La résolution du problème des n-reines par les Algorithmes Exactes et A*

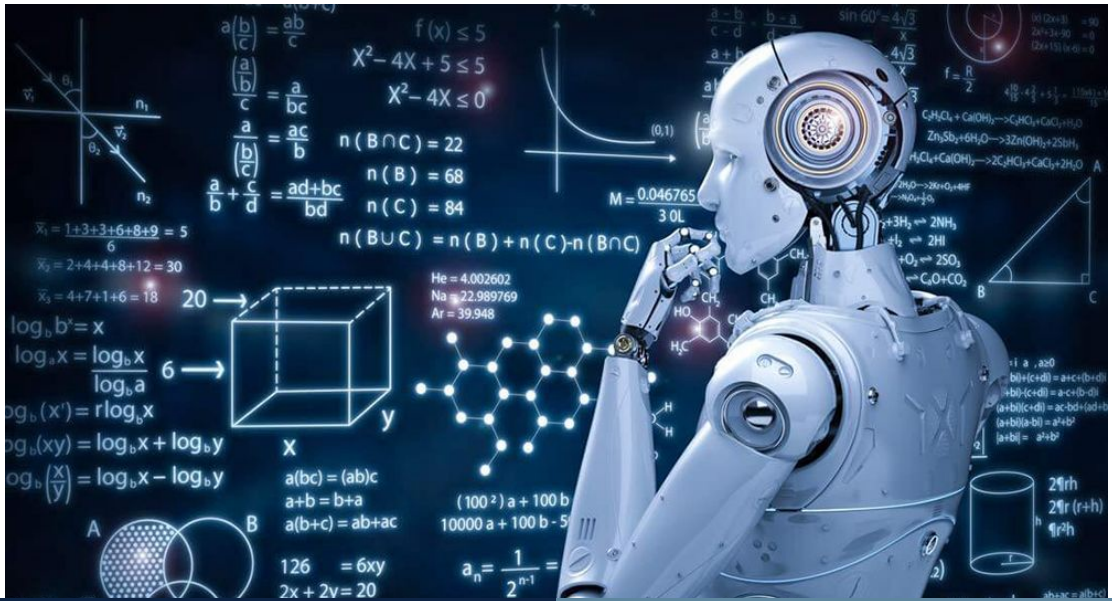
Dr. Hadjer Moulai

Faculty of Computer Science, LRIA laboratory.
USTHB, Algiers. Algeria

2022-2023

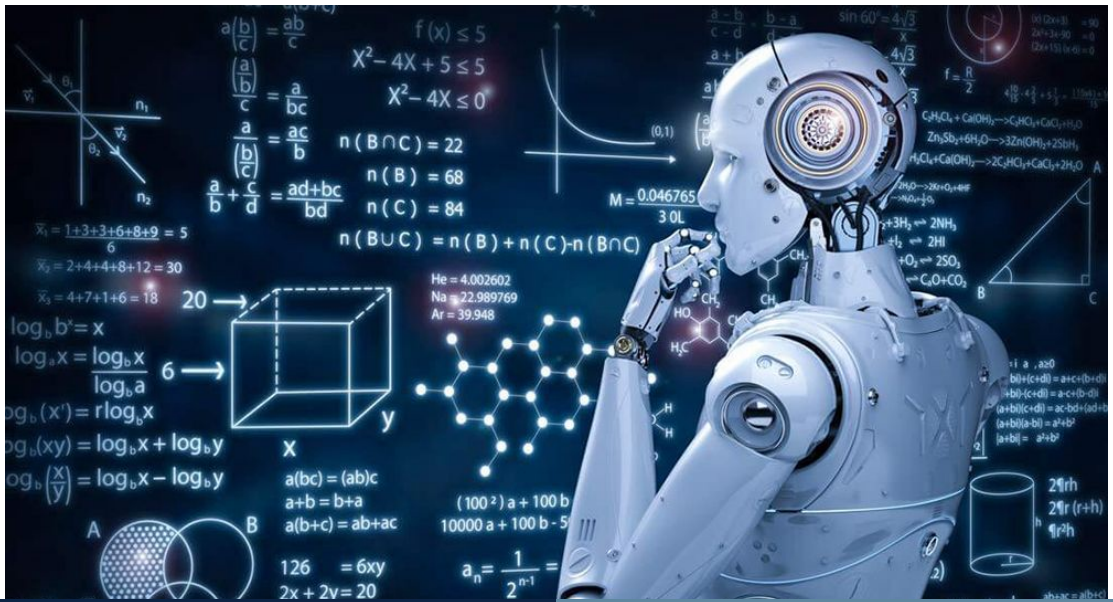
Résolution de problèmes

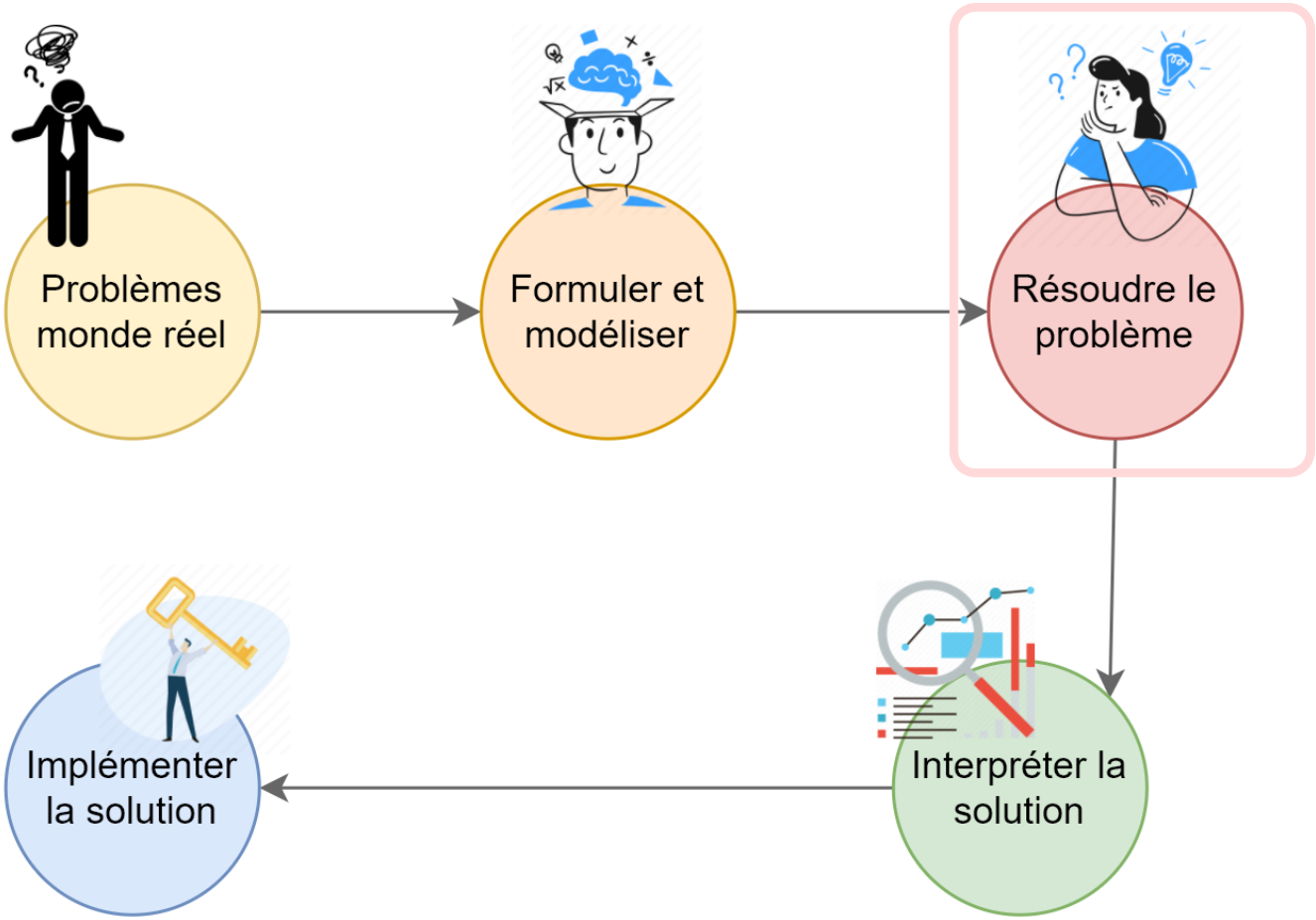
- C'est le processus consistant à sélectionner une séquence d'actions élémentaires afin d'atteindre des buts donnés en respectant les contraintes d'un environnement donné.
- Elle fait référence aux différentes techniques d'intelligence artificielle utilisées pour résoudre un problème donné.

