorsqu'un graphe a "peu" d'arête cette implémentation est plus intéressante en terme d'occupation mémoire que celle par matrice d'adjacence.
- En Python, on utilisera un dictionnaire pour représenter les listes d'adjacences, les clés sont les sommets et les valeurs les listes associées

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

- 1 Ecrire les listes d'adjacences du graphe suivante :

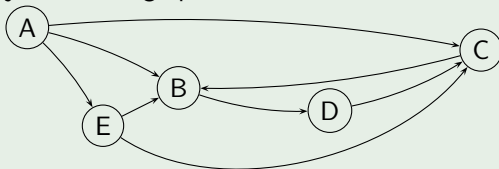


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple

- ① Ecrire les listes d'adjacences du graphe suivante :



- ② Dessiner le graphe représenté par le dictionnaire Python suivante :

```
{  
    'A' : ['C'],  
    'B' : ['D', 'E'],  
    'C' : ['A', 'B'],  
    'D' : ['A', 'C'],  
    'E' : ['B', 'C', 'D']  
}
```

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Parcours d'un graphe

A la base des algorithmes sur les graphes, on trouve les parcours de graphe, c'est à dire l'exploration des sommets. A partir du sommet de départ, on peut :

- explorer tous ses voisins immédiats, puis les voisins des voisins et ainsi de suite. Le graphe est donc exploré en « cercle concentrique » autour du sommet de départ . . . , on parle alors de **parcours en largeur** ou **breadth first search** (*BFS*) en anglais.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Parcours d'un graphe

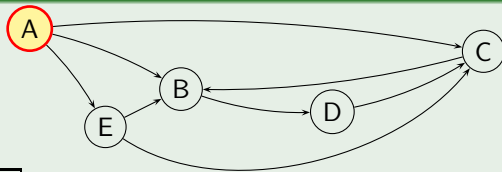
A la base des algorithmes sur les graphes, on trouve les parcours de graphe, c'est à dire l'exploration des sommets. A partir du sommet de départ, on peut :

- explorer tous ses voisins immédiats, puis les voisins des voisins et ainsi de suite. Le graphe est donc exploré en « cercle concentrique » autour du sommet de départ . . . , on parle alors de **parcours en largeur** ou **breadth first search** (*BFS*) en anglais.
- explorer à chaque étape le premier voisin non encore exploré. Lorsque qu'on atteint un sommet dont tous les voisins ont déjà été exploré, on revient en arrière, on parle alors de **parcours en profondeur** ou **depth first search** (*DFS*) en anglais.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en largeur

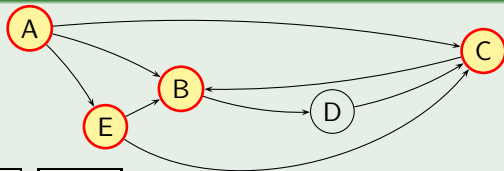


Sommets explorés : A

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en largeur

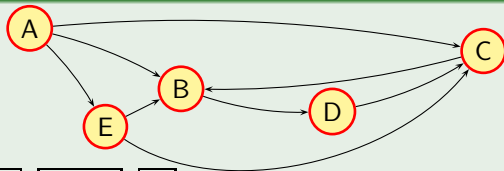


Sommets explorés : A, B,C,E

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en largeur

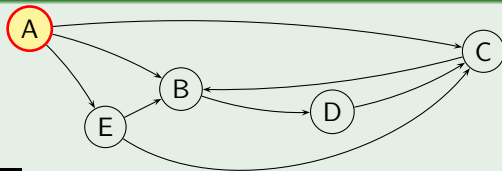


Sommets explorés : A, B,C,E, D.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en profondeur

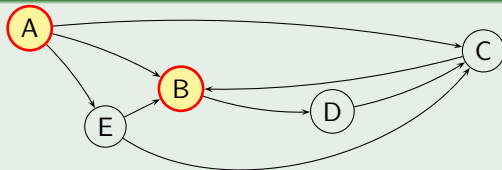


Sommets explorés : A

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en profondeur

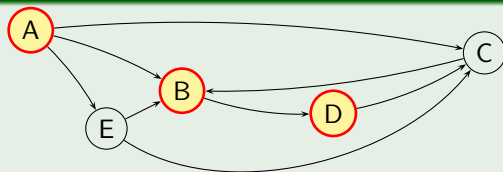


Sommets explorés : A, B

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en profondeur

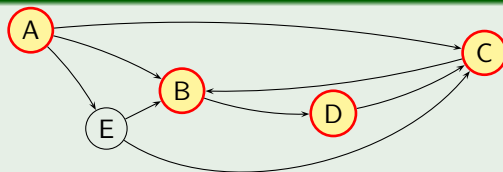


Sommets explorés : A, B, D

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en profondeur

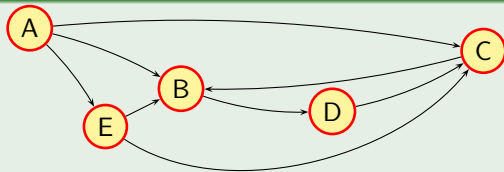


Sommets explorés : A, B, D, C

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Exemple de parcours en profondeur



Sommets explorés : A, B, D, C, E

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins . . . Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins . . . Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins . . . Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.

