

Fondements et programmation de l'intelligence artificielle

Deuxième année BIG DATA ET ANALYSE DE DONNÉES

Plan du cours

1. Introduction à l'intelligence artificielle
2. Formulation d'un problème
3. Résolution d'un problème par recherche aveugle
4. Résolution d'un problème par recherche heuristique
5. Jeux stratégiques et algorithme de recherche

ineskammoun@isimsf.u-sfax.tn

Introduction à l'intelligence artificielle

Introduction

Chapitre 1

Plan du chapitre

1. Introduction
2. C'est quoi l'IA ?
3. Système intelligent
4. Caractéristique de l'IA
5. Type de l'IA
6. Historique
7. Application de l'IA

Introduction

Lien

Question: Qu'est ce que l'intelligence artificielle ?

La réponse dépend de celui à qui on pose la question

Turing: Ce qui rend difficile la distinction entre une tâche réalisée par un être humain ou par une machine.

Darwin: Ce qui permet la survie de l'individu le plus apte, parfaitement adapté à son environnement.

Edison: Tout ce qui fait que cela fonctionne et produit le plus de revenus pour l'entreprise

C'est quoi l'IA ?

« Le but de l'intelligence artificielle est de construire des machines qui réalisent des choses qui requièrent de l'intelligence lorsqu'elles sont faites par des humains. » Boden

« L'IA est la partie de l'informatique concernée par la conception du système informatique intelligent. » Rich

« L'IA et le domaine de l'informatique qui étudie comment faire faire à l'ordinateur des tâches pour lesquelles l'homme est aujourd'hui encore le meilleur. » Feigenbaum

« C'est la science et l'ingénierie de la fabrication de machines intelligentes, en particulier de programmes informatiques intelligents. » McCarthy

C'est quoi l'IA ?

Microsoft: l'IA est la création de logiciels qui imitent les comportements et les capacités humaines.

Les principales charges de travail sont les suivantes :

- **Apprentissage automatique** - C'est souvent la base d'un système d'IA et c'est la façon dont nous « enseignons » à un modèle informatique à faire des prédictions et à tirer des conclusions à partir de données.
- **Détection d'anomalies** - La capacité de détecter automatiquement les erreurs ou les activités inhabituelles dans un système.

C'est quoi l'IA ?

- **Vision par ordinateur** - La capacité des logiciels à interpréter le monde visuellement à travers des caméras, des vidéos et des images.
- **Traitement du langage naturel** - La capacité d'un ordinateur à interpréter le langage écrit ou parlé, et à répondre de la même manière.
- **Exploration des connaissances** - La capacité d'extraire des informations à partir de grands volumes de données souvent non structurées pour créer un magasin de connaissances consultable.

Système intelligent

- La création d'un système intelligent est accomplie:
 - en **étudiant comment le cerveau humain pense** et comment les humains apprennent, décident et travaillent tout en essayant de résoudre un problème
 - puis **en utilisant les résultats de cette étude comme base** pour développer des logiciels intelligents.
- Plusieurs domaines peuvent contribuer à construire un système intelligent :
 - Informatique
 - Biologie
 - Psychologie
 - Linguistique
 - Mathématique
 - cognition

Système intelligent

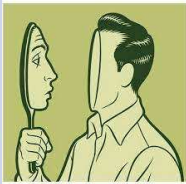

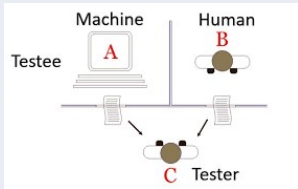

IA Est une science cognitive

- étude des mécanismes de l'intelligence
- l'ordinateur est utilisé comme moyen de simulation
- il tente à imiter le fonctionnement du cerveau humain

IA Est une Branche de l'IT

- rendre l'ordinateur plus habile et plus intelligent
- doter l'ordinateur des capacités de l'intelligence humaine
- consiste à émuler par un programme de comportements intelligents sans reproduire le fonctionnement de l'homme

Caractéristique de l'IA

	Comme un humain	Rationnellement (bon sens)
Penser	<p>Science cognitive</p> <ul style="list-style-type: none"> • Introspection • Expérimentations • Observation du comportement 	<p>Approche logique</p> <ul style="list-style-type: none"> • Traduire la logique (le bon sens humain) vers un code informatique 
Agir	<p>Test de turing</p> <ul style="list-style-type: none"> • Un expert humain pose des questions à une machine et à un humain 	<p>Agir pour atteindre un objectif</p> <ul style="list-style-type: none"> • Un expert informatique crée des actions tout en tenant compte des circonstances actuelles 