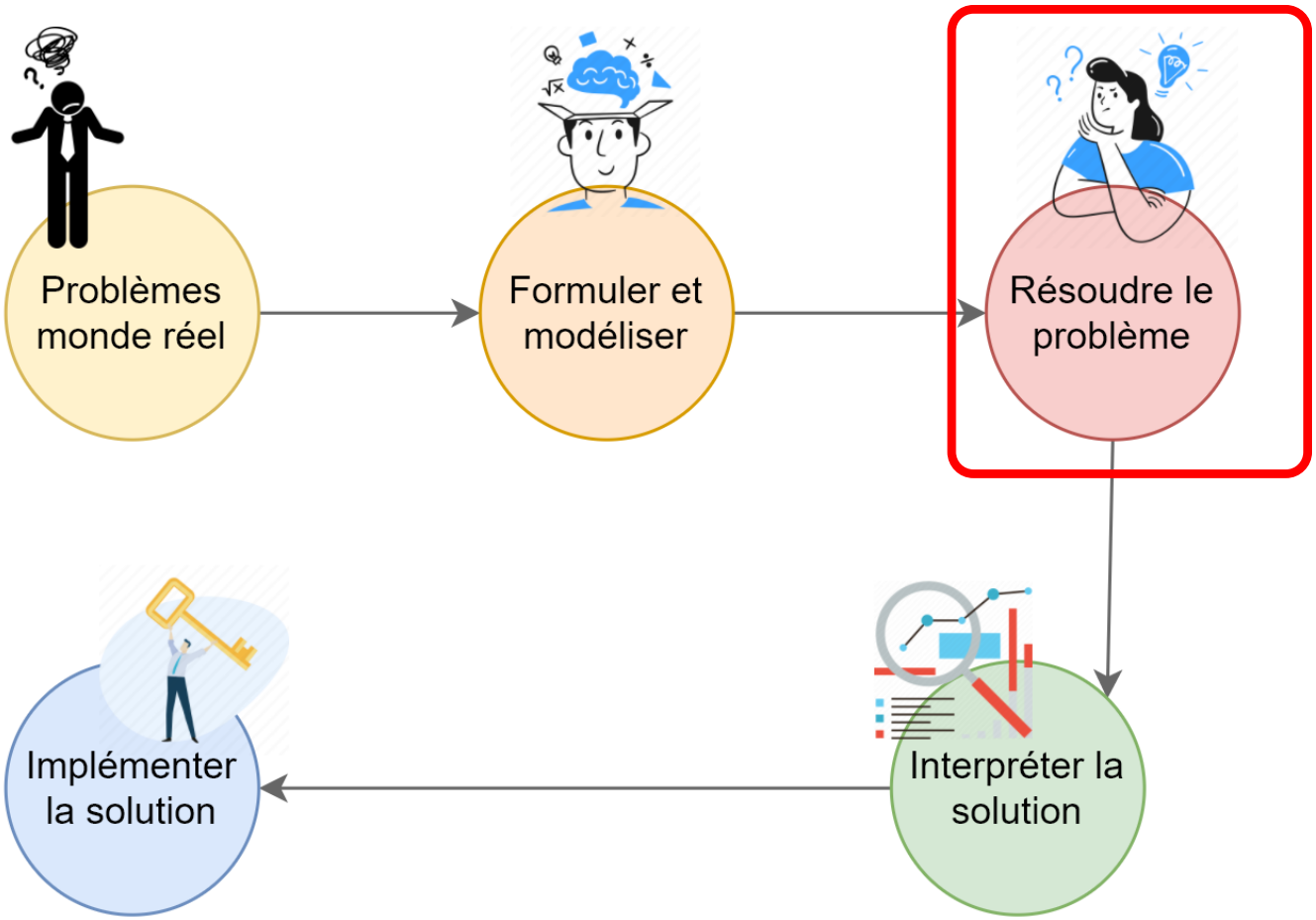


Résolution de problèmes

- C'est le processus consistant à sélectionner une séquence d'actions élémentaires afin d'atteindre des buts donnés en respectant les contraintes d'un environnement donné.
- Elle fait référence aux différentes techniques d'intelligence artificielle utilisées pour résoudre un problème donné.




Problèmes monde réel
Formuler et modéliser
Résoudre le problème
Interpréter la solution
Implémenter la solution


Problèmes monde réel
Formuler et modéliser
Résoudre le problème
Interpréter la solution
Implémenter la solution

Processus

- Un état initial (le problème à résoudre),
- Un ou plusieurs états finaux (la solution au problème),
- Un ensemble d'actions élémentaires permettant de passer d'un état à un autre (opérateur)

Objectif

Trouver une séquence d'actions à exécuter pour passer de l'état initial à l'état final.

Processus

- Un état initial (le problème à résoudre),
- Un ou plusieurs états finaux (la solution au problème),
- Un ensemble d'actions élémentaires permettant de passer d'un état à un autre (opérateur)

Objectif

Trouver une séquence d'actions à exécuter pour passer de l'état initial à l'état final.

Processus

- Un état initial (le problème à résoudre),
- Un ou plusieurs états finaux (la solution au problème),
- Un ensemble d'actions élémentaires permettant de passer d'un état à un autre (opérateur)

Objectif

Trouver une séquence d'actions à exécuter pour passer de l'état initial à l'état final.