<i>E</i>	<i>D</i>	<i>C</i>	<i>B</i>
----------	----------	----------	----------

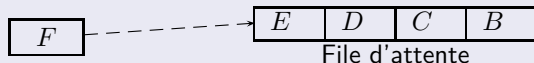
File d'attente

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.

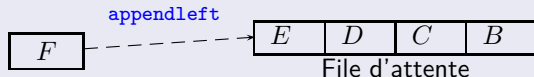


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.

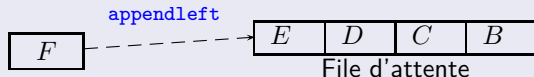


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.

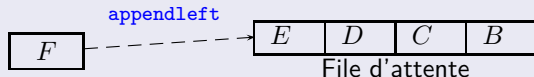


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.

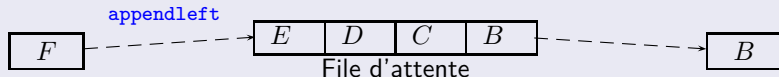


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.

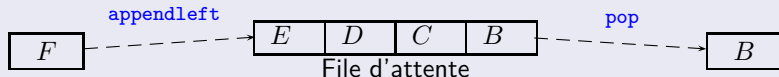


C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en largeur

- Pour un parcours en largeur, on doit stocker dans une structure de données les sommets en attente d'être explorés. C'est à dire les voisins du sommet de départ, puis les voisins des voisins ... Ces sommets doivent être retirés pour exploration, dans leur ordre d'insertion, la structure de données utilisée est donc du type **premier entré, premier sorti** (first in first out (*FIFO*)).
- Ce type de structure de données s'appelle une **file**.
- Pour l'implémentation on doit pouvoir **enfiler** (ajouter un sommet dans la file) et **défiler** (retirer un sommet) de façon efficace donc en $O(1)$.
- Les listes de Python ne sont pas adaptées, on utilisera le module **deque** de Python, enfiler correspond alors à **appendleft** et défiler à **pop**.



C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en profondeur

- Pour un parcours en profondeur, on stocker aussi dans une structure de données les sommets en attente d'être explorés. Mais cette fois, la structure de données utilisée est donc du type **dernier entré, premier sorti** (last in first out (*LIFO*)).

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

File et parcours en profondeur

- Pour un parcours en profondeur, on stocker aussi dans une structure de données les sommets en attente d'être explorés. Mais cette fois, la structure de données utilisée est donc du type **dernier entré, premier sorti** (last in first out (*LIFO*)).
- Ce type de structure de données s'appelle une **pile**.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

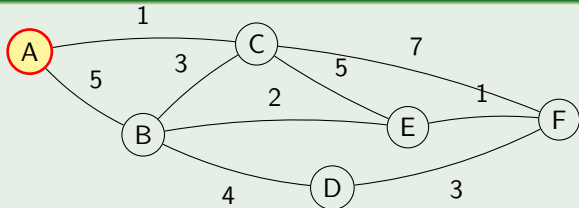
File et parcours en profondeur

- Pour un parcours en profondeur, on stocker aussi dans une structure de données les sommets en attente d'être explorés. Mais cette fois, la structure de données utilisée est donc du type **dernier entré, premier sorti** (last in first out (*LIFO*)).
- Ce type de structure de données s'appelle une **pile**.
- Pour l'implémentation, on se contente d'utiliser la récursivité de façon à ce que la pile des sommets en attente d'être exploré soit gérée de façon automatique par les appels récursifs.

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

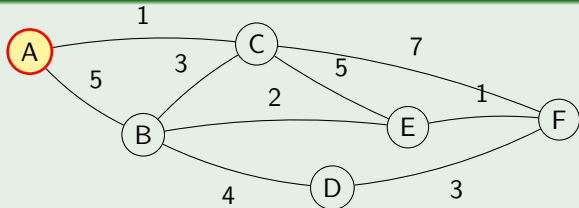


A	B	C	D	E	F	

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

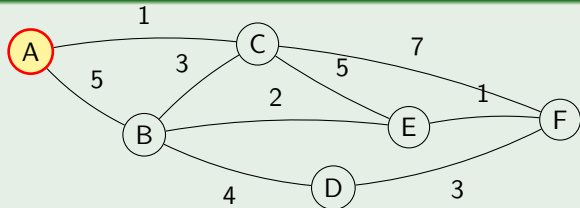


A	B	C	D	E	F	
0 (A)						

