Graphes d'états

Cyril Terrioux

Laboratoire des Sciences de l'Information et des Systèmes

LSIS - UMR CNRS 6168

Plan

1. Espaces d’états
2. Systèmes de production
3. Représentation de l’espace
4. Méthodes de résolution arborescentes

Plan

1. Espaces d’états
2. Systèmes de production
3. Représentation de l’espace
4. Méthodes de résolution arborescentes

Comment modéliser un problème ?

- analyser le problème :
  - identifier les différents objets composant le problème
  - identifier les propriétés de ces objets
  - identifier les relations entre ces objets
- représenter ces informations dans un formalisme donné

La notion d’état

Variable d’état : variable permettant de représenter tout ou partie d’un objet du problème

La notion d'état

Variable d'état : variable permettant de représenter tout ou partie d'un objet du problème

$\Rightarrow$ un problème = un ensemble de variables d'états

La notion d'état

Variable d'état : variable permettant de représenter tout ou partie d'un objet du problème

$\Rightarrow$ un problème = un ensemble de variables d'états

Etat : ensemble des valeurs prises par les variables d'états à un instant donné

La notion d'état

Variable d'état : variable permettant de représenter tout ou partie d'un objet du problème

$\Rightarrow$ un problème = un ensemble de variables d'états

Etat : ensemble des valeurs prises par les variables d'états à un instant donné

Etat initial : état de départ

Etat final : état représentant le but à atteindre

Espace d’états

Espace d’états : l’ensemble des états possibles

Etat possible : état accessible depuis l’état initial

Espace d’états

Espace d’états : l’ensemble des états possibles

Etat possible : état accessible depuis l’état initial

Problème : Déterminer si un état est accessible est parfois impossible.

Espace d'états

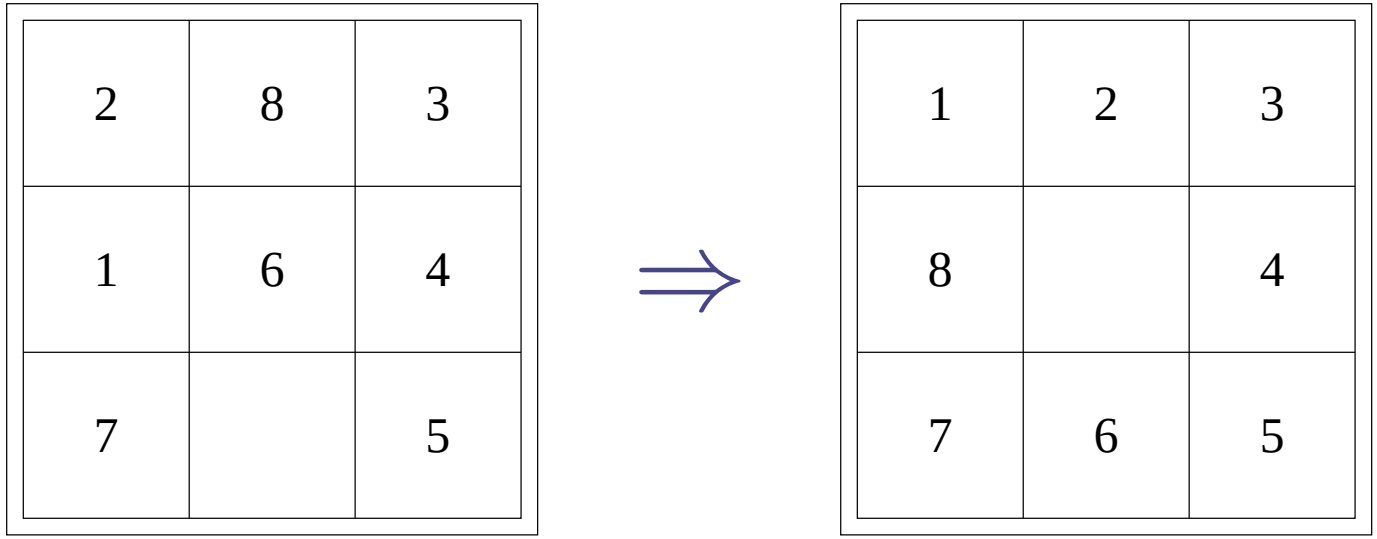
Espace d'états : l'ensemble des états possibles

Etat possible : état accessible depuis l'état initial

Problème : Déterminer si un état est accessible est parfois impossible.

$\Rightarrow$ emploi d'un sur-ensemble de l'espace d'états

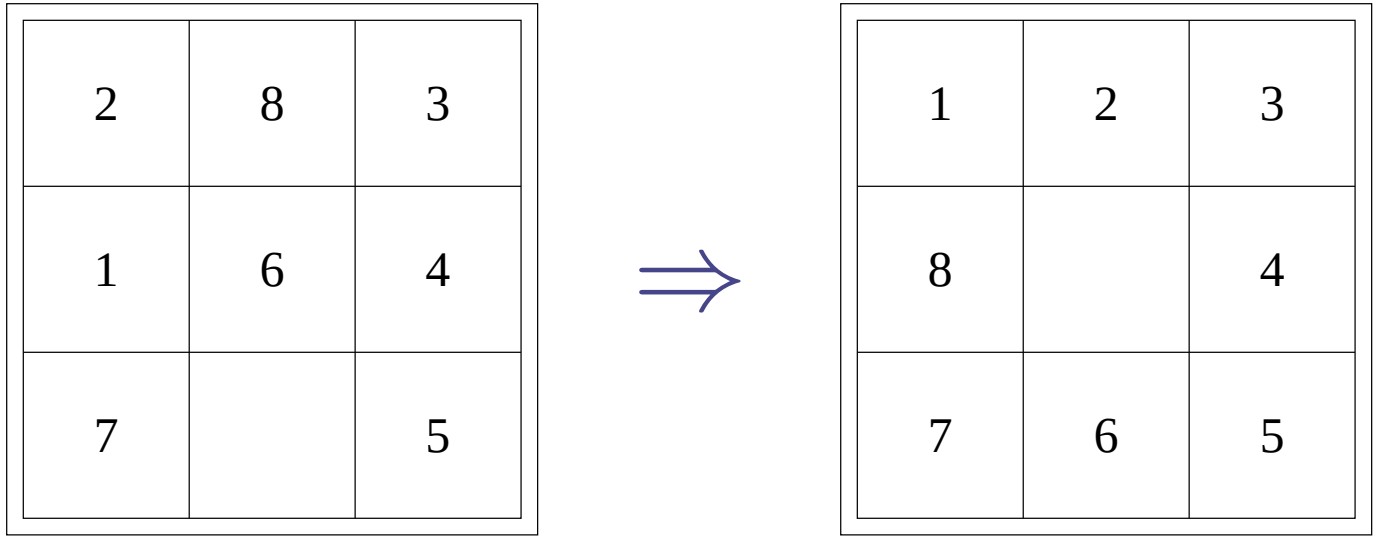
Exemple 1 : Le taquin



état initial

état final

Exemple 1 : Le taquin



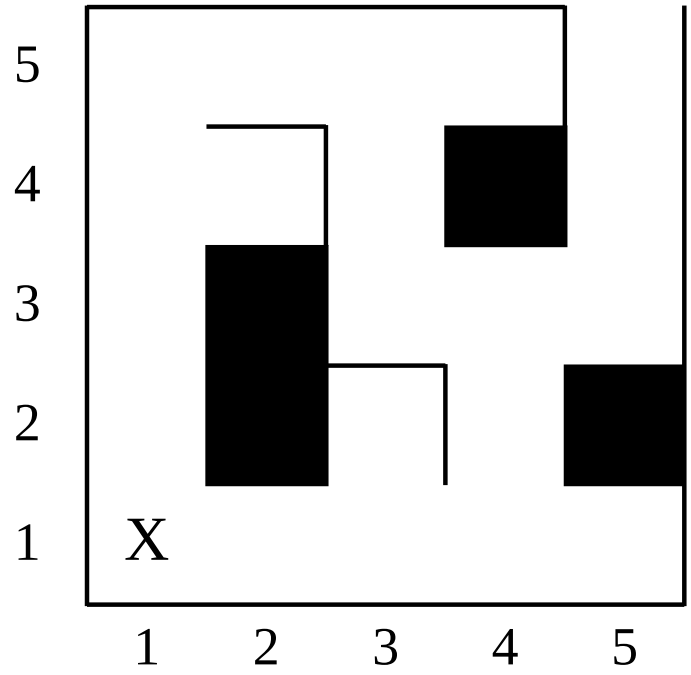
état initial

état final

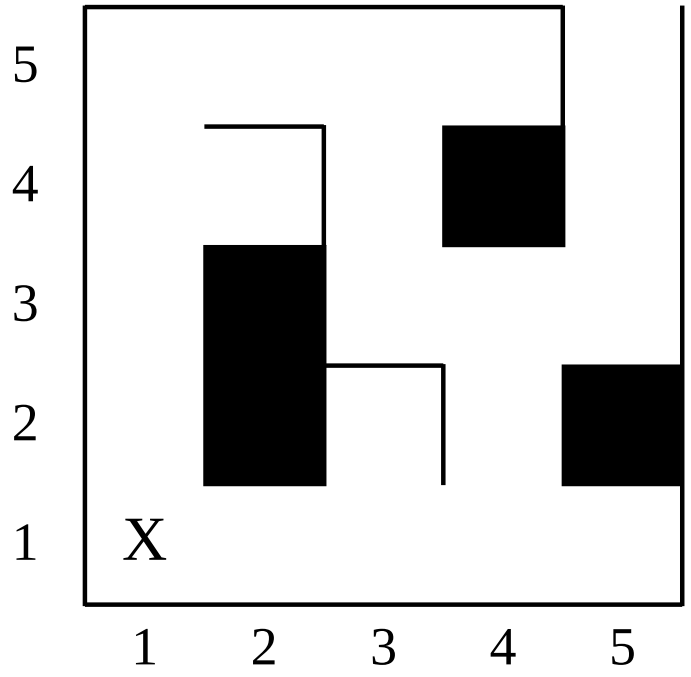
Etat : la position de chaque carré numéroté et celle de la case vide

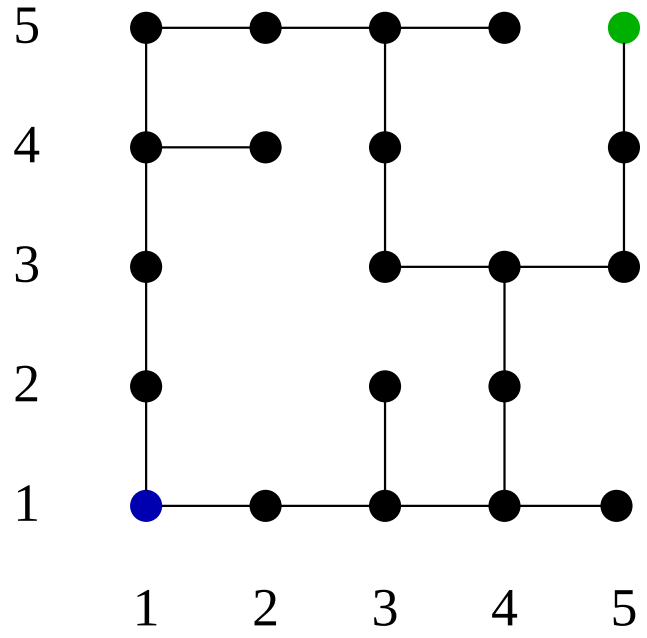
Taille de l'espace d'états = 9! = 362880

Exemple 2 : Le labyrinthe



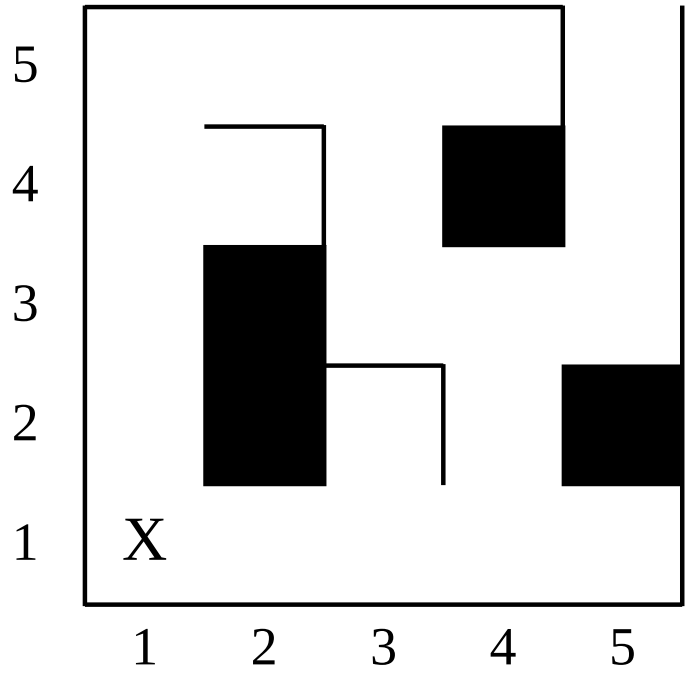
Exemple 2 : Le labyrinthe

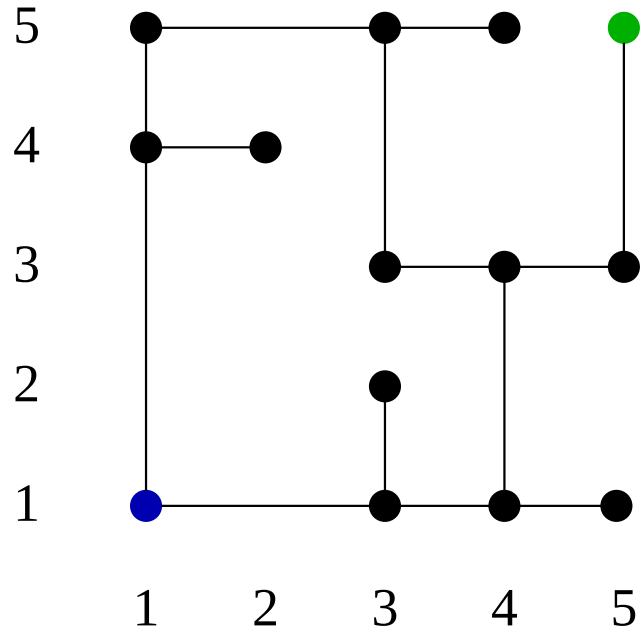




Etat : les coordonnées (x, y) de la position du robot

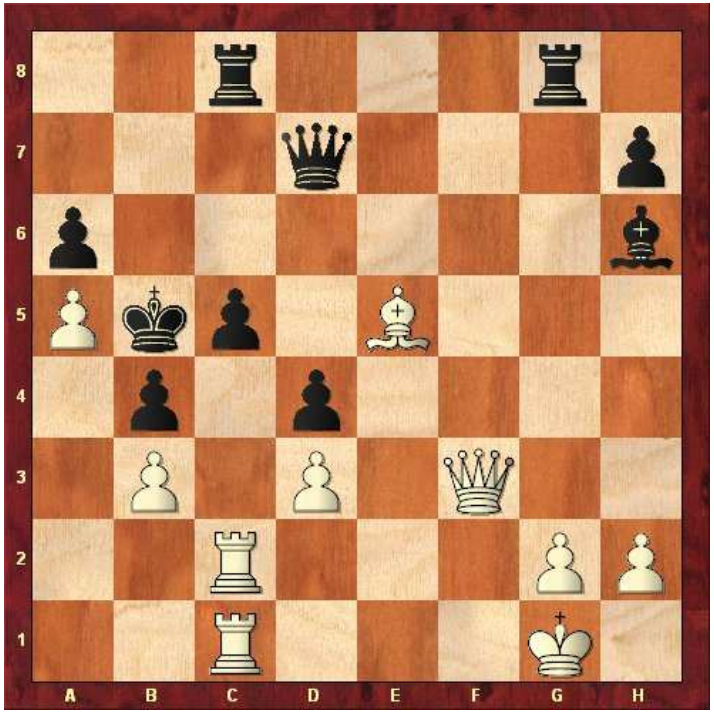
Exemple 2 : Le labyrinthe



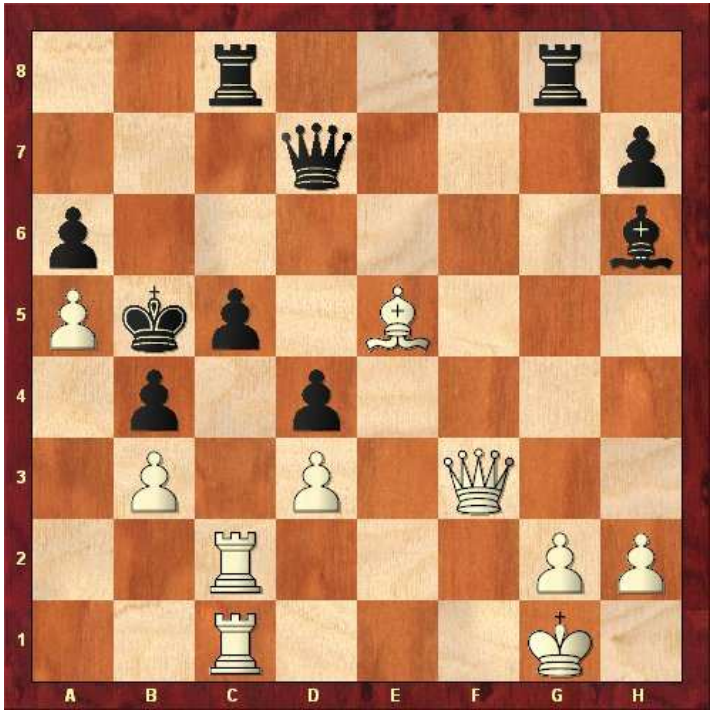


Etat : les coordonnées (x, y) de la position du robot

Exemple 3 : Le jeu d'échecs



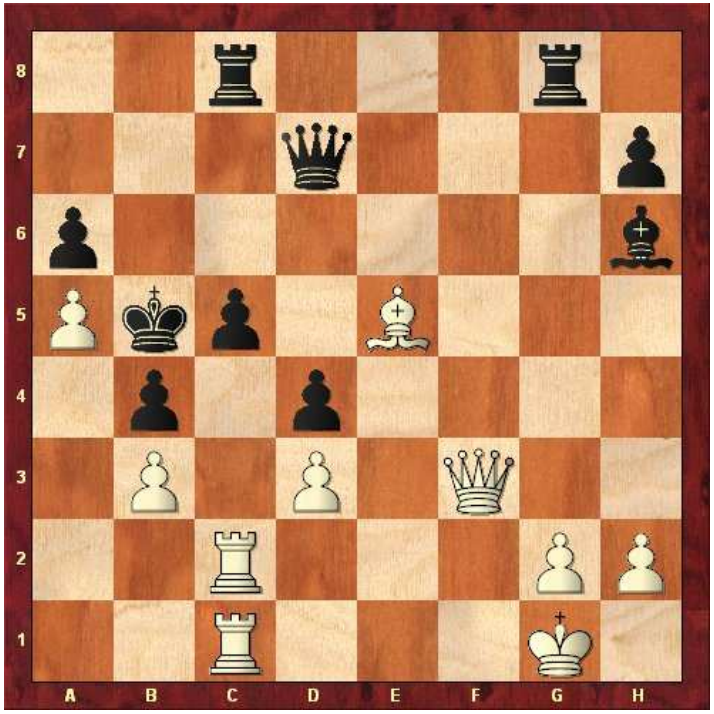
Exemple 3 : Le jeu d'échecs



Etat : le contenu de chacune des 64 cases

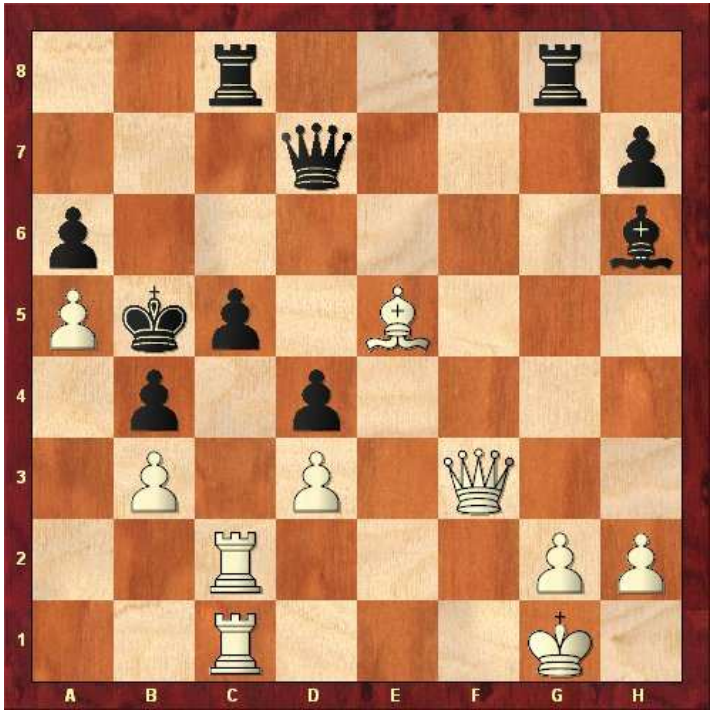
Contenu d'une case : 13 valeurs possibles

Exemple 3 : Le jeu d'échecs



Certains états sont trivialement impossibles.

Exemple 3 : Le jeu d'échecs



Certains états sont trivialement impossibles.

Mais, trouver les autres états impossibles est extrêmement difficile.

Plan

1. Espaces d’états
2. Systèmes de production
3. Représentation de l’espace
4. Méthodes de résolution arborescentes

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

Système de production : ensemble de règles permettant, à partir d'un état donné, de générer tous les états accessibles depuis cet état.

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

Système de production : ensemble de règles permettant, à partir d'un état donné, de générer tous les états accessibles depuis cet état.

Solution déductive : passer de l'état initial à un état final via une succession d'états intermédiaires produits grâce aux règles de production.

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

Système de production : ensemble de règles permettant, à partir d'un état donné, de générer tous les états accessibles depuis cet état.

Solution déductive : passer de l'état initial à un état final via une succession d'états intermédiaires produits grâce aux règles de production.

Exemple : $e_{init} \underset{r_1}{\longrightarrow} e_1$

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

Système de production : ensemble de règles permettant, à partir d'un état donné, de générer tous les états accessibles depuis cet état.

Solution déductive : passer de l'état initial à un état final via une succession d'états intermédiaires produits grâce aux règles de production.

Exemple : $e_{init} \underset{r_1}{\longrightarrow} e_1 \underset{r_2}{\longrightarrow} e_2$

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

Système de production : ensemble de règles permettant, à partir d'un état donné, de générer tous les états accessibles depuis cet état.

Solution déductive : passer de l'état initial à un état final via une succession d'états intermédiaires produits grâce aux règles de production.

Exemple : $e_{init} \underset{r_1}{\longrightarrow} e_1 \underset{r_2}{\longrightarrow} e_2 \ldots \underset{r_{i-1}}{\longrightarrow} e_{i-1}$

Raisonnement déductif ou chaînage avant

Idée : passer de l'état initial à un état final

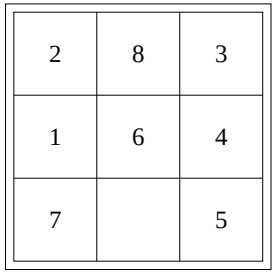
Système de production : ensemble de règles permettant, à partir d'un état donné, de générer tous les états accessibles depuis cet état.

Solution déductive : passer de l'état initial à un état final via une succession d'états intermédiaires produits grâce aux règles de production.

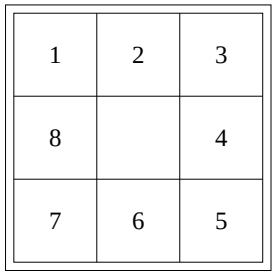
Exemple : $e_{init} \underset{r_1}{\longrightarrow} e_1 \underset{r_2}{\longrightarrow} e_2 \ldots \underset{r_{i-1}}{\longrightarrow} e_{i-1} \underset{r_i}{\longrightarrow} e_{final}$

Exemple 1 : Le taquin

Etat initial :



Etat final :



Règles de productions :

- déplacement d'un carré vers une case vide

Exemple 2 : Le labyrinthe

Etat initial : le point départ (1,1)

Etat final : la sortie (5,5)

Règles de productions :

- déplacement vers la gauche (s'il n'y a pas de mur)
- déplacement vers la droite (s'il n'y a pas de mur)
- déplacement vers le haut (s'il n'y a pas de mur)
- déplacement vers le bas (s'il n'y a pas de mur)

Exemple 3 : Le jeu d’échecs

Etat initial : une position légale

Etats finaux : état "mat", état "match nul", état "pat"

Règles de productions :

- les règles de déplacements de chaque pièce

Raisonnement inductif ou chaînage arrière

Idée : passer d'un état final à l'état initial

Raisonnement inductif ou chaînage arrière

Idée : passer d'un état final à l'état initial

Solution inductive : passer d'un état final à l'état initial via une succession d'états intermédiaires produits grâce aux règles de production.

Raisonnement inductif ou chaînage arrière

Idée : passer d'un état final à l'état initial

Solution inductive : passer d'un état final à l'état initial via une succession d'états intermédiaires produits grâce aux règles de production.

Exemple : $e_{final} \underset{r_1}{\longrightarrow} e_1 \underset{r_2}{\longrightarrow} e_2 \ldots \underset{r_{i-1}}{\longrightarrow} e_{i-1} \underset{r_i}{\longrightarrow} e_{init}$

Exemples

Règles de productions :

- Taquin : mêmes règles

Exemples

Règles de productions :

- Taquin : mêmes règles
- Labyrinthe : mêmes règles

Exemples

Règles de productions :

- Taquin : mêmes règles
- Labyrinthe : mêmes règles
- Echecs :
  - certaines règles restent les mêmes.
  - de nouvelles règles doivent être ajoutées.

Approche intermédiaire

Idée : une partie de la recherche repose sur le chaînage avant, l'autre sur le chaînage arrière

Exemple :

1. décomposer un but en plusieurs sous-buts (chaînage arrière)
2. trouver une solution déductive pour chacun des sous-buts (chaînage avant)

Plan

1. Espaces d’états
2. Systèmes de production
3. Représentation de l’espace
4. Méthodes de résolution arborescentes

Arbres d’états

On relie chaque état généré à son prédécesseur.

Arbres d’états

On relie chaque état généré à son prédécesseur.

Avantage :

- construction rapide

Arbres d’états

On relie chaque état généré à son prédécesseur.

Avantage :

- construction rapide

Inconvénients :

- risque de duplication des états
- risque de ralentissements voire de boucles infinies

Arbres d’états

On relie chaque état généré à son prédécesseur.

