Intelligence Artificielle

Résolution de Problèmes

Emmanuel ADAM

LAMIH UMR CNRS 8530
Université de Valenciennes et du Hainaut-Cambrésis
FRANCE

26 mars 2008

1. Introduction
2. Problèmes
   - Définition d'un problème
   - Types de problèmes
3. Toys Problems
4. Résolution
   - le Problème de la résolution
   - Méthodes de résolution constructive
   - Graphe d'états
   - Evaluation de la recherche
   - Type de recherches
5. Méthodes de recherche aveugles
   - Recherche en largeur
   - Recherche en coût uniforme
   - Recherche en profondeur
   - Recherche en profondeur limitée
   - Recherche par approfondissement itératif
   - Recherche bi-directionnelle
   - Exemple d'algorithmes
6. Méthodes de recherche heuristiques
   - notions d'heuristiques
   - Algorithme A*

- Particularités et propriétés de la recherche par heuristique
- Exemple de résolution par A*

7 Recherche Aléatoire

8 Recherche Exhaustive

Introduction

Résolution de Problèmes

Introduction

Résolution de Problèmes

- Définir les problèmes

Introduction

Résolution de Problèmes

- Définir les problèmes
  - Nature

Introduction

Résolution de Problèmes

- Définir les problèmes
  - Nature
- Représentation des problèmes

Introduction

Résolution de Problèmes

- Définir les problèmes
  - Nature
- Représentation des problèmes
  - Logique

Introduction

Résolution de Problèmes

- Définir les problèmes
  - Nature
- Représentation des problèmes
  - Logique
  - Arbres et Graphes d'états

Introduction

Résolution de Problèmes

- Définir les problèmes
  - Nature
- Représentation des problèmes
  - Logique
  - Arbres et Graphes d'états
- Algorithmes de résolution

Définition d'un problème I

Exemple de problème

Se rendre de son domicile (Ville A) à un lieu précis (Arrivée) d'une autre ville (Ville C), le plus rapidement possible, avant une date donnée (obligation de passer par une ville étape B).
(exemple : de Onnaing au Théâtre L'antre 2 de Lille 2 avant 20h)

Définitions

Espace d'états : description (abstraction) du monde réel définissant le problème

Exemple : réseau des bus, trams, metro . . .

Buts : sous ensemble de l'espace d'états

Exemple : VilleC-Arrivee19h50

Action (opérateurs) : déplacement dans l'espace d'états

Définition d'un problème II

Définitions

Solution du problème : la séquence d'action menant de l'état initial à l'état objectif

Algorithme de Recherche : procédure qui calcule une (ou plusieurs) solution à partir d'un problème (état initial, actions, états objectifs).

Types de problèmes

Les types de problèmes

Types de problèmes

Les types de problèmes

à état unique : déterministe, l'effet de chaque action est connu.

Types de problèmes

Les types de problèmes

à état unique : déterministe, l'effet de chaque action est connu.

à états multiples : déterministe, mais le monde n'est pas complètement accessible (données manquantes initialement)

Types de problèmes

Les types de problèmes

à état unique : déterministe, l'effet de chaque action est connu.

à états multiples : déterministe, mais le monde n'est pas complètement accessible (données manquantes initialement)

de contingence : non déterministe, et monde non complètement accessible initialement (besoin de "capter" les états)

Types de problèmes

Les types de problèmes

à état unique : déterministe, l'effet de chaque action est connu.

à états multiples : déterministe, mais le monde n'est pas complètement accessible (données manquantes initialement)

de contingence : non déterministe, et monde non complètement accessible initialement (besoin de "capter" les états)

d'exploration : espaces d'états inconnu, à découvrir par exploration

Espace d'états de l'exemple

Espace d'états de l'exemple

Action	*Etat*	*Action*	*Etat*
A00 :VilleA-bus18h00	E0.0 :VilleB18h20	A10 :train18h00	E1.0 :VilleC18h40
A01 :VilleA-bus18h10	E0.1 :VilleB18h30	A11 :train18h30	E1.1 :VilleC19h10
A02 :VilleA-bus18h20	E0.2 :VilleB18h40	A12 :train19h00	E1.2 :VilleC19h40
A03 :VilleA-bus18h30	E0.3 :VilleB18h50	A13 :train19h30	E1.3 :VilleC20h10
A04 :VilleA-bus18h40	E0.4 :VilleB19h00	A14 :train20h00	E1.4 :VilleC20h40
A05 :VilleA-bus18h50	E0.5 :VilleB19h10	A15 :train20h30	E1.5 :VilleC21h10

Action	*Etat*
A20 :metro18h50	E2.0 :VilleC-Arrivee19h10
A21 :metro19h05	E2.1 :VilleC-Arrivee19h25
A22 :metro19h20	E2.2 :VilleC-Arrivee19h40
A23 :metro19h35	E2.3 :VilleC-Arrivee19h55
A24 :metro19h50	E2.4 :VilleC-Arrivee20h10
A25 :metro20h05	E2.5 :VilleC-Arrivee20h25

Etat initial E0 = VilleA ; Etat Final EF = E2.3 :VilleC-Arrivee19h55

Problème à état unique

Problème à état unique

Problème à état unique

Problème à état unique

- Le monde est complètement accessible

Problème à état unique

Problème à état unique

- Le monde est complètement accessible
- L'effet de chaque action est connu

Problème à état unique

Problème à état unique

- Le monde est complètement accessible
- L'effet de chaque action est connu

Retour sur l'exemple

Toutes les tables (tram, metro, train, bus) sont connues
Exemple, de l'état E0 Initial, l'action A00 (prendre bus à 18.00) mène à l'état E0.0 (Ville B à 18.20)

Problème à états multiples

Problème à états multiples

Problème à états multiples

Problème à états multiples

- Le monde est n'est pas complètement accessible

Problème à états multiples

Problème à états multiples

- Le monde est n'est pas complètement accessible
- L'effet de chaque action est connu

Problème à états multiples

Problème à états multiples

- Le monde est n'est pas complètement accessible
- L'effet de chaque action est connu

Retour sur l'exemple

Il manque certaines infos sur les tables (tram, metro, train, bus)
Exemple, de l'état E0 Initial, l'action A00 (prendre bus à 18.00) mène à l'état E0.0 VilleB, mais sans information sur l'heure d'arrivée

Problème à états multiples

Problème à états multiples

- Le monde est n'est pas complètement accessible
- L'effet de chaque action est connu

Retour sur l'exemple

Il manque certaines infos sur les tables (tram, metro, train, bus)
Exemple, de l'état E0 Initial, l'action A00 (prendre bus à 18.00) mène à l'état E0.0 VilleB, mais sans information sur l'heure d'arrivée

- On avisera sur place parmis les plusieurs solutions possibles (en espérant qu'il y en ait) pour joindre la ville C

Problème contingent

Problème contingent

Problème contingent

Problème contingent

- Le monde est n'est pas complètement accessible

Problème contingent

Problème contingent

- Le monde est n'est pas complètement accessible
- L'effet des actions est inconnu

Problème contingent

Problème contingent

- Le monde est n'est pas complètement accessible
- L'effet des actions est inconnu
  - Nécessite la faculté de percevoir l'état en cours d'exécution

Problème contingent

Problème contingent

- Le monde est n'est pas complètement accessible
- L'effet des actions est inconnu
  - Nécessite la faculté de percevoir l'état en cours d'exécution

Retour sur l'exemple

Il manque beaucoup d'infos sur les tables (tram, metro, train, bus)
Exemple, de l'état E0 Initial, l'action A00 (prendre bus à 18.00) mène quelque part, on ne sait quand !

Problème contingent

Problème contingent

- Le monde est n'est pas complètement accessible
- L'effet des actions est inconnu
  - Nécessite la faculté de percevoir l'état en cours d'exécution

Retour sur l'exemple

Il manque beaucoup d'infos sur les tables (tram, metro, train, bus)
Exemple, de l'état E0 Initial, l'action A00 (prendre bus à 18.00) mène quelque part, on ne sait quand !

- On prend un bus au hasard ; s'il rapproche de la ville B dans un temps satisfaisant, on y reste, s'il prend la direction opposée, on revient au départ et on prend un autre bus . . .

Problème contingent

Problème contingent

- Le monde est n'est pas complètement accessible
- L'effet des actions est inconnu
  - Nécessite la faculté de percevoir l'état en cours d'exécution

Retour sur l'exemple

Il manque beaucoup d'infos sur les tables (tram, metro, train, bus)
Exemple, de l'état E0 Initial, l'action A00 (prendre bus à 18.00) mène quelque part, on ne sait quand !

- On prend un bus au hasard ; s'il rapproche de la ville B dans un temps satisfaisant, on y reste, s'il prend la direction opposée, on revient au départ et on prend un autre bus . . .
  - Cas concret ? Se rendre au diner de gala d'une conférence dans une ville d'un pays dont on ne connait la langue.