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

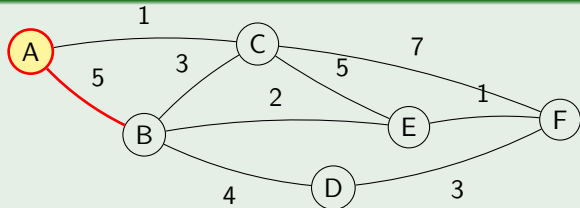


A	B	C	D	E	F	
0 (A)						A
✓						
✓						
✓						
✓						
✓						

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

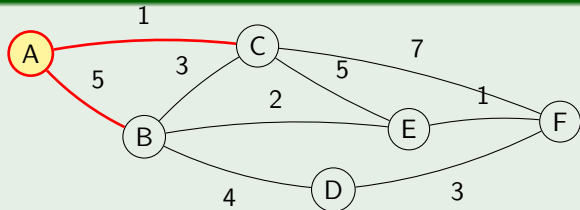


A	B	C	D	E	F	
0 (A)	5 (A)					A
✓						
✓						
✓						
✓						
✓						

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

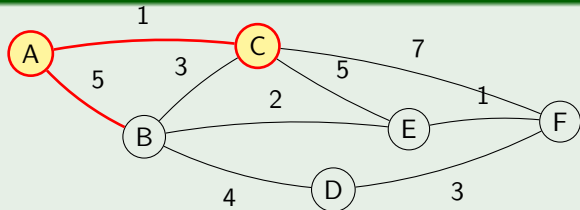


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓						
✓						
✓						
✓						
✓						

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

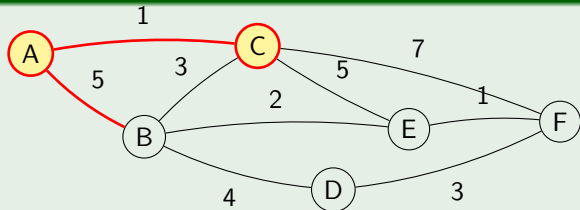


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓		1 (A)				
✓						
✓						
✓						
✓						

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

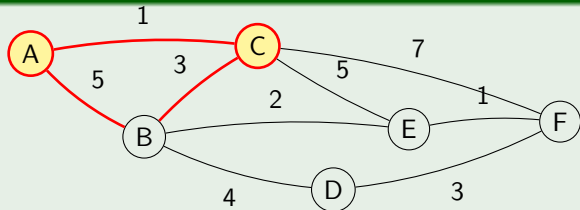


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓		1 (A)				C
✓		✓				
✓		✓				
✓		✓				
✓		✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

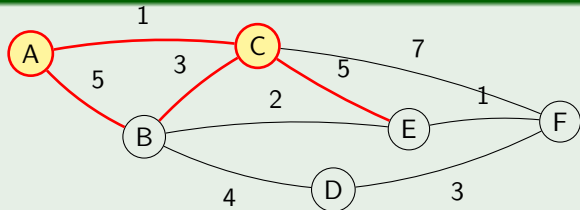


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)				C
✓		✓				
✓		✓				
✓		✓				
✓		✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

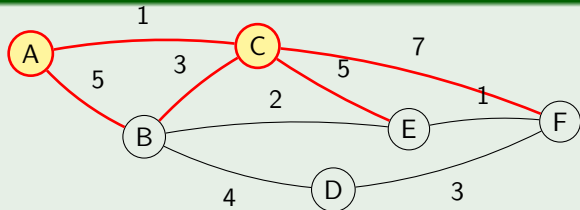


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)		C
✓		✓				
✓		✓				
✓		✓				
✓		✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

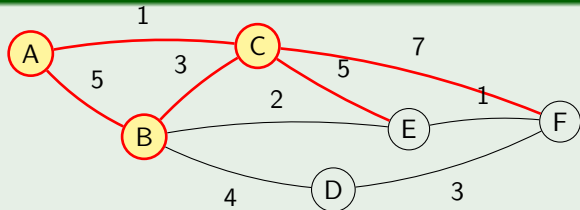


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓		✓				
✓		✓				
✓		✓				
✓		✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

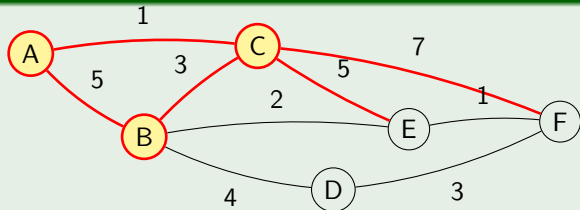


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓				
✓		✓				
✓		✓				
✓		✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