Processus

- Un état initial (le problème à résoudre),
- Un ou plusieurs états finaux (la solution au problème),
- Un ensemble d'actions élémentaires permettant de passer d'un état à un autre (opérateur)

Objectif

Trouver une séquence d'actions à exécuter pour passer de l'état initial à l'état final.

Complexité du processus de résolution

- Mesure le temps (complexité temporel) et l'espace mémoire (complexité spatial) utilisé par un algorithme en fonction de la taille des données.
- La résolution de problème traite des problèmes dont l'espace des solutions est exponentielle. Chercher successivement la/les solution/s optimale/s dans cet espace est une approche fortement combinatoire entrainant une complexité exponentielle. Exemple : concevoir un emploi du temps

Apport de l'IA

L'IA apporte une solution à ce problème : les "heuristiques" : méthodes permettant de guider le processus de recherche en privilégiant les actions ayant le plus de chance d'aboutir.

Complexité du processus de résolution

- Mesure le temps (complexité temporel) et l'espace mémoire (complexité spatial) utilisé par un algorithme en fonction de la taille des données.
- La résolution de problème traite des problèmes dont l'espace des solutions est exponentielle. Chercher successivement la/les solution/s optimale/s dans cet espace est une approche fortement combinatoire entrainant une complexité exponentielle. Exemple : concevoir un emploi du temps

Apport de l'IA

L'IA apporte une solution à ce problème : les "heuristiques" : méthodes permettant de guider le processus de recherche en privilégiant les actions ayant le plus de chance d'aboutir.

Complexité du processus de résolution

- Mesure le temps (complexité temporel) et l'espace mémoire (complexité spatial) utilisé par un algorithme en fonction de la taille des données.
- La résolution de problème traite des problèmes dont l'espace des solutions est exponentielle. Chercher successivement la/les solution/s optimale/s dans cet espace est une approche fortement combinatoire entrainant une complexité exponentielle. Exemple : concevoir un emploi du temps

Apport de l'IA

L'IA apporte une solution à ce problème : les "heuristiques" : méthodes permettant de guider le processus de recherche en privilégiant les actions ayant le plus de chance d'aboutir.

Nature du problème

On distingue deux catégories :

- Problèmes jouets (Toy problems) : problèmes types, concis et facile à formuler, utilisés pour comparer les différentes stratégies de résolution. Exemple : le taquin, jeux d'échec, etc.
- Problèmes du monde réel : plus complexe et difficile à formuler. Exemple : Calcul de routes, voyageur de commerce, navigation de robots, etc.

Nature du problème

On distingue deux catégories :

- Problèmes jouets (Toy problems) : problèmes types, concis et facile à formuler, utilisés pour comparer les différentes stratégies de résolution. Exemple : le taquin, jeux d'échec, etc.
- Problèmes du monde réel : plus complexe et difficile à formuler. Exemple : Calcul de routes, voyageur de commerce, navigation de robots, etc.

Nature du problème

On distingue deux catégories :

- Problèmes jouets (Toy problems) : problèmes types, concis et facile à formuler, utilisés pour comparer les différentes stratégies de résolution. Exemple : le taquin, jeux d'échec, etc.
- Problèmes du monde réel : plus complexe et difficile à formuler. Exemple : Calcul de routes, voyageur de commerce, navigation de robots, etc.

Espace d'états

- On appelle espace d'états (ou espace de recherche) d'un problème l'ensemble des états possibles de ce dernier depuis l'état initial.
- Un espace de recherche peut être représenter par un graphe orienté (arbre)
  - Les sommets/nœuds sont les états
  - Les arcs sont les actions (opérateurs)

La résolution d'un problème revient donc à explorer le graphe dans la recherche du nœud but.

Remarque : Parfois la solution est très difficile à trouver, on est donc amener à trouver une solution approximative.

Espace d'états

- On appelle espace d'états (ou espace de recherche) d'un problème l'ensemble des états possibles de ce dernier depuis l'état initial.
- Un espace de recherche peut être représenter par un graphe orienté (arbre)
  - Les sommets/nœuds sont les états
  - Les arcs sont les actions (opérateurs)

La résolution d'un problème revient donc à explorer le graphe dans la recherche du nœud but.

Remarque : Parfois la solution est très difficile à trouver, on est donc amener à trouver une solution approximative.

Espace d'états

- On appelle espace d'états (ou espace de recherche) d'un problème l'ensemble des états possibles de ce dernier depuis l'état initial.
- Un espace de recherche peut être représenter par un graphe orienté (arbre)
  - Les sommets/nœuds sont les états
  - Les arcs sont les actions (opérateurs)

La résolution d'un problème revient donc à explorer le graphe dans la recherche du nœud but.

Remarque : Parfois la solution est très difficile à trouver, on est donc amener à trouver une solution approximative.

Espace d'états

- On appelle espace d'états (ou espace de recherche) d'un problème l'ensemble des états possibles de ce dernier depuis l'état initial.
- Un espace de recherche peut être représenter par un graphe orienté (arbre)
  - Les sommets/nœuds sont les états
  - Les arcs sont les actions (opérateurs)

La résolution d'un problème revient donc à explorer le graphe dans la recherche du nœud but.

Remarque : Parfois la solution est très difficile à trouver, on est donc amener à trouver une solution approximative.

Espace d'états

- On appelle espace d'états (ou espace de recherche) d'un problème l'ensemble des états possibles de ce dernier depuis l'état initial.
- Un espace de recherche peut être représenter par un graphe orienté (arbre)
  - Les sommets/nœuds sont les états
  - Les arcs sont les actions (opérateurs)

La résolution d'un problème revient donc à explorer le graphe dans la recherche du nœud but.

Remarque : Parfois la solution est très difficile à trouver, on est donc amener à trouver une solution approximative.

Espace d'états

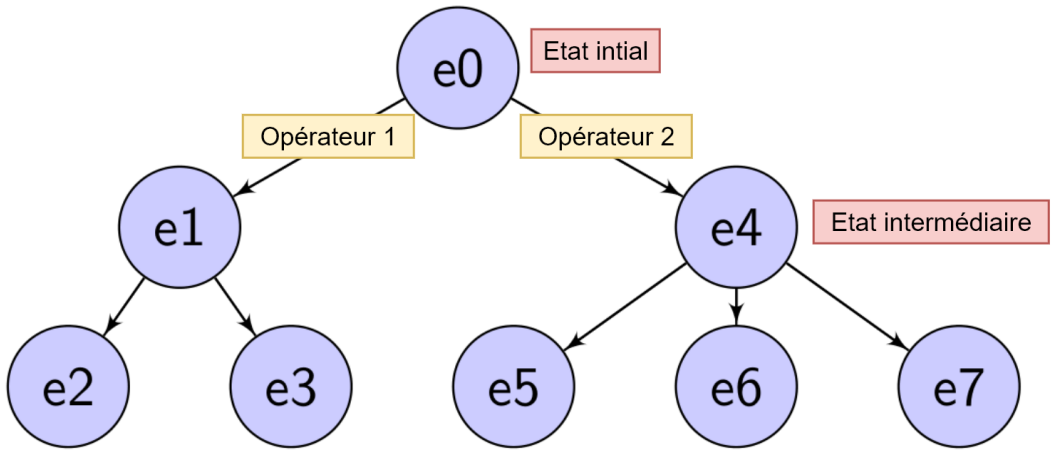
- On appelle espace d'états (ou espace de recherche) d'un problème l'ensemble des états possibles de ce dernier depuis l'état initial.
- Un espace de recherche peut être représenter par un graphe orienté (arbre)
  - Les sommets/nœuds sont les états
  - Les arcs sont les actions (opérateurs)

La résolution d'un problème revient donc à explorer le graphe dans la recherche du nœud but.