Problème d'exploration

Problème d'exploration

Problème d'exploration

Problème d'exploration

- Le monde est inconnu

Problème d'exploration

Problème d'exploration

- Le monde est inconnu
- L'effet des actions est inconnu

Problème d'exploration

Problème d'exploration

- Le monde est inconnu
- L'effet des actions est inconnu
  - Nécessite la faculté d'explorer le monde et de percevoir l'état en cours d'exécution

Problème d'exploration

Problème d'exploration

- Le monde est inconnu
- L'effet des actions est inconnu
  - Nécessite la faculté d'explorer le monde et de percevoir l'état en cours d'exécution

Retour sur l'exemple

Aucune info disponible sur les tables (tram, metro, train, bus)
Exemple, de l'état E0 Initial, marcher au hasard et prendre le premier moyen de locomotion disponible !

Types de problèmes

Retour sur les types de problèmes

Types de problèmes

Retour sur les types de problèmes

à état unique : recherche évidente

Types de problèmes

Retour sur les types de problèmes

à état unique : recherche évidente

à états multiples : recherche facile

Types de problèmes

Retour sur les types de problèmes

à état unique : recherche évidente

à états multiples : recherche facile

de contingence : *le type le plus courant* ; recherche de solution moins évidente, besoin de percevoir l'état en cours ; possibilité de ramener en partie aux types précédents

Types de problèmes

Retour sur les types de problèmes

à état unique : recherche évidente

à états multiples : recherche facile

de contingence : *le type le plus courant* ; recherche de solution moins évidente, besoin de percevoir l'état en cours ; possibilité de ramener en partie aux types précédents

d'exploration : alternance de phases de génération d'états / recherche

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus
- l'aspirateur

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus
- l'aspirateur
- le voyageur de commerce

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus
- l'aspirateur
- le voyageur de commerce
- le labyrinthe

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus
- l'aspirateur
- le voyageur de commerce
- le labyrinthe
- les mots croisés

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus
- l'aspirateur
- le voyageur de commerce
- le labyrinthe
- les mots croisés
- l'arithmétique cryptée

Toys Problems

Exemple de Toys Problems

Les Toys Problems sont des problèmes types utilisés pour tester des algorithmes. Parmi ceux-ci :

- le taquin
- les tours de Hanoi
- le chien, la chèvre et le chou
- le Wumpus
- l'aspirateur
- le voyageur de commerce
- le labyrinthe
- les mots croisés
- l'arithmétique cryptée
- les jeux d'échecs, de dames, ...

Une méthode de résolution

Une méthode de résolution ?

Une méthode de résolution

Une méthode de résolution ?

- Il n'existe de pas de méthode générale de résolution de problème

Une méthode de résolution

Une méthode de résolution ?

- Il n'existe de pas de méthode générale de résolution de problème
- Nécessité

Une méthode de résolution

Une méthode de résolution ?

- Il n'existe de pas de méthode générale de résolution de problème
- Nécessité
  - de décrire formellement le problème

Une méthode de résolution

Une méthode de résolution ?

- Il n'existe de pas de méthode générale de résolution de problème
- Nécessité
  - de décrire formellement le problème
  - de procédure d'évalution de solution

Une méthode de résolution

Une méthode de résolution ?

- Il n'existe de pas de méthode générale de résolution de problème
- Nécessité
  - de décrire formellement le problème
  - de procédure d'évalution de solution
- Résolution constructive = Proposer + Evaluer une solution

Une méthode de résolution

Une méthode de résolution ?

- Il n'existe de pas de méthode générale de résolution de problème
- Nécessité
  - de décrire formellement le problème
  - de procédure d'évalution de solution
- Résolution constructive = Proposer + Evaluer une solution
- Résolution par logique classique (principe de résolution, Herbrand, ...cf. http ://emmanuel.adam.free.fr/site/spip.php ?article35)

Deux méthodes de résolution constructive

Deux méthodes de résolution constructive

Deux méthodes de résolution constructive

Deux méthodes de résolution constructive

- Avancer par étapes, par solutions partielles

Deux méthodes de résolution constructive

Deux méthodes de résolution constructive

- Avancer par étapes, par solutions partielles
  - Exemple : Pb des 8 reines

Deux méthodes de résolution constructive

Deux méthodes de résolution constructive

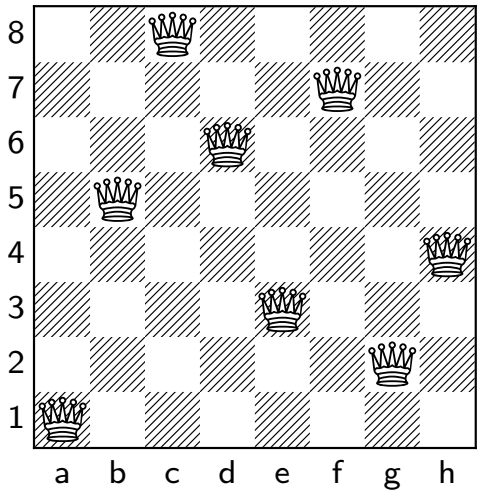
- Avancer par étapes, par solutions partielles
  - Exemple : Pb des 8 reines
- Décomposer en sous-problèmes

Deux méthodes de résolution constructive

Deux méthodes de résolution constructive

- Avancer par étapes, par solutions partielles
  - Exemple : Pb des 8 reines
- Décomposer en sous-problèmes
  - Exemple : Tours de Hanoï

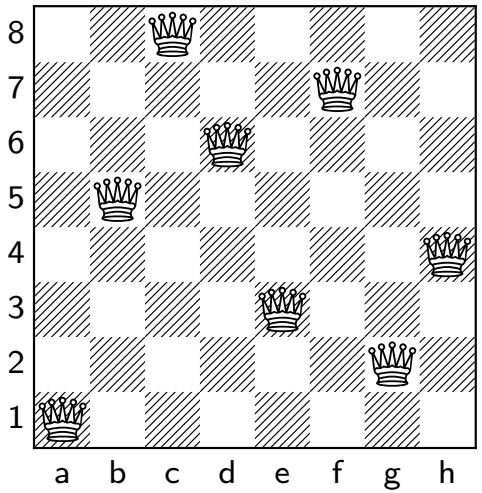
Le problème des 8 reines



Le problème des 8 reines

- But : placer 8 reines sur un échiquier de 8×8, sans attaque possible

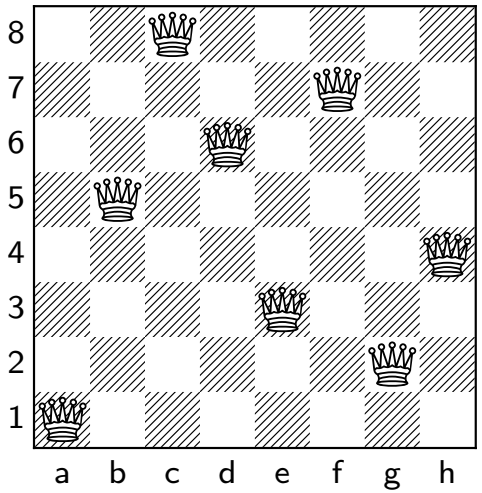
Le problème des 8 reines



Le problème des 8 reines

- But : placer 8 reines sur un échiquier de 8×8, sans attaque possible
- Problème à états multiples

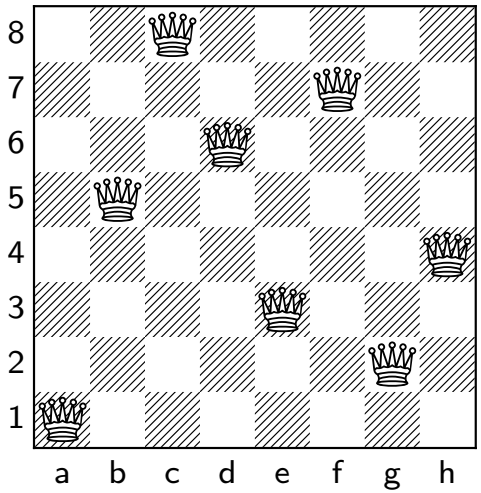
Le problème des 8 reines



Le problème des 8 reines

- But : placer 8 reines sur un échiquier de 8×8, sans attaque possible
- Problème à états multiples
- actions/opérateurs : ajouter une reine sur l'échiquier

Le problème des 8 reines



Le problème des 8 reines

- But : placer 8 reines sur un échiquier de 8×8, sans attaque possible
- Problème à états multiples
- actions/opérateurs : ajouter une reine sur l'échiquier
- Résoudre des solutions partielles :
  - Placer la première reine, puis placer les suivantes dans les cases restantes de l'échiquier

Les tours de Hanoï

Le problème des tours de Hanoï

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.
- Problème à états multiples

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.
- Problème à états multiples
- Actions/opérateurs : déplacer un disque à la fois

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.
- Problème à états multiples
- Actions/opérateurs : déplacer un disque à la fois
- Résoudre des sous problèmes

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.
- Problème à états multiples
- Actions/opérateurs : déplacer un disque à la fois
- Résoudre des sous problèmes
  - Placer N-1 disques de A vers B

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.
- Problème à états multiples
- Actions/opérateurs : déplacer un disque à la fois
- Résoudre des sous problèmes
  - Placer N-1 disques de A vers B
  - Placer le Neme disque de A vers C

Les tours de Hanoï

Le problème des tours de Hanoï

- But : 3 mats (A, B, C) ; Déplacer la pyramide de N disques du mat A au mat C.
- Problème à états multiples
- Actions/opérateurs : déplacer un disque à la fois
- Résoudre des sous problèmes
  - Placer N-1 disques de A vers B
  - Placer le Neme disque de A vers C
  - Placer les N-1 disques de B vers C

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

- But : agencer les pièces en ordre croissant en un coût minimum

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

- But : agencer les pièces en ordre croissant en un coût minimum
- état initial e0 : positions des 8 pièces dans une des 9 cases

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

- But : agencer les pièces en ordre croissant en un coût minimum
- état initial e0 : positions des 8 pièces dans une des 9 cases
- calcul coût : 1 déplacement coûte 1, coût total = nombre de déplacements

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

- But : agencer les pièces en ordre croissant en un coût minimum
- état initial e0 : positions des 8 pièces dans une des 9 cases
- calcul coût : 1 déplacement coûte 1, coût total = nombre de déplacements
- Actions/opérateurs : déplacer la case vide à gauche (L), à droite (R), en haut (U), en bas (D)

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

- But : agencer les pièces en ordre croissant en un coût minimum
- état initial e0 : positions des 8 pièces dans une des 9 cases
- calcul coût : 1 déplacement coûte 1, coût total = nombre de déplacements
- Actions/opérateurs : déplacer la case vide à gauche (L), à droite (R), en haut (U), en bas (D)
- Résoudre par solutions partielles

Le taquin-8

4	3	5		1	2	3
1	6	2	$\longrightarrow$	4	5	6
7	8	.		7	8	.