Caractéristique de l'IA

	Comme un humain	Rationnellement (bon sens)
Penser	<p>Science cognitive</p> <ul style="list-style-type: none">• Introspection• Expérimentations psychologiques• Observation du cerveau humain	<p>Approche logique</p> <ul style="list-style-type: none">• Traduire la logique (le fonctionnement de l'esprit humain) Vers un code qui résout un problème
Agir	<p>Test de turing</p> <ul style="list-style-type: none">• l'agent humain ne devrait idéalement pas être en mesure de conclure qu'il parle à une intelligence artificielle	<p>Agir pour atteindre un objectif</p> <ul style="list-style-type: none">• l'agent rationnel peut traduire ses pensées vers des actions tout en tenant compte des circonstances actuelles

Caractéristique de l'IA

- Manipule des informations symboliques
 - contrairement à l'informatique classique, ces informations représentent des concepts , des règles, des objets, des faits, ...
 - Exemple un système de diagnostic médical: Valeur numérique de la température 39,2 °C implique patient fiévreux
- Utilisation des méthodes heuristiques
 - différente des méthodes algorithmiques
 - méthode de résolution qui en présente des voies non déterministes (succès non garanti)
 - si elle marche : économie de temps et de calcul
 - en cas d'échec : elle revient en arrière et essaye une autre solution

Caractéristique de l'IA

- **Utilisation de l'information diverse**

- L'IA peut résoudre des problèmes quand les informations sont incomplètes ou inexactes
- Elle utilise pour cela des méthodes et des techniques de raisonnement (raisonnement approximatif, raisonnement non monotone) ou des méthodes statistiques

- **Représentation des connaissances**

- Un système général de résolution
- Un système de connaissance pour un domaine particulier, avec une grande quantité de données sur ce domaine
- communique l'expertise à la machine sous forme déclarative: base de connaissance

Type de l'IA

Intelligence artificielle étroite

- Machine learning
- Programmation par contraintes

Intelligence artificielle générale

- Machine intelligente

Super intelligence artificielle

- Machine consciente

Type de l'IA: Intelligence artificielle étroite IAE

- Il s'agit de la forme de l'intelligence artificielle la plus courante
- Ces systèmes d'IA sont conçus pour résoudre un seul problème et serait capables d'exécuter très bien une seule tâche
- Par définition ils ont des capacités étroites comme :
 - recommander un produit à un utilisateur de commerce électronique
 - prédire la météo
- Ils sont capables de se rapprocher du fonctionnement humain dans des contextes très spécifiques et même de les surpasser dans de nombreux cas.
- Mais il n'excellent que dans des environnements très contrôlés avec un ensemble limité de paramètres

Type de l'IA: Intelligence artificielle générale IAG

- Elle est définie comme une intelligence artificielle qui a une fonction cognitive de niveau humain dans une grande variété de domaines tels que :
 - le traitement du langage naturel ,
 - le traitement de l'image ,
 - le fonctionnement et le raisonnement informatique , etc
- Un système IAG devrait comprendre des milliers de systèmes IAE travaillant ensemble et communiquant les uns avec les autres.
- Imitent le raisonnement humain

Type de l'IA: Super Intelligence artificielle SIA

- C'est presque la science-fiction, mais SIA est considérée comme la progression logique de IAG.
- un système SIA serait capable de surpasser toutes les capacités humaines
- Cela inclurait
 - la prise de décision,
 - la prise de décision rationnelle,
 - établir des relations émotionnelles, etc
- Une fois que nous aurons maîtrisé l'IAG, les systèmes de d'IA pourraient rapidement améliorer leur capacités et progresser dans des domaines dont nous n'aurions peut-être même pas rêvé
- Le long voyage qui nous attend vers la SIA lui-même donne l'impression que cela ressemble à un concept qui s'étend loin dans le futur

Historique

50's: Naissance de l'IA

- Optimisme exagéré
- Echecs (Jeu d'échecs, reconnaissance vocale, etc.)

60's: véritable démarrage de l'IA

- Nombreux projets
- Des résultats significatifs
- Algorithmes heuristiques

70's: Explosion des travaux de l'IA

- Représentation des connaissances
- raisonnement
- système à base de connaissances
- compréhension du langage naturel
- robotique

80's: entrée de l'IA dans la vie économique

- réalisation pratique dans plusieurs domaines
- de nombreux travaux de recherche avec des projets ambitieux

90's: commercialisation de l'IA

- usage à domicile (aspirateur, Google Now, etc.)
- les assistants intelligents
- les jeux vidéo
- voiture autonome

Application de l'IA

- L'IA est utilisée dans différents domaines pour donner un aperçu du comportement des utilisateurs et donner des recommandations basées sur les données



L'algorithme de recherche prédictive de Google a utilisé les données des utilisateurs passés pour prédire ce que un utilisateur tapera ensuite dans la barre de recherche



Netflix utilise les données des utilisateurs passés pour recommander le film qu'un utilisateur pourrait vouloir voir ensuite ce qui rend l'utilisateur accroché à la plateforme et augmente le temps de visionnage