Avantage :

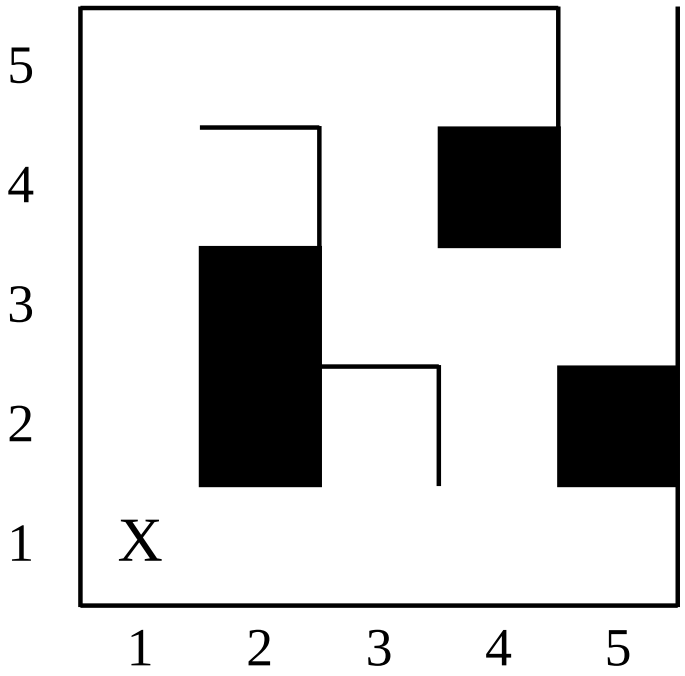
- construction rapide

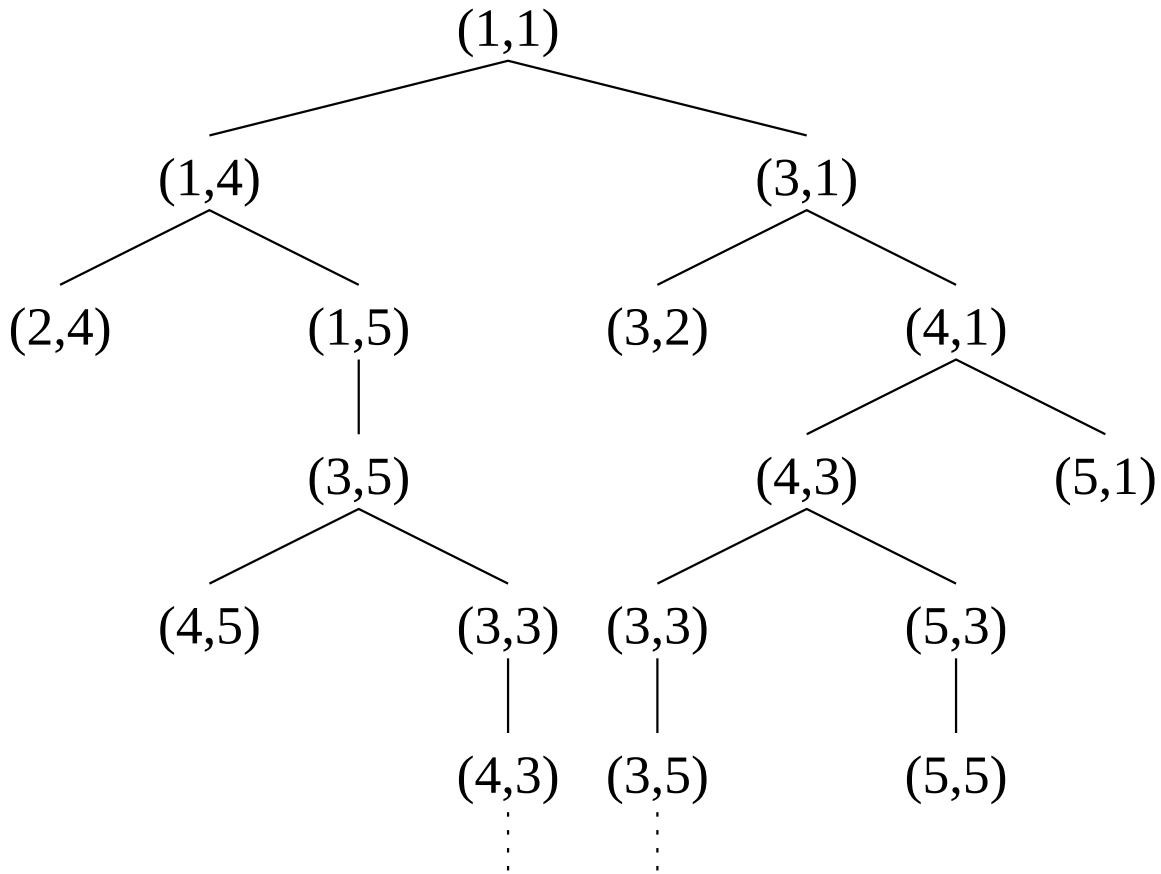
Inconvénients :

- risque de duplication des états
- risque de ralentissements voire de boucles infinies

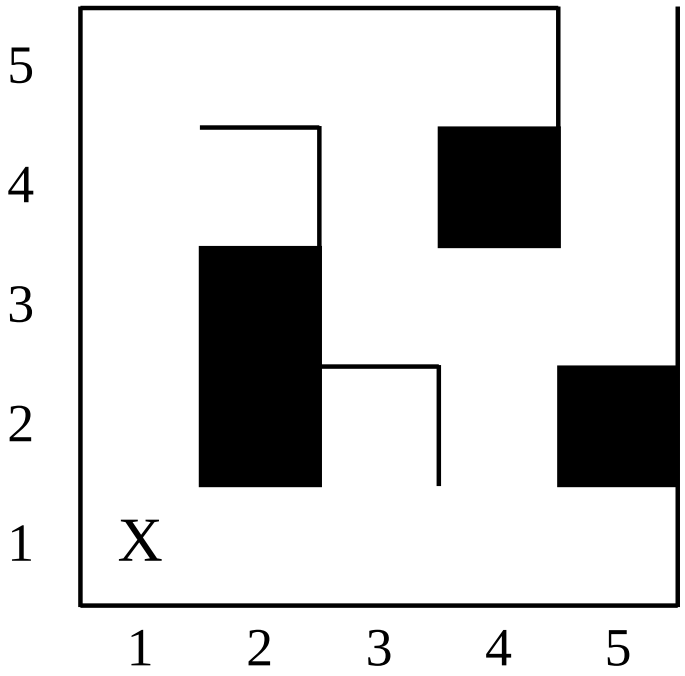
Utilisé essentiellement en théorie des jeux

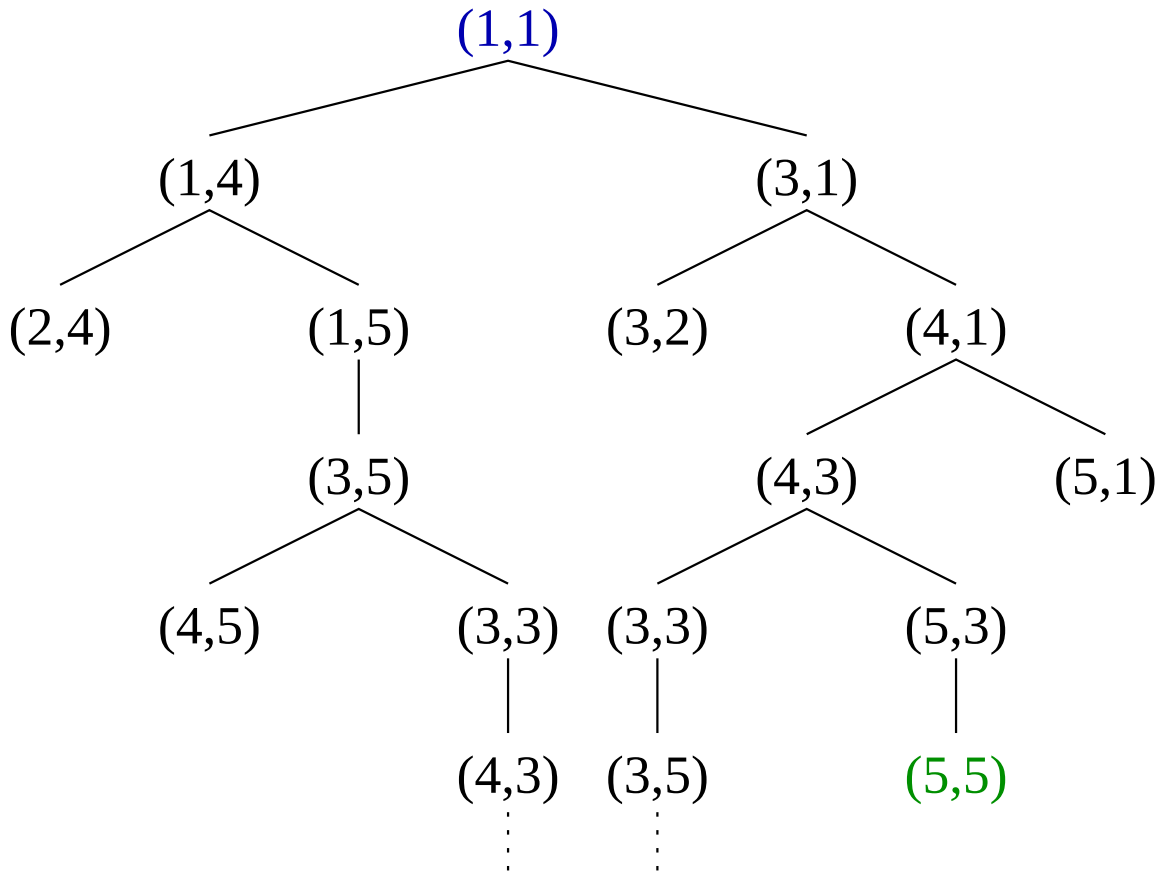
Exemple d'arbre d'états



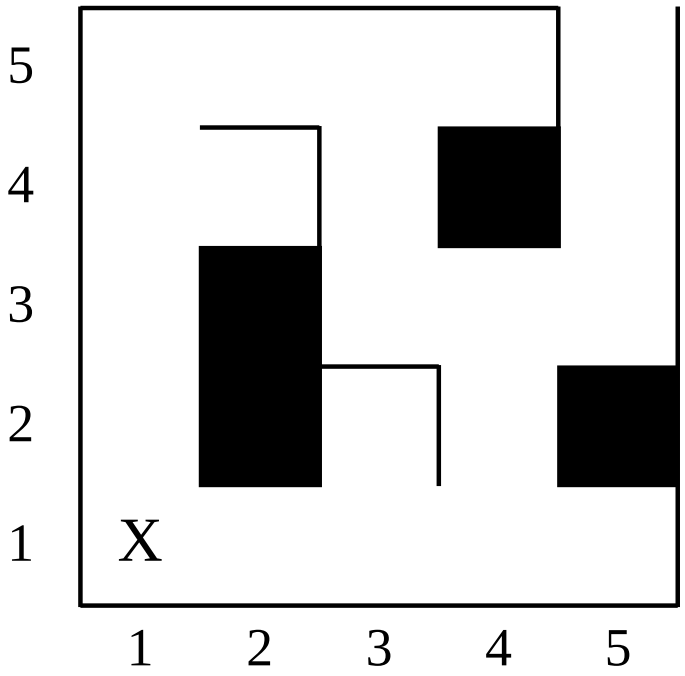


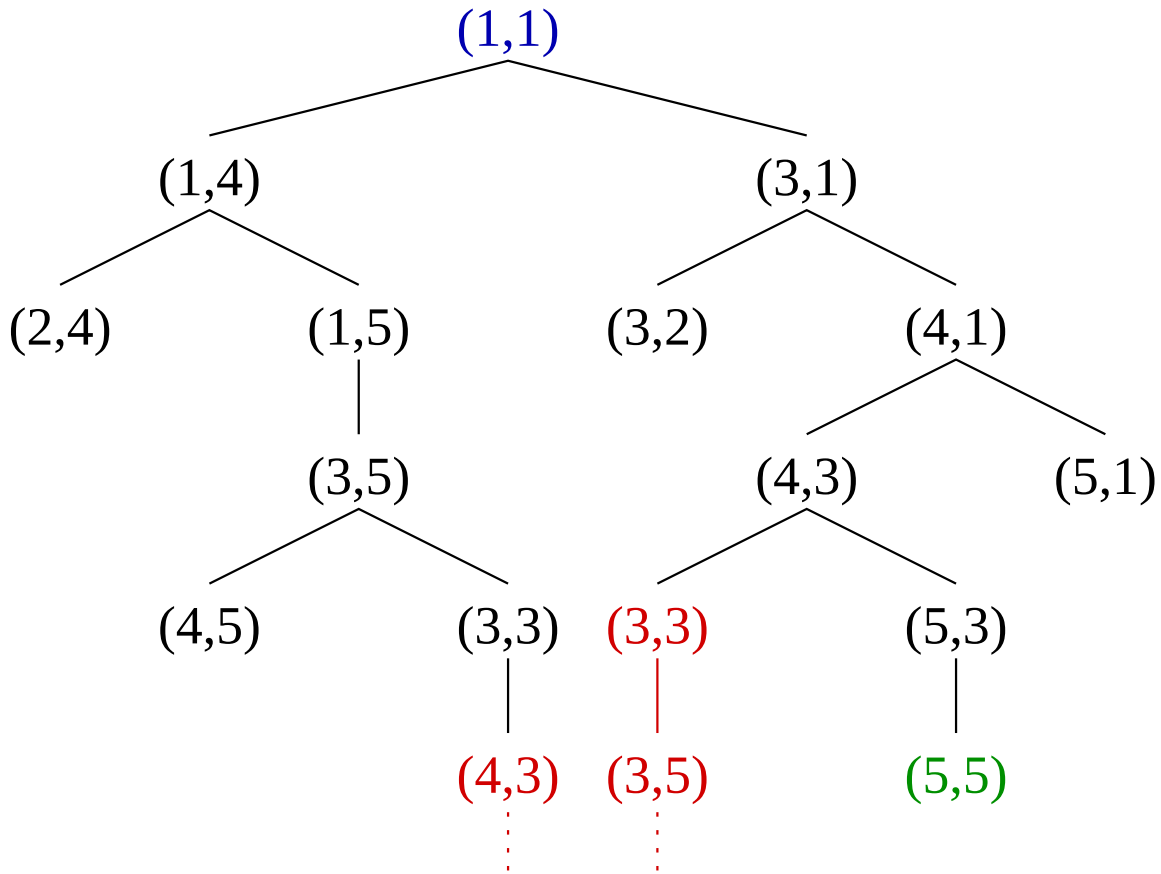
Exemple d'arbre d'états





Exemple d'arbre d'états





Graphes d’états

Lors de la génération d’un état :

1. on teste si l’état existe déjà
2. s’il n’existe pas, on le crée
3. on ajoute un arc allant du prédécesseur vers cet état.

Graphes d’états

Lors de la génération d’un état :

1. on teste si l’état existe déjà
2. s’il n’existe pas, on le crée
3. on ajoute un arc allant du prédécesseur vers cet état.

Avantage :

- unicité des états

Graphes d’états

Lors de la génération d’un état :

1. on teste si l’état existe déjà
2. s’il n’existe pas, on le crée
3. on ajoute un arc allant du prédécesseur vers cet état.

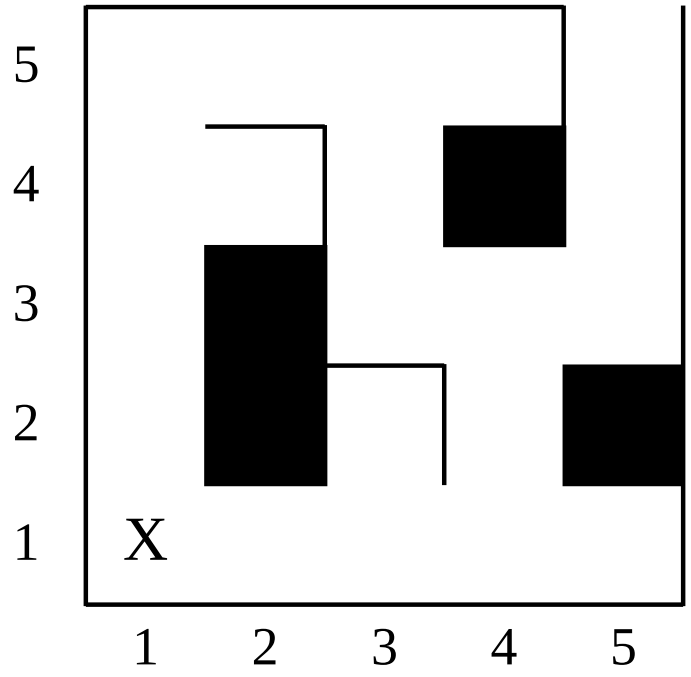
Avantage :

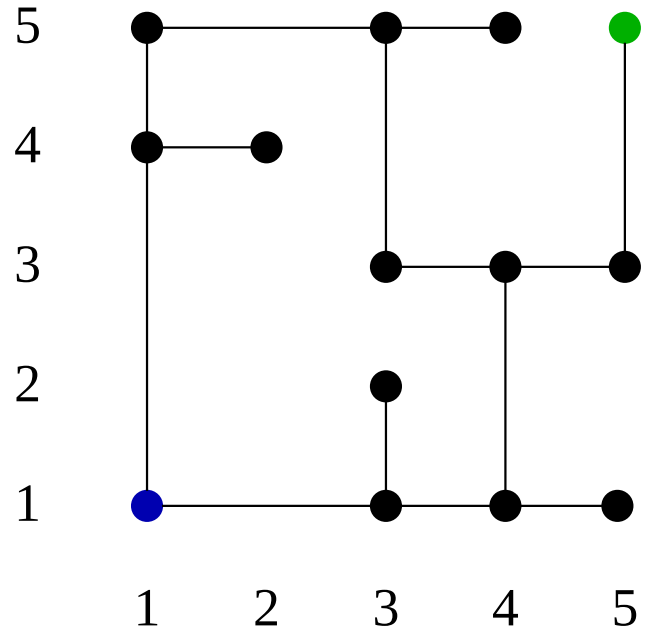
- unicité des états

Inconvénient :

- coût du test d’occurrence

Exemple de graphe d'états





Plan

1. Espaces d’états
2. Systèmes de production
3. Représentation de l’espace
4. Méthodes de résolution arborescentes

L'algorithme type de recherche arborescente

$Ouvert \leftarrow \{e_{init}\}$

$[Fermé \leftarrow \emptyset]$

TantQue $tête(Ouvert)$ n'est pas un but **Et** $Ouvert \neq \emptyset$

- $e \leftarrow tête(Ouvert)$
- $Ouvert \leftarrow Ouvert - \{e\}$
- $[Fermé \leftarrow Fermé \cup \{e\}]$
- générer tous les voisins possibles de e et les insérer dans $Ouvert$ s'ils ne sont pas déjà dans $Ouvert$ [ni dans $Fermé$]

FinTantQue

Si $Ouvert = \emptyset$ **Alors** il n'existe pas de but accessible

Sinon l'élément $tête(Ouvert)$ est un but

Recherche arborescente en profondeur

On examine les états les plus récents d'abord.

$Ouvert$ a la structure d'une pile.

Recherche arborescente en profondeur

On examine les états les plus récents d'abord.

$Ouvert$ a la structure d'une pile.

Avantages :

- éloignement rapide de l'état initial
- coût en mémoire limité

Recherche arborescente en profondeur

On examine les états les plus récents d'abord.

$Ouvert$ a la structure d'une pile.

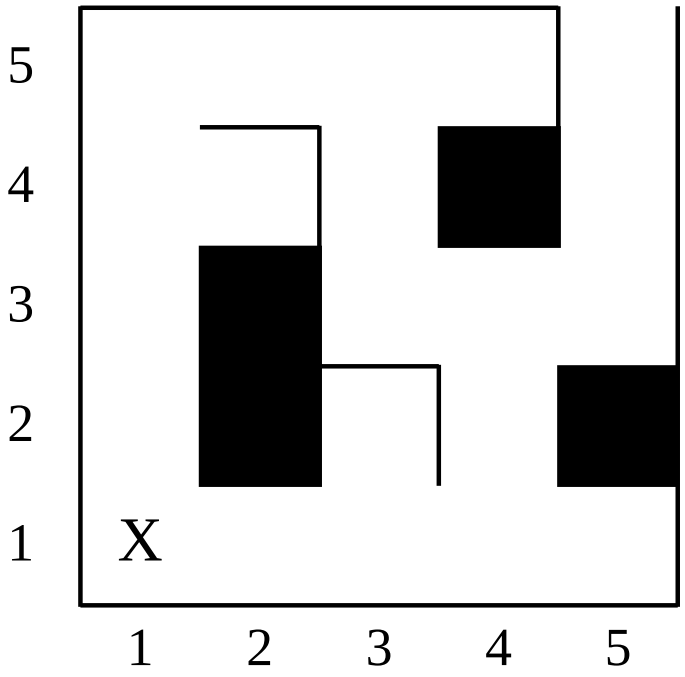
Avantages :

- éloignement rapide de l'état initial
- coût en mémoire limité

Inconvénients :

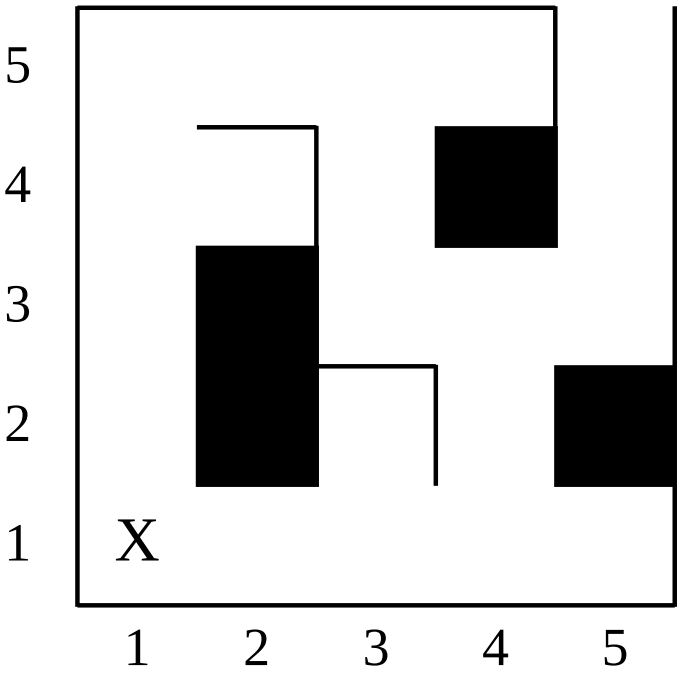
- risque de parcours intégral de l'espace de recherche
- risque de branches infinies si on n'utilise pas $Fermé$

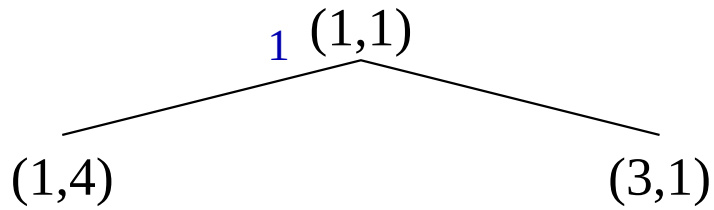
Recherche arborescente en profondeur



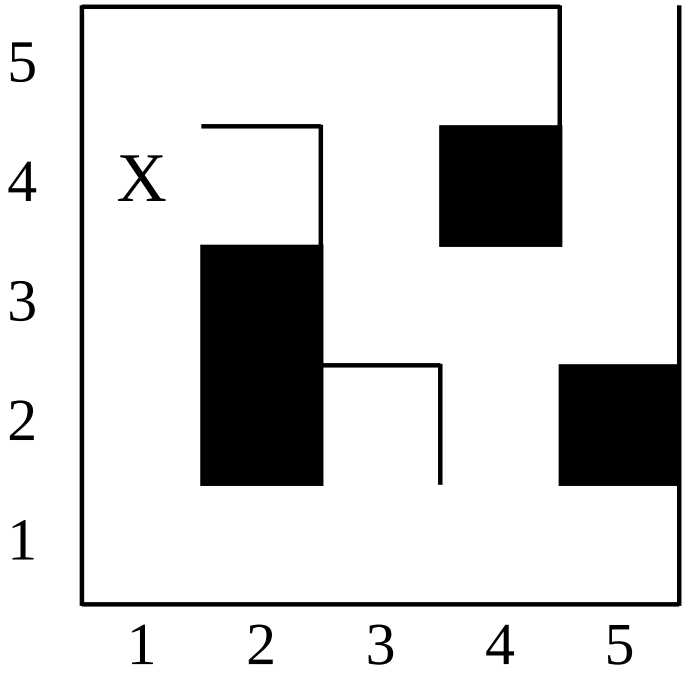
1 (1,1)