Le problème du taquin 8

- But : agencer les pièces en ordre croissant en un coût minimum
- état initial e0 : positions des 8 pièces dans une des 9 cases
- calcul coût : 1 déplacement coûte 1, coût total = nombre de déplacements
- Actions/opérateurs : déplacer la case vide à gauche (L), à droite (R), en haut (U), en bas (D)
- Résoudre par solutions partielles
- remarque : trouver solution optimale pour un taquin-n est NP-difficile

Représentation par graphe d'états

Graphe d'états

Représentation par graphe des états du problème :

Ex. ac les bus...

Représentation par graphe d'états

Graphe d'états

Représentation par graphe des états du problème :

- Les sommets représentent les états

Ex. ac les bus...

Représentation par graphe d'états

Graphe d'états

Représentation par graphe des états du problème :

- Les sommets représentent les états
- Un arc (i,j) représentent l'opération/l'action permettant de l'état i à l'état j

Ex. ac les bus...

Représentation par graphe d'états

Graphe d'états

Représentation par graphe des états du problème :

- Les sommets représentent les états
- Un arc (i,j) représentent l'opération/l'action permettant de l'état i à l'état j
- Solution = chemin entre l'état initial et l'état final

Ex. ac les bus...

Représentation par graphe d'états

Graphe d'états

Représentation par graphe des états du problème :

- Les sommets représentent les états
- Un arc (i,j) représentent l'opération/l'action permettant de l'état i à l'état j
- Solution = chemin entre l'état initial et l'état final
- Recherche de Solution = Recherche du/d'un chemin entre l'état initial et l'état final

Ex. ac les bus...

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état $=$ 1 case

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état $=$ 1 case
  - 2^{64} états possibles : $case(i, j) \in \{reine, \emptyset\}$

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i, j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne
  - 2^8 états possibles : $ligne(i) \in \{1 \ldots 8, \emptyset\}$
    $ligne(i) =$

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne
  - 2^8 états possibles : $ligne(i) \in \{1 \ldots 8, \emptyset\}$
    $ligne(i) =$
    - j si reine dans la case (i,j)

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne
  - 2^8 états possibles : $ligne(i) \in \{1 \ldots 8, \emptyset\}$
    $ligne(i) =$
    - j si reine dans la case (i,j)
    - $\emptyset$ si aucune reine sur la ligne i

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne
  - 2^8 états possibles : $ligne(i) \in \{1 \ldots 8, \emptyset\}$
    $ligne(i) =$
    - j si reine dans la case (i,j)
    - $\emptyset$ si aucune reine sur la ligne i
  - Action/Opérateur : placerReine(i,j) :

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne
  - 2^8 états possibles : $ligne(i) \in \{1 \ldots 8, \emptyset\}$
    $ligne(i) =$
    - j si reine dans la case (i,j)
    - $\emptyset$ si aucune reine sur la ligne i
  - Action/Opérateur : placerReine(i,j) :
    - SI $ligne(i) = \emptyset$ ALORS $ligne(i) \leftarrow j$

Importance du choix de représentation

Choix de Graphe d'états

Reprenons le problèmes des 8 reines sur un échiquer 8×8

- Si 1 état = 1 case
  - 2^{64} états possibles : $case(i,j) \in \{reine, \emptyset\}$
  - Action/Opérateur : placerReine(i,j) :
    - SI $case(i,j) = \emptyset$ ALORS $case(i,j) \leftarrow reine$
  - Test : PriseImpossible, ToutesReinesPlacees
- Si 1 état = 1 ligne
  - 2^8 états possibles : $ligne(i) \in \{1 \ldots 8, \emptyset\}$
    $ligne(i) =$
    - j si reine dans la case (i,j)
    - $\emptyset$ si aucune reine sur la ligne i
  - Action/Opérateur : placerReine(i,j) :
    - SI $ligne(i) = \emptyset$ ALORS $ligne(i) \leftarrow j$
  - Test : PriseImpossible, ToutesReinesPlacees

Evaluation de la recherche

Critères d'évaluation

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Optimalité : la solution retenue est la meilleure

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Optimalité : la solution retenue est la meilleure

Complexité

Les complexités en temps et en espace dépendent de :

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Optimalité : la solution retenue est la meilleure

Complexité

Les complexités en temps et en espace dépendent de :

- b = facteur de branchement maximum de l'arbre de recherche,

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Optimalité : la solution retenue est la meilleure

Complexité

Les complexités en temps et en espace dépendent de :

- b = facteur de branchement maximum de l'arbre de recherche,
- d = profondeur à laquelle se trouve le (meilleur) noeud-solution,

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Optimalité : la solution retenue est la meilleure

Complexité

Les complexités en temps et en espace dépendent de :

- b = facteur de branchement maximum de l'arbre de recherche,
- d = profondeur à laquelle se trouve le (meilleur) noeud-solution,
- m = profondeur maximum de l'espace de recherche (espace d'états).

Evaluation de la recherche

Critères d'évaluation

Complétude : si une solution existe, la méthode de recherche la trouve toujours

Complexité en temps : temps nécessaire pour trouver la solution

Complexité en espace : espace mémoire nécessaire pour effectuer la recherche

Optimalité : la solution retenue est la meilleure

Complexité

Les complexités en temps et en espace dépendent de :

- b = facteur de branchement maximum de l'arbre de recherche,
- d = profondeur à laquelle se trouve le (meilleur) noeud-solution,
- m = profondeur maximum de l'espace de recherche (espace d'états).
  - Remarque : il est possible d'avoir $m = \infty$

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul
  - son évolution en fonction des données entrées

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul
  - son évolution en fonction des données entrées
  - le pire cas

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul
  - son évolution en fonction des données entrées
  - le pire cas

Taux de croissance

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul
  - son évolution en fonction des données entrées
  - le pire cas

Taux de croissance

- Le temps de calcul $T(n)$ est fonction de l'ordre du taux de croissance $f(n)$

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul
  - son évolution en fonction des données entrées
  - le pire cas

Taux de croissance

- Le temps de calcul $T(n)$ est fonction de l'ordre du taux de croissance $f(n)$
  - On a $T(n) = O(f(n))$ ssi $\lim_{n\to\infty} \frac{T(n)}{f(n)} = constante$

Evaluation de la recherche

Questions sur la complexité

Comment évaluer le temps indépendemment de la machine ?

- On note $T(n)$ le temps de calcul d'un algorithme en fonction de la taille des données d'entrées.
- utiliser de préférence :
  - l'ordre de grandeur du temps de calcul
  - son évolution en fonction des données entrées
  - le pire cas

Taux de croissance

- Le temps de calcul $T(n)$ est fonction de l'ordre du taux de croissance $f(n)$
  - On a $T(n) = O(f(n))$ ssi $\lim_{n\to\infty} \frac{T(n)}{f(n)} = constante$
- ex : $T(n) = O(5n^2)$ ou $T(n) = O(2n^2 + 7n)$, ces deux temps se simplifient grossièrement en $T(n) = O(n^2)$

Evolution du temps de recherche

(données de O.Gasquet, P. Muller ; "Représentation et résolution de problèmes"

Evolution du temps par rapport au nombre de données
(base : $T(x) = 1 \Rightarrow 1\mu s$)

$T(n)/n$	10	20	30	40	50	60
n	$10\mu s$	$20\mu s$	$30\mu s$	$40\mu s$	$50\mu s$	$60\mu s$
$\log n$	$1\mu s$	$1.3\mu s$	$1.5\mu s$	$1.6\mu s$	$1.7\mu s$	$1.8\mu s$
$n \log n$	$10\mu s$	$26\mu s$	$44\mu s$	$64\mu s$	$85\mu s$	$107\mu s$
n^2	$100\mu s$	$400\mu s$	$900\mu s$	1.6 ms	2.5 ms	3.5 ms
n^3	1 ms	8 ms	27 ms	64 ms	125 ms	216 ms
n^5	0.1 s	3.2 s	24.3 s	1.7 mn	5.2 mn	13 mn
2^n	1 ms	1 s	18 mn	13 jours	36 ans	366 siècles
3^n	59 ms	58 mn	6 mn	3855 siècles	2.10^8 siècles	$1,3.10^{13}$ siècles