Facebook utilise les données passées des utilisateurs pour donner automatiquement des suggestions pour taguer vos amis en fonction de leurs caractéristiques faciales dans leurs images

Application de l'IA

- L'IA dans les soins de santé

- Les plus gros paris sont sur l'amélioration des résultats pour les patients et la réduction des coûts
- les entreprises appliquent l'apprentissage automatique pour établir les diagnostics meilleurs et plus rapides que les humains
- Utilisation des robots pour effectuer des opérations chirurgicales

- L'IA en entreprise

- l'automatisation robotique des processus est appliquée à des tâches hautement répétitives normalement effectuées par des humains
- des algorithmes d'apprentissage automatique sont intégrés aux plateformes d'analyse et de CRM pour découvrir les informations sur la façon de mieux servir les clients

Application de l'IA

- L'IA dans l'éducation

- L'IA peut automatiser la notation ce qui économise le temps aux enseignants
- L'IA peut évaluer les élèves et s'adapter à leurs besoins les aider à travailler à leur propre rythme s'assurer qu'ils restent sur la bonne voie
- L'IA pourrait changer où et comment les élèves apprennent peut-être même remplacer certains enseignants

- L'IA dans la fabrication

- c'est un domaine qui a été l'avant-garde de l'intégration des robots dont le flux de travail
- les robots industriels exécutaient des tâches uniques et étaient séparées des travailleurs humains mais à mesure que la technologie progressait cela a changé