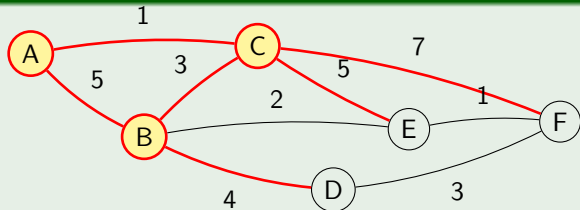


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓				B
✓	✓	✓				
✓	✓	✓				
✓	✓	✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

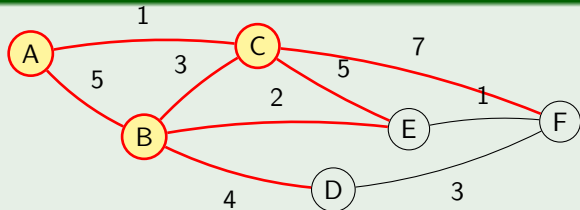


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)			B
✓	✓	✓				
✓	✓	✓				
✓	✓	✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

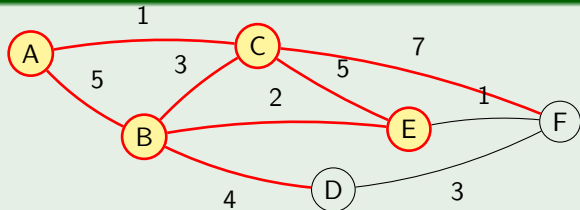


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓				
✓	✓	✓				
✓	✓	✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

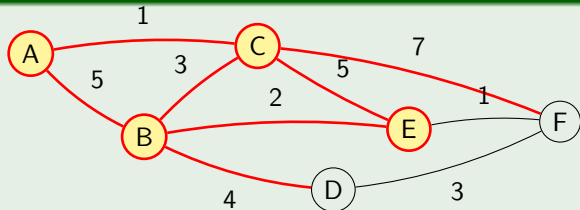


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)		
✓	✓	✓				
✓	✓	✓				

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

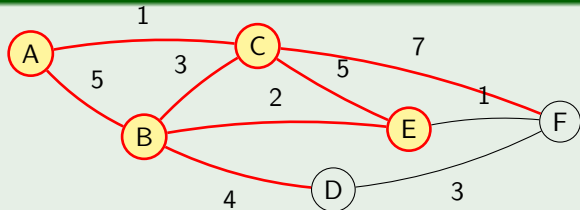


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)		E
✓	✓	✓		✓		
✓	✓	✓		✓		

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

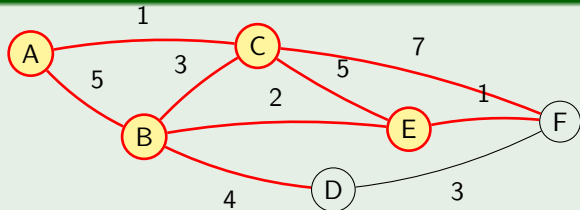


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)		E
✓	✓	✓		✓		
✓	✓	✓		✓		

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

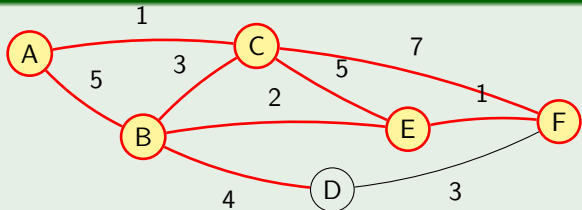


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)	7 (E)	E
✓	✓	✓		✓		
✓	✓	✓		✓		

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

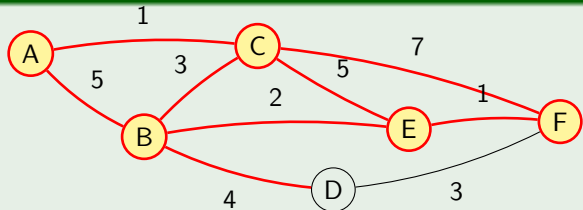


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)	7 (E)	E
✓	✓	✓		✓	7 (E)	
✓	✓	✓		✓	✓	

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

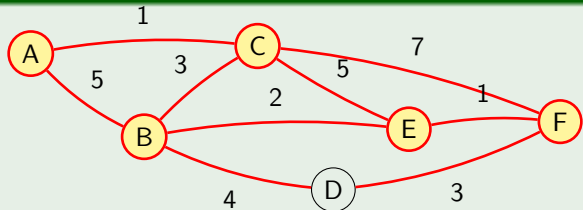


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)	7 (E)	E
✓	✓	✓		✓	7 (E)	F
✓	✓	✓		✓	✓	

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

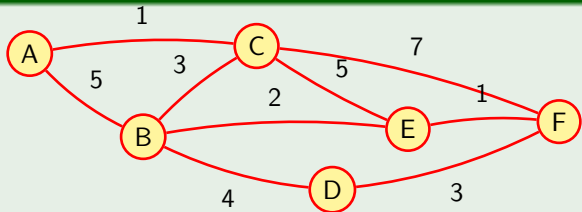


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)	7 (E)	E
✓	✓	✓	11 (F)	✓	7 (E)	F
✓	✓	✓		✓	✓	