Type de recherches

Type de recherches

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur
  - recherche en coût uniforme

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur
  - recherche en coût uniforme
  - recherche en profondeur

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur
  - recherche en coût uniforme
  - recherche en profondeur
  - recherche en profondeur limitée

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur
  - recherche en coût uniforme
  - recherche en profondeur
  - recherche en profondeur limitée
  - recherche par approfondissement itératif

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur
  - recherche en coût uniforme
  - recherche en profondeur
  - recherche en profondeur limitée
  - recherche par approfondissement itératif
  - recherche bi-directionnelle

Type de recherches

Type de recherches

- Méthodes de recherche aveugles, sans utilisation de connaissances sur le problème :
  - recherche en largeur
  - recherche en coût uniforme
  - recherche en profondeur
  - recherche en profondeur limitée
  - recherche par approfondissement itératif
  - recherche bi-directionnelle
- Méthodes de recherche informées (heuristiques)

Recherche en largeur

Recherche en largeur

Recherche en largeur

Recherche en largeur

- Stratégie :

Recherche en largeur

Recherche en largeur

- Stratégie :
  - étendre totalement le noeud sur 1 niveau

Recherche en largeur

Recherche en largeur

- Stratégie :
  - étendre totalement le noeud sur 1 niveau
  - effectuer une recherche en largeur sur chacun des noeuds générés pris séquentiellement

Recherche en largeur

Recherche en largeur

- Stratégie :
  - étendre totalement le noeud sur 1 niveau
  - effectuer une recherche en largeur sur chacun des noeuds générés pris séquentiellement
- Complétude : Oui ! (si b (branchement) est fini)

Recherche en largeur

Recherche en largeur

- Stratégie :
  - étendre totalement le noeud sur 1 niveau
  - effectuer une recherche en largeur sur chacun des noeuds générés pris séquentiellement
- Complétude : Oui ! (si b (branchement) est fini)
- Complexité en temps : $ct = 1 + b + b \times b + \cdots + b^d \rightarrow$ en $O(b^d)$

Recherche en largeur

Recherche en largeur

- Stratégie :
  - étendre totalement le noeud sur 1 niveau
  - effectuer une recherche en largeur sur chacun des noeuds générés pris séquentiellement
- Complétude : Oui ! (si b (branchement) est fini)
- Complexité en temps : $ct = 1 + b + b \times b + \cdots + b^d \rightarrow$ en $O(b^d)$
- Complexité en espace : en $O(b^d)$ (il faut garder en mémoire les noeuds)

Recherche en largeur

Recherche en largeur

- Stratégie :
  - étendre totalement le noeud sur 1 niveau
  - effectuer une recherche en largeur sur chacun des noeuds générés pris séquentiellement
- Complétude : Oui ! (si b (branchement) est fini)
- Complexité en temps : $ct = 1 + b + b \times b + \cdots + b^d \rightarrow$ en $O(b^d)$
- Complexité en espace : en $O(b^d)$ (il faut garder en mémoire les noeuds)
- Optimalité : pas nécessairement

Recherche en coût uniforme

Recherche en coût uniforme

Recherche en coût uniforme

Recherche en coût uniforme

- Stratégie :

Recherche en coût uniforme

Recherche en coût uniforme

- Stratégie :
  - en largeur mais n'étend que le noeud de coup le plus faible

Recherche en coût uniforme

Recherche en coût uniforme

- Stratégie :
  - en largeur mais n'étend que le noeud de coup le plus faible
- Complétude : Oui

Recherche en coût uniforme

Recherche en coût uniforme

- Stratégie :
  - en largeur mais n'étend que le noeud de coup le plus faible
- Complétude : Oui
- Complexité en temps : en $O(b^d)$

Recherche en coût uniforme

Recherche en coût uniforme

- Stratégie :
  - en largeur mais n'étend que le noeud de coup le plus faible
- Complétude : Oui
- Complexité en temps : en $O(b^d)$
- Complexité en espace : en $O(b^d)$

Recherche en coût uniforme

Recherche en coût uniforme

- Stratégie :
  - en largeur mais n'étend que le noeud de coup le plus faible
- Complétude : Oui
- Complexité en temps : en $O(b^d)$
- Complexité en espace : en $O(b^d)$
- Optimalité : oui !

Recherche en profondeur

Recherche en profondeur

Recherche en profondeur

Recherche en profondeur

- Stratégie :

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds
  - placer les autres noeuds en file d'attente

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds
  - placer les autres noeuds en file d'attente
  - éliminer des noeuds de la file d'attente les noeuds rencontrés

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds
  - placer les autres noeuds en file d'attente
  - éliminer des noeuds de la file d'attente les noeuds rencontrés
- Complétude : Oui (si espaces d'états finis et acycliques)

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds
  - placer les autres noeuds en file d'attente
  - éliminer des noeuds de la file d'attente les noeuds rencontrés
- Complétude : Oui (si espaces d'états finis et acycliques)
- Complexité en temps : en $O(b^m)$

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds
  - placer les autres noeuds en file d'attente
  - éliminer des noeuds de la file d'attente les noeuds rencontrés
- Complétude : Oui (si espaces d'états finis et acycliques)
- Complexité en temps : en $O(b^m)$
- Complexité en espace : en $O(b \times m)$ (intéressant)

Recherche en profondeur

Recherche en profondeur

- Stratégie :
  - étendre le 1er noeud
  - effectuer une recherche en profondeur sur le premier des sous-noeuds
  - placer les autres noeuds en file d'attente
  - éliminer des noeuds de la file d'attente les noeuds rencontrés
- Complétude : Oui (si espaces d'états finis et acycliques)
- Complexité en temps : en $O(b^m)$
- Complexité en espace : en $O(b \times m)$ (intéressant)
- Optimalité : non

Recherche en profondeur limitée

Recherche en profondeur limitée

Recherche en profondeur limitée

Recherche en profondeur limitée

- Stratégie :

Recherche en profondeur limitée

Recherche en profondeur limitée

- Stratégie :
  - recherche en profondeur avec une limite d'exploration L

Recherche en profondeur limitée

Recherche en profondeur limitée

- Stratégie :
  - recherche en profondeur avec une limite d'exploration L
- Complétude : Oui (si $L \geq d$)

Recherche en profondeur limitée

Recherche en profondeur limitée

- Stratégie :
  - recherche en profondeur avec une limite d'exploration L
- Complétude : Oui (si $L \geq d$)
- Complexité en temps : en $O(b^L)$

Recherche en profondeur limitée

Recherche en profondeur limitée

- Stratégie :
  - recherche en profondeur avec une limite d'exploration L
- Complétude : Oui (si $L \geq d$)
- Complexité en temps : en $O(b^L)$
- Complexité en espace : en $O(b \times L)$

Recherche en profondeur limitée

Recherche en profondeur limitée

- Stratégie :
  - recherche en profondeur avec une limite d'exploration L
- Complétude : Oui (si $L \geq d$)
- Complexité en temps : en $O(b^L)$
- Complexité en espace : en $O(b \times L)$
- Optimalité : non

Recherche par approfondissement itératif

Recherche par approfondissement itératif

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :
  - recherche en profondeur limité en incrémentant L itérativement, à partir de $L = 0$

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :
  - recherche en profondeur limité en incrémentant L itérativement, à partir de $L = 0$
  - combine donc recherche en largeur et recherche en profondeur

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :
  - recherche en profondeur limité en incrémentant L itérativement, à partir de $L = 0$
  - combine donc recherche en largeur et recherche en profondeur
- Complétude : Oui

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :
  - recherche en profondeur limité en incrémentant L itérativement, à partir de $L = 0$
  - combine donc recherche en largeur et recherche en profondeur
- Complétude : Oui
- Complexité en temps : en $O(b^d) := (d+1)b^0 + db + (d-1)b^2 + \cdots + b^d$

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :
  - recherche en profondeur limité en incrémentant L itérativement, à partir de $L = 0$
  - combine donc recherche en largeur et recherche en profondeur
- Complétude : Oui
- Complexité en temps : en $O(b^d) := (d+1)b^0 + db + (d-1)b^2 + \cdots + b^d$
- Complexité en espace : en $O(b \times d)$

Recherche par approfondissement itératif

Recherche par approfondissement itératif

- Stratégie :
  - recherche en profondeur limité en incrémentant L itérativement, à partir de $L = 0$
  - combine donc recherche en largeur et recherche en profondeur
- Complétude : Oui
- Complexité en temps : en $O(b^d) := (d+1)b^0 + db + (d-1)b^2 + \cdots + b^d$
- Complexité en espace : en $O(b \times d)$
- Optimalité : oui (si coût par noeud $= 1$)

Recherche bi-directionnelle

Recherche bi-directionnelle

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial
  - Rechercher en arrière à partir de l'état final

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial
  - Rechercher en arrière à partir de l'état final
  - Arrêter la recherche lorsque les 2 sous-recherches se rencontrent