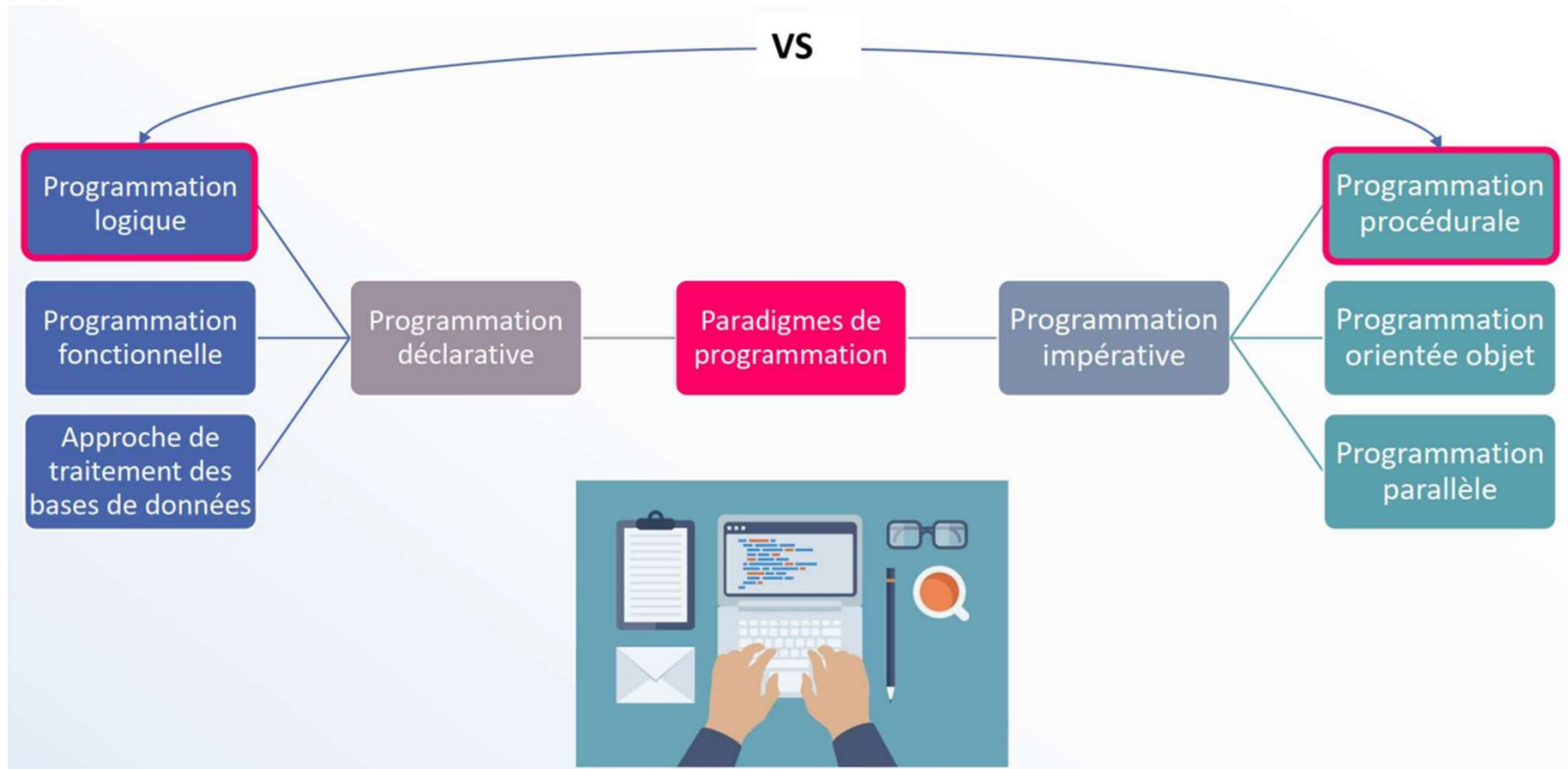
Formulation d'un problème

Chapitre 2

Plan du chapitre

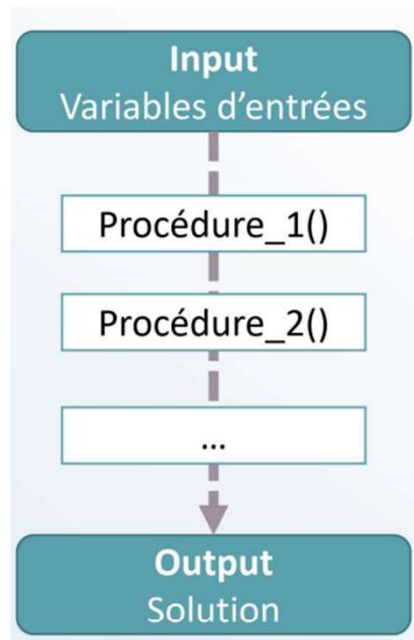
1. Paradigmes de programmation
2. Rappels de logique
 1. Calcul propositionnel
 1. Syntaxe
 2. Sémantique
 3. Inférence
 2. Calcul des prédicats
3. Langage PROLOG
 1. Présentation
 2. Premiers pas
 3. Base des connaissances
 4. Questions et réponses
 5. Exemples sur la récursivité
 6. Synthèse