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple

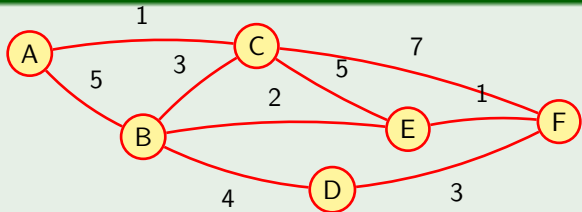


A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)	7 (E)	E
✓	✓	✓	11 (F)	✓	7 (E)	F
✓	✓	✓	8 (B)	✓	✓	

C4 Algorithmique pour l'IA et l'étude des jeux

1. Rappels sur les graphes

Algorithme de Dijkstra : exemple



A	B	C	D	E	F	
0 (A)	5 (A)	1 (A)				A
✓	4 (C)	1 (A)		6 (C)	8 (C)	C
✓	4 (C)	✓	8 (B)	6 (B)		B
✓	✓	✓		6 (B)	7 (E)	E
✓	✓	✓	11 (F)	✓	7 (E)	F
✓	✓	✓	8 (B)	✓	✓	D

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Généralités

- On s'intéresse maintenant à des jeux à deux joueurs ou chaque joueur joue en alternance jusqu'à la victoire de l'un deux ou un match nul. On supposera que :

Généralités

- On s'intéresse maintenant à des jeux à deux joueurs ou chaque joueur joue en alternance jusqu'à la victoire de l'un des deux ou un match nul. On supposera que :
 - le jeu est à **somme nulle**, c'est à dire que la somme des pertes et des gains des deux joueurs vaut 0,

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Généralités

- On s'intéresse maintenant à des jeux à deux joueurs ou chaque joueur joue en alternance jusqu'à la victoire de l'un des deux ou un match nul. On supposera que :
 - le jeu est à **somme nulle**, c'est à dire que la somme des pertes et des gains des deux joueurs vaut 0,
 - le jeu est à **information complète** c'est à dire que les joueurs disposent de toutes les informations pour effectuer leur choix.

Généralités

- On s'intéresse maintenant à des jeux à deux joueurs ou chaque joueur joue en alternance jusqu'à la victoire de l'un des deux ou un match nul. On supposera que :
 - le jeu est à **somme nulle**, c'est à dire que la somme des pertes et des gains des deux joueurs vaut 0,
 - le jeu est à **information complète** c'est à dire que les joueurs disposent de toutes les informations pour effectuer leur choix.

Le jeu du morpion, des dames ou des échecs sont des exemples bien connus.

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Généralités

- On s'intéresse maintenant à des jeux à deux joueurs ou chaque joueur joue en alternance jusqu'à la victoire de l'un des deux ou un match nul. On supposera que :
 - le jeu est à **somme nulle**, c'est à dire que la somme des pertes et des gains des deux joueurs vaut 0,
 - le jeu est à **information complète** c'est à dire que les joueurs disposent de toutes les informations pour effectuer leur choix.

Le jeu du morpion, des dames ou des échecs sont des exemples bien connus.

- Un tel jeu se modélise par **un graphe orienté biparti** $G = (S, A)$ c'est à dire un graphe dans lequel on peut partitionner l'ensemble des sommets en deux ensembles disjoints S_1 et S_2 tels que chaque arc possède une extrémité dans S_1 (états du jeu contrôlés par le joueur 1) et l'autre dans S_2 (états du jeu contrôlés par le joueur 2).

Exemple : le jeu de Juniper Green

On choisit un entier N , puis le jeu ne comporte que trois règles.

Exemple : le jeu de Juniper Green

On choisit un entier N , puis le jeu ne comporte que trois règles.

- 1 Le joueur 1 choisit un *nombre pair* entre 1 et N .

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Exemple : le jeu de Juniper Green

On choisit un entier N , puis le jeu ne comporte que trois règles.

- 1 Le joueur 1 choisit un *nombre pair* entre 1 et N .
- 2 A tour de rôle, chaque joueur doit choisir un nombre dans $\llbracket 1; N \rrbracket$ qui est un multiple ou un diviseur du nombre choisi au tour précédent.

Exemple : le jeu de Juniper Green

On choisit un entier N , puis le jeu ne comporte que trois règles.

- 1 Le joueur 1 choisit un *nombre pair* entre 1 et N .
- 2 A tour de rôle, chaque joueur doit choisir un nombre dans $\llbracket 1; N \rrbracket$ qui est un multiple ou un diviseur du nombre choisi au tour précédent.
- 3 Un nombre ne peut-être joué qu'une seule fois.