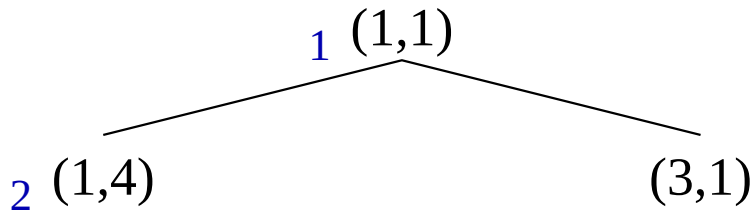
Recherche arborescente en profondeur



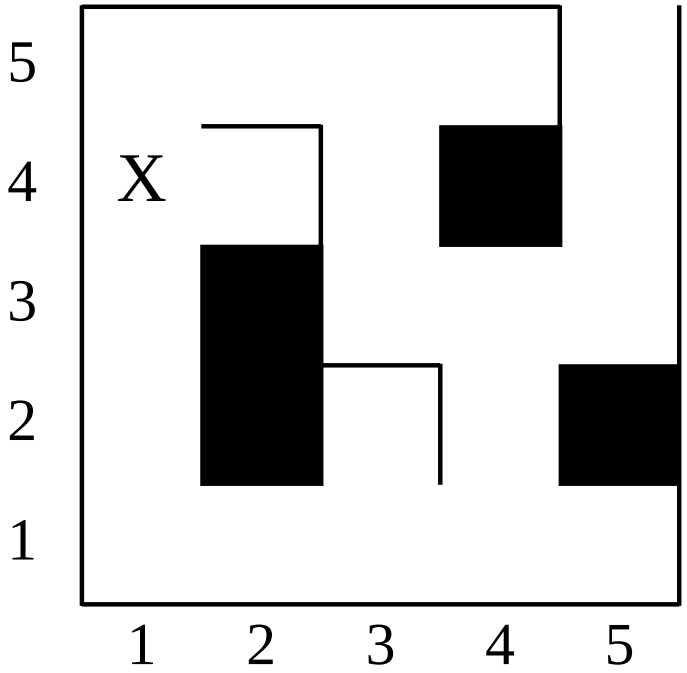


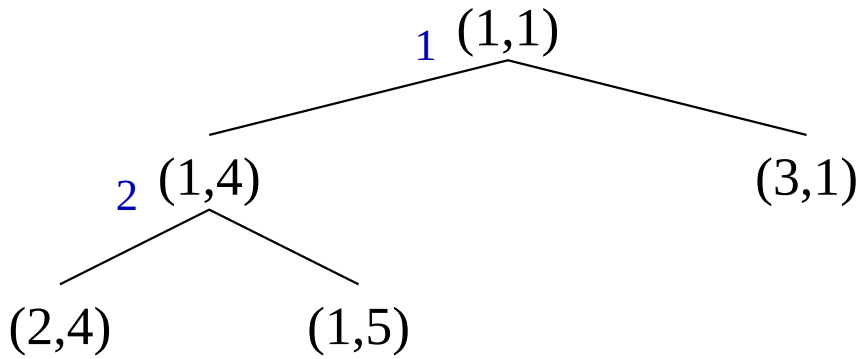
Recherche arborescente en profondeur



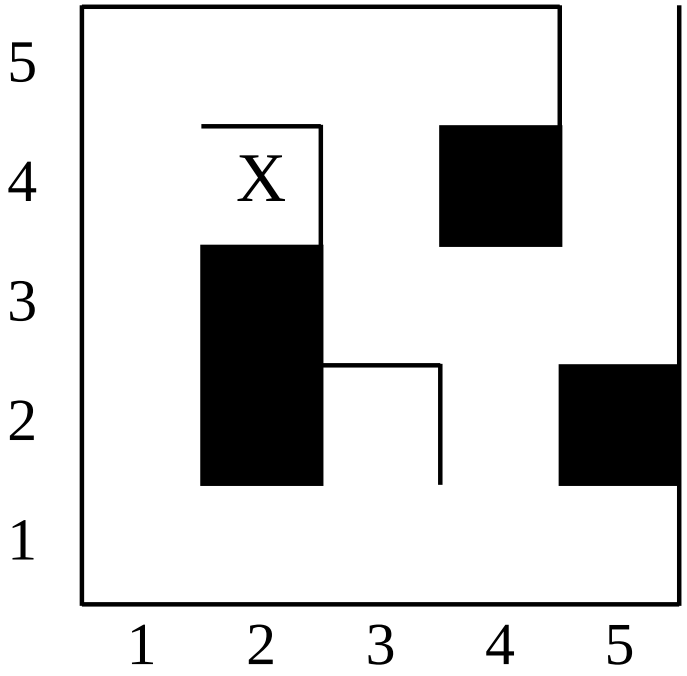


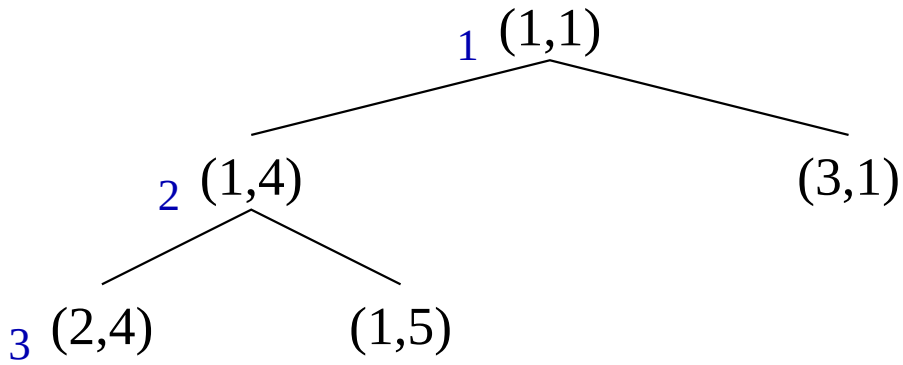
Recherche arborescente en profondeur



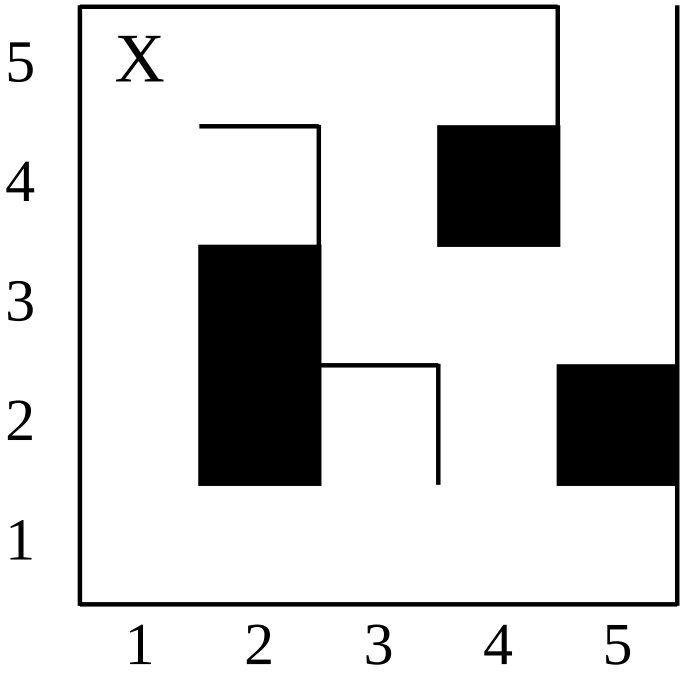


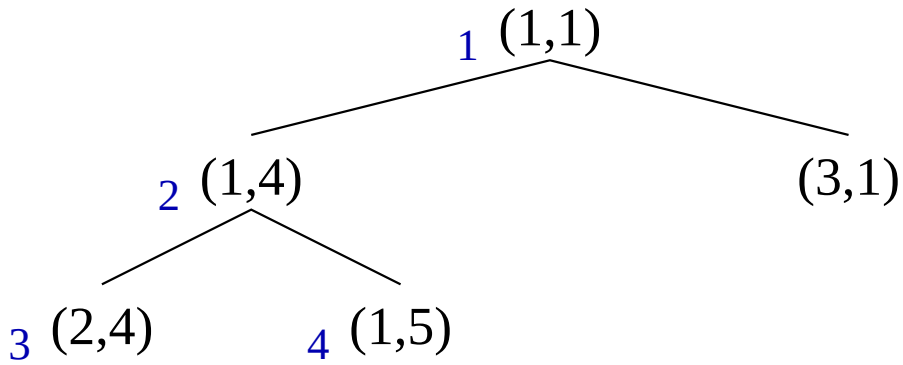
Recherche arborescente en profondeur



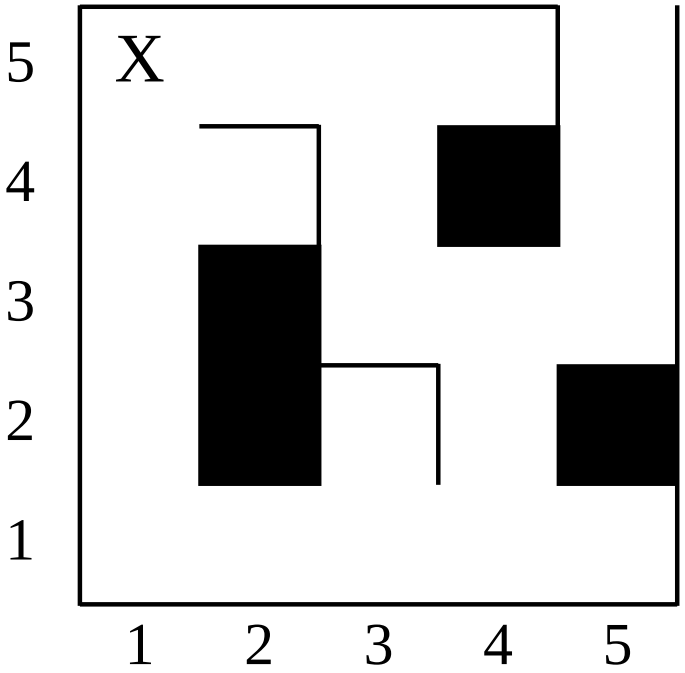


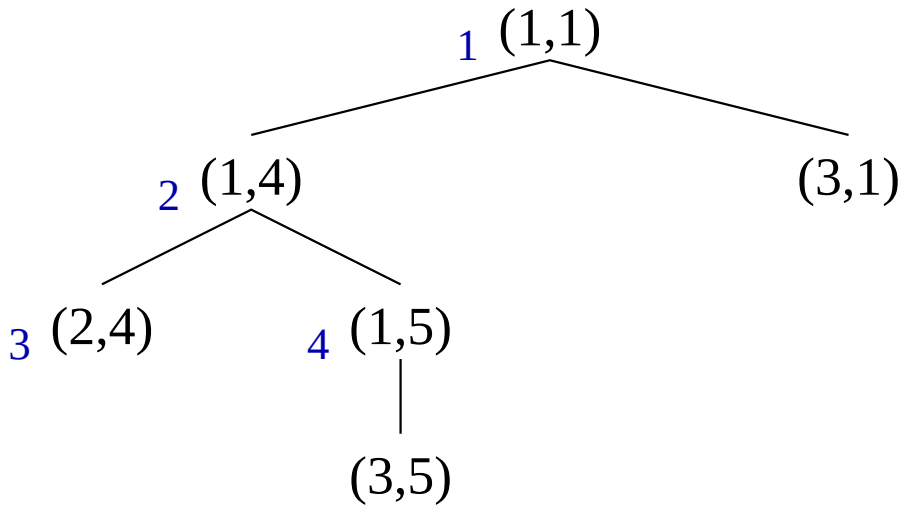
Recherche arborescente en profondeur



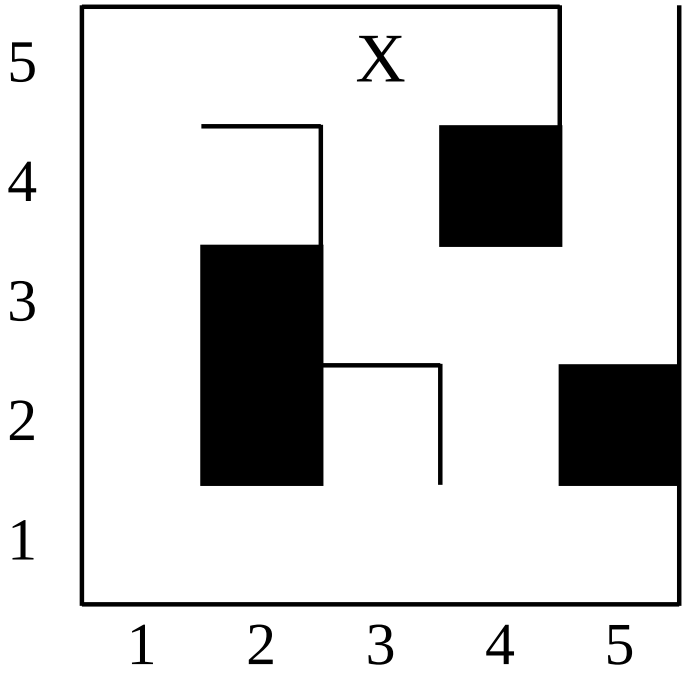


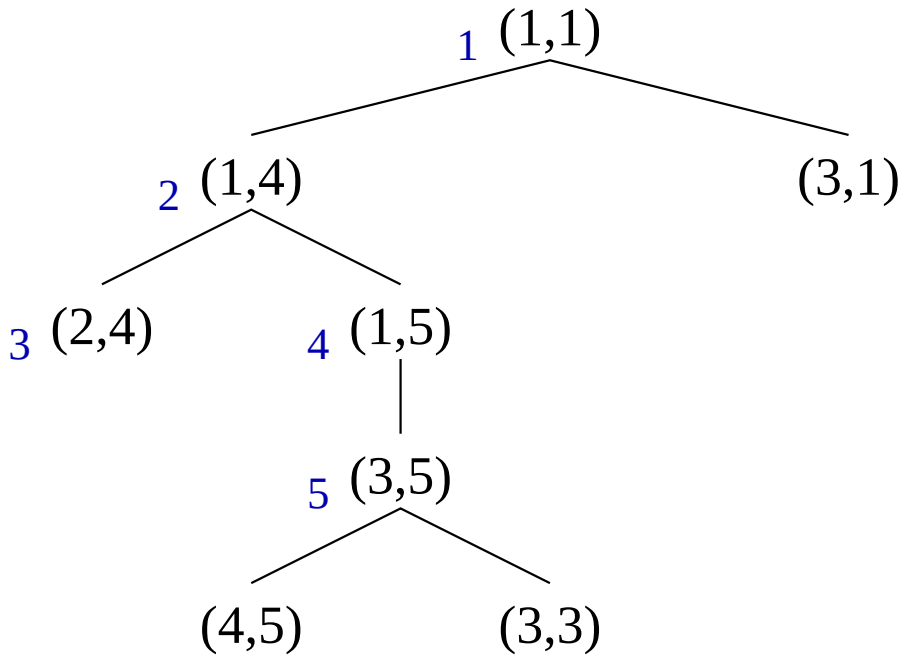
Recherche arborescente en profondeur



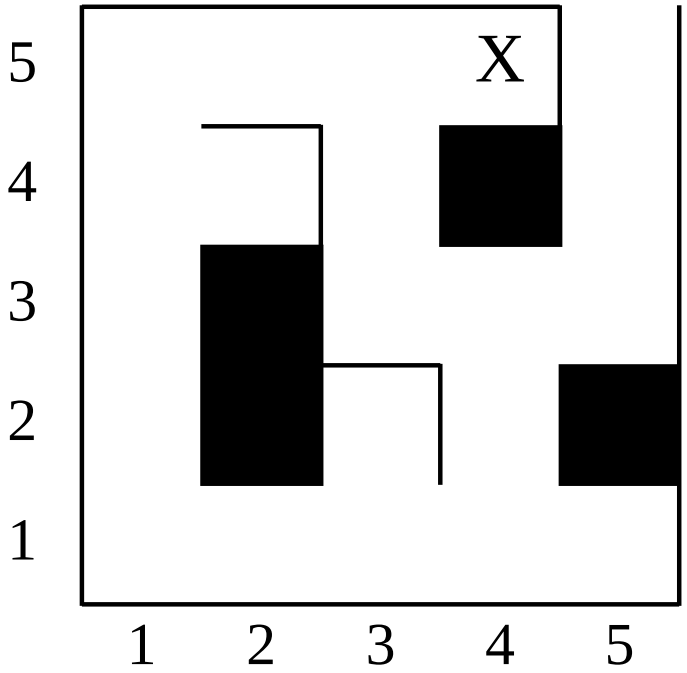


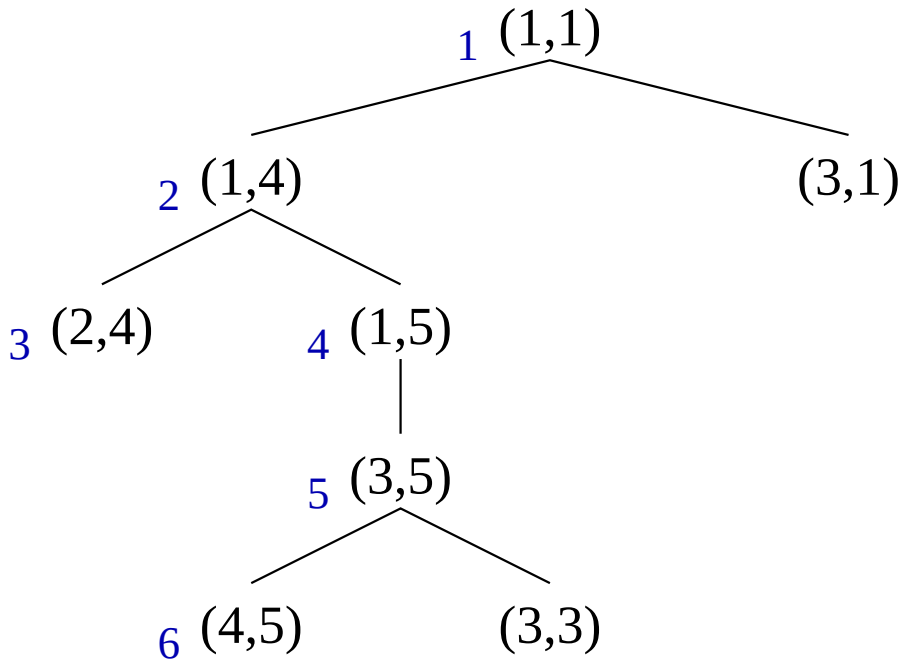
Recherche arborescente en profondeur



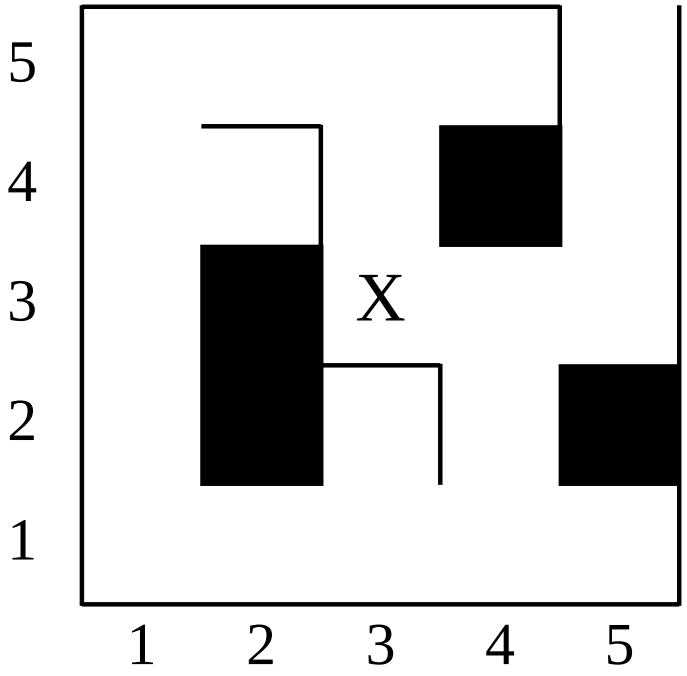


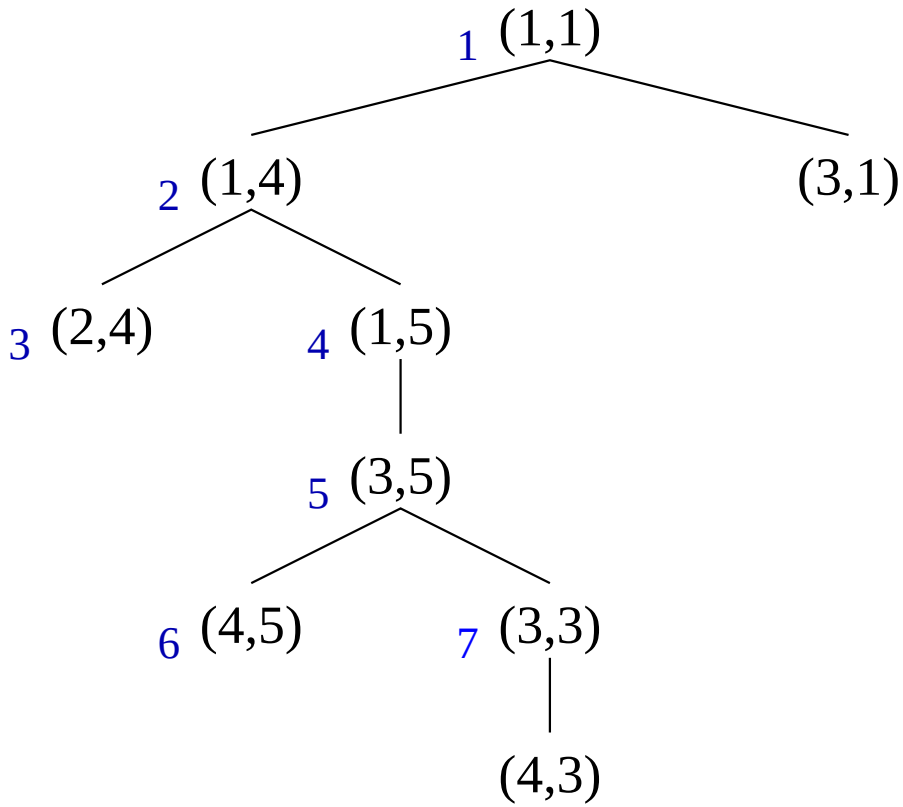
Recherche arborescente en profondeur



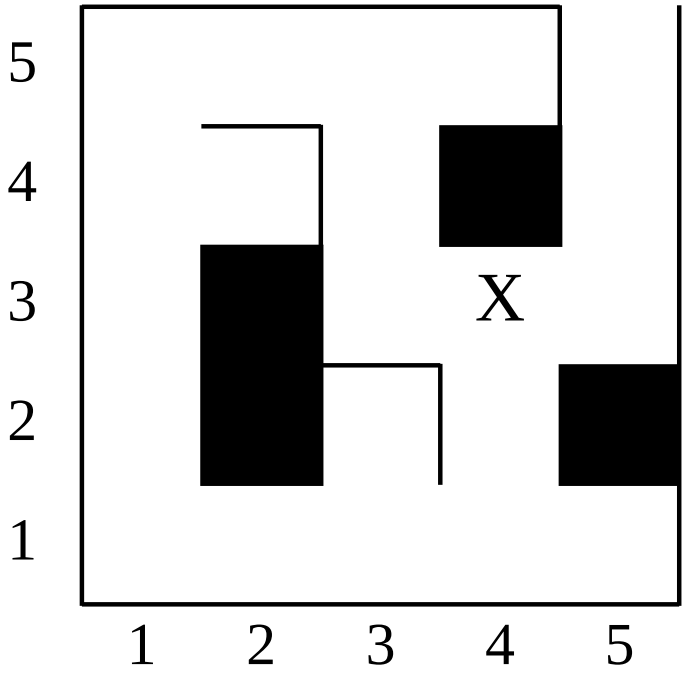


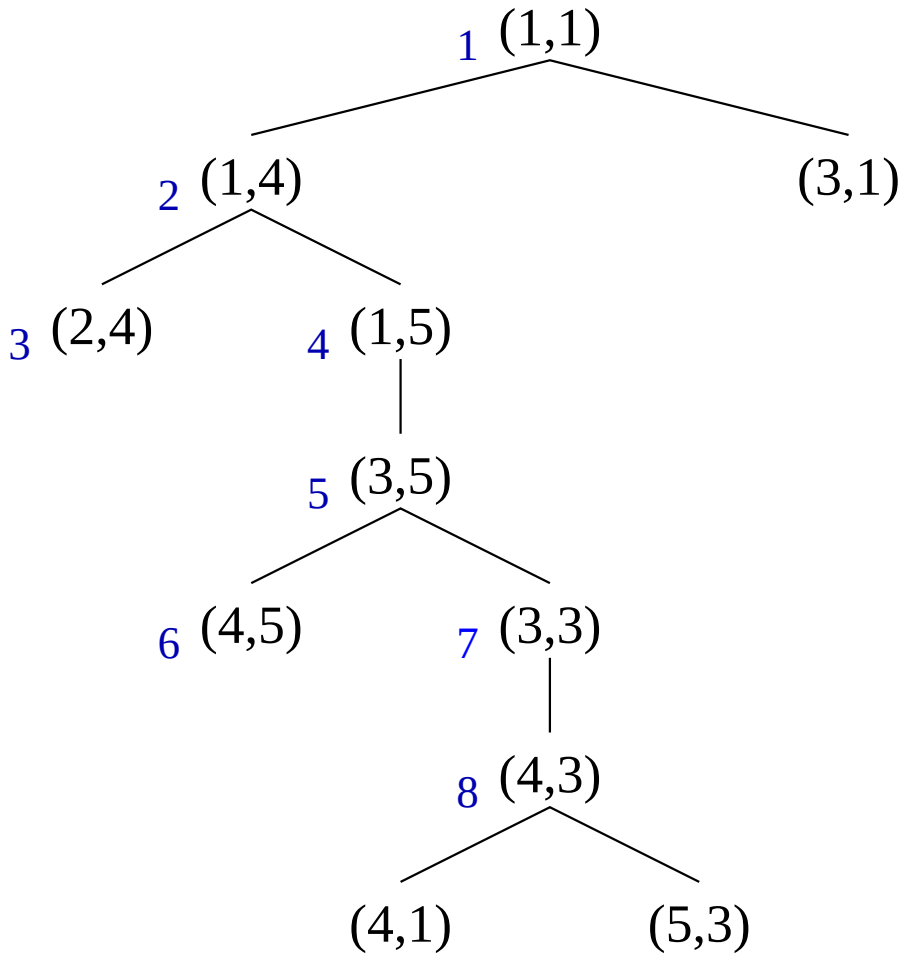
Recherche arborescente en profondeur



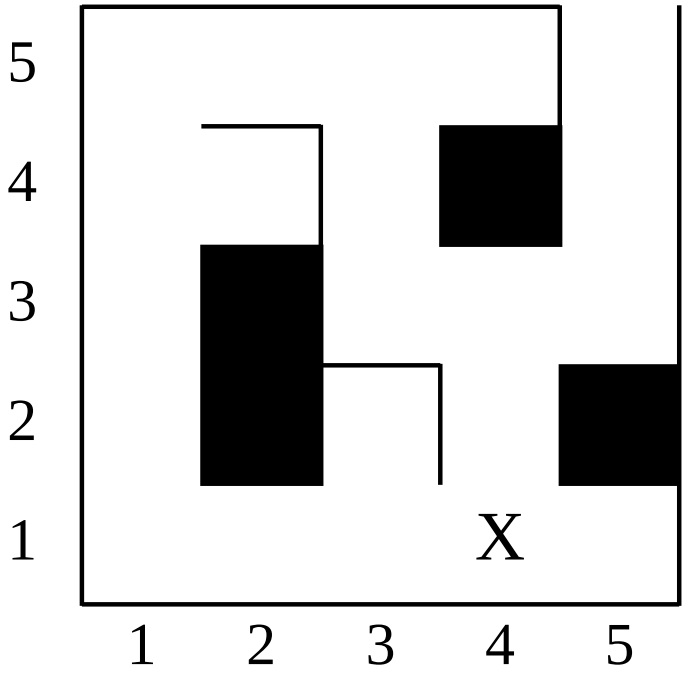


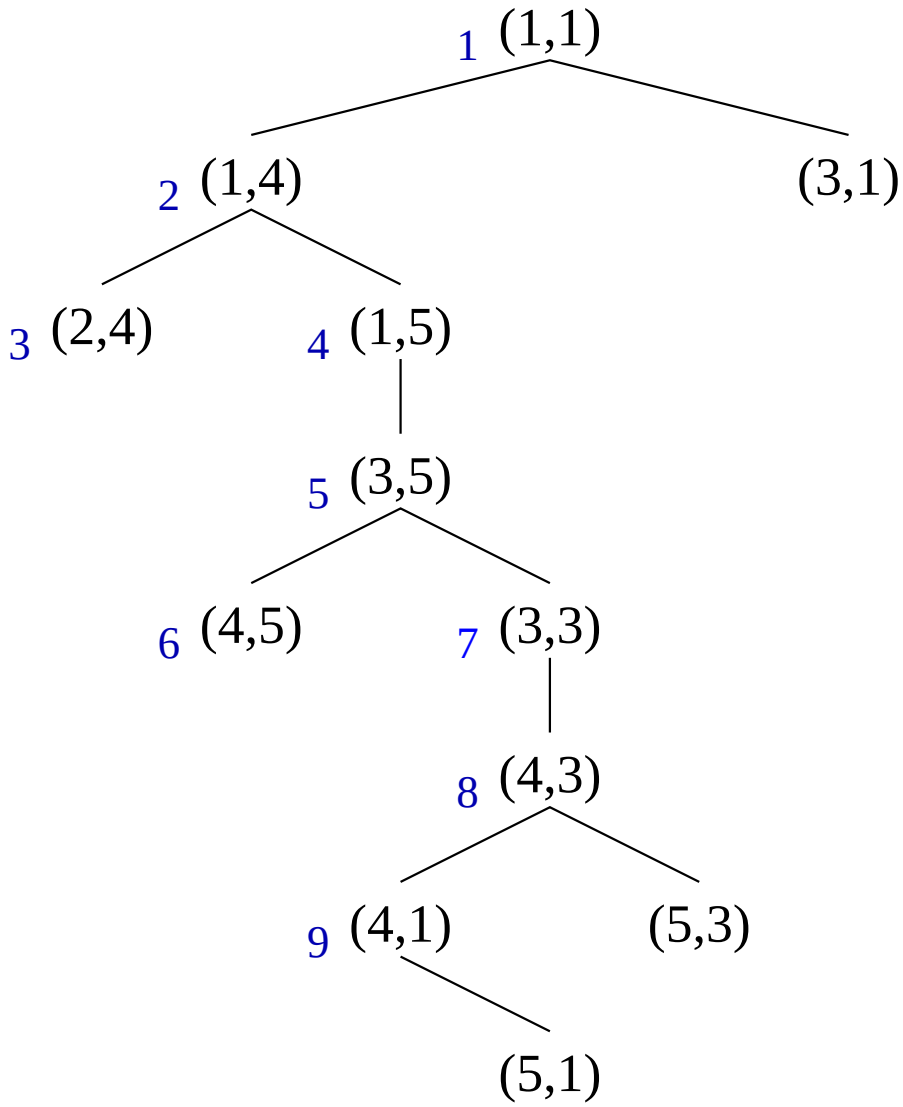
Recherche arborescente en profondeur