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial
  - Rechercher en arrière à partir de l'état final
  - Arrêter la recherche lorsque les 2 sous-recherches se rencontrent
- Complétude : Oui

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial
  - Rechercher en arrière à partir de l'état final
  - Arrêter la recherche lorsque les 2 sous-recherches se rencontrent
- Complétude : Oui
- Complexité en temps : en $O(b^d/2)$

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial
  - Rechercher en arrière à partir de l'état final
  - Arrêter la recherche lorsque les 2 sous-recherches se rencontrent
- Complétude : Oui
- Complexité en temps : en $O(b^d/2)$
- Complexité en espace : en $O(b^d/2)$

Recherche bi-directionnelle

Recherche bi-directionnelle

- Stratégie :
  - Rechercher en avant à partir de l'état initial
  - Rechercher en arrière à partir de l'état final
  - Arrêter la recherche lorsque les 2 sous-recherches se rencontrent
- Complétude : Oui
- Complexité en temps : en $O(b^d/2)$
- Complexité en espace : en $O(b^d/2)$
- Optimalité : oui

Exemple d'algorithme avec Retour Arrière

```
procedure RECHERCHERA(etat)
    solution ← ∅
    if ESTFINAL(etat) then retourner chemin ← ∅
    else
        operateurs ← OPERATEURSAPPLICABLES(etat)
        for all op ∈ operateurs do
            nouvelEtat ← OP(etat)
            chemin ← RECHERCHERA(nouvelEtat)
            if chemin ≠ ECHEC then
                solution ← chemin ∪ {op, etat}
                retourner solution
            end if
        end for
        retourner ECHEC
    end if
end procedure
```

Exemple d'algorithme avec Graphe

procedure RECHERCHE
  noeudsLibres $\leftarrow$ *etatInitial*
  noeudsClos $\leftarrow \emptyset$
  succes $\leftarrow$ *faux*
  while *noeudsLibres* $\neq \emptyset \wedge$ (non(succes)) **do**
    n $\leftarrow$ CHOISIRNOEUD(noeudsLibres)
    if ESTFINAL(n) **then**
      succes $\leftarrow$ vrai
    else
      noeudsLibres $\leftarrow$ *noeudsLibres* $- \{n\}$
      noeudsClos $\leftarrow$ *noeudsClos* $\cup \{n\}$
      for all $s \in n.successeurs$ **do**
        if $s \notin$ *noeudsLibres* $\vee$ $s \notin$ *noeudsClos* **then**
          noeudsLibres $\leftarrow$ *noeudsLibres* $\cup \{s\}$
          $s.pere \leftarrow n$
        end if
      end for
    end if
  end while
end procedure

Heuristiques

Notion d'heuristiques

Heuristiques

Notion d'heuristiques

- Méthodes en aveugle trop gourmandes en mémoire et/ou en temps

Heuristiques

Notion d'heuristiques

- Méthodes en aveugle trop gourmandes en mémoire et/ou en temps
- Une solution : orienter la recherche par une information heuristique

Heuristiques

Notion d'heuristiques

- Méthodes en aveugle trop gourmandes en mémoire et/ou en temps
- Une solution : orienter la recherche par une information heuristique
- Une *heuristique* doit guider le choix des états à tester et les ordonner selon leurs "promesse de rapprocher d'un but".

Heuristiques

Notion d'heuristiques

- Méthodes en aveugle trop gourmandes en mémoire et/ou en temps
- Une solution : orienter la recherche par une information heuristique
- Une *heuristique* doit guider le choix des états à tester et les ordonner selon leurs "promesse de rapprocher d'un but".
  - Une heuristique dépend fortement du problème à traiter

Heuristiques

Notion d'heuristiques

- Méthodes en aveugle trop gourmandes en mémoire et/ou en temps
- Une solution : orienter la recherche par une information heuristique
- Une *heuristique* doit guider le choix des états à tester et les ordonner selon leurs "promesse de rapprocher d'un but".
  - Une heuristique dépend fortement du problème à traiter
  - Une heuristique *pauvre* basée sur des propriétés trop simples du problème sera peu efficace,

Heuristiques

Notion d'heuristiques

- Méthodes en aveugle trop gourmandes en mémoire et/ou en temps
- Une solution : orienter la recherche par une information heuristique
- Une *heuristique* doit guider le choix des états à tester et les ordonner selon leurs "promesse de rapprocher d'un but".
  - Une heuristique dépend fortement du problème à traiter
  - Une heuristique *pauvre* basée sur des propriétés trop simples du problème sera peu efficace,
  - Une heuristique *riche* basée sur des propriétés approfondies du problème sera efficace, mais est difficile à établir.

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

Il est donc nécessaire de définir :

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

- $g^*(n)$ est défini comme le le coût minimum entre le noeud de départ $n0$ et le noeud n

Il est donc nécessaire de définir :

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

- $g^*(n)$ est défini comme le le coût minimum entre le noeud de départ $n0$ et le noeud n
- $h^*(n)$ est le minimum du coût des chemins du noeud n à un noeud but quelconque.

Il est donc nécessaire de définir :

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

- $g^*(n)$ est défini comme le le coût minimum entre le noeud de départ $n0$ et le noeud n
- $h^*(n)$ est le minimum du coût des chemins du noeud n à un noeud but quelconque.
- $f^*(n) = g^*(n) + h^*(n)$ est le coût du chemin solution optimal passant par n.

Il est donc nécessaire de définir :

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

- $g^*(n)$ est défini comme le le coût minimum entre le noeud de départ $n0$ et le noeud n
- $h^*(n)$ est le minimum du coût des chemins du noeud n à un noeud but quelconque.
- $f^*(n) = g^*(n) + h^*(n)$ est le coût du chemin solution optimal passant par n.

Il est donc nécessaire de définir :

- une heuristique h(n) qui *estime* $h^*(n)$

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

- $g^*(n)$ est défini comme le le coût minimum entre le noeud de départ $n0$ et le noeud n
- $h^*(n)$ est le minimum du coût des chemins du noeud n à un noeud but quelconque.
- $f^*(n) = g^*(n) + h^*(n)$ est le coût du chemin solution optimal passant par n.

Il est donc nécessaire de définir :

- une heuristique h(n) qui *estime* $h^*(n)$
- g(n) le gain, le coût effectif du meilleur chemin connu pour aller de $n0$ à n (c'est un estimateur de $g^*(n)$)

Recherche Heuristique

Définitions pour la recherche par heuristique

Soit n un noeud du graphe

- $g^*(n)$ est défini comme le le coût minimum entre le noeud de départ $n0$ et le noeud n
- $h^*(n)$ est le minimum du coût des chemins du noeud n à un noeud but quelconque.
- $f^*(n) = g^*(n) + h^*(n)$ est le coût du chemin solution optimal passant par n.

Il est donc nécessaire de définir :

- une heuristique h(n) qui *estime* $h^*(n)$
- g(n) le gain, le coût effectif du meilleur chemin connu pour aller de $n0$ à n (c'est un estimateur de $g^*(n)$)
- $f(n) = g(n) + h(n)$, la fonction d'évaluation du noeud n

Exemple d'algorithme A* I

procedure RECHERCHEA*
  noeudsLibres ← *etatInitial*
  noeudsClos ← ∅
  succes ← *faux*
  while *noeudsLibres* ≠ ∅ ∧ (non(succes)) **do**
    ▷ choisir n dans les noeudsLibres tel que f(n) est minimum
    n ← CHOISIRNOEUDHEURISTIQUE(noeudsLibres)
    if ESTFINAL(n) **then**
      succes ← vrai
    else
      noeudsLibres ← *noeudsLibres* − {*n*}
      noeudsClos ← *noeudsClos* ∪ {*n*}
      for all *s* ∈ *n.successeurs* **do**
        if *s* ∉ *noeudsLibres* ∨ *s* ∉ *noeudsClos* **then**
          noeudsLibres ← *noeudsLibres* ∪ {*s*}
          s.pere ← *n*
          $g(s) \leftarrow g(n) + cout(n, s)$
        else

Exemple d'algorithme A* II

if $g(s) \geq g(n) + cout(n, s)$ **then**
$s.pere \leftarrow n$
$g(s) \leftarrow g(n) + cout(n, s)$
if $s \in noeudsClos$ **then**
$noeudsLibres \leftarrow noeudsLibres \cup \{s\}$
$noeudsClos \leftarrow noeudsClos - \{s\}$
end if
end if
end if
end for
end if
end while
end procedure

propriétés de la recherche par heuristique

Particularités

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;
  - si $\forall n, f(n) = h(n)$, l'algorithme est équivalent à une recherche en profondeur

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;
  - si $\forall n, f(n) = h(n)$, l'algorithme est équivalent à une recherche en profondeur

Propriétés

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;
  - si $\forall n, f(n) = h(n)$, l'algorithme est équivalent à une recherche en profondeur

Propriétés

- h est dite *minorante* si $\forall n, h(n) \leq h^*(n)$

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;
  - si $\forall n, f(n) = h(n)$, l'algorithme est équivalent à une recherche en profondeur

Propriétés

- h est dite *minorante* si $\forall n, h(n) \leq h^*(n)$
- h est dite *monotone* si
  $\forall n, \forall s \in successeur(n), h(n) - h(s) \leq cout(n, s)$