Paradigmes de programmation

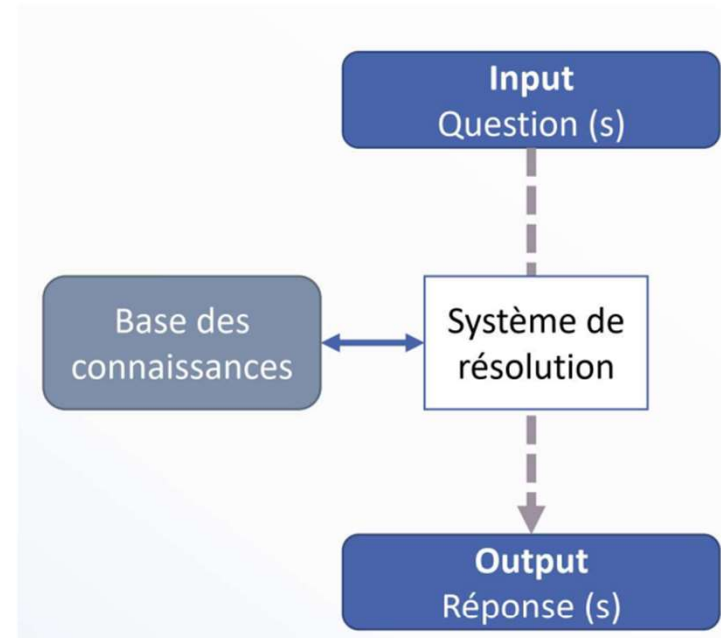


Paradigmes de programmation

Programmation procédurale

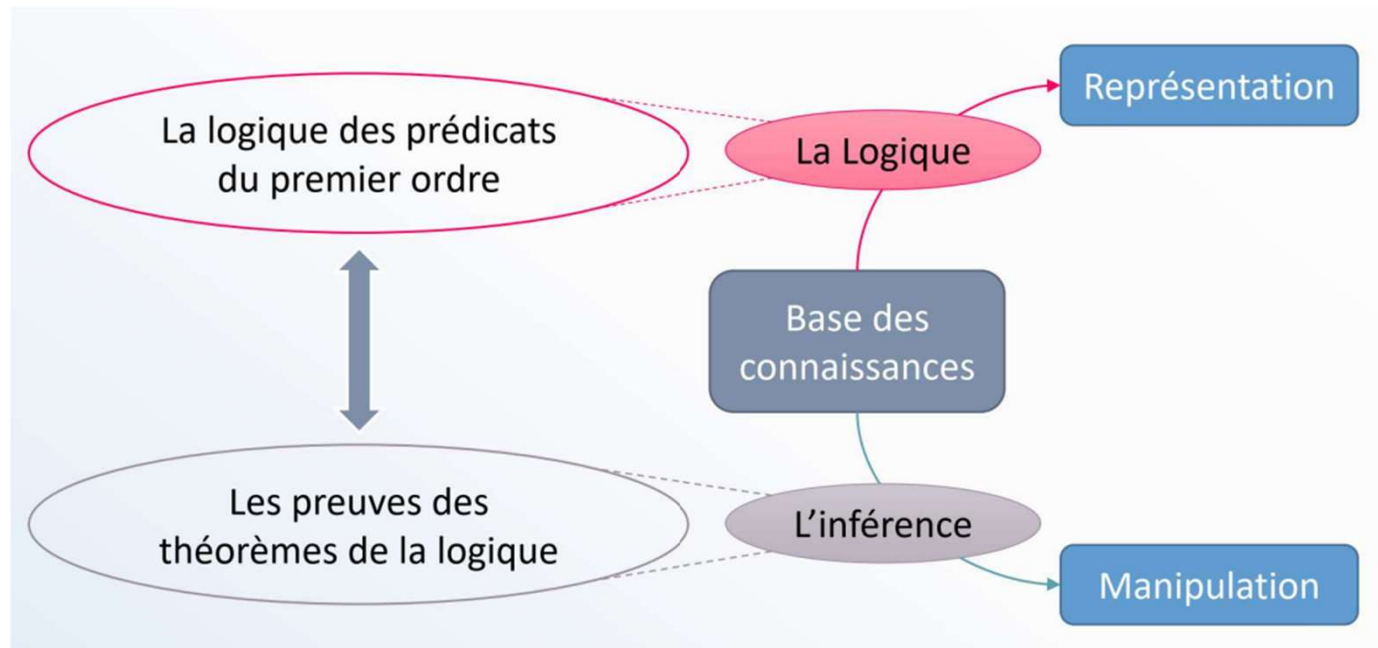


Programmation logique



Paradigmes de programmation

- La programmation logique est une méthode que les informaticiens utilisent pour essayer de permettre aux machines de raisonner car elle est utile pour la représentation des connaissances



Calcul propositionnel

- Le **calcul propositionnel** modélise la façon dont les mathématiciens raisonnent sur la vérité et la fausseté en faisant abstraction des propositions spécifiques qui forment le raisonnement correct
- La **logique propositionnelle** est:
 - un langage formel pour exprimer des connaissances
 - la forme la plus simple des logiques mathématiques
 - un langage symbolique pour décrire des propositions et de raisonner avec
- une **proposition** (ou **assertion** ou **formule**) et une phrase déclarative qui peut être soit vrai soit fausse (mais pas les deux)

Calcul propositionnel

Syntaxe

- **Vocabulaire**: un ensemble de symboles (dits propositionnels) et de connecteurs
- **Langage**: un ensemble de formules propositionnelles. Il est obtenu à partir des règles qui combinent ces symboles

Sémantique

- Sens associé aux symboles et connecteurs
- Tables de vérités

Inférence

- Ce sont des **règles** qui permettent de dériver de nouvelles propositions à partir des propositions considérées comme **vraies**

Calcul propositionnel: Syntaxe

Une formule (ou bien une proposition) consiste aux éléments suivants :

- Une **constante** qui possède soit la valeur **faux** \perp soit la valeur **vrai** \top
- Un ensemble de **variables propositionnelles** noté $P = \{p, q, r, x, \dots\}$

Une variable propositionnelle est un symbole pouvant avoir la valeur faux ou vraie

- Un ensemble de **connecteurs logiques** noté $C = \{\neg, \wedge, \vee, \supset, \rightarrow, \leftrightarrow\}$ permettant de connecter des variables propositionnelles et ou des constantes:

\neg négation

\wedge conjonction

\vee disjonction

\supset, \rightarrow implication $a \rightarrow b$ est équivalent à $\neg a \vee b$

\leftrightarrow équivalence logique

Calcul propositionnel: Syntaxe

- x est une formule si $x \in P$ ou bien x est une constante
- $\neg(x)$ est une formule si x est une formule
- $(x)\Delta (y)$ est une formule si x et y sont des formules et si $\Delta \in \mathcal{C}$
- Exemple de formules:
 - $x \wedge y$
 - $x \vee y$
 - $x \supset y$

Calcul propositionnel: Sémantique

- **Interprétation:** Les formules sont interprétées dans {vrai, faux}

On note $\delta(p)$ la valeur de vérité de la variable p

$$\delta(p) \in \{\perp, \top\}$$

On définit l'interprétation associée à chaque connecteur grâce aux tables de vérité

a	b	$\neg a$	$a \wedge b$	$a \vee b$	$a \rightarrow b$	$a \leftrightarrow b$
\top	\top	\perp	\top	\top	\top	\top
\top	\perp	\perp	\perp	\top	\perp	\perp
\perp	\top	\top	\perp	\top	\top	\perp
\perp	\perp	\top	\perp	\perp	\top	\top

Calcul propositionnel: Sémantique

- **Tautologie**: ce sont des formules toujours vraies, sa table de vérité ne contient que des T. On l'appelle aussi une formule **valide**
- Exemple: $p \vee \neg p$

p	$\neg p$	$p \vee \neg p$
\perp	T	T
T	\perp	T

Calcul propositionnel: Sémantique

La formule suivante est-elle une tautologie ?