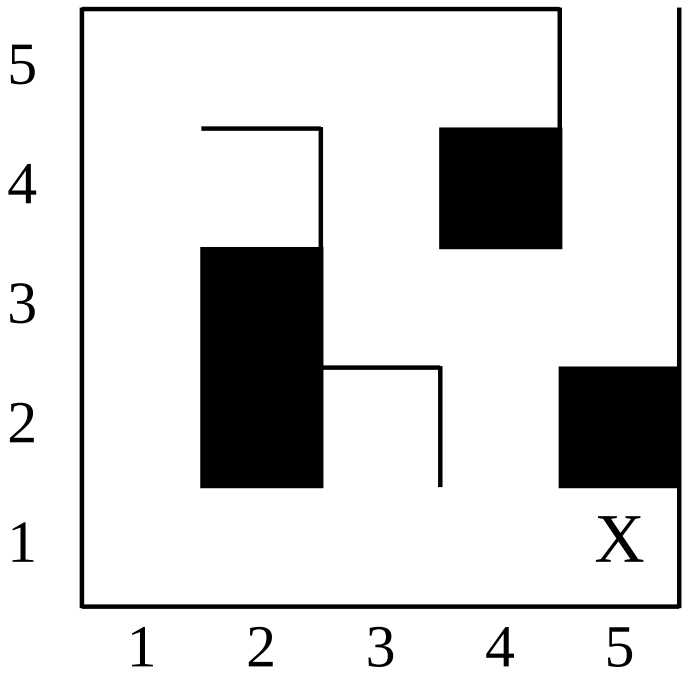


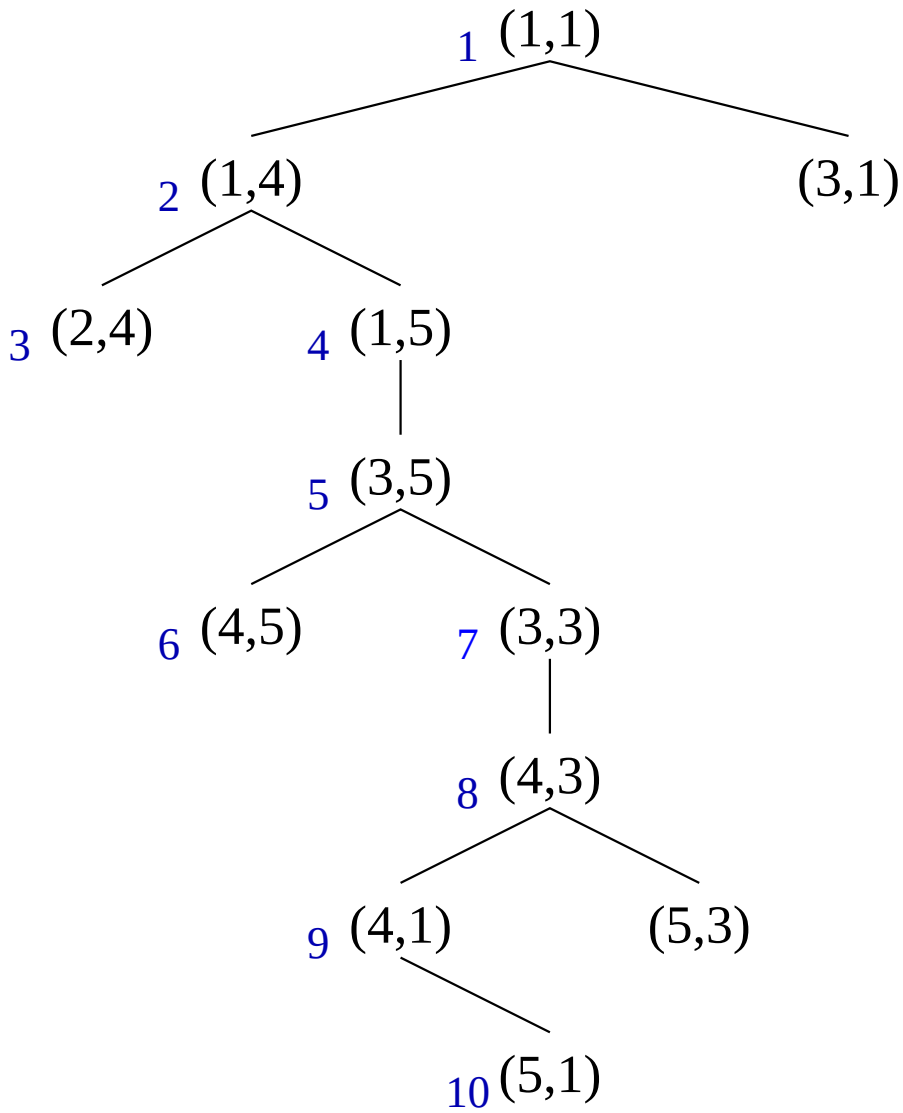
Recherche arborescente en profondeur



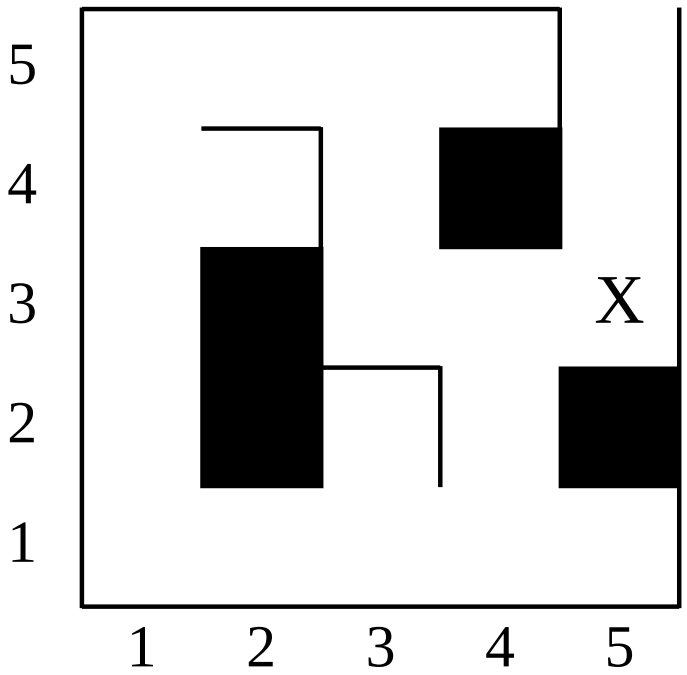


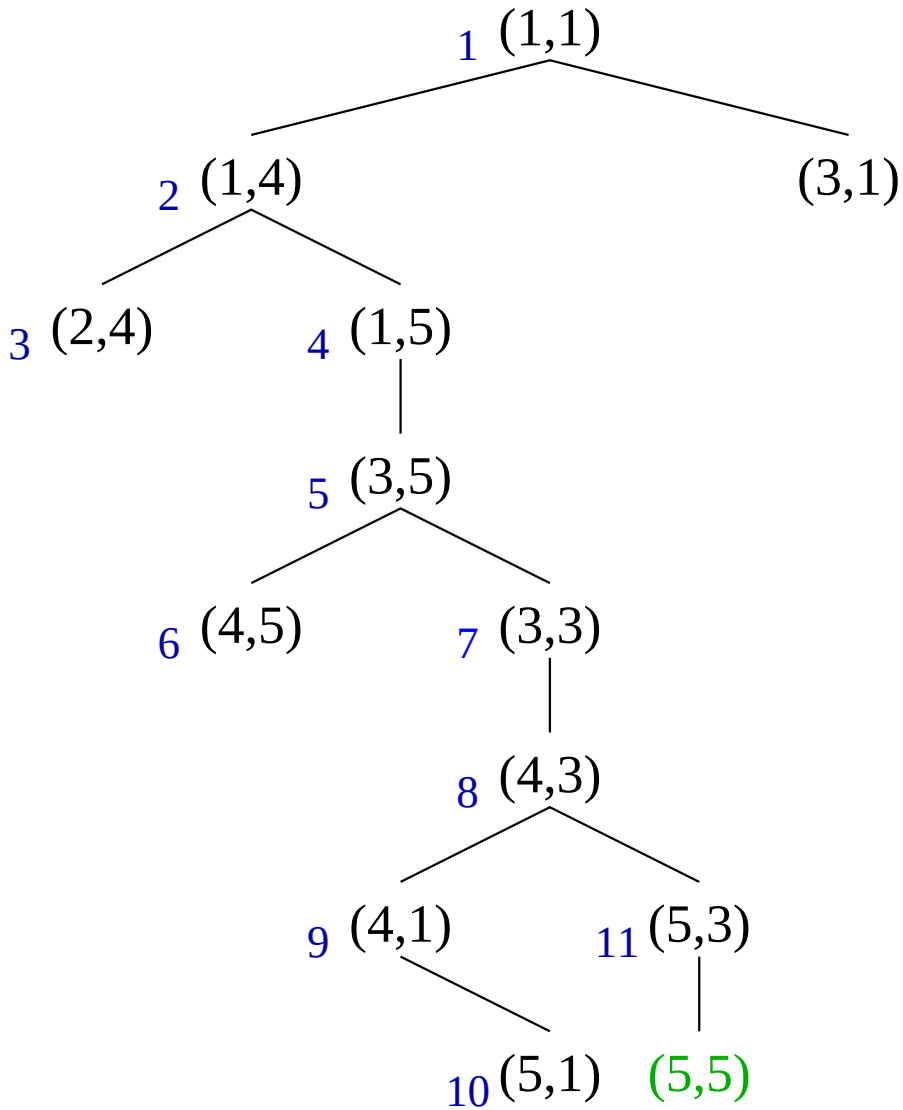
Recherche arborescente en profondeur





Recherche arborescente en profondeur





Recherche arborescente en largeur

On examine les états les moins récents d'abord.

$Ouvert$ a la structure d'une file.

Recherche arborescente en largeur

On examine les états les moins récents d'abord.

$Ouvert$ a la structure d'une file.

Avantages :

- garantie de trouver une solution s'il en existe une
- la première solution trouvée est la moins profonde

Recherche arborescente en largeur

On examine les états les moins récents d'abord.

$Ouvert$ a la structure d'une file.

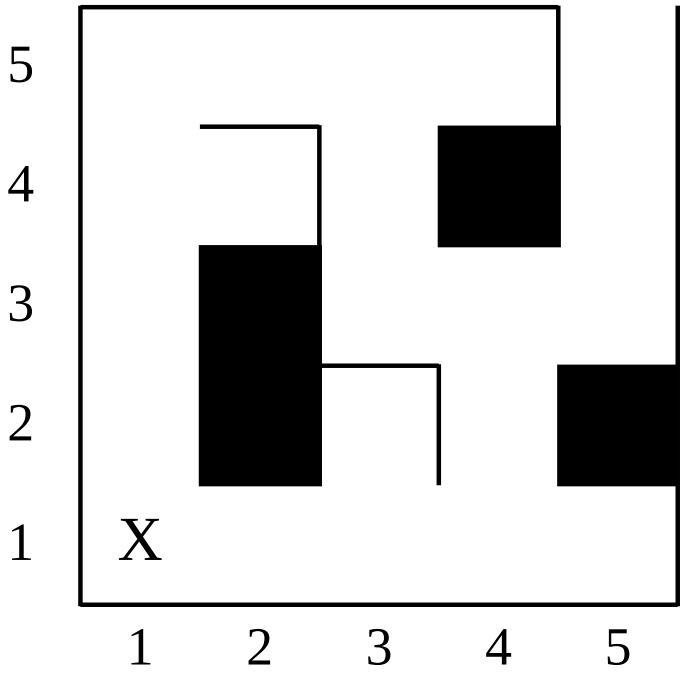
Avantages :

- garantie de trouver une solution s'il en existe une
- la première solution trouvée est la moins profonde

Inconvénients :

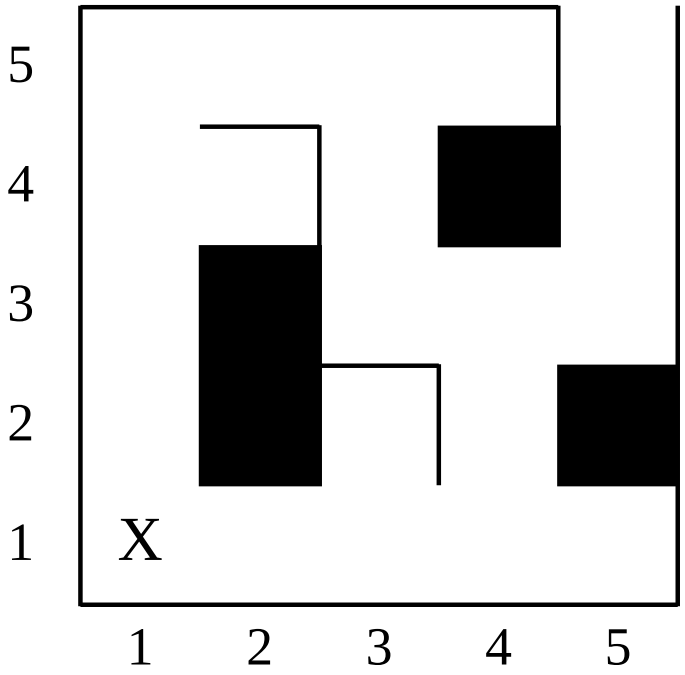
- risque de parcours intégral de l'espace de recherche
- coût en mémoire très important voire prohibitif

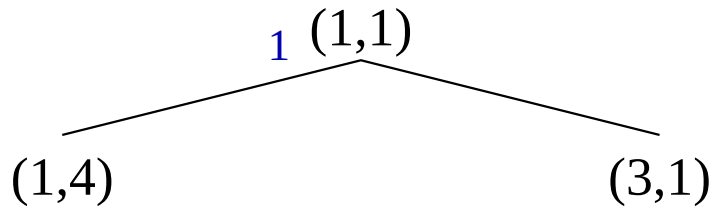
Recherche arborescente en largeur



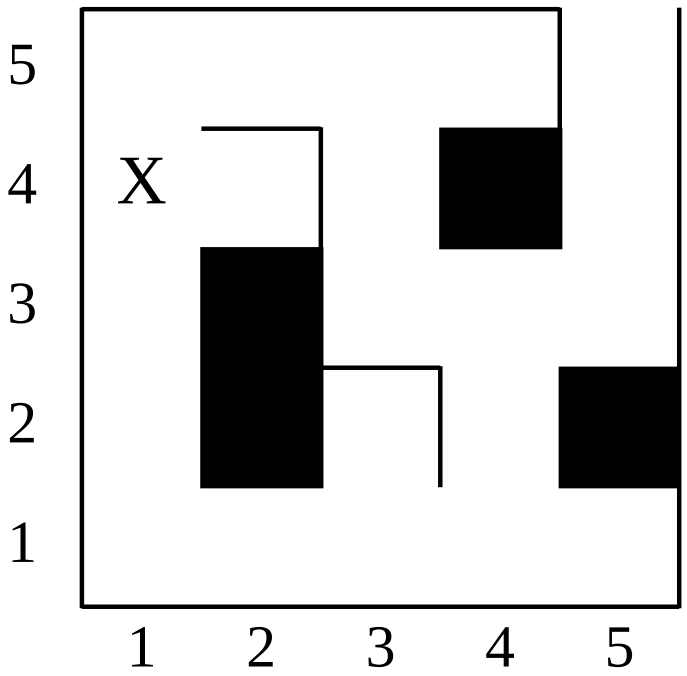
1 (1,1)