Remarque : Parfois la solution est très difficile à trouver, on est donc amener à trouver une solution approximative.

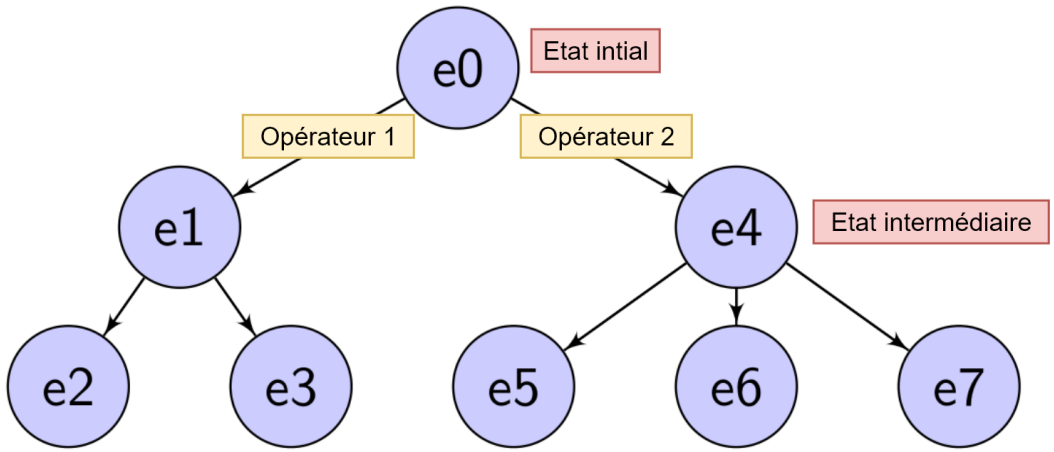
L'application des opérateurs sur les états en démarrant de l'état initial conduit à la construction d'une arborescence.

- La racine = l'état initial du problème.
- Les feuilles = des états sans successeur ou des nœuds qui n'ont pas encore été développés.
- Un chemin = une séquence de nœuds partant de la racine à une feuille.



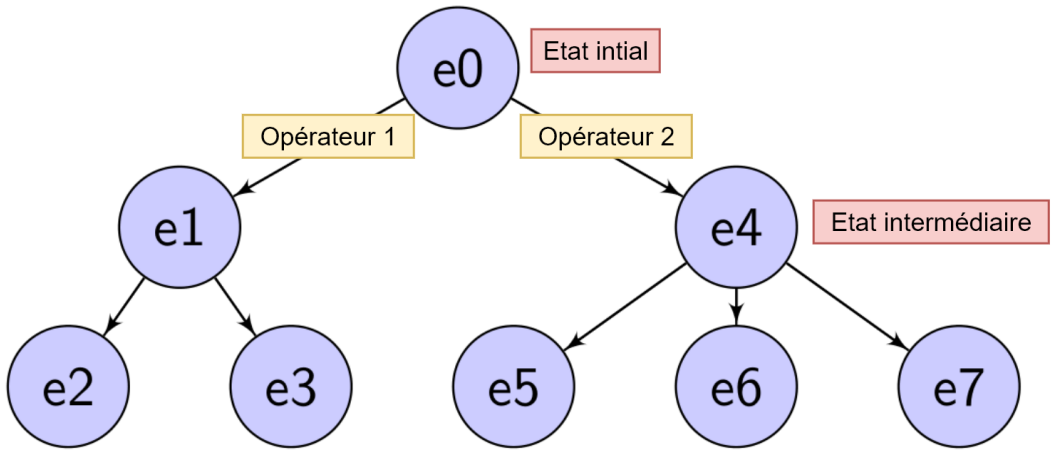
L'application des opérateurs sur les états en démarrant de l'état initial conduit à la construction d'une arborescence.

- La racine = l'état initial du problème.
- Les feuilles = des états sans successeur ou des nœuds qui n'ont pas encore été développés.
- Un chemin = une séquence de nœuds partant de la racine à une feuille.



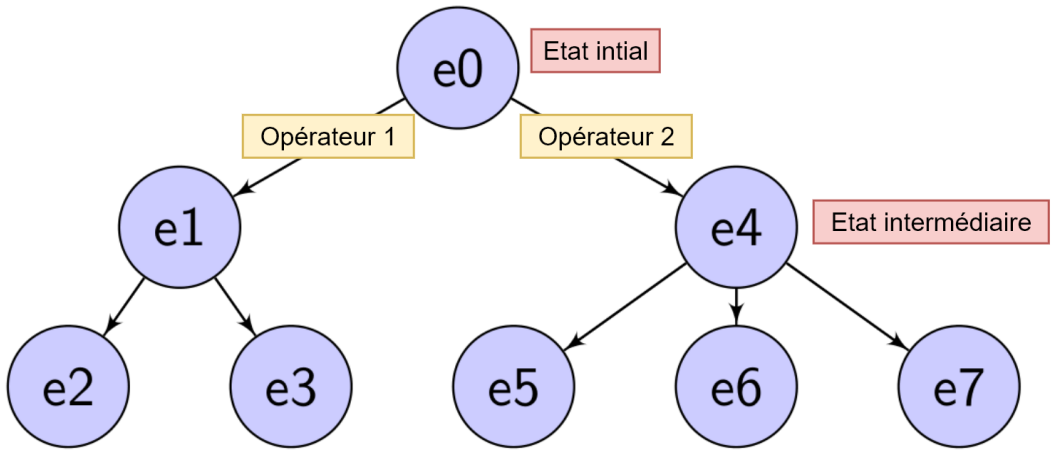
L'application des opérateurs sur les états en démarrant de l'état initial conduit à la construction d'une arborescence.

- La racine = l'état initial du problème.
- Les feuilles = des états sans successeur ou des nœuds qui n'ont pas encore été développés.
- Un chemin = une séquence de nœuds partant de la racine à une feuille.



L'application des opérateurs sur les états en démarrant de l'état initial conduit à la construction d'une arborescence.