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;
  - si $\forall n, f(n) = h(n)$, l'algorithme est équivalent à une recherche en profondeur

Propriétés

- h est dite *minorante* si $\forall n, h(n) \leq h^*(n)$
- h est dite *monotone* si
  $\forall n, \forall s \in successeur(n), h(n) - h(s) \leq cout(n, s)$
- si h est *minorante*, alors A* est *admissible*

propriétés de la recherche par heuristique

Particularités

- Cas particuliers :
  - si $\forall n, h(n) = 0$, alors f(n)=g(n) ; l'algorithme est équivalent à une recherche en largeur ;
  - si $\forall n, f(n) = h(n)$, l'algorithme est équivalent à une recherche en profondeur

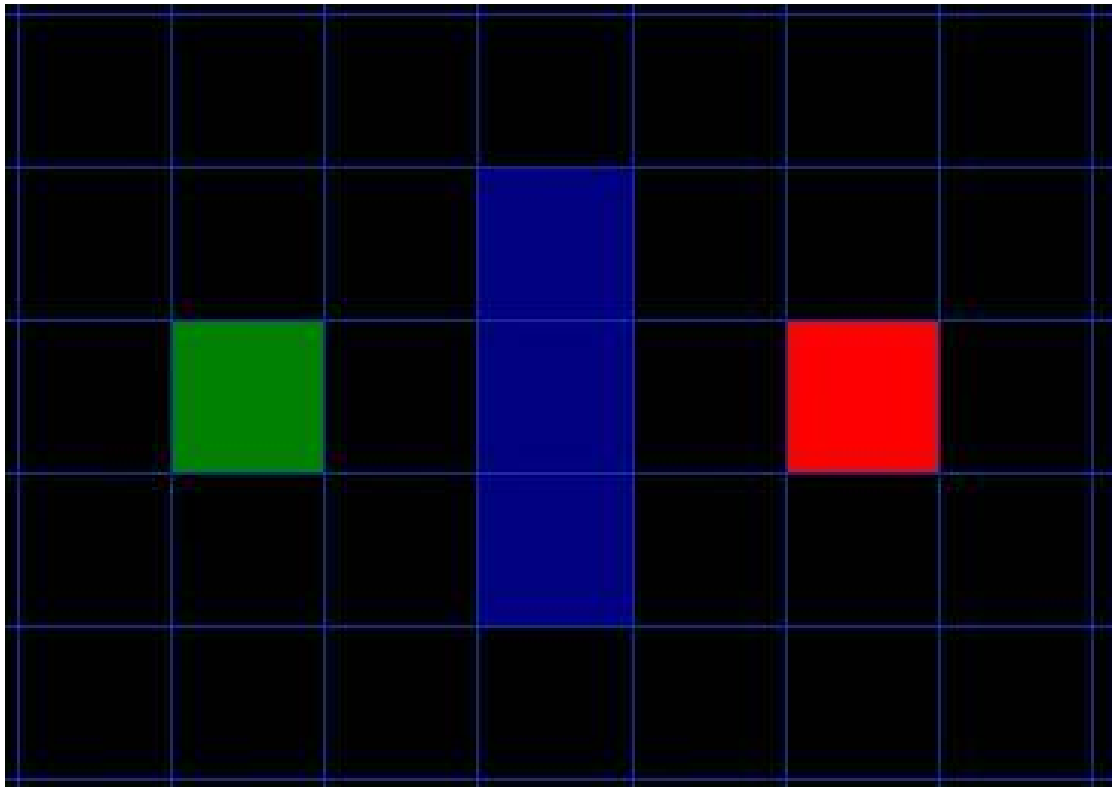
Propriétés

- h est dite *minorante* si $\forall n, h(n) \leq h^*(n)$
- h est dite *monotone* si
  $\forall n, \forall s \in successeur(n), h(n) - h(s) \leq cout(n, s)$
- si h est *minorante*, alors A* est *admissible*
- si h est *monotone* alors la première rencontre d'un noeud fournit le meilleur chemin pour y arriver (on est dispensé des mises à jour dans l'algorithme).

Recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

But : Se rendre de la case verte à la case rouge en évitant les case bleues



Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

- Définir l'espace d'états

Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

- Définir l'espace d'états
  - Décomposition de la zone en grille, l'espace est constitué des cellules

Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

- Définir l'espace d'états
  - Décomposition de la zone en grille, l'espace est constitué des cellules
- Etat initial = cellule verte A en c(1,2)

Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

- Définir l'espace d'états
  - Décomposition de la zone en grille, l'espace est constitué des cellules
- Etat initial = cellule verte A en c(1,2)
- Etat but/final = cellule rouge B en c(5,2)

Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

- Définir l'espace d'états
  - Décomposition de la zone en grille, l'espace est constitué des cellules
- Etat initial = cellule verte A en c(1,2)
- Etat but/final = cellule rouge B en c(5,2)
- Branchement = 8 :chaque cellule est liée à ses 8 cellules voisines

Initialisation de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Analyse du problème

- Définir l'espace d'états
  - Décomposition de la zone en grille, l'espace est constitué des cellules
- Etat initial = cellule verte A en c(1,2)
- Etat but/final = cellule rouge B en c(5,2)
- Branchement = 8 :chaque cellule est liée à ses 8 cellules voisines
- Action = se déplacer d'une case (H, B, D, G, HG, HD, BD, BG) si possible

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

- Définir $f = g + h$

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

- Définir $f = g + h$
  - $g(C)$: détermine le coût du trajet de A vers C en suivant le chemin généré

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

- Définir $f = g + h$
  - $g(C)$: détermine le coût du trajet de A vers C en suivant le chemin généré
  - $h(C)$: coût estimé du mouvement de C vers le but B

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

- Définir $f = g + h$
  - $g(C)$: détermine le coût du trajet de A vers C en suivant le chemin généré
  - $h(C)$: coût estimé du mouvement de C vers le but B
- Choix effectué pour

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

- Définir $f = g + h$
  - $g(C)$: détermine le coût du trajet de A vers C en suivant le chemin généré
  - $h(C)$: coût estimé du mouvement de C vers le but B
- Choix effectué pour
  - $g(C)$: coût du trajet entre cases adjacentes $= 10$; en diagonale $= 14$ $(int(10 \times \sqrt{2}))$

Coût et Heuristiques de la recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Définir les coûts et heuristiques

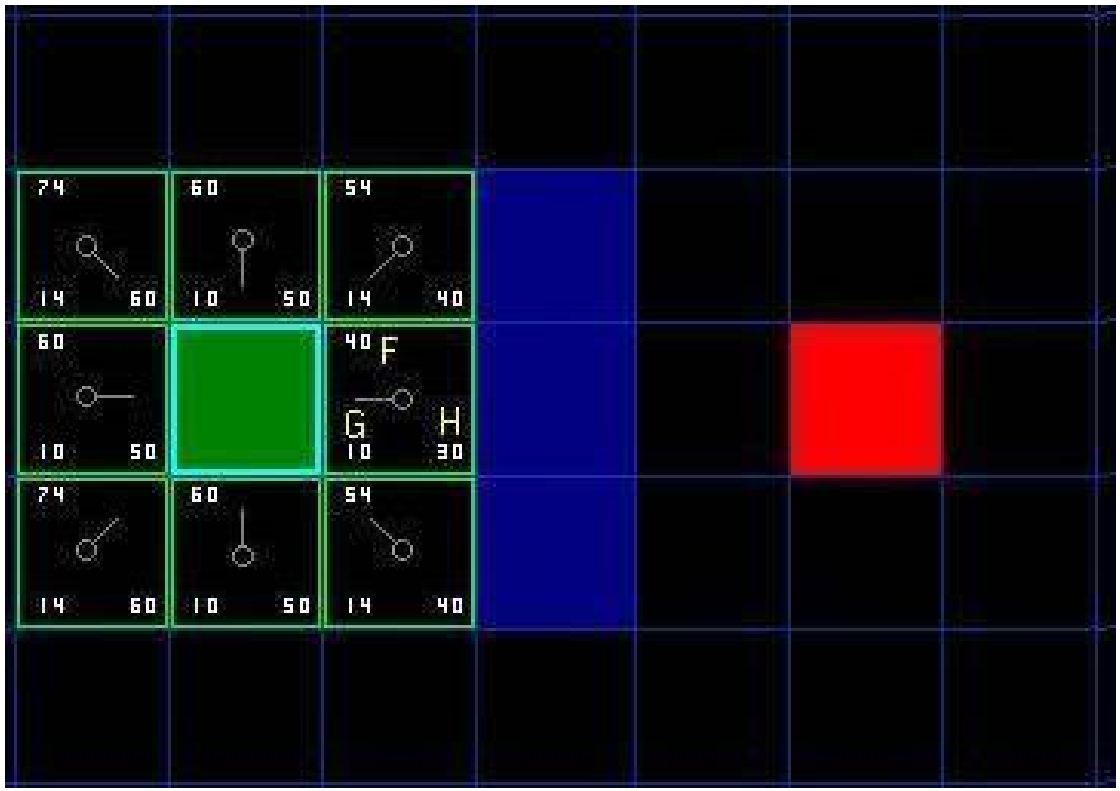
- Définir $f = g + h$
  - $g(C)$: détermine le coût du trajet de A vers C en suivant le chemin généré
  - $h(C)$: coût estimé du mouvement de C vers le but B
- Choix effectué pour
  - $g(C)$: coût du trajet entre cases adjacentes $= 10$; en diagonale $= 14$ $(int(10 \times \sqrt{2}))$
  - $h(C)$: $|x_C - x_B| + |y_C - y_B|$

Recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Valeurs de f, g, h autour de A

valeur de f dans le coin gauche-haut, valeur de g dans le coin gauche-bas et valeur de h dans le coin bas-droite



Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos
- Analyse des cases voisines à la case retenue

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos
- Analyse des cases voisines à la case retenue
  - seules 4 cases non closes (par Haut, Bas, HautGauche, BasGauche) ;

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos
- Analyse des cases voisines à la case retenue
  - seules 4 cases non closes (par Haut, Bas, HautGauche, BasGauche) ;
  - ces cases sont déjà dans la liste des cases libres

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos
- Analyse des cases voisines à la case retenue
  - seules 4 cases non closes (par Haut, Bas, HautGauche, BasGauche) ;
  - ces cases sont déjà dans la liste des cases libres
  - on teste alors si l'accès par la nouvelle case est plus intéressant que l'accès déjà calculé à partir de A

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos
- Analyse des cases voisines à la case retenue
  - seules 4 cases non closes (par Haut, Bas, HautGauche, BasGauche) ;
  - ces cases sont déjà dans la liste des cases libres
  - on teste alors si l'accès par la nouvelle case est plus intéressant que l'accès déjà calculé à partir de A
    - prenons la case dessous, on a déjà $g(s) = 14$, en passant par la case intermédaire on a $g(n) + cout(n, s) = 20$: plus coûteux donc n'est pas retenu

Déroulement de l'algorithme A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Déroulement de l'algorithme A*

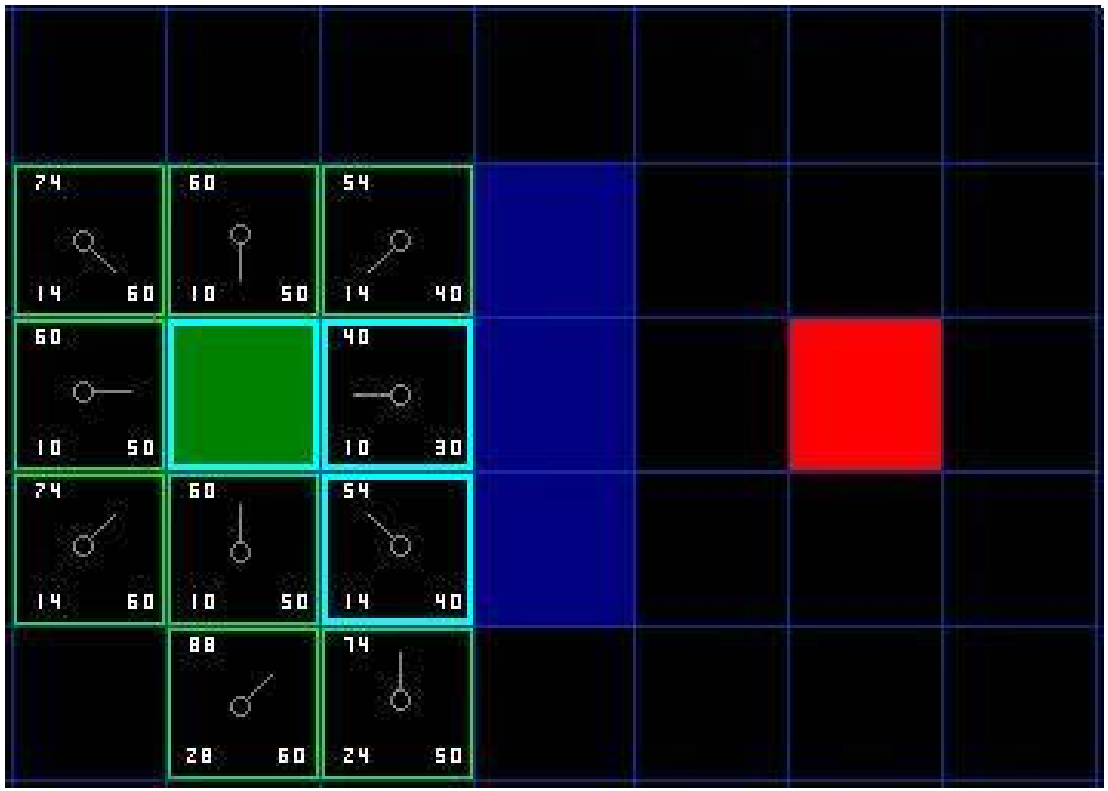
- Les noeuds libres sont les cases voisines de A
- Choix du meilleur noeud, de la meilleure case (ici avec $f = 40$)
  - on retire la case des noeuds libres
  - on la place dans les noeuds clos
- Analyse des cases voisines à la case retenue
  - seules 4 cases non closes (par Haut, Bas, HautGauche, BasGauche) ;
  - ces cases sont déjà dans la liste des cases libres
  - on teste alors si l'accès par la nouvelle case est plus intéressant que l'accès déjà calculé à partir de A
    - prenons la case dessous, on a déjà $g(s) = 14$, en passant par la case intermédaire on a $g(n) + cout(n, s) = 20$: plus coûteux donc n'est pas retenu
- Choix alors d'un autre meilleur noeud (ici un des noeuds avec $f = 54$)

Recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Sélection du second noeud intéressant

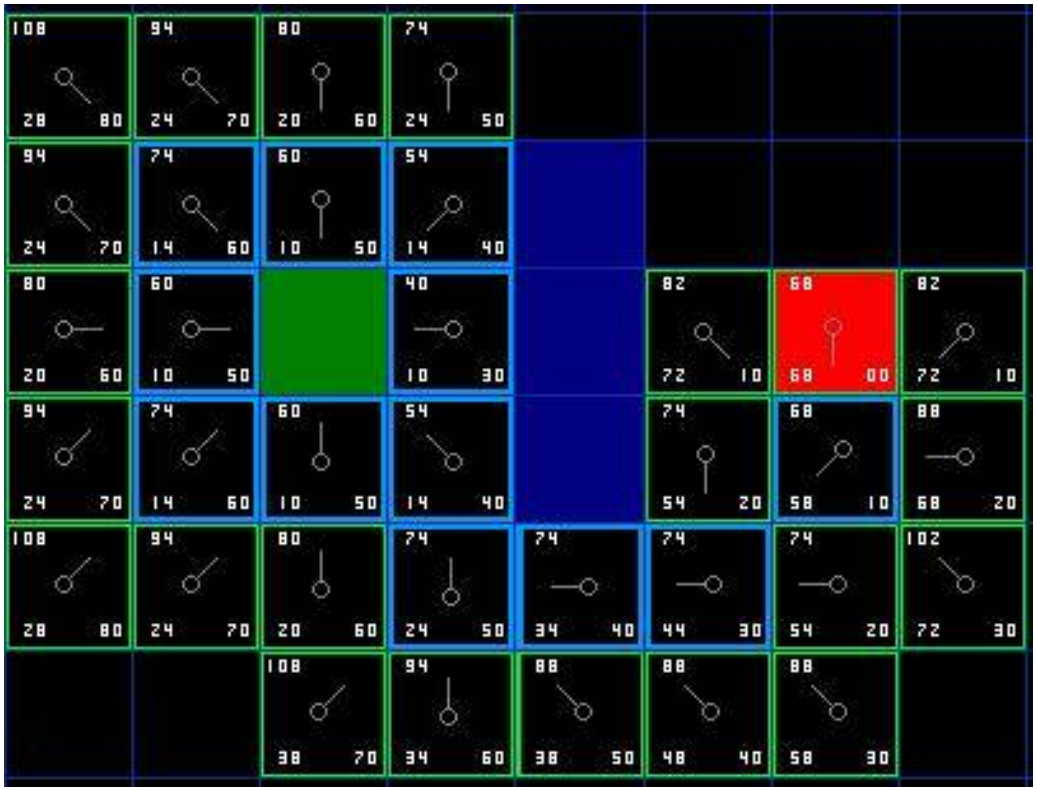
Seuls 2 noeuds libres ajoutés (la diagonale bas-droite est impossible à cause du mur)



Recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

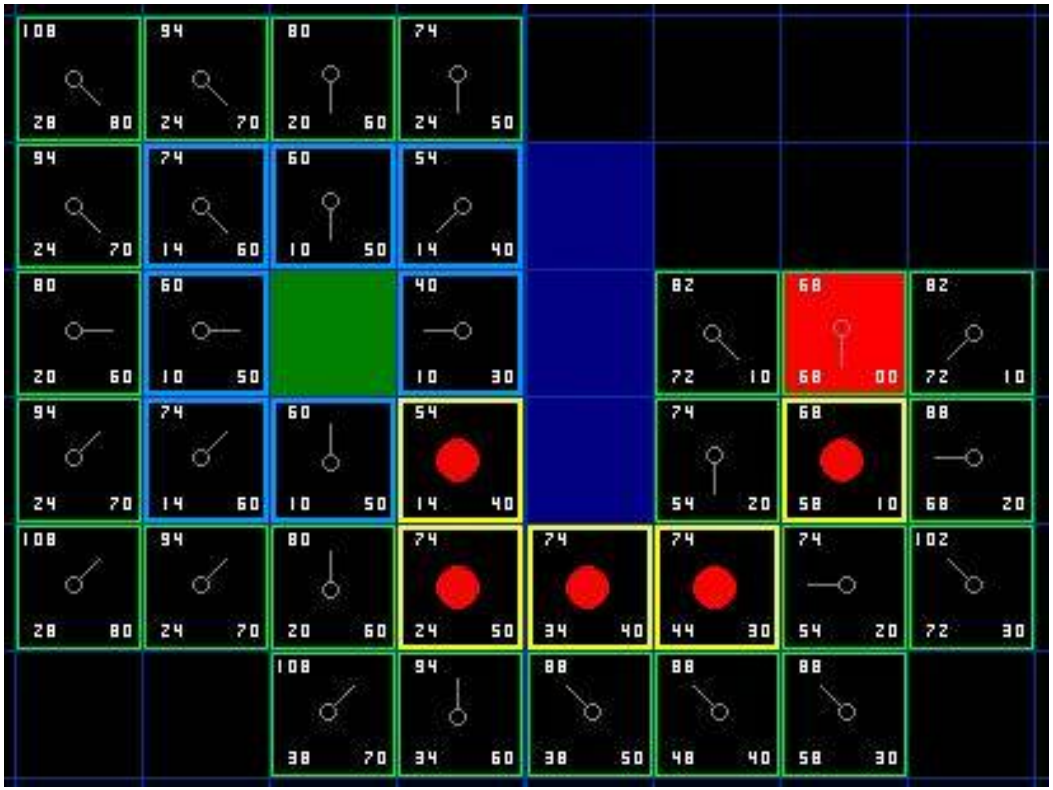
Répétition de l'algorithme jusqu'au but



Recherche du plus court chemin par A*

d'après "A* Pathfinding for Beginners" (Patrick Lester)

Etablissement de la solution à partir de l'arrivée par suivi du noeud père



Recherche Aléatoire

Concept de Recherche Aléatoire

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$
  - à partir d'une liste L de 6 nombres
    $nb_i \in values = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$
  - à partir d'une liste L de 6 nombres
    $nb_i \in values = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$
  - du choix de deux nombres $(a, b) \in L^2$
    et de 4 actions : $action \in \{a + b, a - b, a \times b, a \div b\}$

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$
  - à partir d'une liste L de 6 nombres
    $nb_i \in values = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$
  - du choix de deux nombres $(a, b) \in L^2$
    et de 4 actions : $action \in \{a + b, a - b, a \times b, a \div b\}$
    - chaque nombre généré par une action est utilisable et ajouté à L

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$
  - à partir d'une liste L de 6 nombres
    $nb_i \in values = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$
  - du choix de deux nombres $(a, b) \in L^2$
    et de 4 actions : $action \in \{a + b, a - b, a \times b, a \div b\}$
    - chaque nombre généré par une action est utilisable et ajouté à L
    - chaque nombre utilisé par une action n'est plus utilisable et retiré de L

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$
  - à partir d'une liste L de 6 nombres
    $nb_i \in values = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$
  - du choix de deux nombres $(a, b) \in L^2$
    et de 4 actions : $action \in \{a + b, a - b, a \times b, a \div b\}$
    - chaque nombre généré par une action est utilisable et ajouté à L
    - chaque nombre utilisé par une action n'est plus utilisable et retiré de L
    - une action peut être utilisée plusieurs fois

Recherche Aléatoire

Concept de Recherche Aléatoire

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est trop important pour balayer le tout
- La stratégie consiste alors à tenter des solutions au hasard
- Exemple : Le compte est bon (il est en fait possible de balayer le tout)
  - l'espace d'état est : $+\aleph$
  - à partir d'une liste L de 6 nombres
    $nb_i \in values = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$
  - du choix de deux nombres $(a, b) \in L^2$
    et de 4 actions : $action \in \{a + b, a - b, a \times b, a \div b\}$
    - chaque nombre généré par une action est utilisable et ajouté à L
    - chaque nombre utilisé par une action n'est plus utilisable et retiré de L
    - une action peut être utilisée plusieurs fois
  - il s'agit d'obtenir ou de se rapprocher au mieux d'un nombre de 3 chiffres

Recherche Aléatoire

Exemple de résolution de compte

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$
  - Exécution n°2 : but atteint après 1783 etapes

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$
  - Exécution n°2 : but atteint après 1783 etapes
    - $25 + 2 = 27; 6 + 27 = 33; 1 + 75 = 76; 10 * 76 = 760; 33 + 760 = 793$

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$
  - Exécution n°2 : but atteint après 1783 etapes
    - $25 + 2 = 27; 6 + 27 = 33; 1 + 75 = 76; 10 * 76 = 760; 33 + 760 = 793$
  - Exécution n°3 : but atteint après 3149 etapes

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$
  - Exécution n°2 : but atteint après 1783 etapes
    - $25 + 2 = 27; 6 + 27 = 33; 1 + 75 = 76; 10 * 76 = 760; 33 + 760 = 793$
  - Exécution n°3 : but atteint après 3149 etapes
    - $2 + 25 = 27; 6 + 1 = 7; 7 + 75 = 82; 10 * 82 = 820; 820 - 27 = 793$

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$
  - Exécution n°2 : but atteint après 1783 etapes
    - $25 + 2 = 27; 6 + 27 = 33; 1 + 75 = 76; 10 * 76 = 760; 33 + 760 = 793$
  - Exécution n°3 : but atteint après 3149 etapes
    - $2 + 25 = 27; 6 + 1 = 7; 7 + 75 = 82; 10 * 82 = 820; 820 - 27 = 793$
  - Exécution n°4 : but atteint après 355 etapes

Recherche Aléatoire

Exemple de résolution de compte

- Etat initial : $[1, 2, 6, 25, 75, 10]$
- Etat Final : 793
  - Exécution n°1 : but atteint après 540 etapes
    - $75 + 1 = 76; 76 * 10 = 760; 6 + 2 = 8; 25 + 760 = 785; 8 + 785 = 793$
  - Exécution n°2 : but atteint après 1783 etapes
    - $25 + 2 = 27; 6 + 27 = 33; 1 + 75 = 76; 10 * 76 = 760; 33 + 760 = 793$
  - Exécution n°3 : but atteint après 3149 etapes
    - $2 + 25 = 27; 6 + 1 = 7; 7 + 75 = 82; 10 * 82 = 820; 820 - 27 = 793$
  - Exécution n°4 : but atteint après 355 etapes
    - $1 + 10 = 11; 75 - 11 = 64; 64 * 2 = 128; 6 * 128 = 768; 25 + 768 = 793$

Recherche Exhaustive

Concept de Recherche Exhaustive

Recherche Exhaustive

Concept de Recherche Exhaustive

- Pour certains problèmes, l'heuristique est impossible ou trop complexe

Recherche Exhaustive

Concept de Recherche Exhaustive

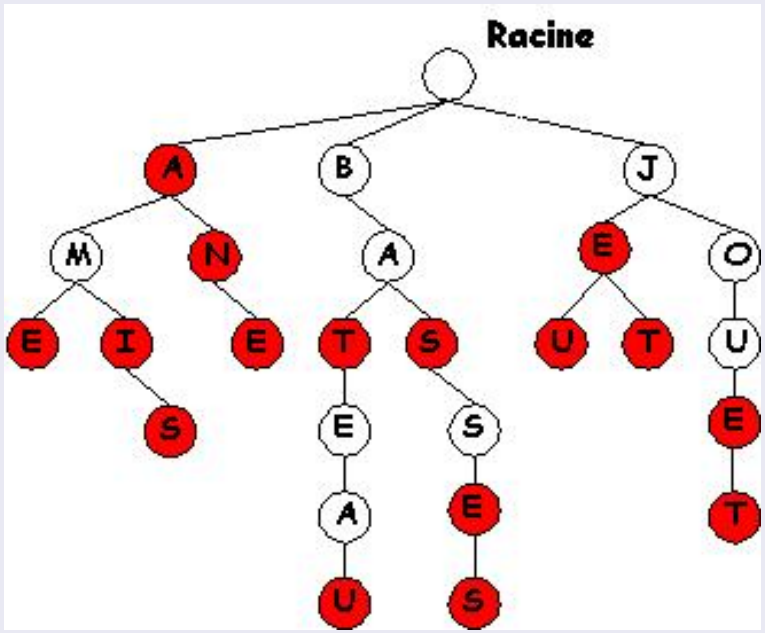
- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est important, mais il est nécessaire de balayer l'ensemble des combinaisons

Recherche Exhaustive

Concept de Recherche Exhaustive

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est important, mais il est nécessaire de balayer l'ensemble des combinaisons
- Exemple : Le solitaire, Le Scrabble

Recherche Exhaustive

Concept de Recherche Exhaustive

- Pour certains problèmes, l'heuristique est impossible ou trop complexe
- L'espace d'états est important, mais il est nécessaire de balayer l'ensemble des combinaisons
- Exemple : Le solitaire, Le Scrabble
  - Pour le scrable, utilisation d'un arbre lexicographique généralisé (cf. http ://lwh.free.fr/pages/algo/minmax/scrabble.htm)