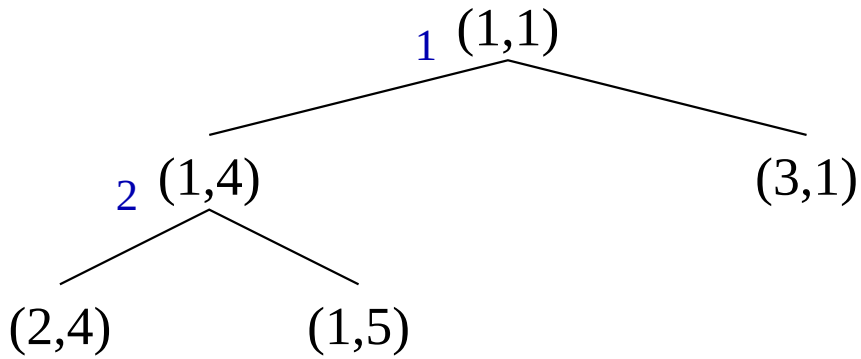
Recherche arborescente en largeur



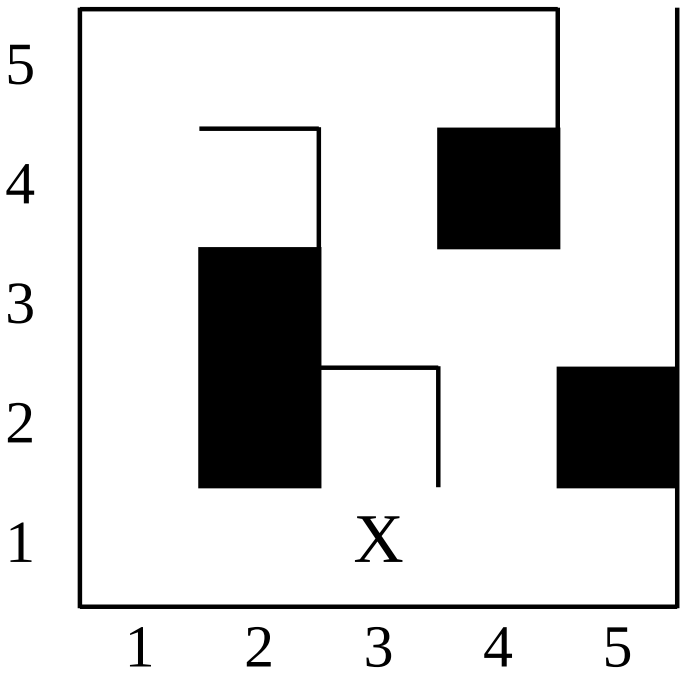


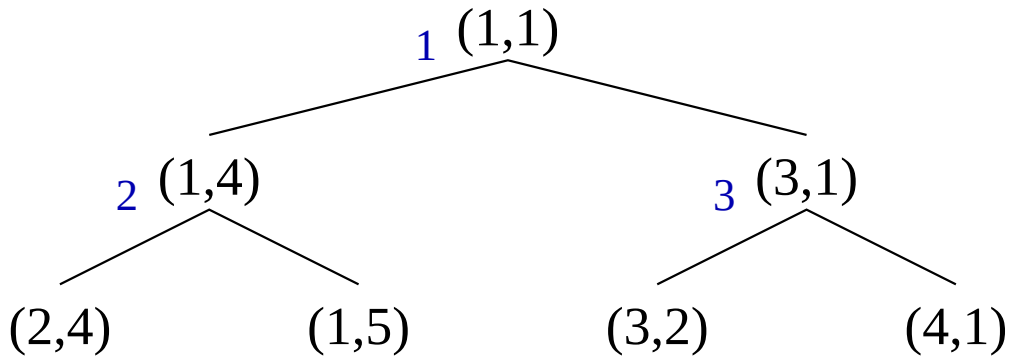
Recherche arborescente en largeur



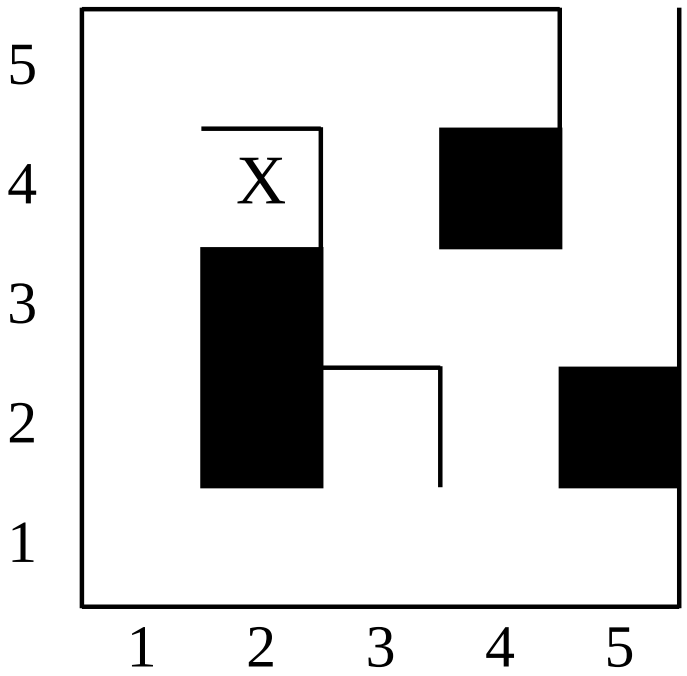


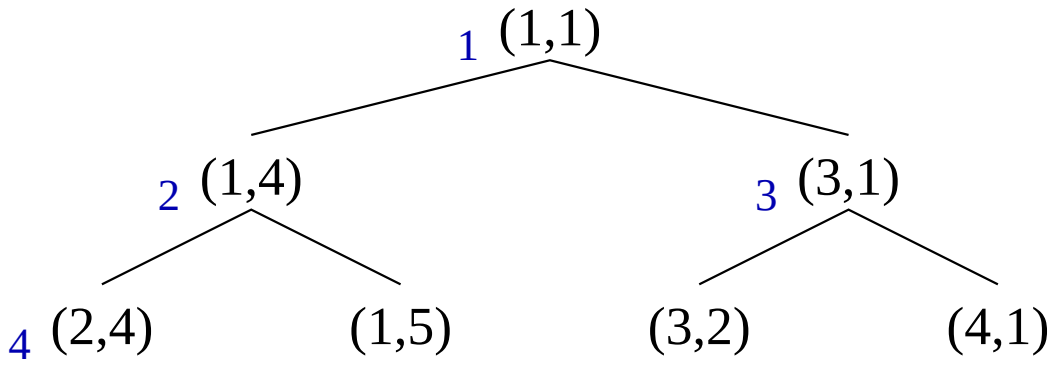
Recherche arborescente en largeur



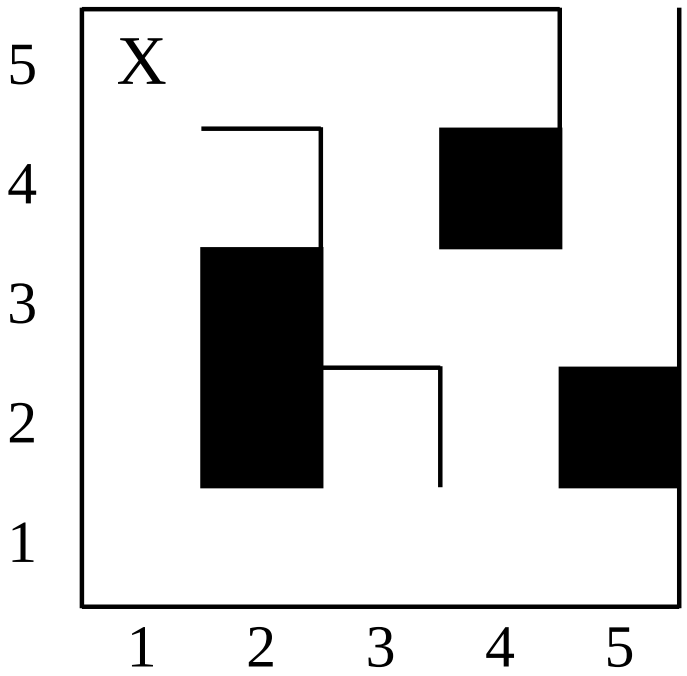


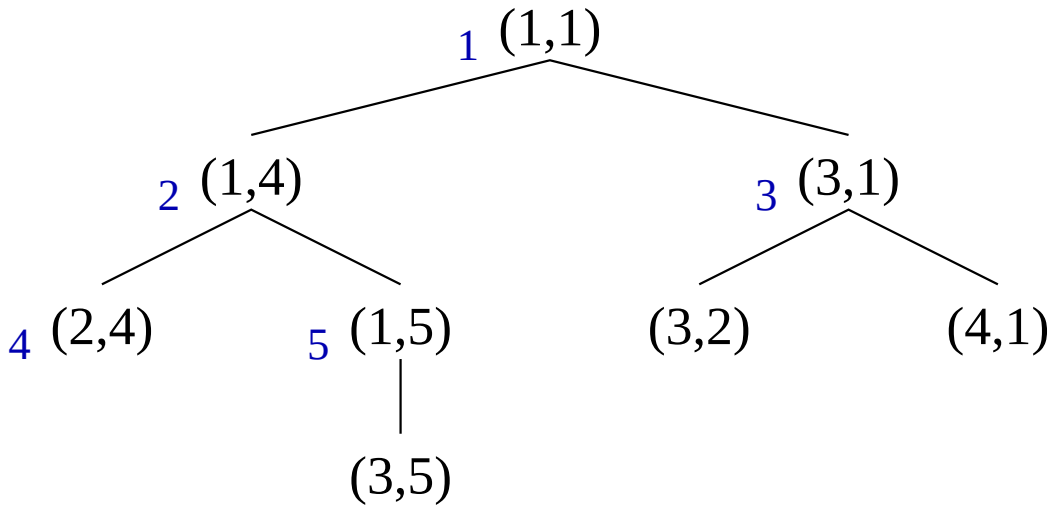
Recherche arborescente en largeur



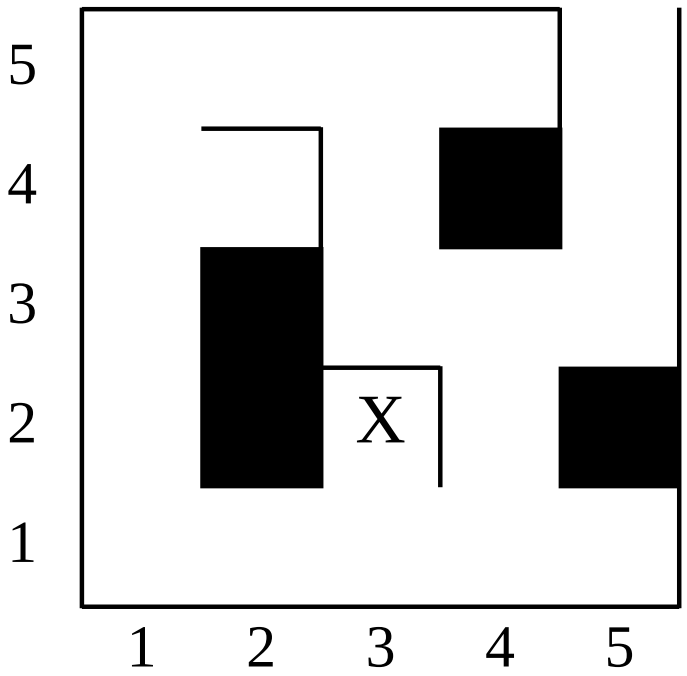


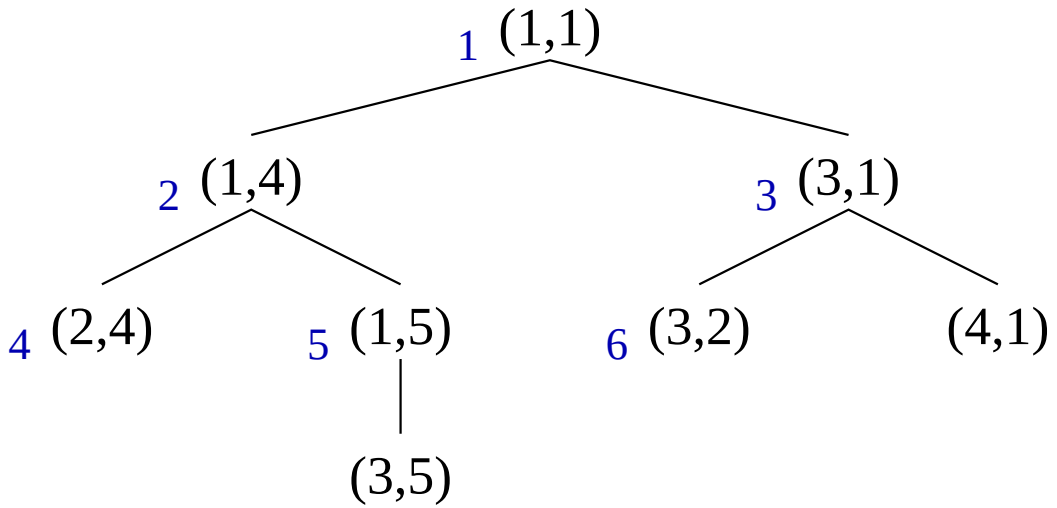
Recherche arborescente en largeur



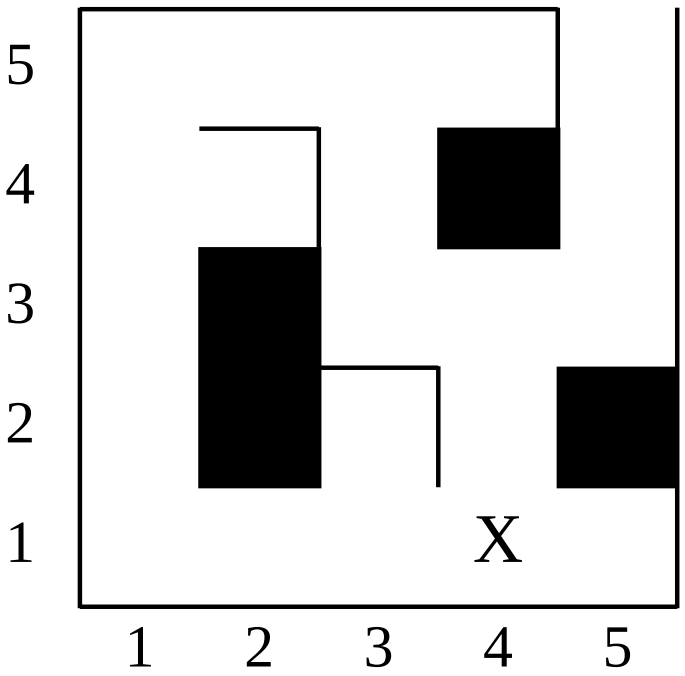


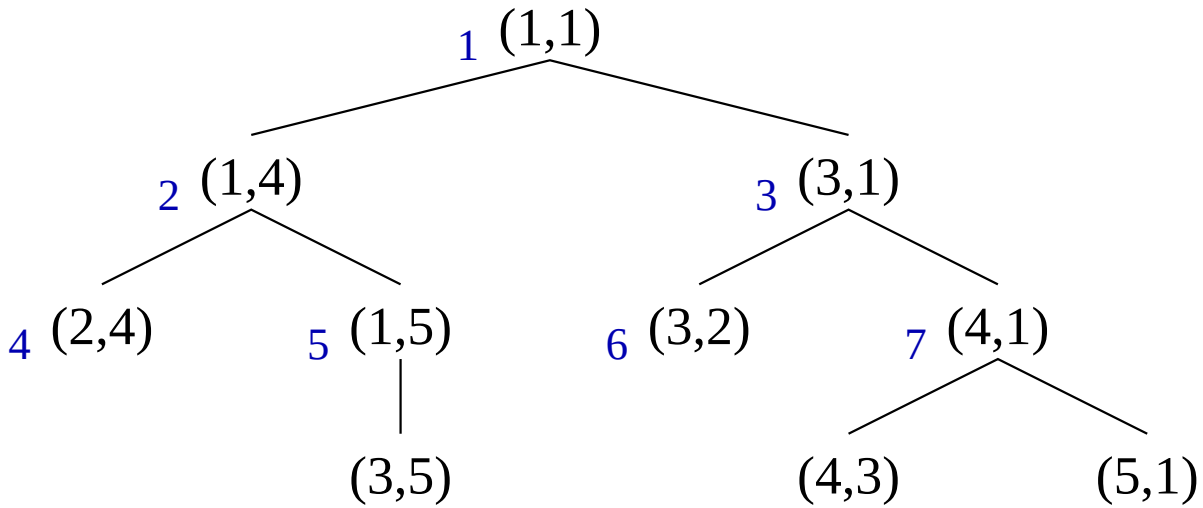
Recherche arborescente en largeur



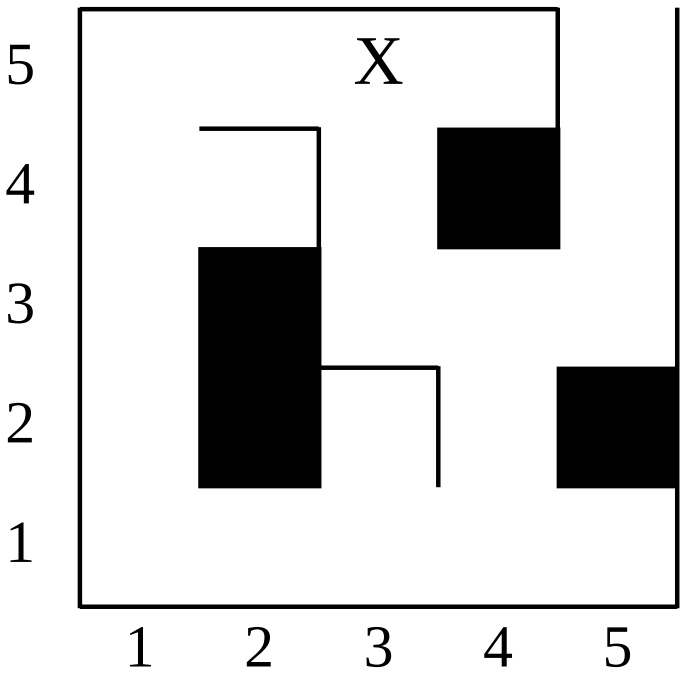


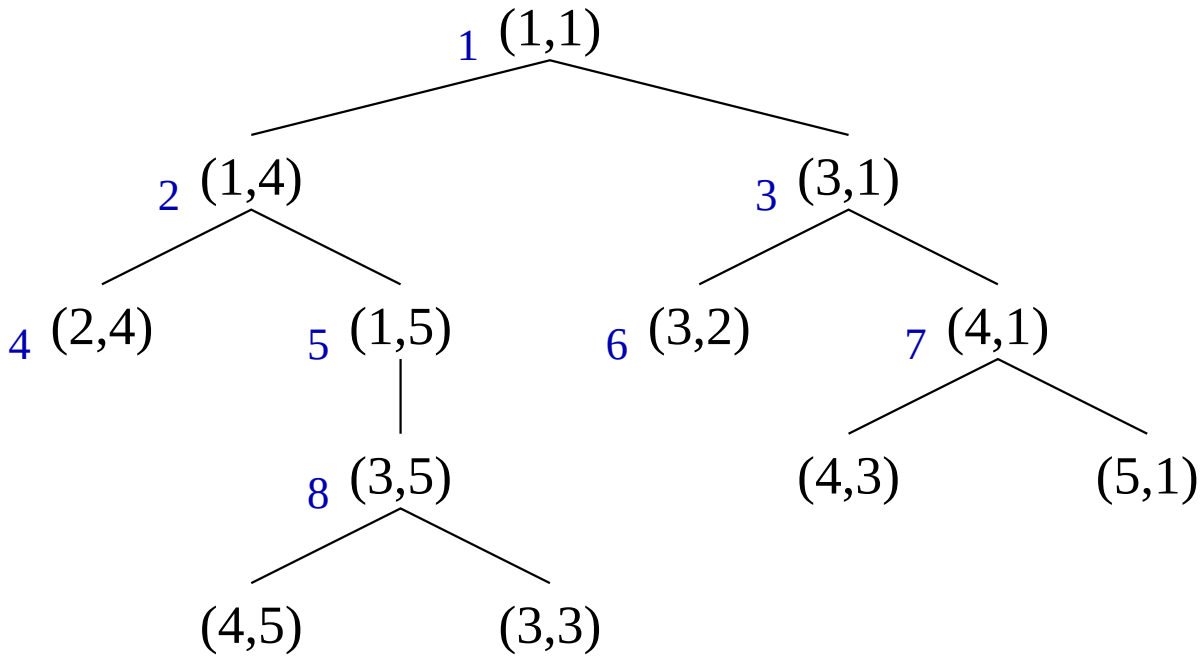
Recherche arborescente en largeur



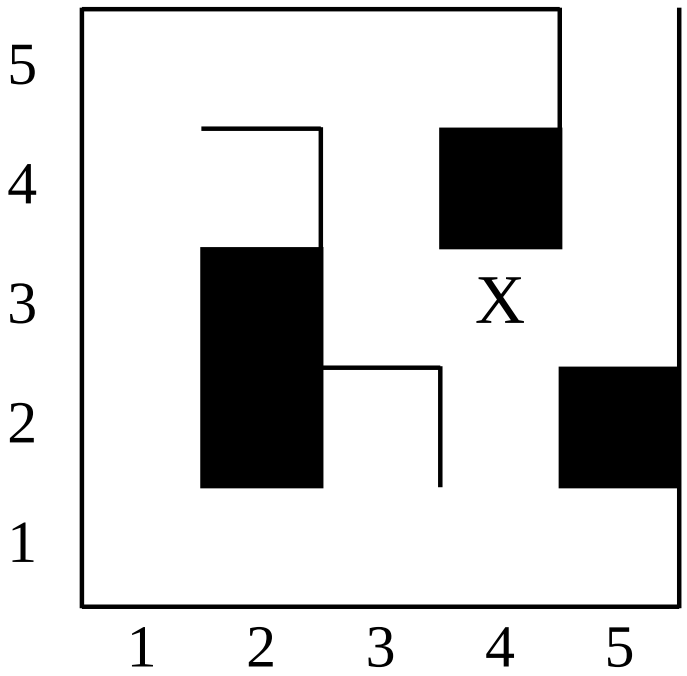


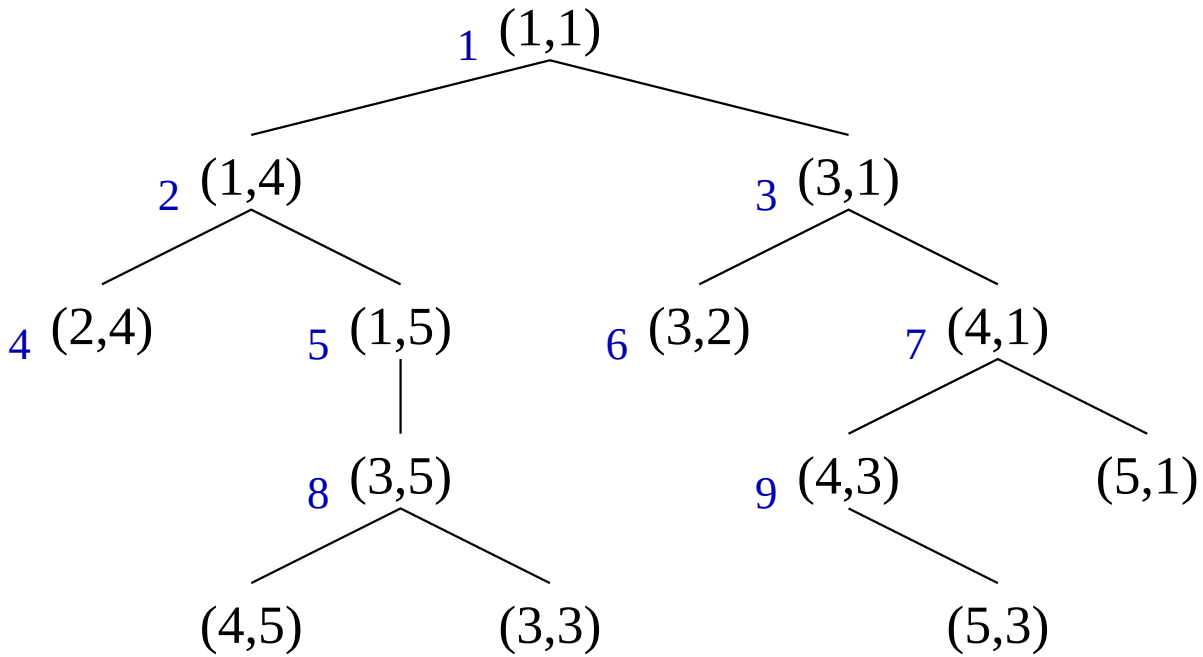
Recherche arborescente en largeur



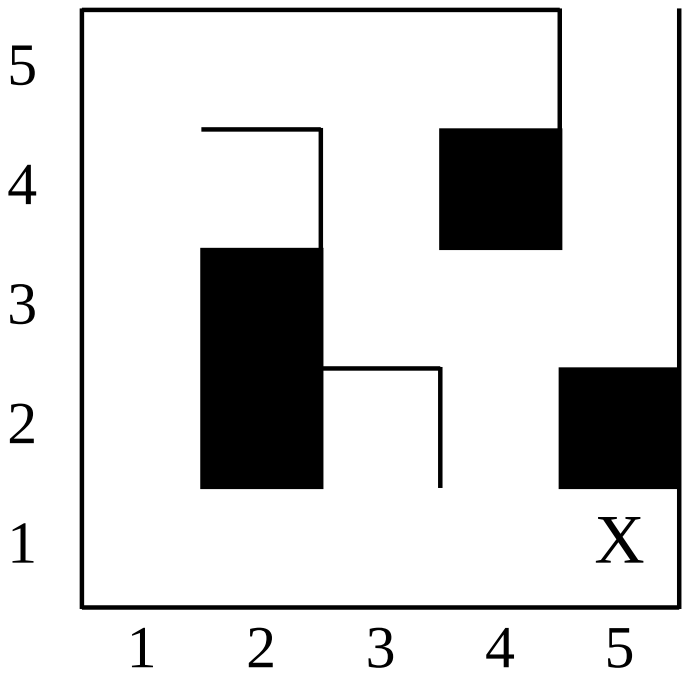


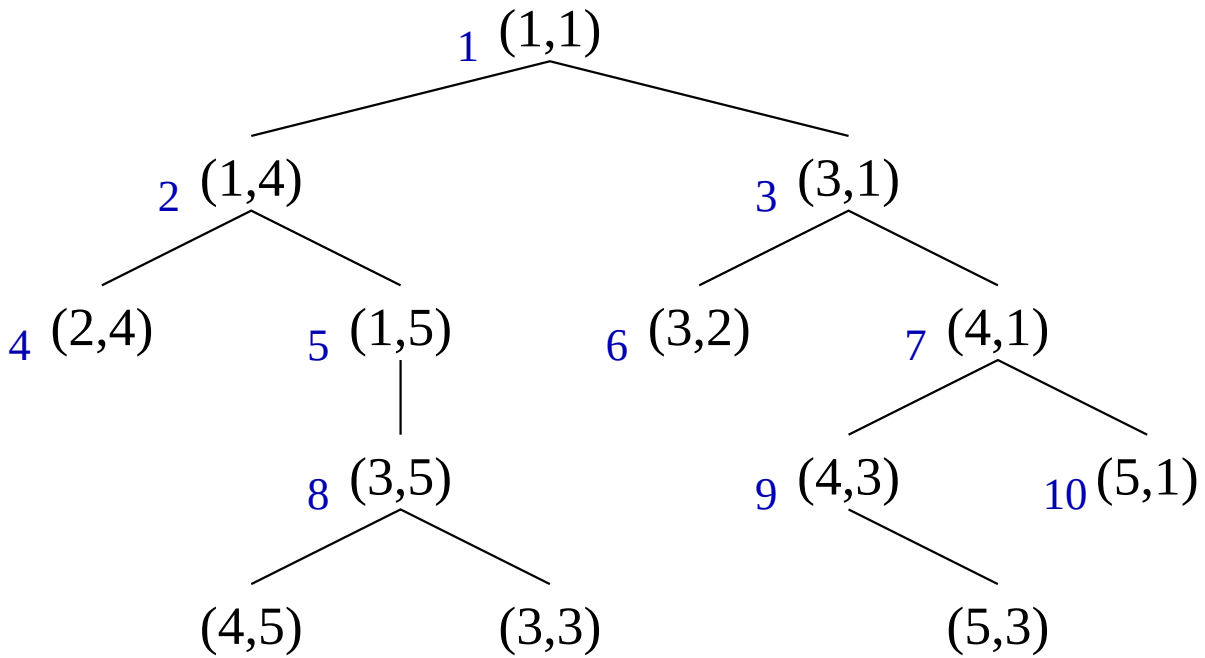
Recherche arborescente en largeur