- La racine = l'état initial du problème.
- Les feuilles = des états sans successeur ou des nœuds qui n'ont pas encore été développés.
- Un chemin = une séquence de nœuds partant de la racine à une feuille.



Méthode de résolution

Il n'existe de pas de méthode générale de résolution de problème, ça nécessite :

- de décrire formellement le problème (les états, les actions)
- une procédure d'évaluation de solution

Méthode de résolution

Il n'existe de pas de méthode générale de résolution de problème, ça nécessite :

- de décrire formellement le problème (les états, les actions)
- une procédure d'évaluation de solution

Méthode de résolution

Il n'existe de pas de méthode générale de résolution de problème, ça nécessite :

- de décrire formellement le problème (les états, les actions)
- une procédure d'évaluation de solution

Démarche générale pour la résolution d'un problème

1. Choix de l'approche de résolution :
   - Espace des états
   - Réduction de problèmes
   - Démonstration de théorèmes
2. Choix d'une représentation :
   - Pour un même problème, il peut exister plusieurs représentations : logique, arbre, graphe d'états.
3. Choix de la méthode de résolution : recherche aveugle, heuristique ou méta-heuristique.

Démarche générale pour la résolution d'un problème

1. Choix de l'approche de résolution :
   - Espace des états
   - Réduction de problèmes
   - Démonstration de théorèmes
2. Choix d'une représentation :
   - Pour un même problème, il peut exister plusieurs représentations : logique, arbre, graphe d'états.
3. Choix de la méthode de résolution : recherche aveugle, heuristique ou méta-heuristique.

Démarche générale pour la résolution d'un problème

1. Choix de l'approche de résolution :
   - Espace des états
   - Réduction de problèmes
   - Démonstration de théorèmes
2. Choix d'une représentation :
   - Pour un même problème, il peut exister plusieurs représentations : logique, arbre, graphe d'états.
3. Choix de la méthode de résolution : recherche aveugle, heuristique ou méta-heuristique.

Démarche générale pour la résolution d'un problème

1. Choix de l'approche de résolution :
   - Espace des états
   - Réduction de problèmes
   - Démonstration de théorèmes
2. Choix d'une représentation :
   - Pour un même problème, il peut exister plusieurs représentations : logique, arbre, graphe d'états.
3. Choix de la méthode de résolution : recherche aveugle, heuristique ou méta-heuristique.

Démarche générale pour la résolution d'un problème

1. Choix de l'approche de résolution :
   - Espace des états
   - Réduction de problèmes
   - Démonstration de théorèmes
2. Choix d'une représentation :
   - Pour un même problème, il peut exister plusieurs représentations : logique, arbre, graphe d'états.
3. Choix de la méthode de résolution : recherche aveugle, heuristique ou méta-heuristique.

Démarche générale pour la résolution d'un problème

1. Choix de l'approche de résolution :
   - Espace des états
   - Réduction de problèmes
   - Démonstration de théorèmes
2. Choix d'une représentation :
   - Pour un même problème, il peut exister plusieurs représentations : logique, arbre, graphe d'états.
3. Choix de la méthode de résolution : recherche aveugle, heuristique ou méta-heuristique.

Méthodes de résolution

De nombreux problèmes peuvent être formulés comme des problèmes de recherche.

1. Méthodes de recherche aveugles (algorithmes exactes), sans utilisation de connaissances sur le problème :
   - recherche en largeur
   - recherche en profondeur...
2. Méthodes de recherche informées (heuristiques)
   - Algorithme glouton (greedy)
   - Algorithme A*

Méthodes de résolution

De nombreux problèmes peuvent être formulés comme des problèmes de recherche.

1. Méthodes de recherche aveugles (algorithmes exactes), sans utilisation de connaissances sur le problème :
   - recherche en largeur
   - recherche en profondeur...
2. Méthodes de recherche informées (heuristiques)
   - Algorithme glouton (greedy)
   - Algorithme A*

Méthodes de résolution

De nombreux problèmes peuvent être formulés comme des problèmes de recherche.

1. Méthodes de recherche aveugles (algorithmes exactes), sans utilisation de connaissances sur le problème :
   - recherche en largeur
   - recherche en profondeur...
2. Méthodes de recherche informées (heuristiques)
   - Algorithme glouton (greedy)
   - Algorithme A*

Méthodes de résolution

De nombreux problèmes peuvent être formulés comme des problèmes de recherche.

1. Méthodes de recherche aveugles (algorithmes exactes), sans utilisation de connaissances sur le problème :
   - recherche en largeur
   - recherche en profondeur...
2. Méthodes de recherche informées (heuristiques)
   - Algorithme glouton (greedy)
   - Algorithme A*

Méthodes de résolution

De nombreux problèmes peuvent être formulés comme des problèmes de recherche.

1. Méthodes de recherche aveugles (algorithmes exactes), sans utilisation de connaissances sur le problème :
   - recherche en largeur
   - recherche en profondeur...
2. Méthodes de recherche informées (heuristiques)
   - Algorithme glouton (greedy)
   - Algorithme A*

Algorithmes Exactes :

Les algorithmes exactes sont des algorithmes de recherche exhaustive dont le principe est de parcourir l'espace de recherche dans sa totalité jusqu'à arriver à la solution. En dépit de leur efficacité indiscutable à trouver la solution optimale, leur majeur inconvénient est leur complexité.
En effet, utiliser une méthode exacte pour résoudre un problème fortement combinatoire peut rapidement épuiser toutes les ressources dont on dispose.
Cependant, on a la garantie d'obtenir la solution optimale.

Dans ce projet, deux méthodes exactes sont à implémenter :

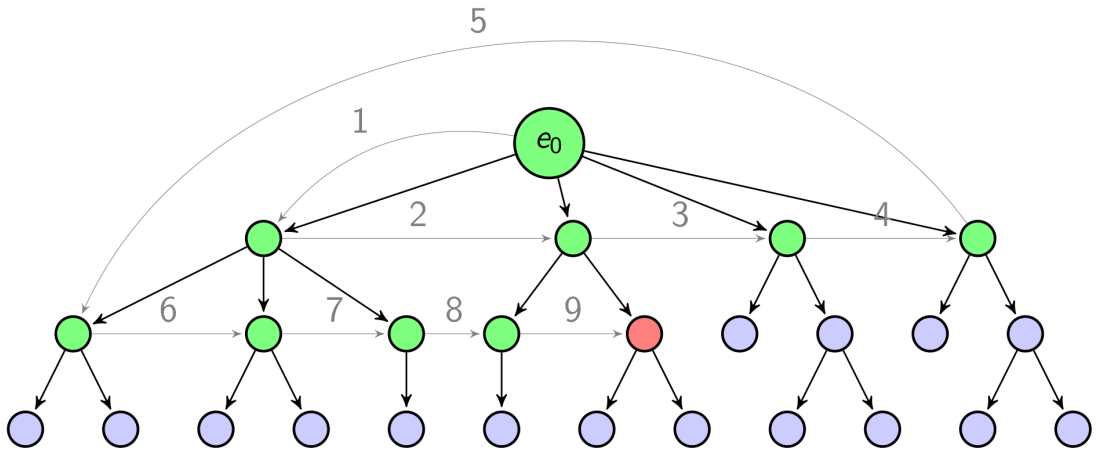
- **Largeur d'abord (Breadth First Search)**
- **Profondeur d'abord (Depth First Search)**