Exemple : le jeu de Juniper Green

On choisit un entier N , puis le jeu ne comporte que trois règles.

- 1 Le joueur 1 choisit un *nombre pair* entre 1 et N .
- 2 A tour de rôle, chaque joueur doit choisir un nombre dans $\llbracket 1; N \rrbracket$ qui est un multiple ou un diviseur du nombre choisi au tour précédent.
- 3 Un nombre ne peut-être joué qu'une seule fois.

Un joueur perd dès qu'il ne peut plus jouer, c'est donc un jeu sans match nul.

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Exemple : le jeu de Juniper Green

On choisit un entier N , puis le jeu ne comporte que trois règles.

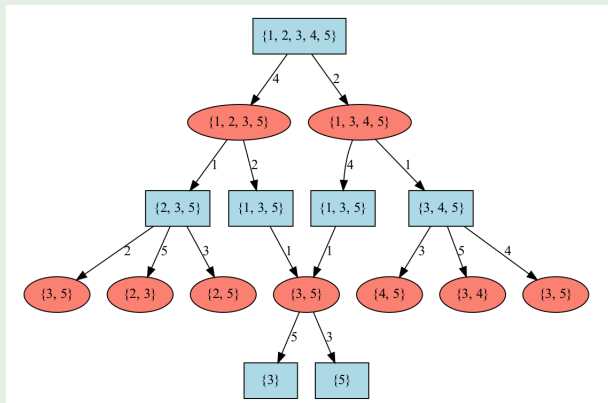
- 1 Le joueur 1 choisit un *nombre pair* entre 1 et N .
- 2 A tour de rôle, chaque joueur doit choisir un nombre dans $\llbracket 1; N \rrbracket$ qui est un multiple ou un diviseur du nombre choisi au tour précédent.
- 3 Un nombre ne peut-être joué qu'une seule fois.

Un joueur perd dès qu'il ne peut plus jouer, c'est donc un jeu sans match nul. Dans le cas simple ou $N = 5$, on peut tracer le graphe complet du jeu

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

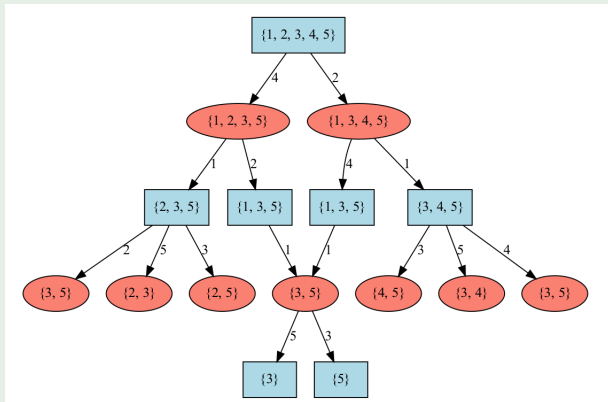
Graphe de Juniper-Green pour $N = 5$



C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Graphe de Juniper-Green pour $N = 5$



En partant des états finaux, on peut remonter dans le graphe et déterminer les positions gagnantes pour un joueur donné. Ce sont les **attracteurs** pour ce joueur.

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Définition des attracteurs

On note W_1 l'ensemble des états terminaux gagnants pour le joueur 1, on note A_i l'ensemble des positions gagnantes pour le joueur 1 qui permettent une victoire en au plus i coups. Alors :

$$\begin{cases} A_0 &= W_1 \\ A_{i+1} &= A_i \cup \{v \in V_2 \text{ ayant tous les arcs sortants dans } A_i\} \\ &\quad \cup \{v \in V_1 \text{ ayant au moins un arc sortant dans } A_i\} \end{cases}$$

Stratégie gagnante

Un **stratégie** pour le joueur i est une application de V dans V et une stratégie est **gagnante** si elle conduit à la victoire du joueur quelle que soit les coups de l'adversaire. Lorsqu'il existe une stratégie gagnante à partir d'un état du jeu, cet état est une **position gagnante**.

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Exemple

Dans le jeu de Juniper Green avec $N = 5$, la position de départ est gagnante pour le joueur 2 (si le joueur 1 joue 2, il joue 4 et inversement, cela impose au joueur 1 de jouer 1 ce qui conduit à sa défaite).

On voit donc qu'une position gagnante pour le joueur 2, n'est pas forcément une position où c'est à lui de jouer.

C4 Algorithmique pour l'IA et l'étude des jeux

2. Jeux à deux joueurs

Exemple

Dans le jeu de Juniper Green avec $N = 5$, la position de départ est gagnante pour le joueur 2 (si le joueur 1 joue 2, il joue 4 et inversement, cela impose au joueur 1 de jouer 1 ce qui conduit à sa défaite).

On voit donc qu'une position gagnante pour le joueur 2, n'est pas forcément une position où c'est à lui de jouer.