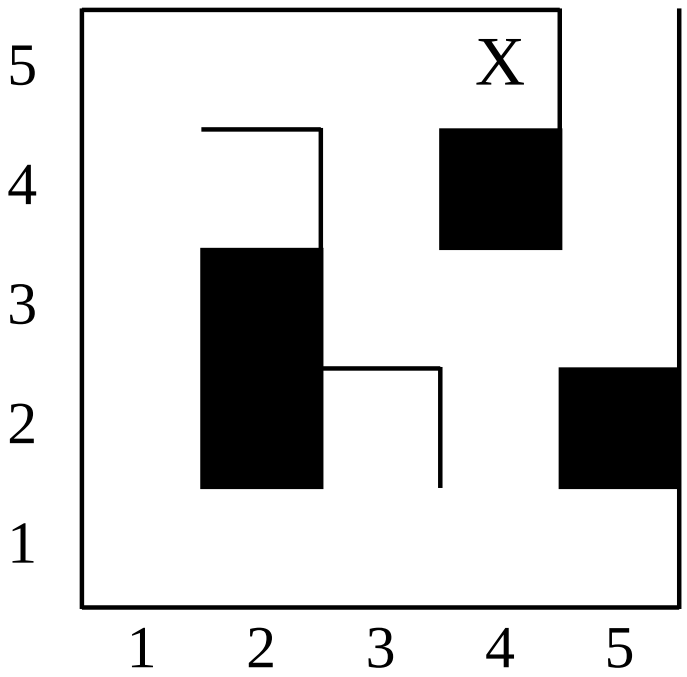


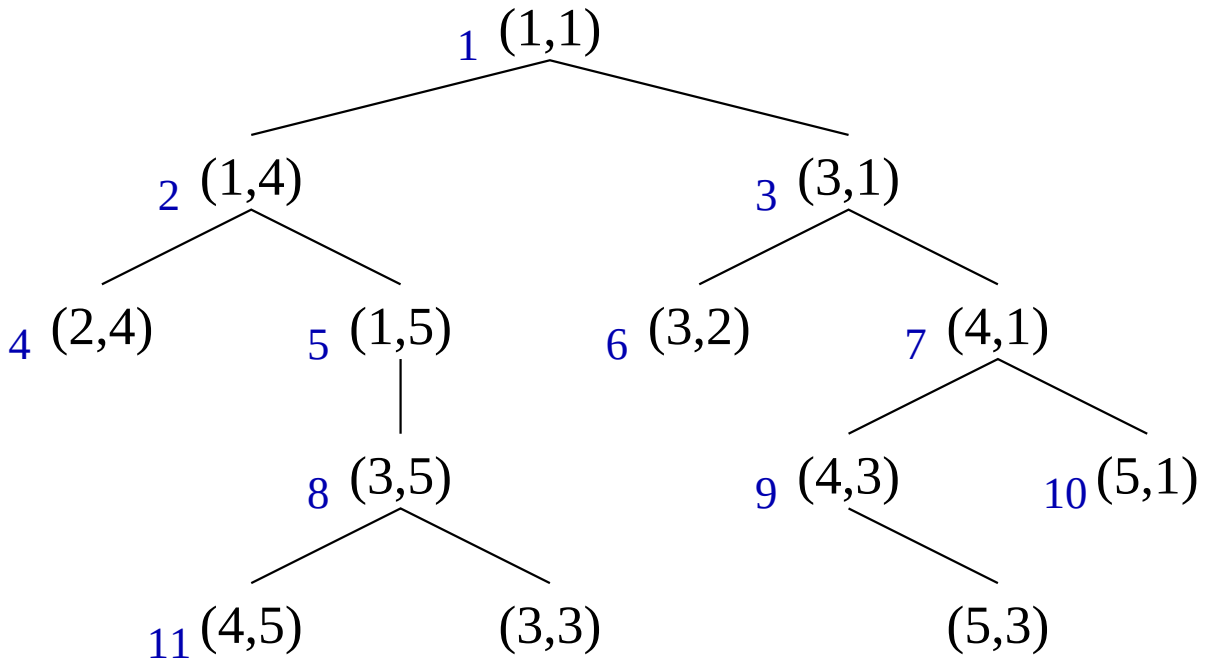
Recherche arborescente en largeur



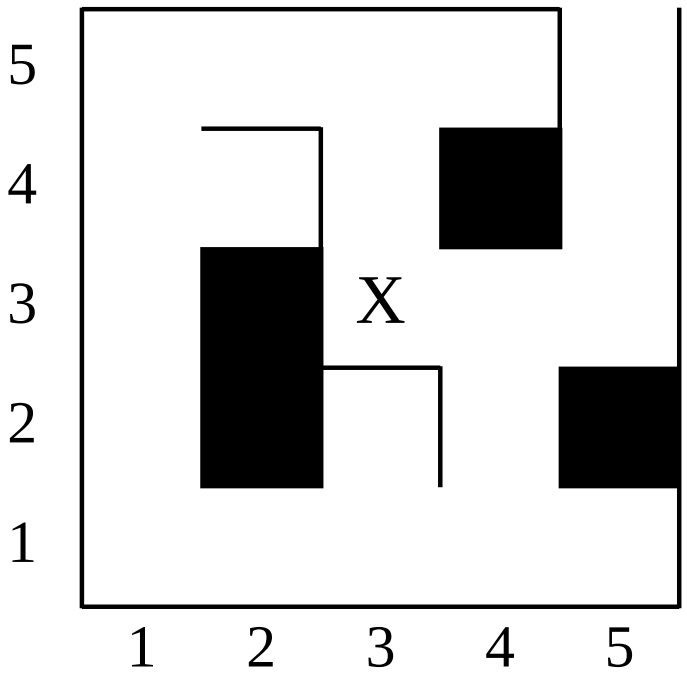


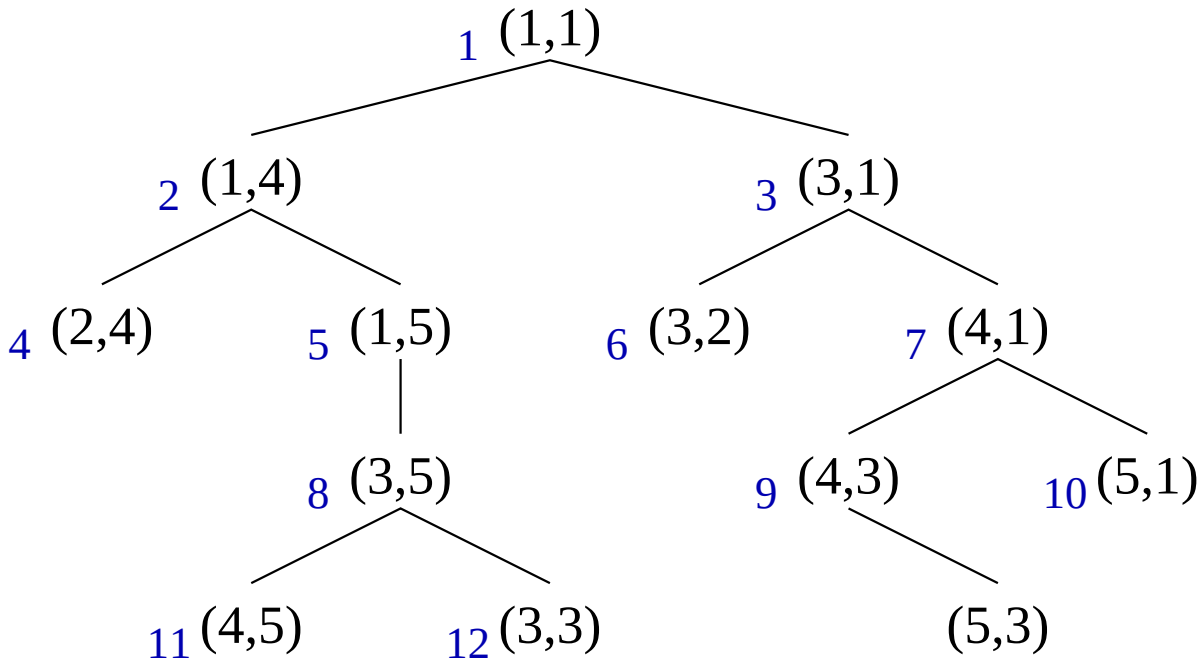
Recherche arborescente en largeur



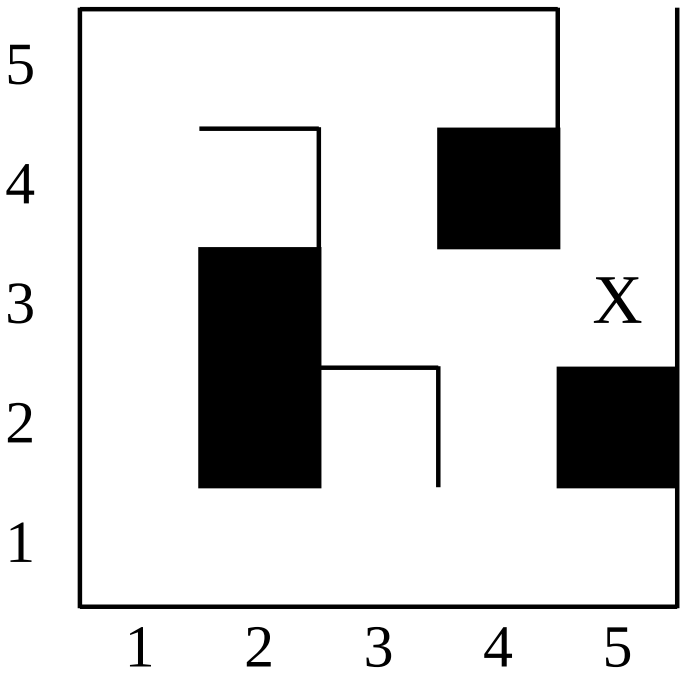


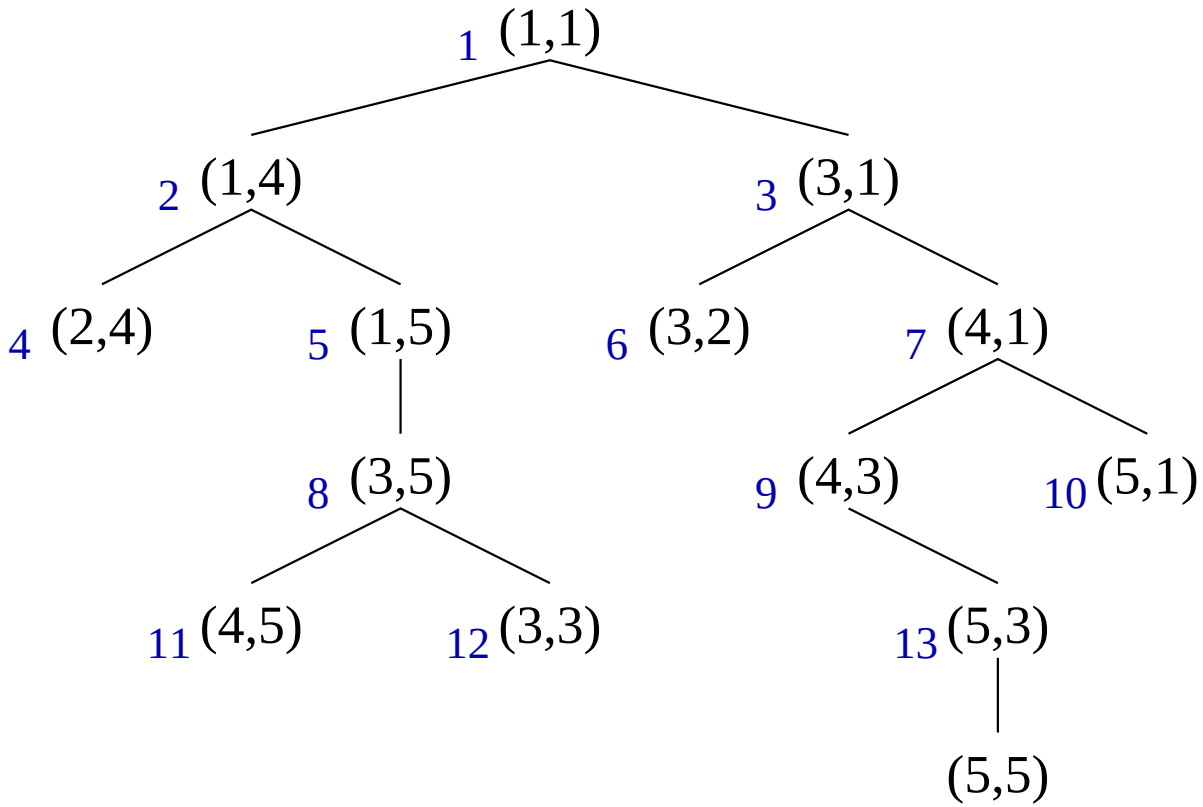
Recherche arborescente en largeur





Recherche arborescente en largeur





Les heuristiques

Heuristique = méthode approximative

Les heuristiques

Heuristique = méthode approximative

Objectif = guider la recherche afin de réduire le temps de résolution

Les heuristiques

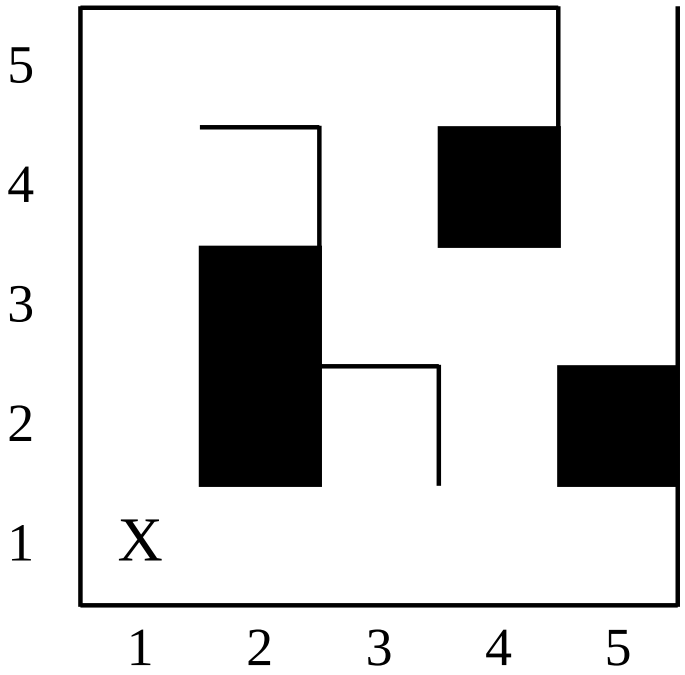
Heuristique = méthode approximative

Objectif = guider la recherche afin de réduire le temps de résolution

Mise en œuvre : estimation de la proximité d'un état par rapport à un état final.

Utilisation pour ordonner les états lors de l'insertion dans $Ouvert$

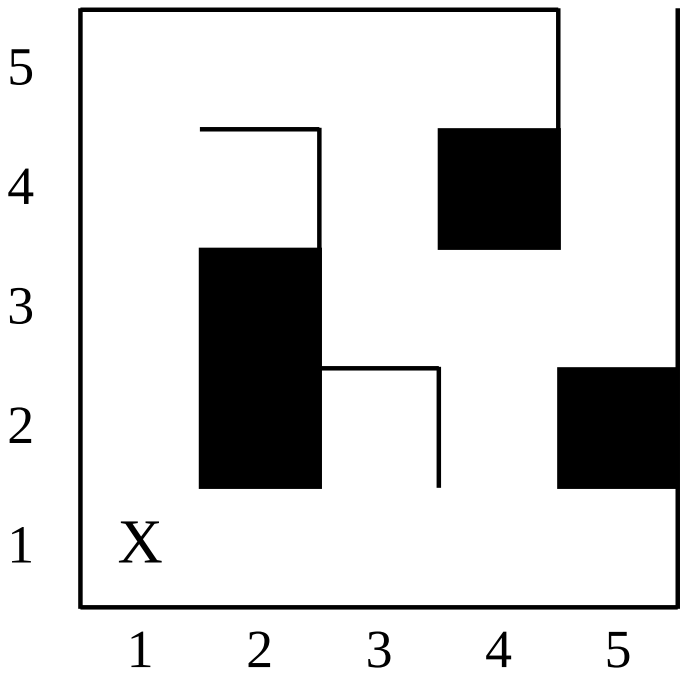
Exemple pour la recherche en profondeur

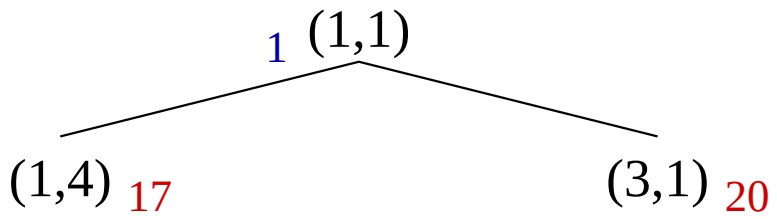


1 (1,1)

Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

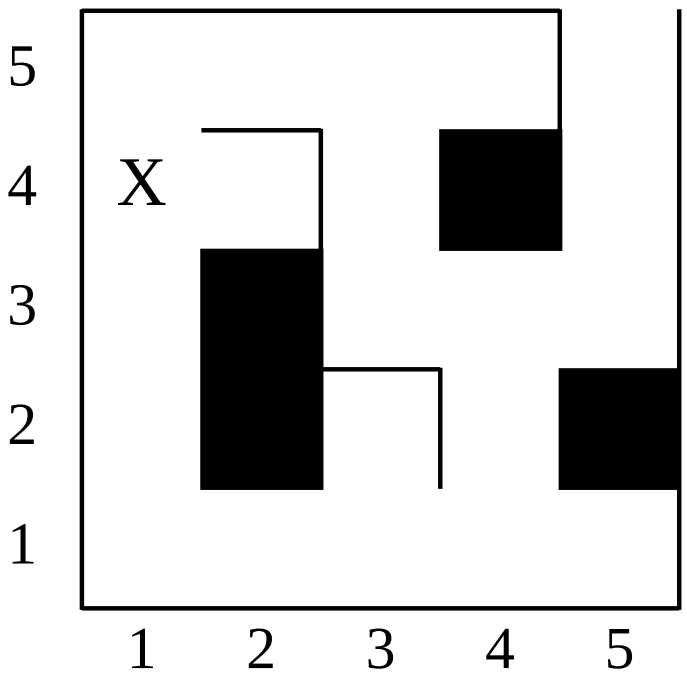
Exemple pour la recherche en profondeur

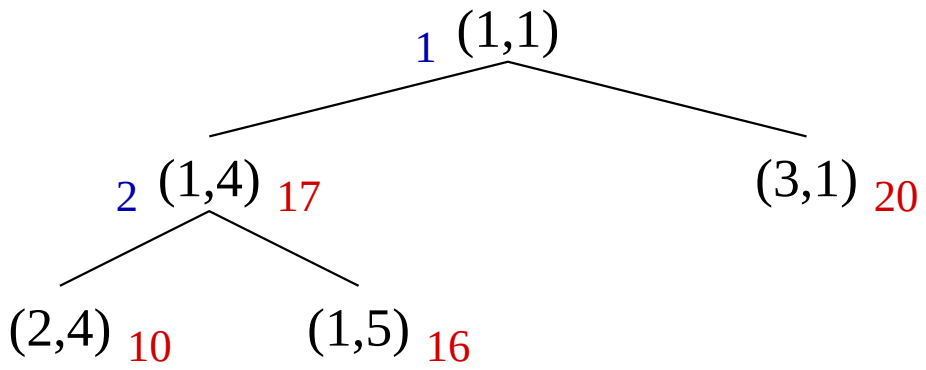




Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

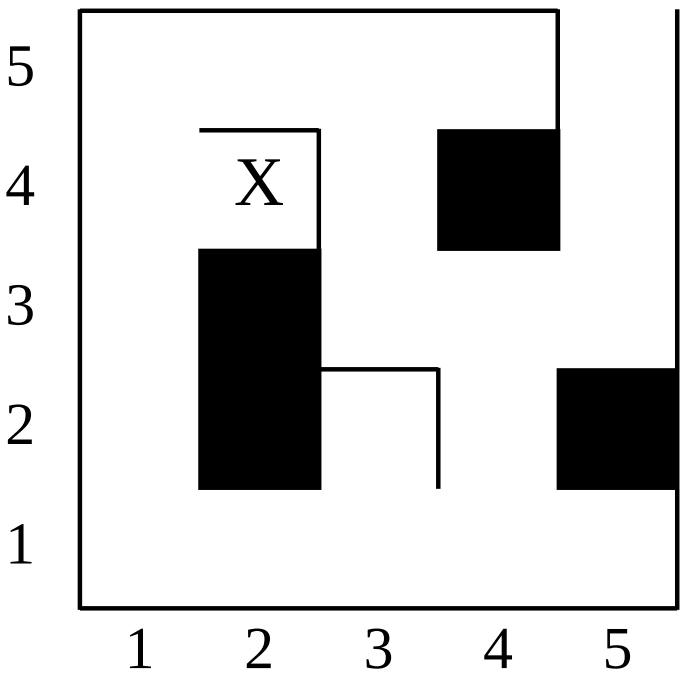
Exemple pour la recherche en profondeur

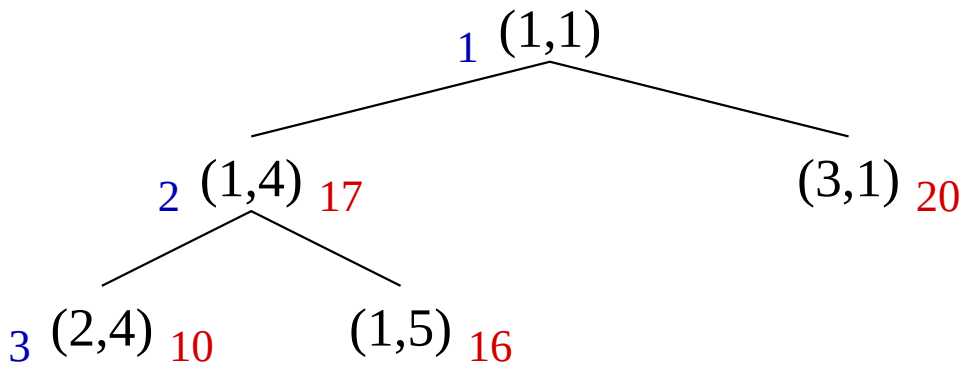




Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

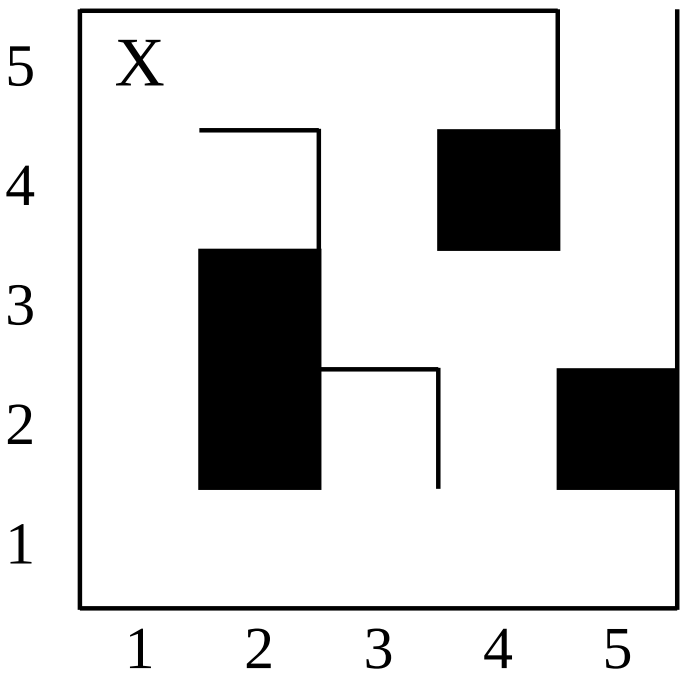
Exemple pour la recherche en profondeur

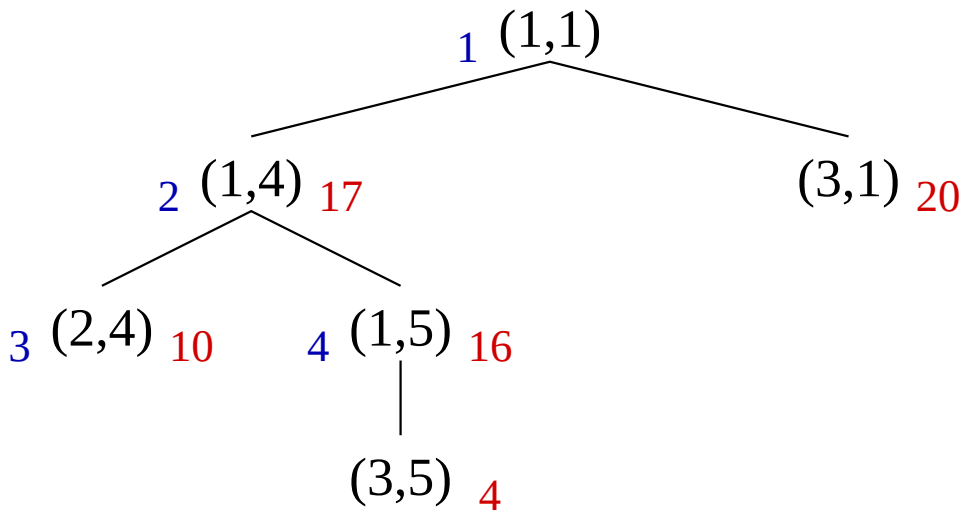




Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

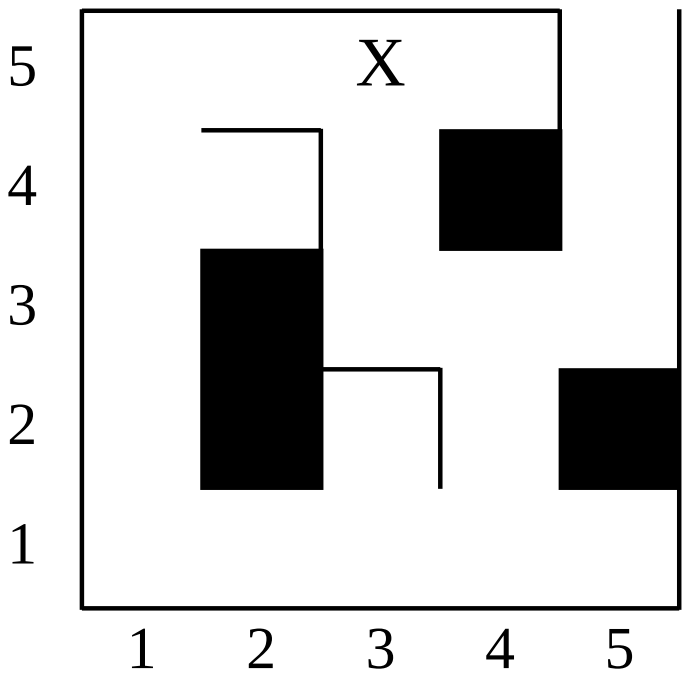
Exemple pour la recherche en profondeur

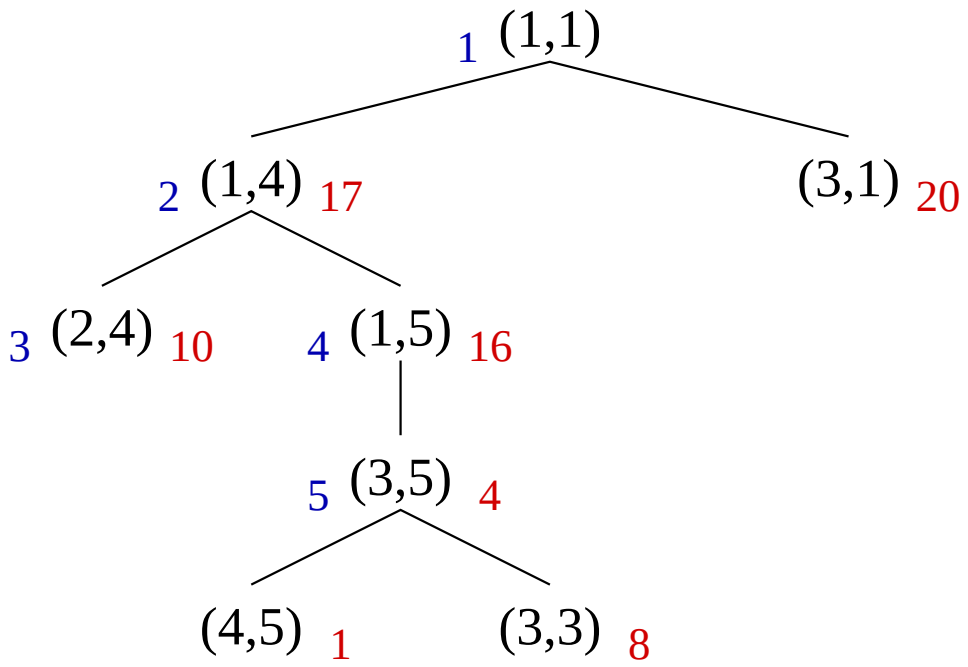




Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

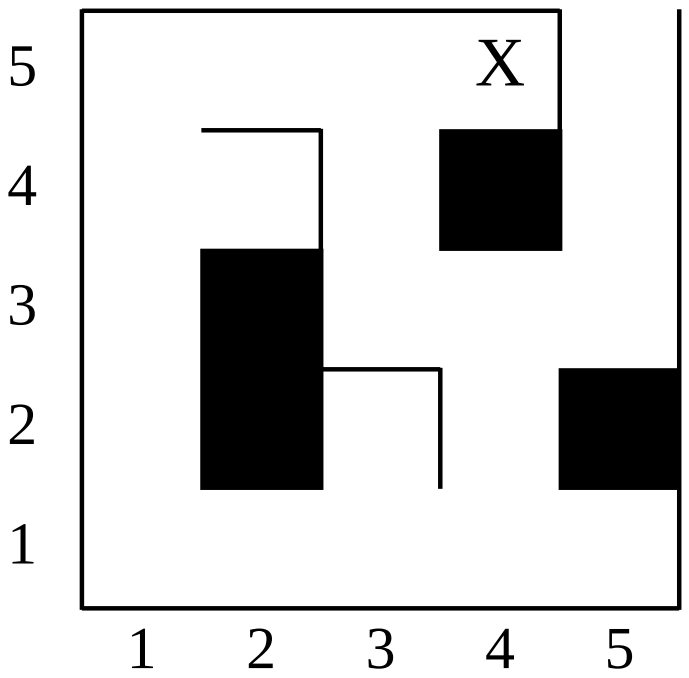
Exemple pour la recherche en profondeur

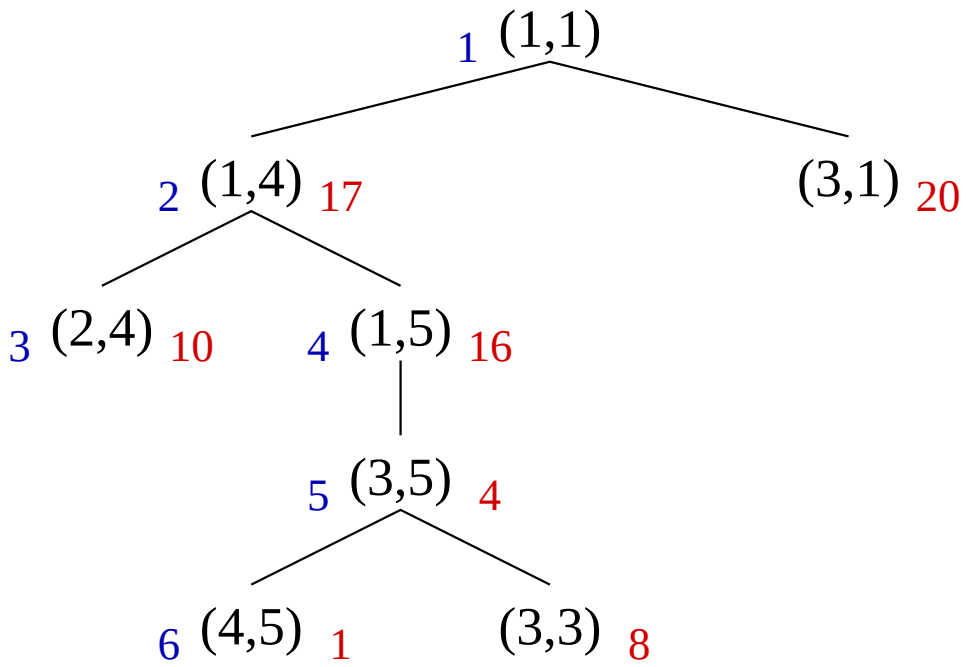




Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

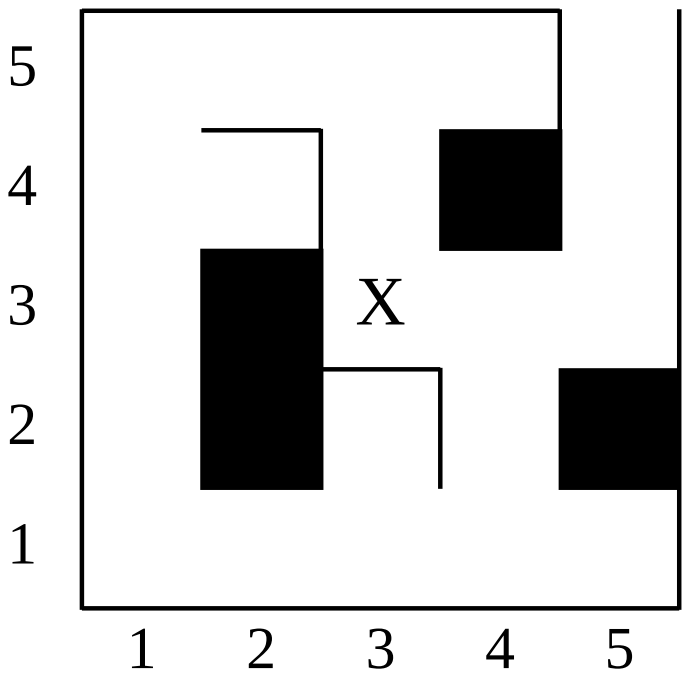
Exemple pour la recherche en profondeur

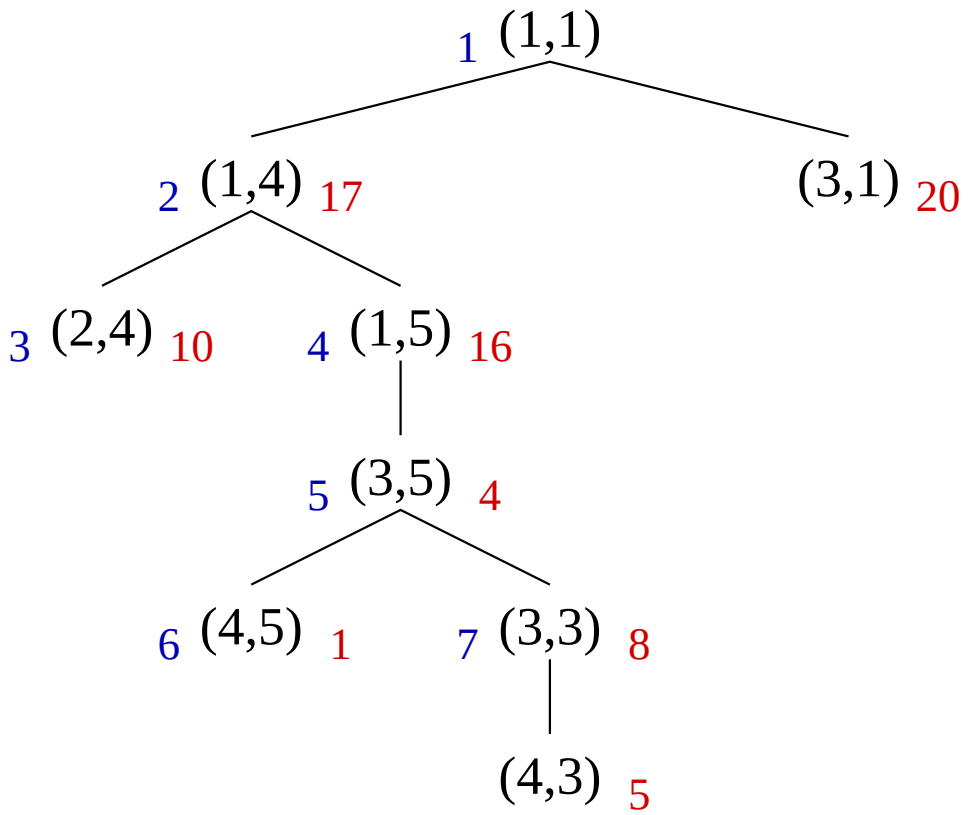




Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

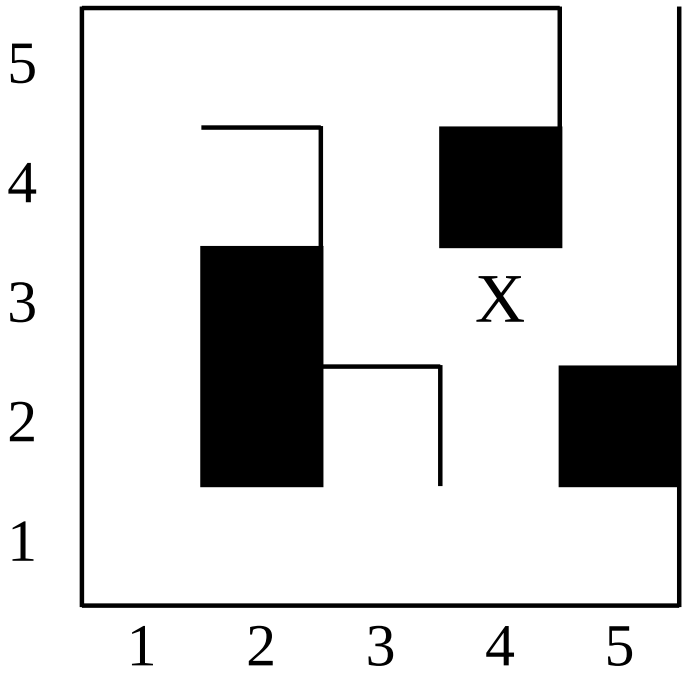
Exemple pour la recherche en profondeur

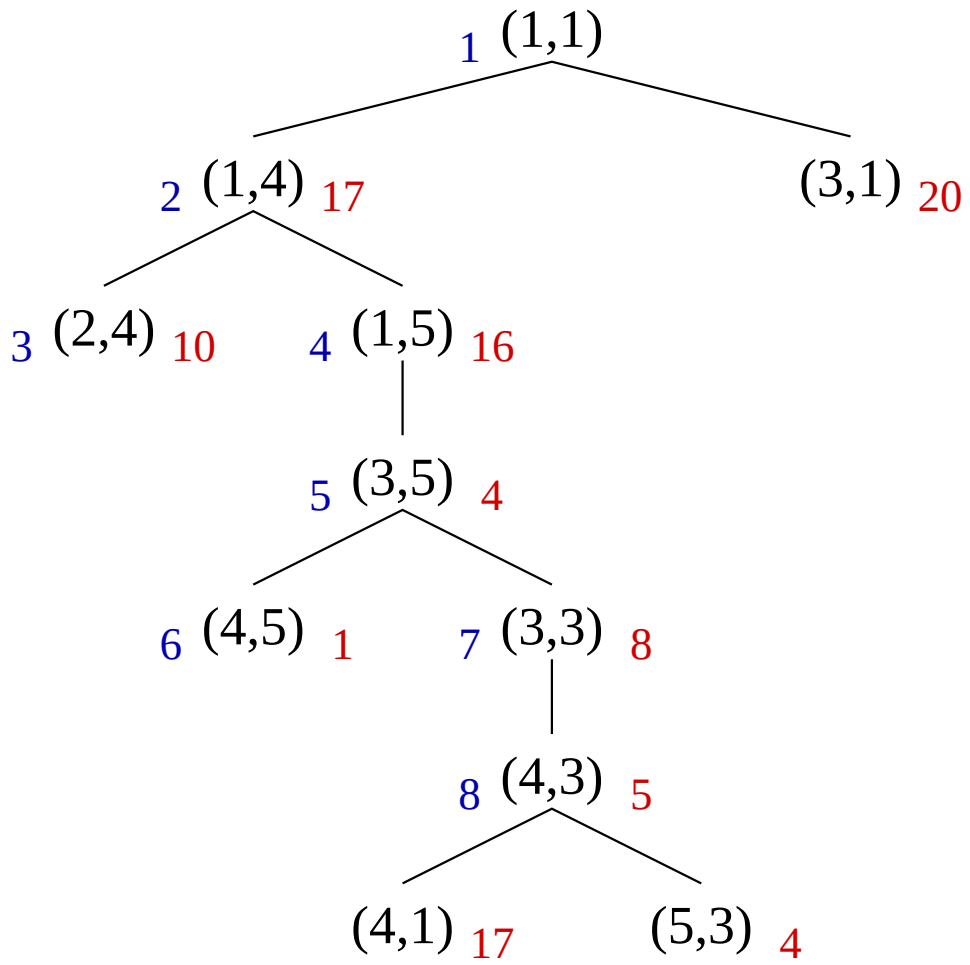




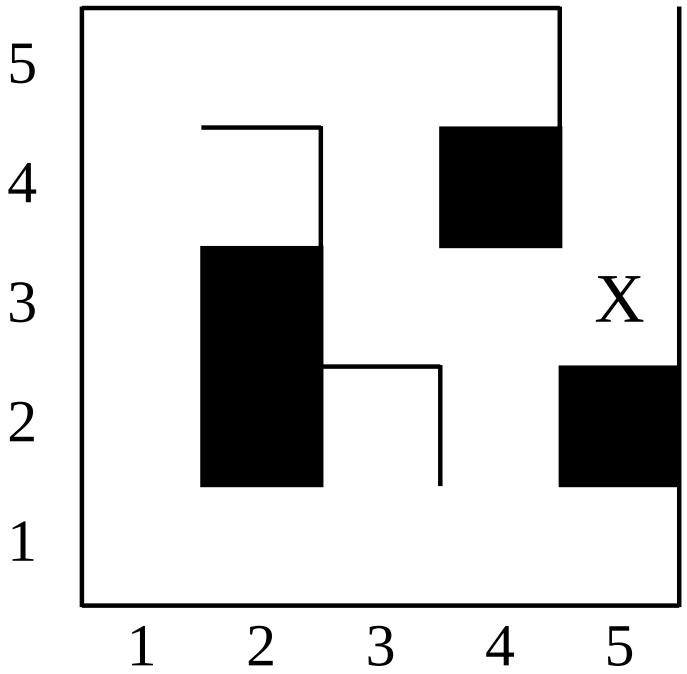
Heuristique : on ajoute en premier l'état qui minimise la valeur $h(e) = (5-x)^2 + (5-y)^2$

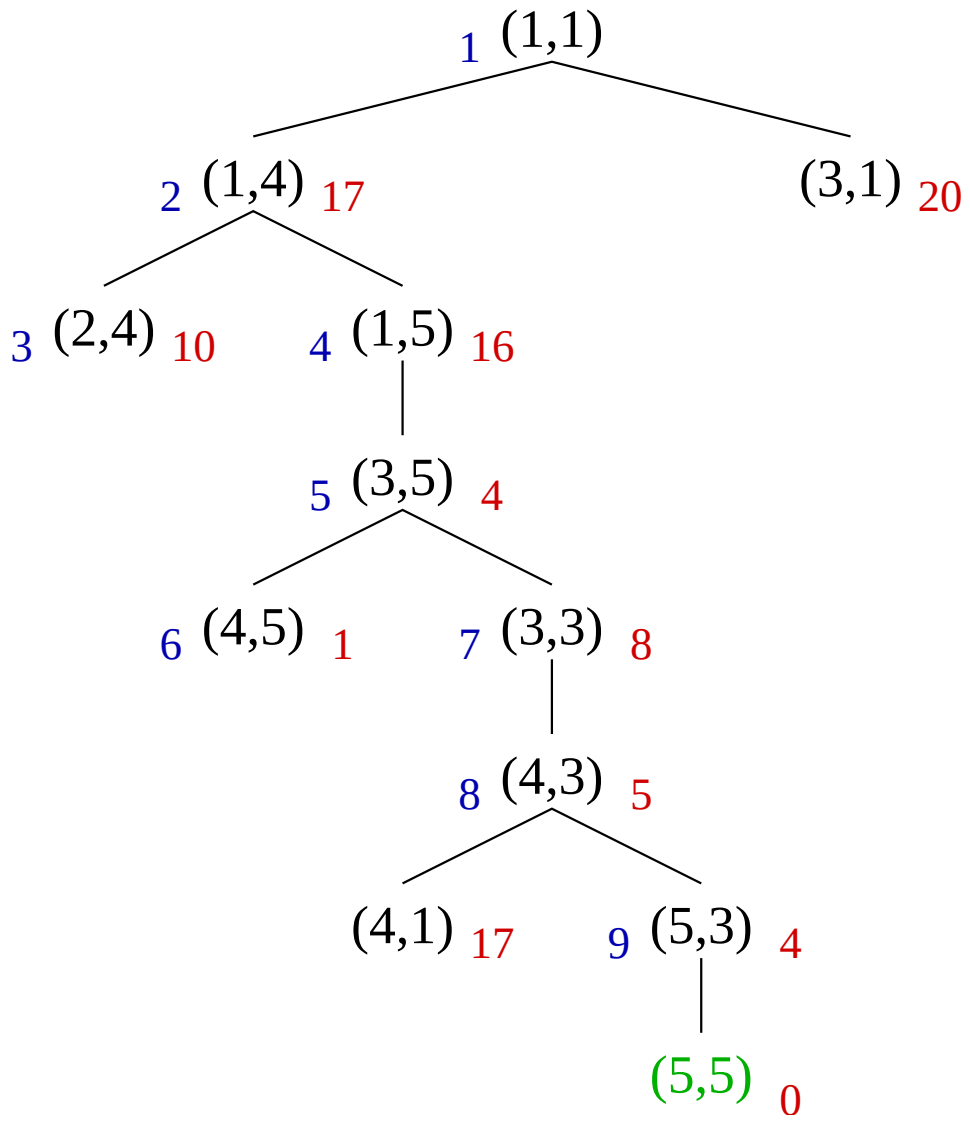
Exemple pour la recherche en profondeur





Exemple pour la recherche en profondeur





Recherche arborescente le meilleur d'abord

On examine les états les plus prometteurs d'abord selon une heuristique donnée.

$Ouvert$ a la structure d'une liste triée.

Recherche arborescente le meilleur d'abord

On examine les états les plus prometteurs d'abord selon une heuristique donnée.

$Ouvert$ a la structure d'une liste triée.

Avantage :

- une bonne efficacité si l'heuristique est de bonne qualité

Recherche arborescente le meilleur d'abord

On examine les états les plus prometteurs d'abord selon une heuristique donnée.

$Ouvert$ a la structure d'une liste triée.

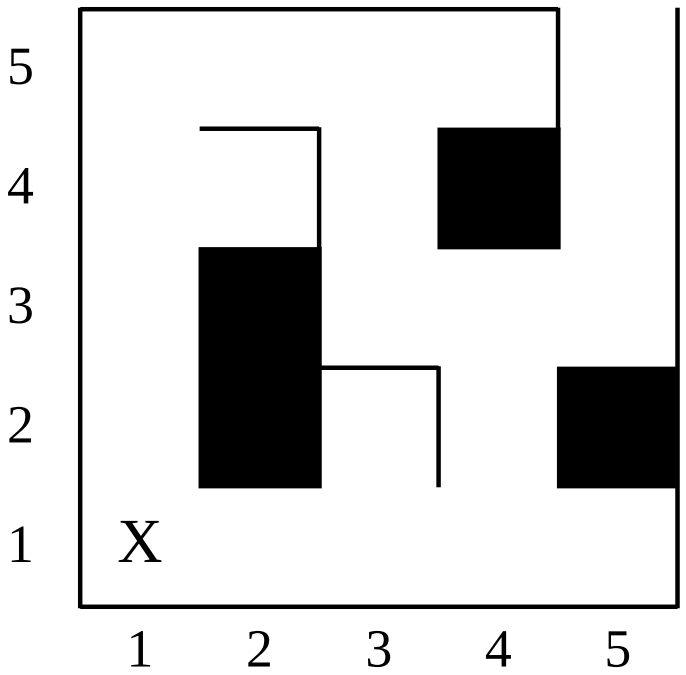
Avantage :

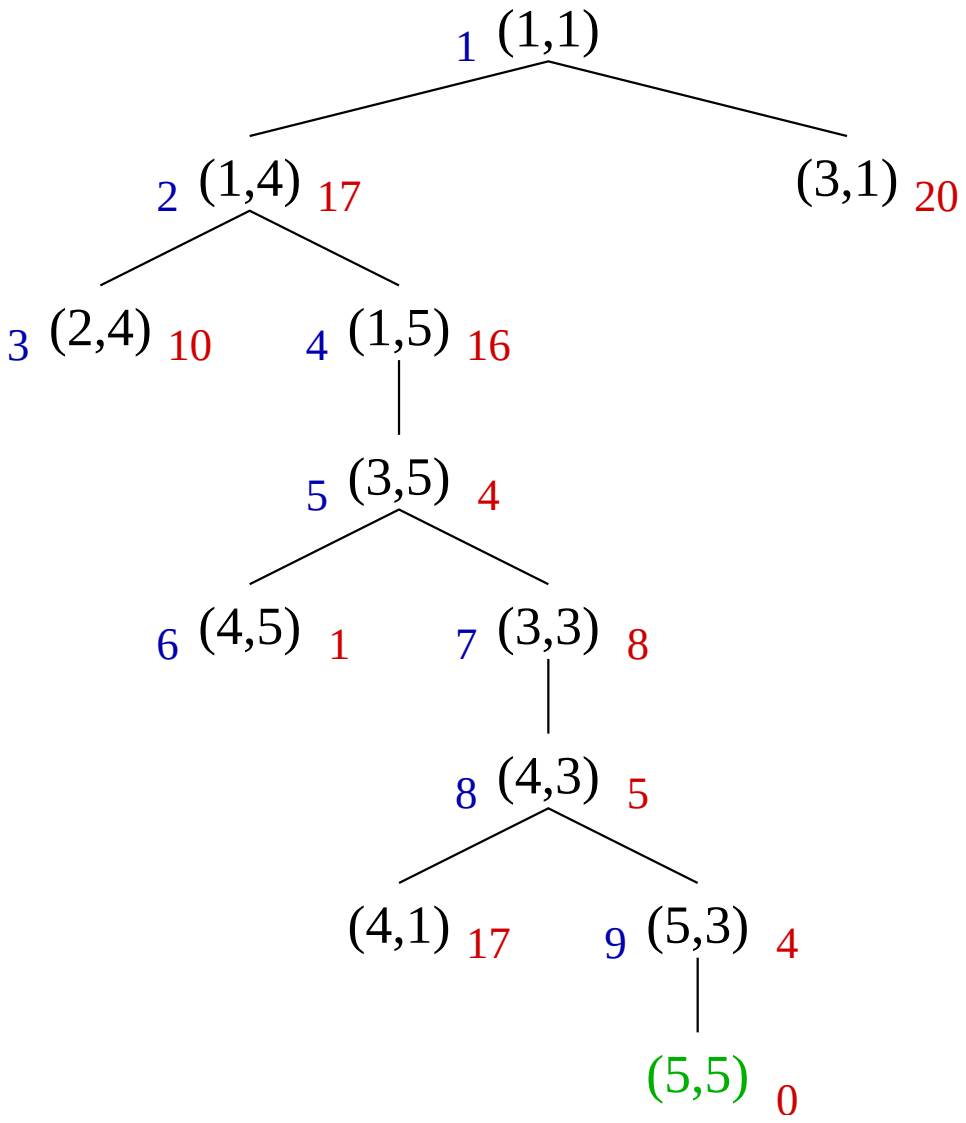
- une bonne efficacité si l'heuristique est de bonne qualité

Inconvénients :

- risque de parcours intégral de l'espace de recherche
- coût en mémoire très important voire prohibitif

Recherche arborescente le meilleur d'abord





Iterative Deepening

Un compromis entre recherche en largeur et recherche en profondeur

Iterative Deepening

Un compromis entre recherche en largeur et recherche en profondeur

Principe :

- borner la recherche en profondeur :

  seuls les états situés à une distance d de l'état initial sont générés

Iterative Deepening

Un compromis entre recherche en largeur et recherche en profondeur

Principe :

- borner la recherche en profondeur :

  seuls les états situés à une distance d de l'état initial sont générés

- lancer itérativement plusieurs recherches en profondeur avec des distances d croissantes

Iterative Deepening

Avantages :

- garantie de trouver une solution s'il en existe une
- la première solution trouvée est la moins profonde
- coût en mémoire limité

Iterative Deepening

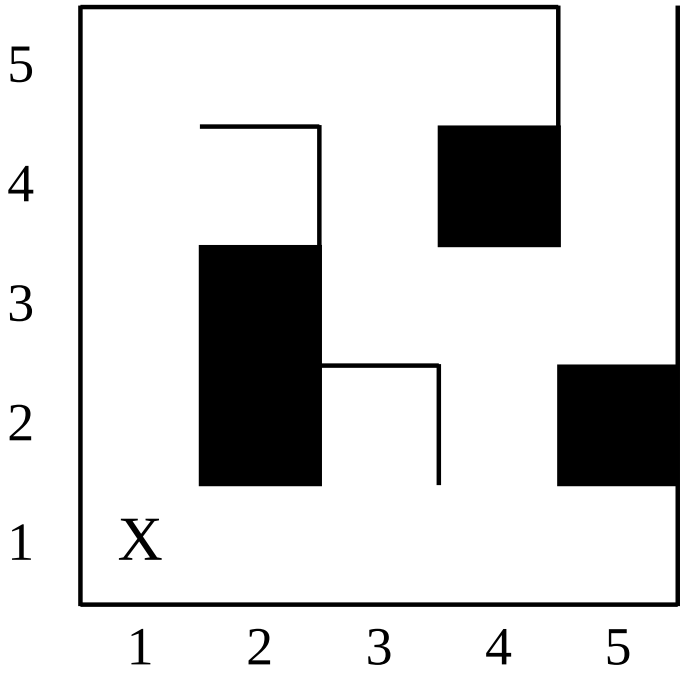
Avantages :

- garantie de trouver une solution s'il en existe une
- la première solution trouvée est la moins profonde
- coût en mémoire limité

Inconvénients :

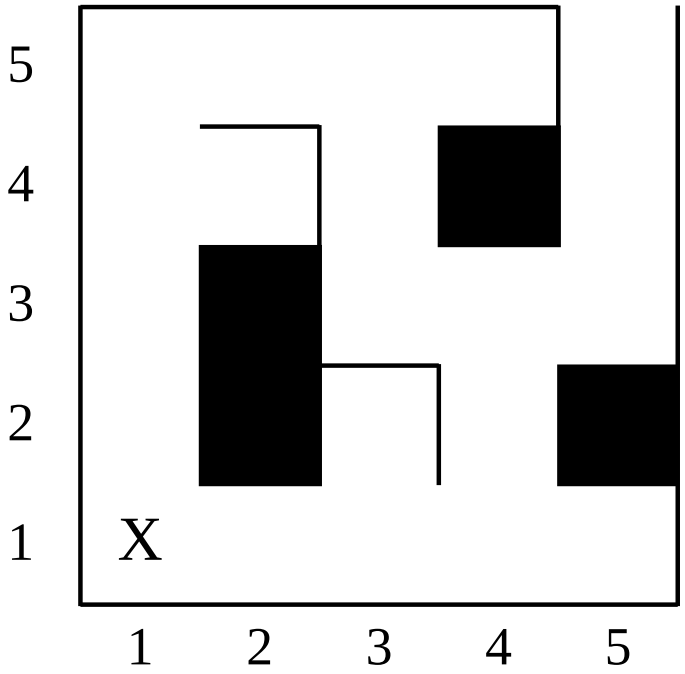
- risque de parcours intégral de l'espace de recherche
- des redondances : un état peut être examiné plusieurs fois

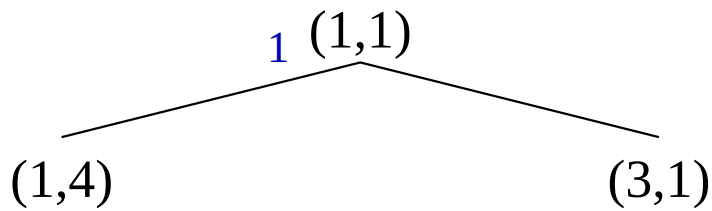
Exemple $d = 1$



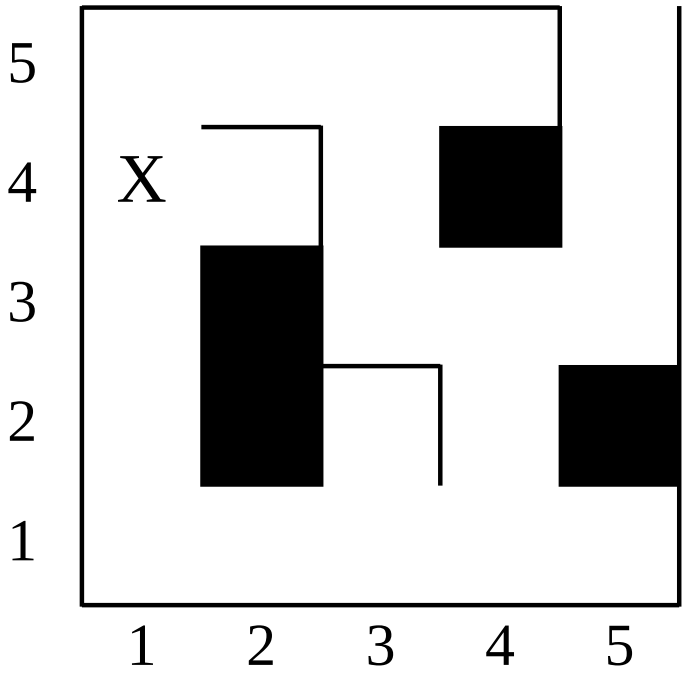
1 (1,1)