Dans ce projet, deux méthodes exactes sont à implémenter :

- **Largeur d'abord (Breadth First Search)**
- **Profondeur d'abord (Depth First Search)**

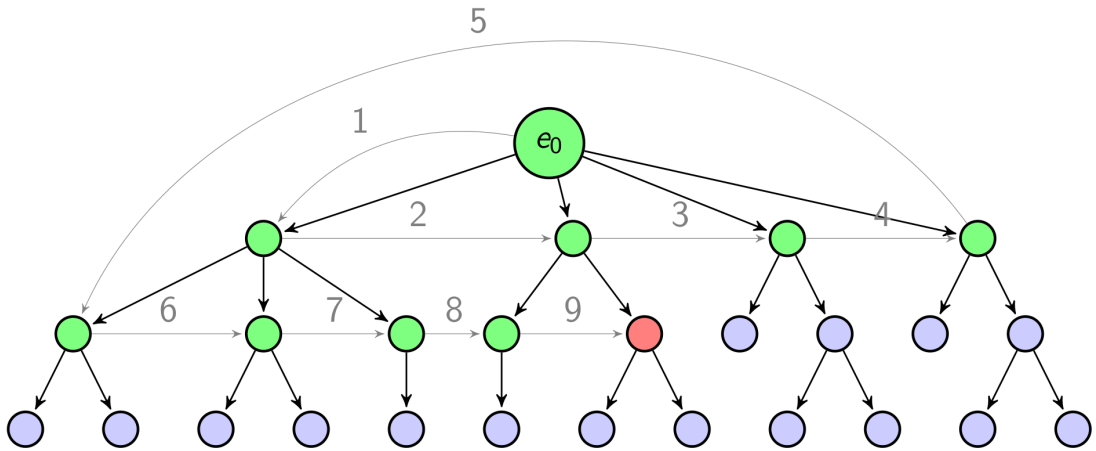
Largeur d'abord

- Stratégie :
  - L'expansion des nœuds se fait selon l'ordre dans lequel ils ont été engendrés.
  - le parcours de l'arbre s'effectue niveau par niveau et donc de manière horizontale.



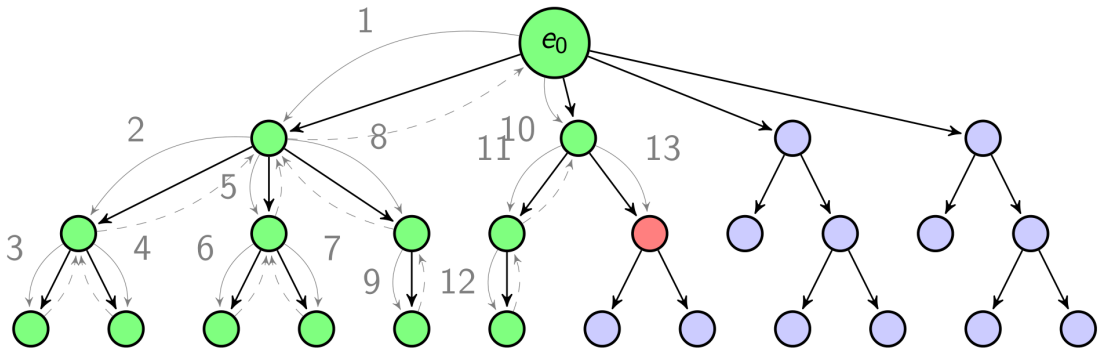
Largeur d'abord

- Stratégie :
  - L'expansion des nœuds se fait selon l'ordre dans lequel ils ont été engendrés.
  - le parcours de l'arbre s'effectue niveau par niveau et donc de manière horizontale.



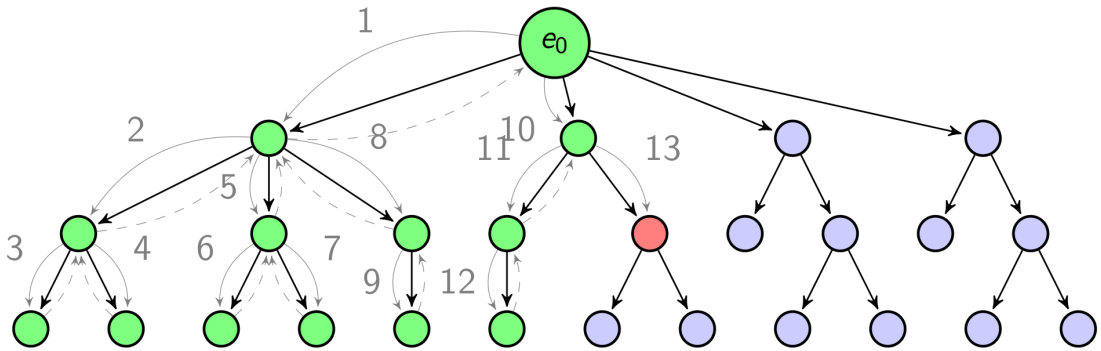
Profondeur d'abord

- Stratégie :
  - L'expansion considère les nœuds les plus récemment engendrés.
  - ainsi le parcours de l'arbre s'effectue en profondeur jusqu'au bout de l'arbre.



Profondeur d'abord

- Stratégie :
  - L'expansion considère les nœuds les plus récemment engendrés.
  - ainsi le parcours de l'arbre s'effectue en profondeur jusqu'au bout de l'arbre.



A* : recherche par heuristique

- n = noeud représentant un état, la fonction d'évaluation de n sera de la forme : $f(n) = g(n) + h(n)$
- $g(n)$ est le coût du chemin déjà parcouru (coût consenti)
- on suppose connue une fonction heuristique $h(n)$, estimant par défaut le coût optimal inconnu de n jusqu'à un nœud but (coût restant).

La difficulté de A* réside dans la conception de la fonction heuristique h qui doit être admissible. C'est-à-dire qu'elle ne surestime jamais le coût réel pour atteindre le but, A* est donc assuré de retourner le chemin de coût minimum.

A* : recherche par heuristique

- n = noeud représentant un état, la fonction d'évaluation de n sera de la forme : $f(n) = g(n) + h(n)$
- $g(n)$ est le coût du chemin déjà parcouru (coût consenti)
- on suppose connue une fonction heuristique $h(n)$, estimant par défaut le coût optimal inconnu de n jusqu'à un nœud but (coût restant).

La difficulté de A* réside dans la conception de la fonction heuristique h qui doit être admissible. C'est-à-dire qu'elle ne surestime jamais le coût réel pour atteindre le but, A* est donc assuré de retourner le chemin de coût minimum.