$$F = \neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$$

\wedge	\perp	\top
\perp	\perp	\perp
\top	\perp	\top

\vee	\perp	\top
\perp	\perp	\top
\top	\top	\top

\leftrightarrow	\perp	\top
\perp	\top	\perp
\top	\perp	\top

p	q	$\neg p$	$\neg q$	$(p \wedge q)$	$\neg(p \wedge q)$	$(\neg p \vee \neg q)$	F
\perp	\perp	\top	\top	\perp	\top	\top	\top
\perp	\top	\top	\perp	\perp	\top	\top	\top
\top	\perp	\perp	\top	\perp	\top	\top	\top
\top	\top	\perp	\perp	\top	\perp	\perp	\top

Cette formule est **valide**, c'est une **tautologie**

Calcul propositionnel: Sémantique

- **Formules consistantes**: Une formule est dite consistante s'il existe une interprétation dans la quelle elle est vraie: $\exists \delta, \delta(p) = T$
- **Formules inconsistantes**: Une formule est dite inconsistante si elle est toujours fausse: la table de vérité ne contient que des \perp . Exemple:
 $p \wedge \neg p$
- **Formules tautologiquement équivalentes**: sont les formules qui ont les mêmes tables de vérités

$$p =_{tauto} q \text{ noté également } p \equiv q \text{ si } \forall \delta, \delta(p) = \delta(q)$$

Calcul propositionnel: Sémantique

Loi de Morgan:

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

Calcul propositionnel: inférence

- Une base de connaissances modélise un problème du monde réel
- Elle exprime les connaissances disponibles sur un problème donné
- Elle est composée d'un ensemble de formules propositionnelles



Calcul propositionnel: inférence

- Quelles sont les bonnes pratiques pour bien composer une base de connaissances?
 - **Littéral** : un atome ou la négation d'un atome.
 - **Clause** : une disjonction de littéraux.
 - **Cube** : une conjonction de littéraux.
 - **CNF** : forme normale conjonction, une conjonction de clauses.
 - **DNF** : forme normale disjonctive, une disjonction de cubes.

$$(p \wedge (\neg p \wedge q)) \vee ((r \vee \neg q) \wedge (\neg p \vee q))$$

DNF

CNF

Calcul propositionnel: formes normales

Toute formule peut être mise, de manière équivalente, sous la forme de CNF ou de DNF

Forme Normale Disjonctive (DNF)	Forme Normale Conjonctive (CNF)
<p>C'est une disjonction de conjonctions de variables propositionnelles et leurs négations</p> <p>Exemple:</p> $(a \wedge b) \vee (\neg a \wedge b) \vee (a \wedge \neg b)$	<p>C'est une conjonction de disjonctions de variables propositionnelles et leurs négations</p> <p>Exemple:</p> $(a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b)$

But:

- Avoir une représentation uniforme des formules du calcul prépositionnel
- Limiter le nombre des différents connecteurs utilisés
- Limiter l'allure des formules rencontrées

Calcul propositionnel: propriétés des connecteurs logiques

Négation	$\neg (\neg p) \equiv p$	$\neg p \vee p \equiv \text{T}$	$\neg p \wedge p \equiv \perp$
	ET logique « \wedge »	OU logique « \vee »	
Associativité	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$	
Distributivité	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	
Élément neutre	$p \wedge \text{T} \equiv p$	$p \vee \perp \equiv p$	
Élément absorbant	$p \wedge \perp \equiv \perp$	$p \vee \text{T} \equiv \text{T}$	
Implication matérielle	$p \rightarrow q \equiv \neg(p \vee q)$		
Équivalence logique	$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$		

Calcul propositionnel: limites

- La logique propositionnelle ne parle que de vérité, et elle ne permet pas de faire référence à des objets ou à des notions
- Comment modéliser ?
 - Les lampes sont faites pour éclairer
 - Quelques lampes éclairent très mal
 - Quelques objets, qui sont faits pour éclairer, le font très mal
- La **logique du premier ordre** (logique des **prédicats**) est un langage qui permet de dépasser ces limites