Remarque

Le calcul explicite des attracteurs et donc la détermination d'une stratégie gagnante n'est plus possible lorsque le nombre de sommets du graphe est trop important. Pour $N = 24$, il y a déjà 540 568 sommets dans le graphe du jeu de Juniper Green ! Pour les échecs le nombre de sommets est estimé à environ 10^{50} ...

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .
- Le but du jouer 1 va être alors de rechercher la position qui **maximimise** h .

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .
- Le but du jouer 1 va être alors de rechercher la position qui **maximimise** h .
- Alors que le jouer 2 va rechercher la position qui **minimise** h .

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .
- Le but du jouer 1 va être alors de rechercher la position qui **maximimise** h .
- Alors que le jouer 2 va rechercher la position qui **minimise** h .
- L'algorithme fixe un niveau maximal de profondeur de recherche dans le graphe du jeu et remonte à la racine une valeur minmax calculée récursivement par :

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .
- Le but du jouer 1 va être alors de rechercher la position qui **maximimise** h .
- Alors que le jouer 2 va rechercher la position qui **minimise** h .
- L'algorithme fixe un niveau maximal de profondeur de recherche dans le graphe du jeu et remonte à la racine une valeur minmax calculée récursivement par :
 - $\text{minmax}(s) = h(s)$ si s est une feuille où que s est à la profondeur maximale.

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .
- Le but du jouer 1 va être alors de rechercher la position qui **maximimise** h .
- Alors que le jouer 2 va rechercher la position qui **minimise** h .
- L'algorithme fixe un niveau maximal de profondeur de recherche dans le graphe du jeu et remonte à la racine une valeur minmax calculée récursivement par :
 - $\text{minmax}(s) = h(s)$ si s est une feuille où que s est à la profondeur maximale.
 - $\text{minmax}(s) = \max(\text{minmax}(t_1, \dots, t_n))$ si $s \in S_1$ (où $(s, t_i) \in A$)

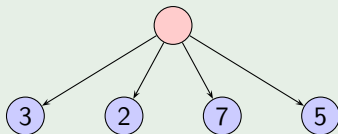
C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Principe

- Lorsqu'il n'est pas envisageable d'explorer la totalité des états du jeu, on se donne une **heuristique** c'est à dire une fonction h permettant d'évaluer un état du jeu telle que $h(s_1) > h(s_2)$ si le joueur 1 considère que l'état s_1 est plus favorable pour lui que l'état s_2 .
- Le but du jouer 1 va être alors de rechercher la position qui **maximimise** h .
- Alors que le jouer 2 va rechercher la position qui **minimise** h .
- L'algorithme fixe un niveau maximal de profondeur de recherche dans le graphe du jeu et remonte à la racine une valeur minmax calculée récursivement par :
 - $\text{minmax}(s) = h(s)$ si s est une feuille où que s est à la profondeur maximale.
 - $\text{minmax}(s) = \max(\text{minmax}(t_1, \dots, t_n))$ si $s \in S_1$ (où $(s, t_i) \in A$)
 - $\text{minmax}(s) = \min(\text{minmax}(t_1, \dots, t_n))$ si $s \in S_2$ (où $(s, t_i) \in A$)

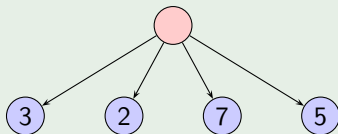
Exemple (profondeur 1)



C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 1)

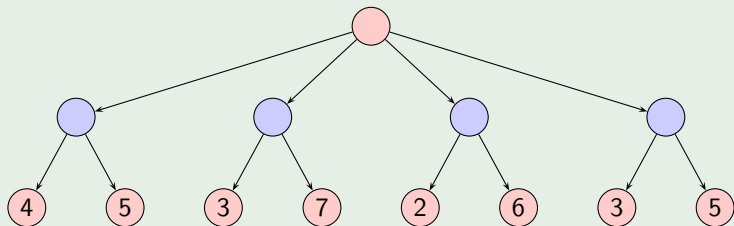


Le joueur 1 maximise l'heuristique, il va donc sélectionner le coup menant à la valeur maximale 7.