A* : recherche par heuristique

- n = noeud représentant un état, la fonction d'évaluation de n sera de la forme : $f(n) = g(n) + h(n)$
- $g(n)$ est le coût du chemin déjà parcouru (coût consenti)
- on suppose connue une fonction heuristique $h(n)$, estimant par défaut le coût optimal inconnu de n jusqu'à un nœud but (coût restant).

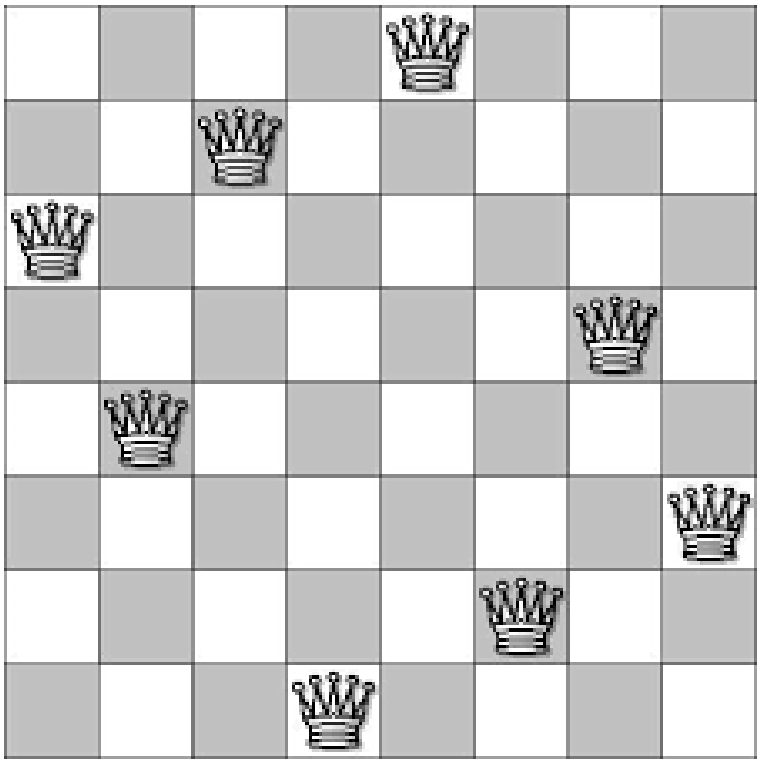
La difficulté de A* réside dans la conception de la fonction heuristique h qui doit être admissible. C'est-à-dire qu'elle ne surestime jamais le coût réel pour atteindre le but, A* est donc assuré de retourner le chemin de coût minimum.

A* : recherche par heuristique

- n = noeud représentant un état, la fonction d'évaluation de n sera de la forme : $f(n) = g(n) + h(n)$
- $g(n)$ est le coût du chemin déjà parcouru (coût consenti)
- on suppose connue une fonction heuristique $h(n)$, estimant par défaut le coût optimal inconnu de n jusqu'à un nœud but (coût restant).

La difficulté de A* réside dans la conception de la fonction heuristique h qui doit être admissible. C'est-à-dire qu'elle ne surestime jamais le coût réel pour atteindre le but, A* est donc assuré de retourner le chemin de coût minimum.

Le problème initialement posé par K.F. Gauss en 1842 est le suivant : est-il possible de placer 8 reines sur un échiquier sans qu'aucune reine n'en menace une autre ? Une reine menace toutes les pièces de l'échiquier qui sont situées sur la même : ligne, colonne, diagonale.



Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Formulation :

Comment atteindre un état final ?

- **Etats :** Une configuration de 0 à 8 reines sur l'échiquier.
- **Etat initial :** Aucune reine sur l'échiquier.
- **Actions :** Ajouter une reine sur une case vide.
- **Fonction de successeur :** Cette fonction spécifie les états résultants des différentes actions.
- **Test de but :** Les 8 reines sont placés sur l'échiquier sans attaque.
- **Coût des actions** : Chaque déplacement d'une reine a un coût de 1 (pas d'intérêt particulier).

Résolution dans l'espace des états

Chaque nœud dans l'arbre de recherche est caractérisé par :

- **État :** Une configuration du jeu.
- **Noeud parent :** Le noeud dans l'arbre de recherche qui a généré ce noeud.
- **Noeuds enfants :** Les noeuds obtenu en appliquant des opérateurs.
- **Action :** L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
- **Coût du chemin :** Le coût du chemin à partir de l'état initial jusqu'à ce noeud : g(n).
- **Profondeur :** La profondeur du noeud dans l'arbre de recherche.

Résolution dans l'espace des états

Chaque nœud dans l'arbre de recherche est caractérisé par :

- **État :** Une configuration du jeu.
- **Noeud parent :** Le noeud dans l'arbre de recherche qui a généré ce noeud.
- **Noeuds enfants :** Les noeuds obtenu en appliquant des opérateurs.
- **Action :** L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
- **Coût du chemin :** Le coût du chemin à partir de l'état initial jusqu'à ce noeud : g(n).
- **Profondeur :** La profondeur du noeud dans l'arbre de recherche.

Résolution dans l'espace des états

Chaque nœud dans l'arbre de recherche est caractérisé par :

- **État :** Une configuration du jeu.
- **Noeud parent :** Le noeud dans l'arbre de recherche qui a généré ce noeud.
- **Noeuds enfants :** Les noeuds obtenu en appliquant des opérateurs.
- **Action :** L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
- **Coût du chemin :** Le coût du chemin à partir de l'état initial jusqu'à ce noeud : g(n).
- **Profondeur :** La profondeur du noeud dans l'arbre de recherche.

Résolution dans l'espace des états

Chaque nœud dans l'arbre de recherche est caractérisé par :

- **État :** Une configuration du jeu.
- **Noeud parent :** Le noeud dans l'arbre de recherche qui a généré ce noeud.
- **Noeuds enfants :** Les noeuds obtenu en appliquant des opérateurs.
- **Action :** L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
- **Coût du chemin :** Le coût du chemin à partir de l'état initial jusqu'à ce noeud : g(n).
- **Profondeur :** La profondeur du noeud dans l'arbre de recherche.

Résolution dans l'espace des états

Chaque nœud dans l'arbre de recherche est caractérisé par :

- **État :** Une configuration du jeu.
- **Noeud parent :** Le noeud dans l'arbre de recherche qui a généré ce noeud.
- **Noeuds enfants :** Les noeuds obtenu en appliquant des opérateurs.
- **Action :** L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
- **Coût du chemin :** Le coût du chemin à partir de l'état initial jusqu'à ce noeud : g(n).
- **Profondeur :** La profondeur du noeud dans l'arbre de recherche.

Résolution dans l'espace des états

Chaque nœud dans l'arbre de recherche est caractérisé par :

- **État :** Une configuration du jeu.
- **Noeud parent :** Le noeud dans l'arbre de recherche qui a généré ce noeud.
- **Noeuds enfants :** Les noeuds obtenu en appliquant des opérateurs.
- **Action :** L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
- **Coût du chemin :** Le coût du chemin à partir de l'état initial jusqu'à ce noeud : g(n).
- **Profondeur :** La profondeur du noeud dans l'arbre de recherche.