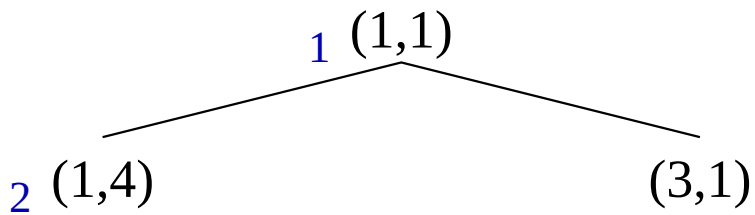
Exemple $d = 1$



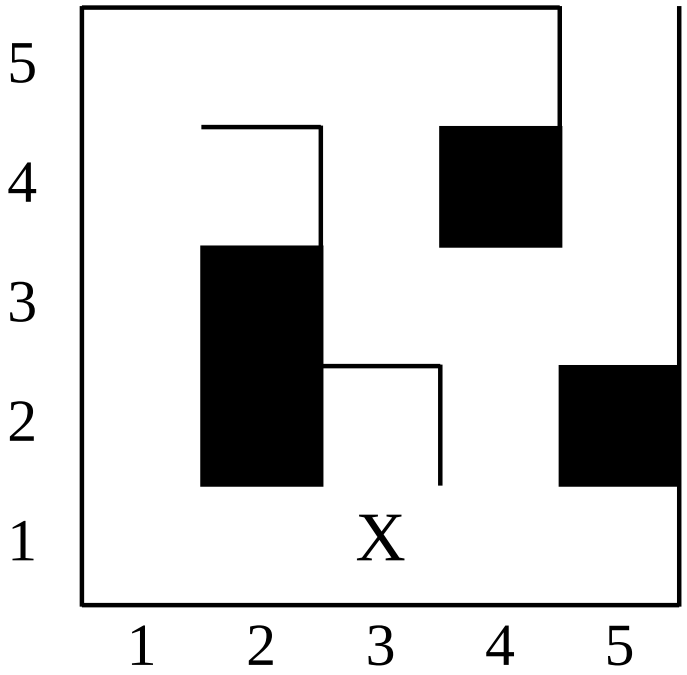


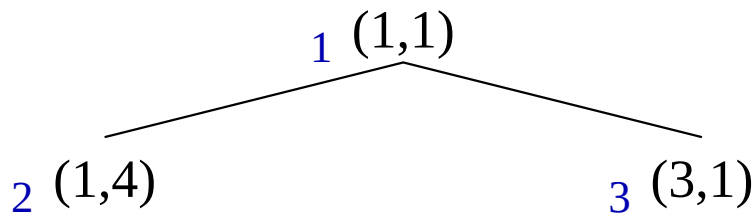
Exemple $d = 1$



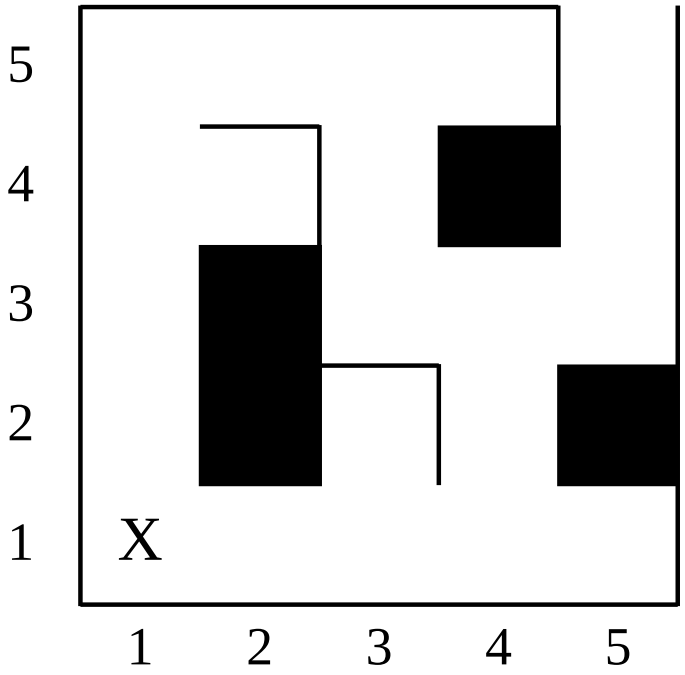


Exemple $d = 1$



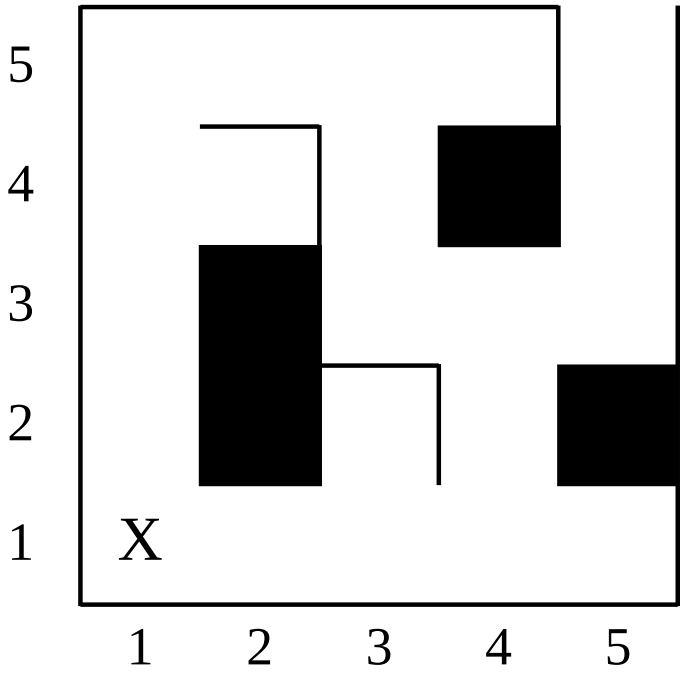


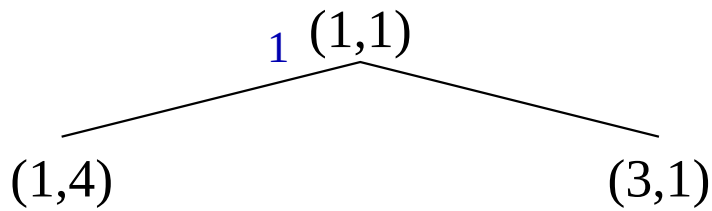
Exemple $d = 2$



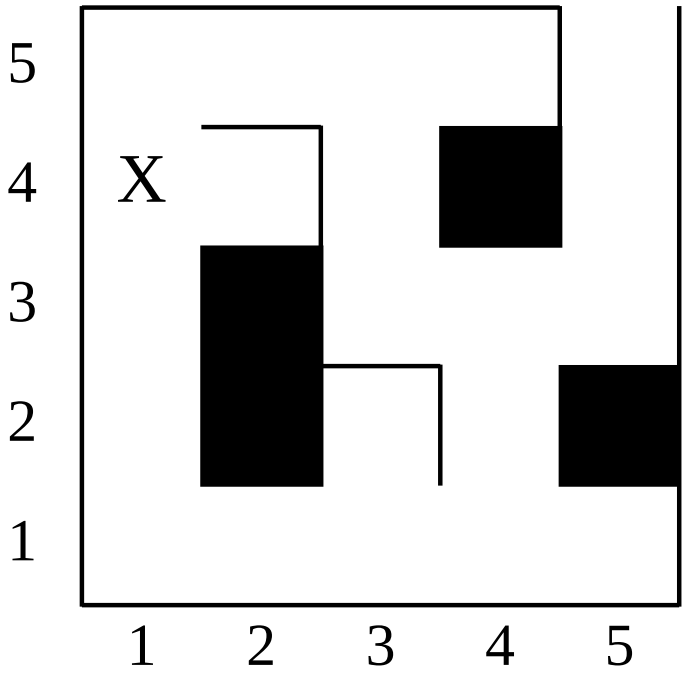
1 (1,1)

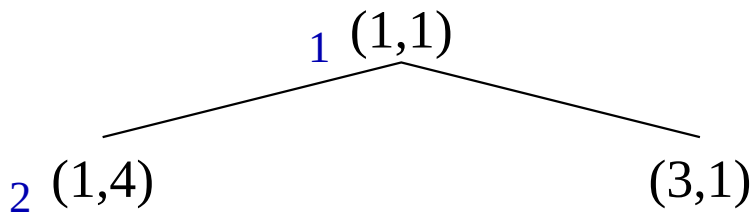
Exemple $d = 2$



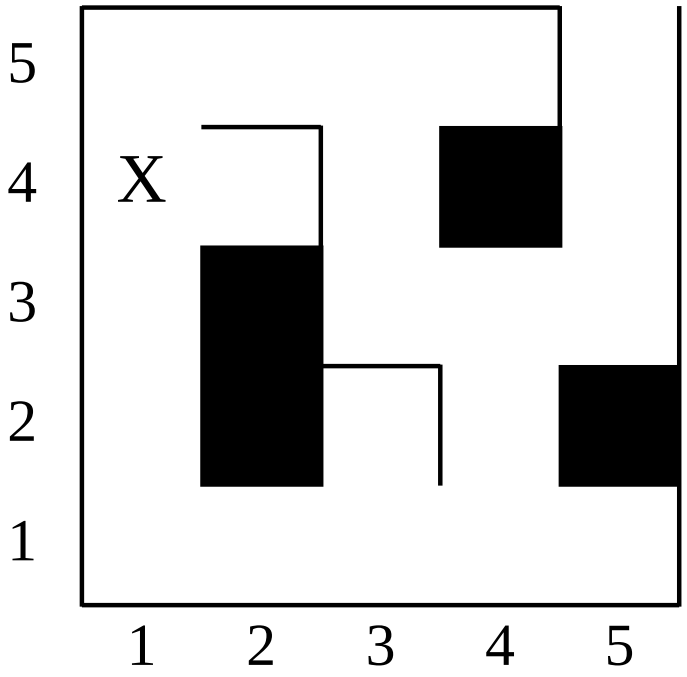


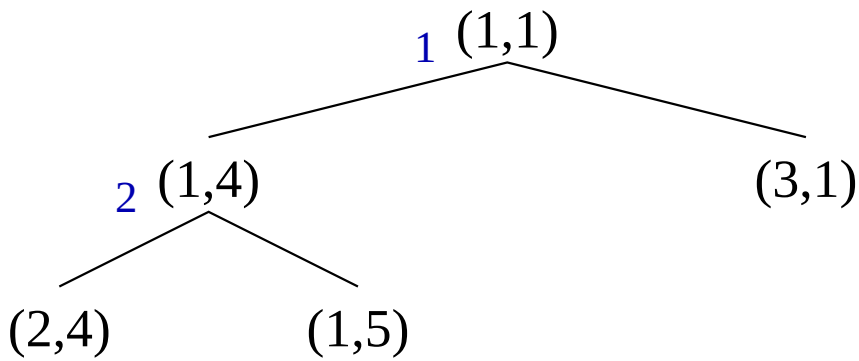
Exemple $d = 2$



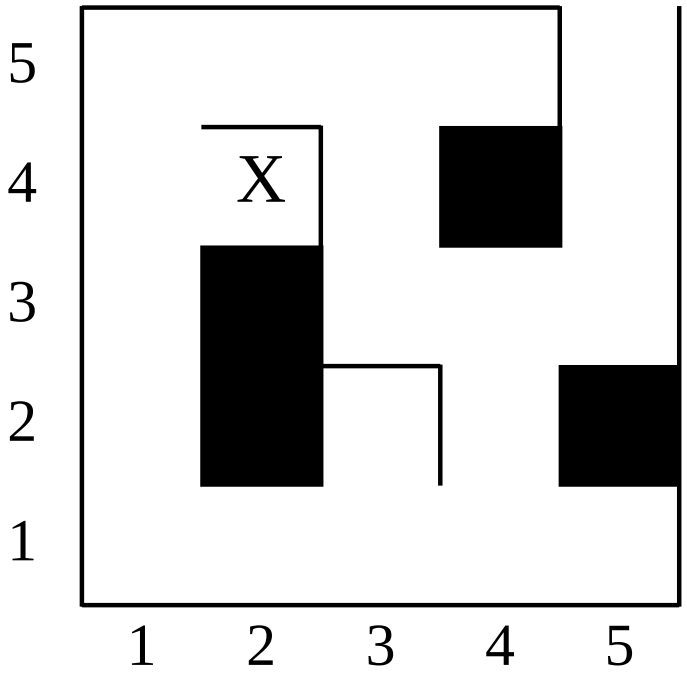


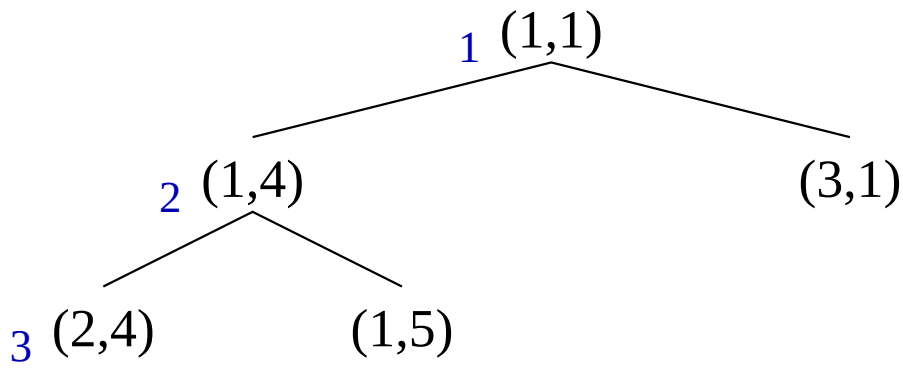
Exemple $d = 2$



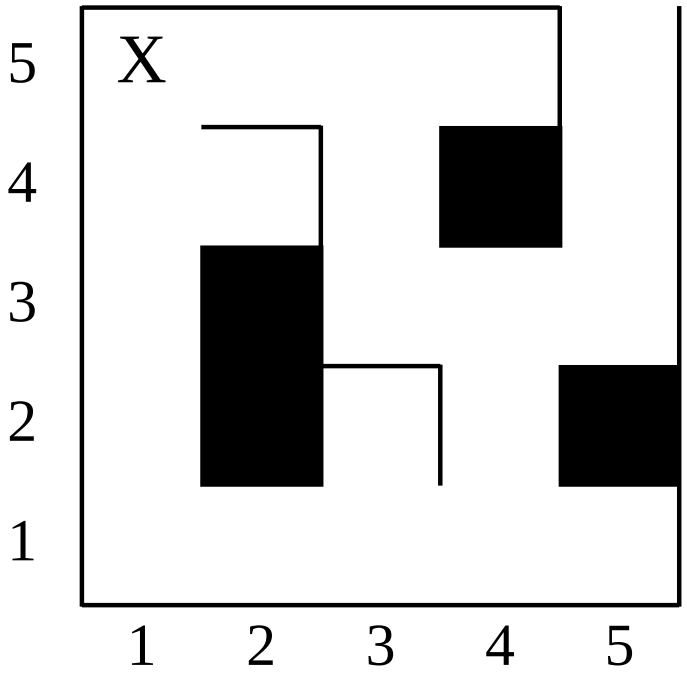


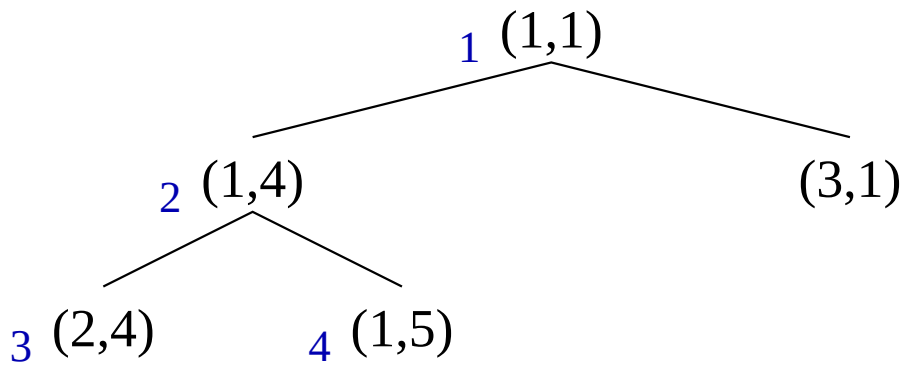
Exemple $d = 2$



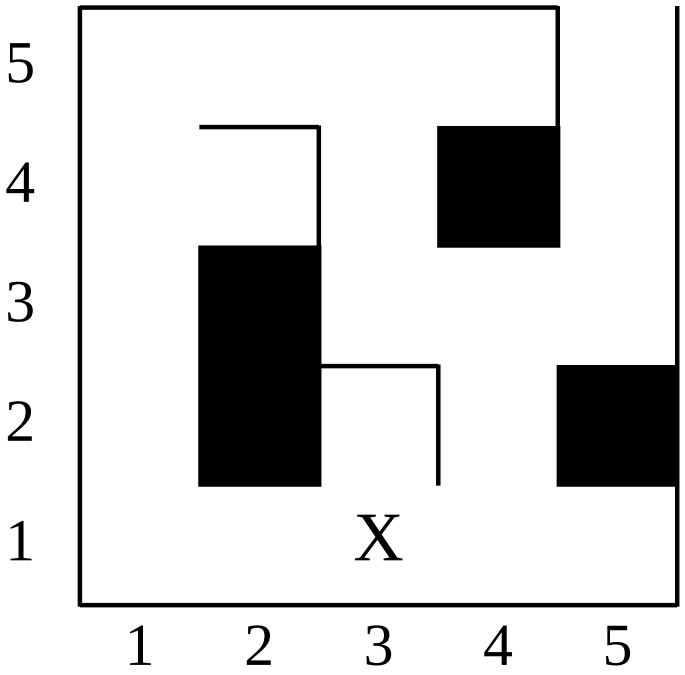


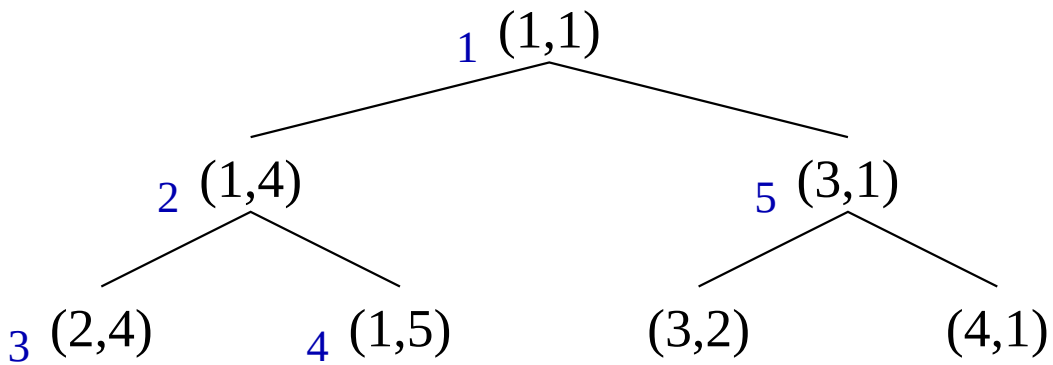
Exemple $d = 2$



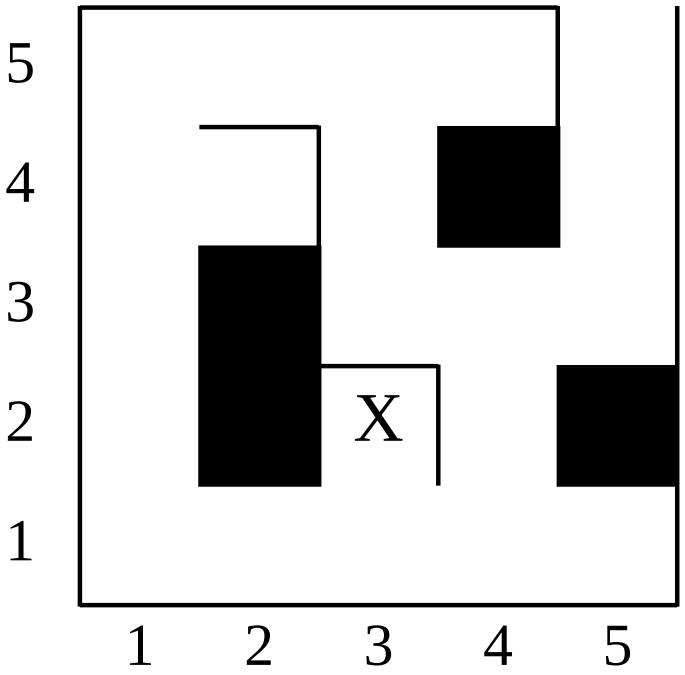


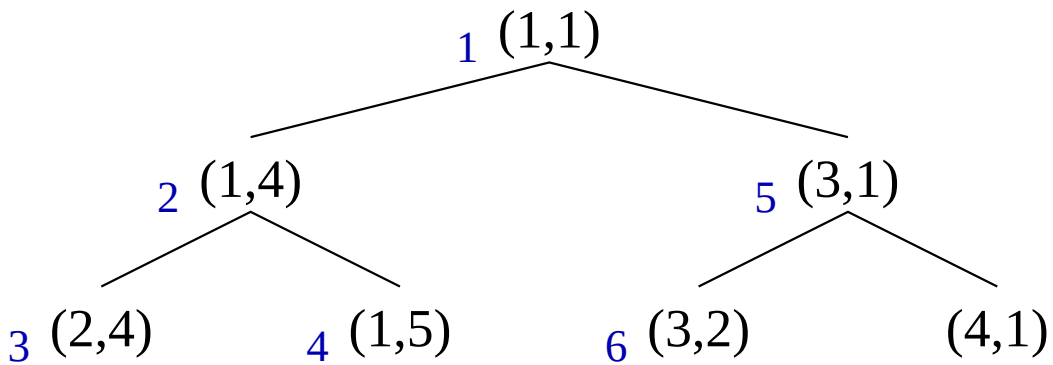
Exemple $d = 2$



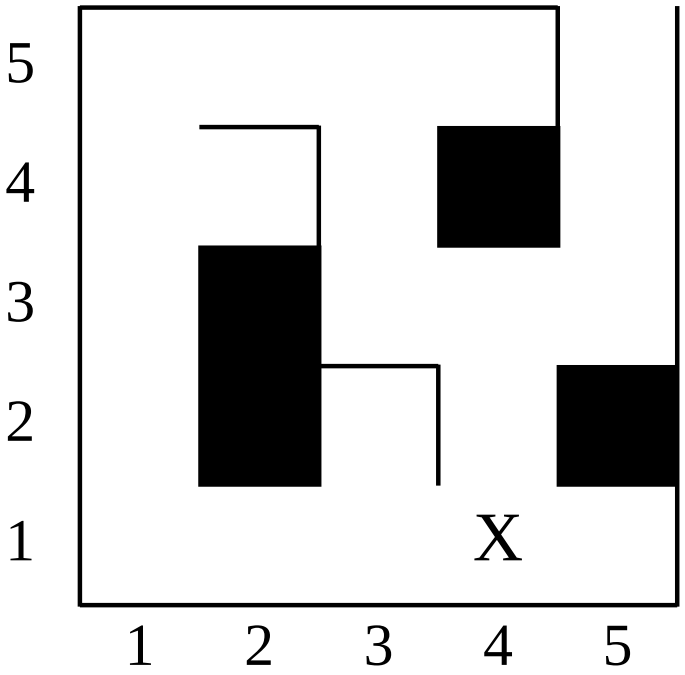


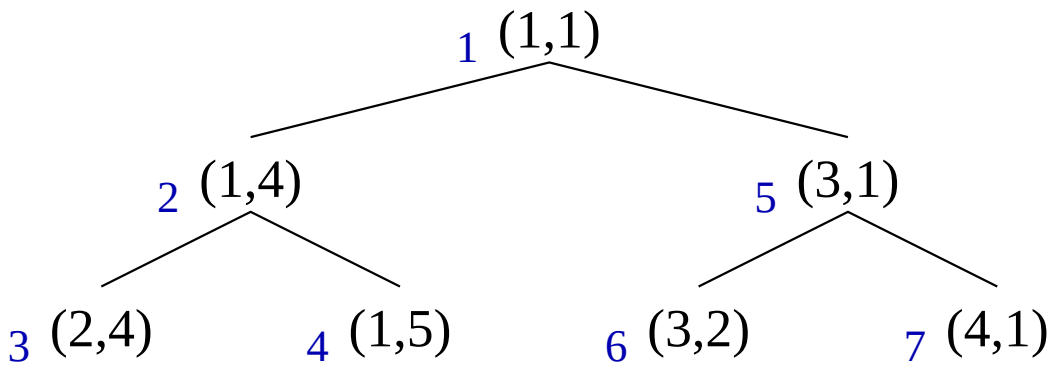
Exemple $d = 2$



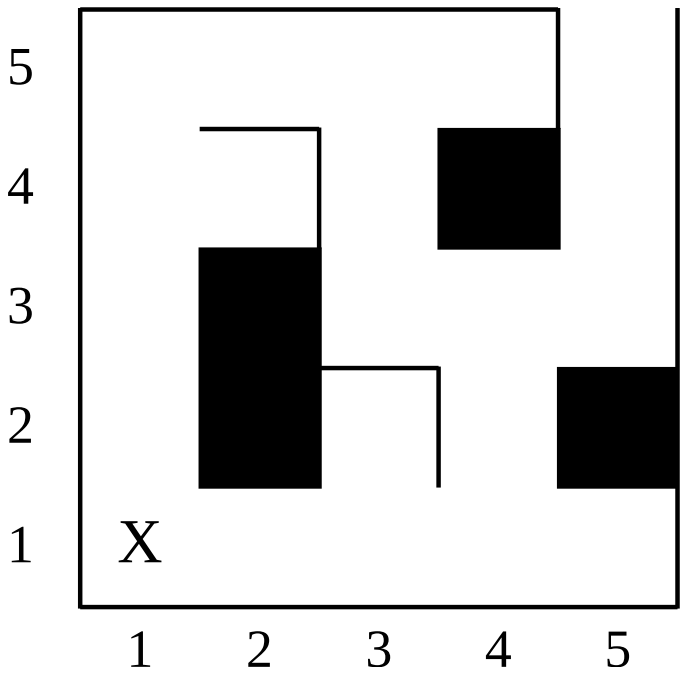


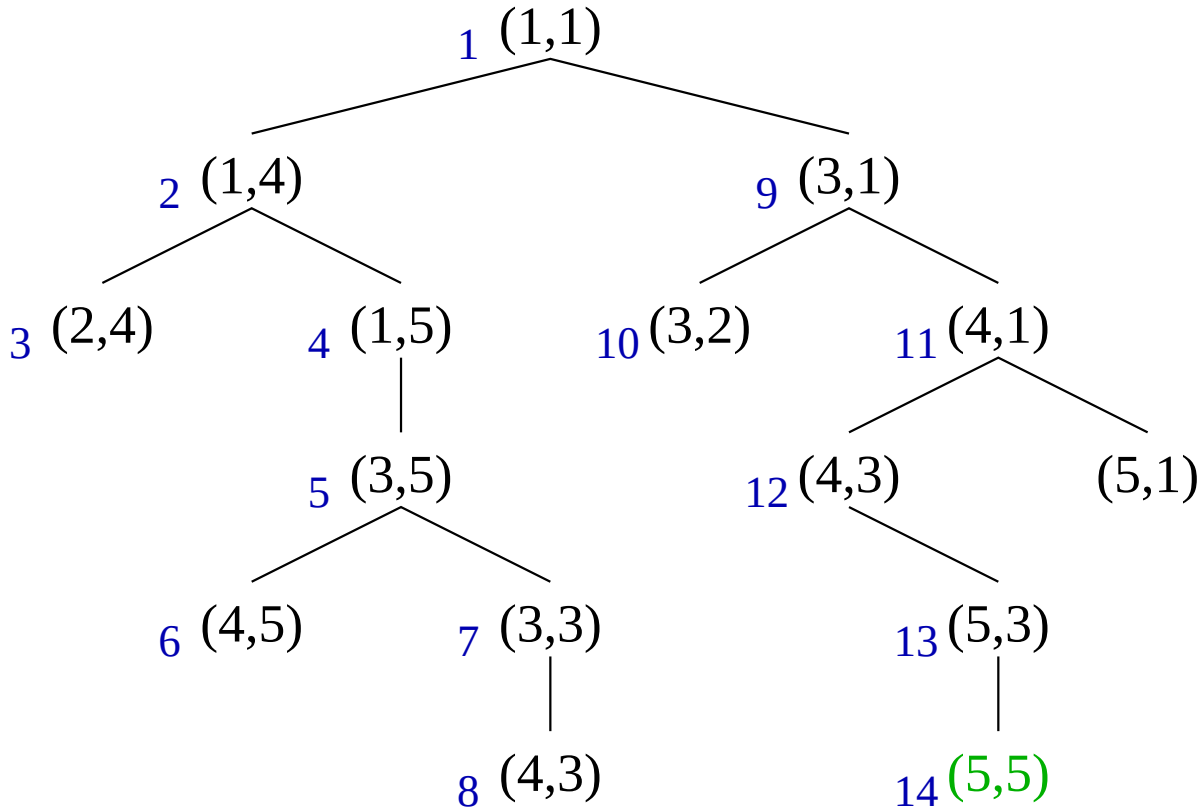
Exemple $d = 2$





Exemple $d = 5$


5
4
3
2
1
X
1
2
3
4
5


1 (1,1)
2 (1,4)
9 (3,1)
3 (2,4)
4 (1,5)
10 (3,2)
11 (4,1)
5 (3,5)
12 (4,3)
(5,1)
6 (4,5)
7 (3,3)
13 (5,3)
8 (4,3)
14 (5,5)