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

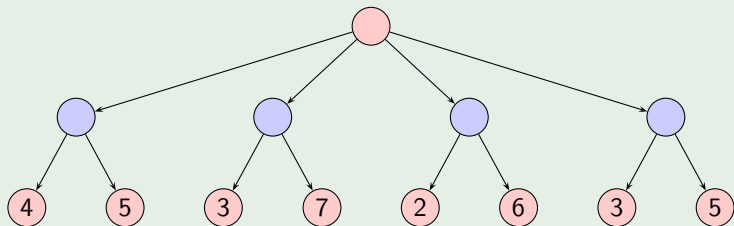
Exemple (profondeur 2)



C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 2)

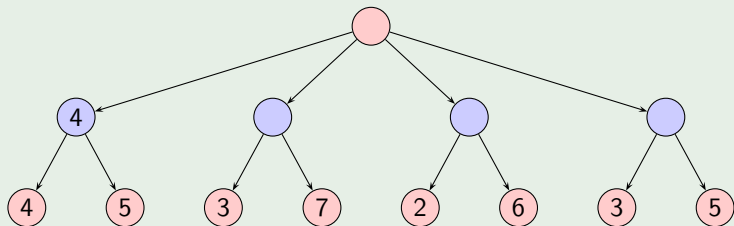


La valeur *minimale* des feuilles remonte car on suppose que le joueur 2 joue le coup qui minimise l'heuristique

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 2)

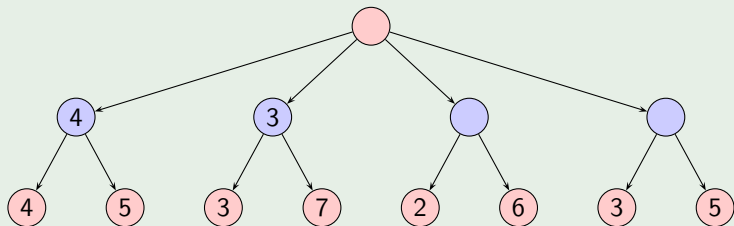


La valeur *minimale* des feuilles remonte car on suppose que le joueur 2 joue le coup qui minimise l'heuristique

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 2)

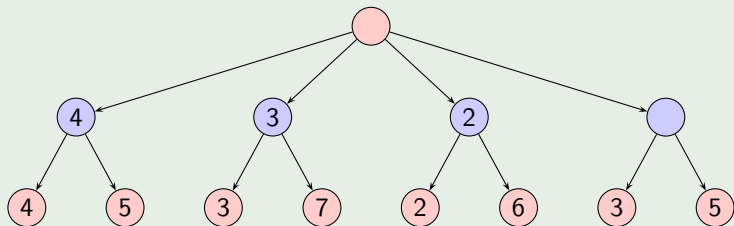


La valeur *minimale* des feuilles remonte car on suppose que le joueur 2 joue le coup qui minimise l'heuristique

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 2)

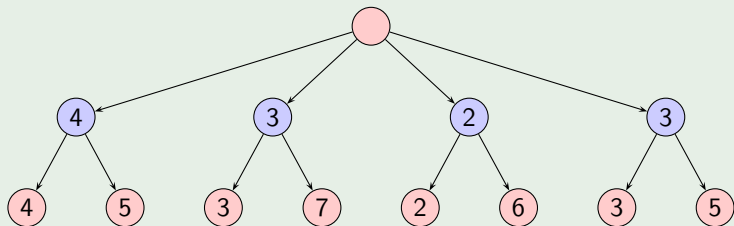


La valeur *minimale* des feuilles remonte car on suppose que le joueur 2 joue le coup qui minimise l'heuristique

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 2)

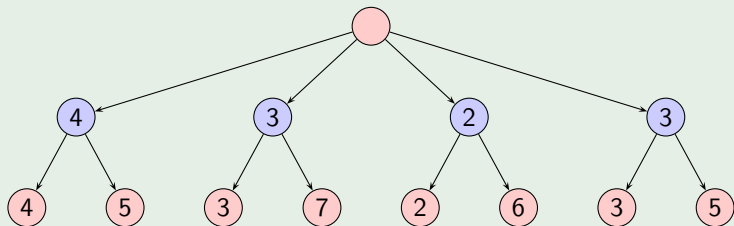


La valeur *minimale* des feuilles remonte car on suppose que le joueur 2 joue le coup qui minimise l'heuristique

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

Exemple (profondeur 2)



La valeur *minimale* des feuilles remonte car on suppose que le joueur 2 joue le coup qui minimise l'heuristique

Le coup menant à la valeur maximale est sélectionné

C4 Algorithmique pour l'IA et l'étude des jeux

3. Algorithme du min-max

```
1 def minmax(grille, joueur, profondeur, heuristique):
2     coups = coups_possibles(grille)
3     if profondeur==0 or coups == []:
4         return (heuristique(grille), None)
5     if joueur == A:
6         max_eval, best = -float('inf'), None
7         for c in coups:
8             ng = grille.copy()
9             jouer(ng, joueur, c)
10            e, z = minmax(ng, B, profondeur-1, heuristique)
11            if e >= max_eval:
12                max_eval, best = e, c
13        return max_eval, best
14    if joueur == B:
15        min_eval, best = +float('inf'), None
16        for c in coups:
17            ng = grille.copy()
18            jouer(ng, joueur, c)
19            e, z = minmax(ng, A, profondeur-1, heuristique)
20            if e <= min_eval:
21                min_eval, best = e, c
22    return min_eval, best
```