Noeud vs Etat

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- Conclusion :
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états.
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents).

Noeud vs Etat

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- Conclusion :
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états.
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents).

Noeud vs Etat

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- Conclusion :
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états.
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents).

Noeud vs Etat

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- Conclusion :
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états.
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents).

Noeud vs Etat

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- Conclusion :
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états.
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents).

Noeud vs Etat

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- Conclusion :
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états.
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents).

Principe d'exploration

- Commencer par le nœud racine
- Développer le nœud en lui appliquant la fonction successeur
- Les nœuds successeurs non encore développés sont mis dans une la liste ouverte
- La stratégie d'exploration est la fonction qui détermine le nœud suivant à développer dans cet ensemble (largeur, profondeur, etc.)
- Une fonction d'exploration peut être utilisée afin d'examiner chacun des éléments de la liste ouverte et choisir le meilleur (A*).

Principe d'exploration

- Commencer par le nœud racine
- Développer le nœud en lui appliquant la fonction successeur
- Les nœuds successeurs non encore développés sont mis dans une la liste ouverte
- La stratégie d'exploration est la fonction qui détermine le nœud suivant à développer dans cet ensemble (largeur, profondeur, etc.)
- Une fonction d'exploration peut être utilisée afin d'examiner chacun des éléments de la liste ouverte et choisir le meilleur (A*).

Principe d'exploration

- Commencer par le nœud racine
- Développer le nœud en lui appliquant la fonction successeur
- Les nœuds successeurs non encore développés sont mis dans une la liste ouverte
- La stratégie d'exploration est la fonction qui détermine le nœud suivant à développer dans cet ensemble (largeur, profondeur, etc.)
- Une fonction d'exploration peut être utilisée afin d'examiner chacun des éléments de la liste ouverte et choisir le meilleur (A*).

Principe d'exploration

- Commencer par le nœud racine
- Développer le nœud en lui appliquant la fonction successeur
- Les nœuds successeurs non encore développés sont mis dans une la liste ouverte
- La stratégie d'exploration est la fonction qui détermine le nœud suivant à développer dans cet ensemble (largeur, profondeur, etc.)
- Une fonction d'exploration peut être utilisée afin d'examiner chacun des éléments de la liste ouverte et choisir le meilleur (A*).

Principe d'exploration

- Commencer par le nœud racine
- Développer le nœud en lui appliquant la fonction successeur
- Les nœuds successeurs non encore développés sont mis dans une la liste ouverte
- La stratégie d'exploration est la fonction qui détermine le nœud suivant à développer dans cet ensemble (largeur, profondeur, etc.)
- Une fonction d'exploration peut être utilisée afin d'examiner chacun des éléments de la liste ouverte et choisir le meilleur (A*).

Comment représenter une solution au problème ?

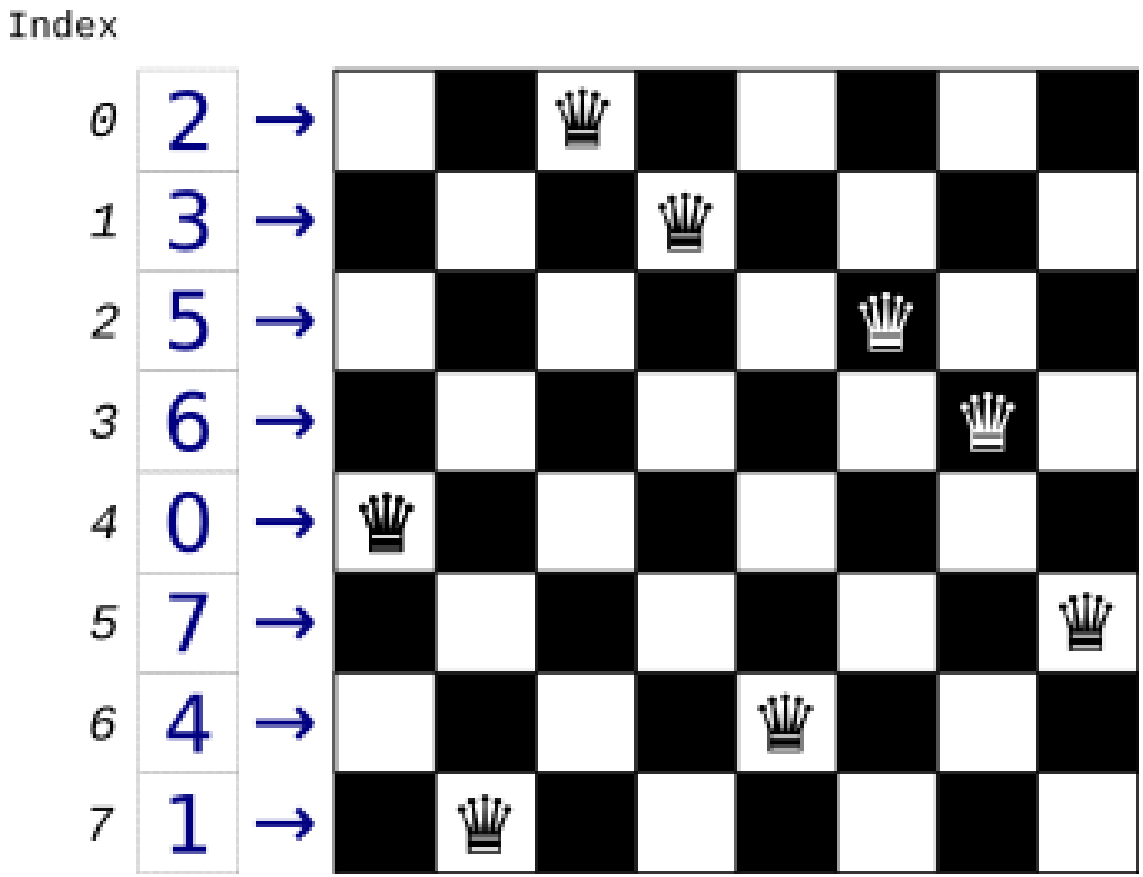


FIGURE: Modélisation d'une solution

Implémentation des listes ouverte et fermée

- Vérifier l'existence d'un état dans la lister ouverte pour ne pas refaire le même chemin déjà parcouru. Ceci implique de vérifier l'existence d'un élément dans une structure de donnée pouvant inclure des milliers d'éléments. Comment éviter cela ?

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.

Les classes principales :

- Classe Node : représente un nœud décrit par ses différents attributs et fonctions.
- Classe Main : classe principale où l'exécution aura lieu.
- Classe DFS
- Classe BFS
- Classe A*

Les structures de données nécessaires :

- Les listes ouverte et fermée auxquelles chaque il faut accéder à chaque itération.