Calcul des prédicats: exemple

- Comment modéliser ?
 - Les lampes sont faites pour éclairer

$$\forall x, \text{lampe}(x) \rightarrow \text{éclaire}(x)$$

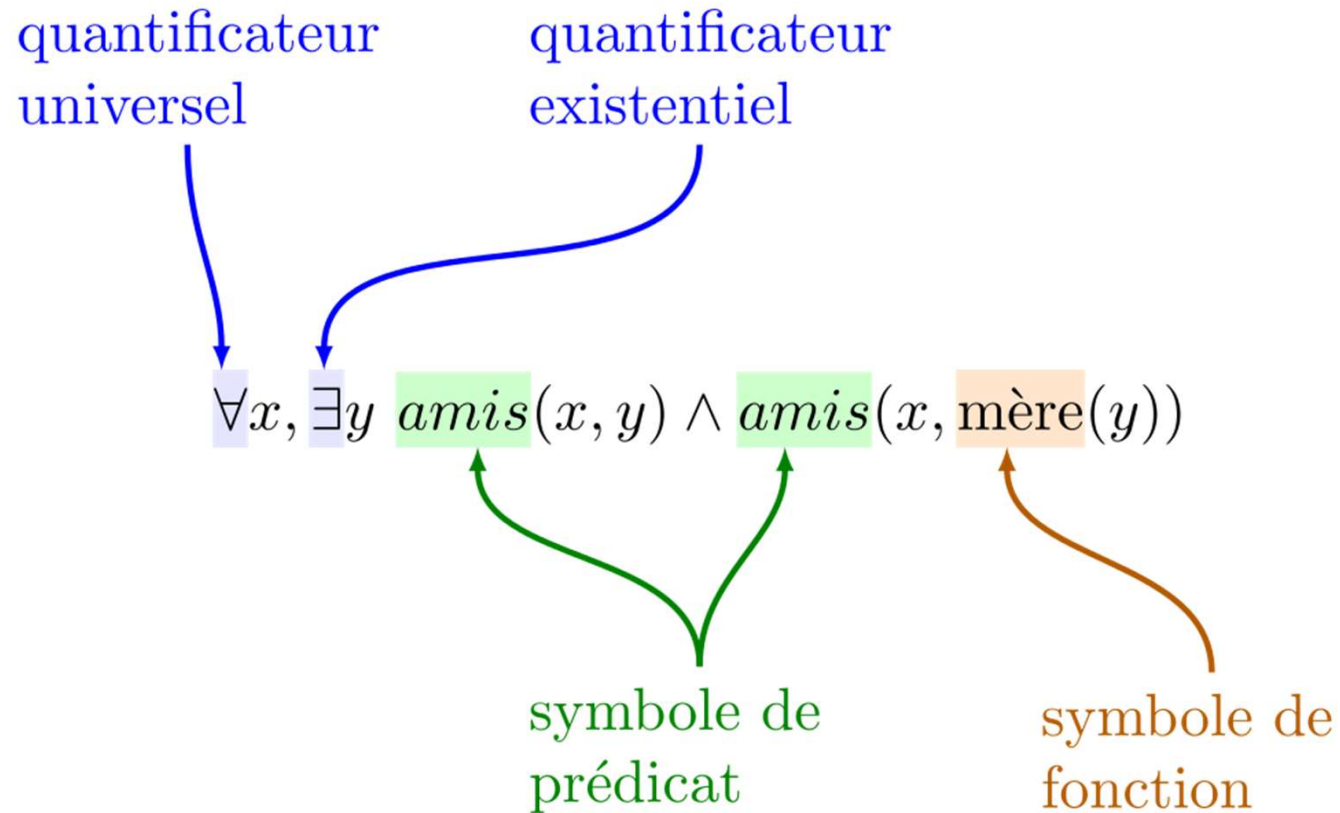
- Quelques lampes éclairent très mal

$$\exists x, \text{lampe}(x) \wedge \text{éclaireMal}(x)$$

- Quelques objets, qui sont faits pour éclairer, le font très mal

$$\exists x, \text{éclaire}(x) \wedge \text{éclaireMal}(x)$$

Calcul des prédicats: exemple



Calcul des prédicats: syntaxe

- Connecteurs: $\neg, \wedge, \vee, \rightarrow$ et \leftrightarrow
- Qualificateurs: \forall, \exists
- Variables: x, y, p, q, a, b
- Prédicats (relations): $R, S, \text{éclaire}, \text{lampe}, \text{mère}, \text{amis}$

Calcul des prédicats: définitions

- Proposition: est un énoncé qui est soit vrai soit faux

Exemple:

- Ali est le père de Saleh
- Ali est heureux

- Prédicat: est une fonction booléenne

Exemple

- Père(X,Y)= vrai, si X est le père de Y, faux sinon
- Heureux(X)= vrai si X est heureux, faux sinon

Calcul des prédicats: exemples

Tout est relatif:

Une porte est soit ouverte, soit fermée:

$$\forall x, porte(x) \rightarrow ouvert(x) \vee fermé(x)$$

Tout ce qui brille n'est pas or

$$\forall x, brille(x) \rightarrow \neg or(x)$$

Tout ce qui brille n'est pas nécessairement or

$$\exists x, brille(x) \rightarrow \neg or(x)$$

Calcul des prédicats: exercices

Traduisez les propositions suivantes vers des prédicats:

- Tous les chemins mènent à Rome

$$\forall x, \text{chemin}(x) \rightarrow \text{mène_à}(x, \text{Rome})$$

- Les amis d'un ami sont mes amis

$$\forall x, \forall y, \text{amis}(\text{self}, x) \wedge \text{amis}(x, y) \rightarrow \text{amis}(\text{self}, y)$$

- Pour tout entier, il existe un plus grand entier

$$\forall x, \text{entier}(x) \rightarrow \exists y, \text{entier}(y) \wedge \text{plus_grand}(y, x)$$

Calcul des prédicats: définitions

Un **fait** est un prédicat évalué en une valeur

Exemple:

Père(Ali, Saleh)

Heureux(Ali)

Mère(Ali, Mariem)

Atomes

Père, Heureux,
Mère

Termes

Ali, Saleh, Mariem

Calcul des prédicats: définitions

Termes: des objets manipulés par un programme logique. Ce sont les données du programme.

On distingue trois types de termes:

1. **Les variables:** représentent des objets inconnus de l'univers: x, y, a, b
2. **Les termes élémentaires:** ou termes atomiques: représentant les objets simples connus de l'univers. Exemple: les nombres, les identificateurs, les chaînes de caractères
3. **Les termes composés:** représentent les objets composés (structurés) de l'univers

Exemple:

`Habite(Ali, adresse(8, "rue de la paix", Sfax))`

Calcul des prédicats: définitions

Atome logique: exprime une relation entre les termes qui peut être vraie ou fausse

Syntaxiquement:

$$atome_relation(t_1, t_2, \dots, t_n)$$

Le nombre d'arguments n est appelé **arité de l'atome logique**

Littéral: est un atome (positif) ou la négation d'un atome(négatif)

Clause: est une formule qui a la forme d'une disjonction de littéraux

Exemple: $p(x, y) \vee \neg q(z)$

Une clause concrète: est une clause sans variables

Une clause de Horn: est une clause qui comporte **au plus un littéral positif**

Remarque: On peut toujours transformer une formule en un ensemble (conjonction) de clauses

Langage PROLOG

Exercice

On considère une base de faits décrivant des vols d'une compagnie aérienne

`vol(numero du vol,ville de depart,ville d'arrivee,heure de depart,heure d'arrivee,nombre de passagers)`

Décrire les vols suivants:

Un vol qui part de brest à 14h pour arriver à paris à 15h transportant 200 voyageurs

`vol (1 , brest , paris ,1400 ,1500 ,200).`

Un vol qui part de brest à 6h pour arriver à lyon à 8h transportant 100 voyageurs

`vol (2 , brest , lyon ,0600 ,0800 ,100).`

Un vol qui part de brest à 6h30 pour arriver à londre à 8h transportant 75 voyageurs

`vol (3 , brest , londres ,0630 ,0800 ,75).`

Un vol qui part de lyon à 12h pour arriver à paris à 13h transportant 250 voyageurs

`vol (4 , lyon , paris ,1200 ,1300 ,250).`

Un vol qui part de lyon à 13h pour arriver à paris à 14h transportant 250 voyageurs

`vol (5 , lyon , paris ,1300 ,1400 ,250).`

Quelles questions Prolog doit-on poser pour déterminer la liste des

Question	Réponse
au départ de la ville de brest ?	?- vol(N,brest,_,_,_,_).
qui arrivent dans la ville paris ?	?- vol(N,_,paris,_,_,_).
au départ de brest et qui partent avant midi ?	?- vol(N,brest,_,H,_,_), H =< 1200.
arrivant dans paris à partir de 14 h ?	?- vol(N,_,paris,_,H,_,_), H > 1400.
qui ont plus de 100 passagers et qui arrivent dans paris avant 17 h ?	?- vol(N,_,paris,_,H,P), H < 1700, P > 100.
qui partent à la même heure de deux villes différentes ?	?- vol(N1,V1,_,H,_,_), vol(N2,V2,_,H,_,_), V1 \== V2.
qui durent plus de deux heures ?	?- vol(N,_,_,D,A,_,_), A-D > 0200.

Exercice 2

On considère un réseau routier décrit par la base de faits suivante :

route(s,a). % une route relie la ville s à la ville a
route(s,d).
route(a,b).
route(a,s).
route(b,c).
route(b,e).
route(d,e).
route(c,b).

1. Représenter le réseau directionnel ainsi défini
2. Définir le prédicat voisins(X,Y) %s'il existe une route qui relie X et Y dans les deux sens%
3. Que doit-on avoir comme réponses aux questions suivantes:
 1. ?- voisins(s,a).
 2. ?- route(d,e).
 3. ?- route(a,X)

Résolution de problème en IA

Chapitre 3

Plan du chapitre

1. Introduction
2. Résolution des problèmes en IA
 1. Principe
 2. General Problem Solver
 3. Types de problèmes
 4. Méthodes de résolution de problèmes
3. Représentation et résolution d'un problèmes
 1. Description d'un problème
 2. Principe et représentation graphique
 3. Exercices et solutions
 4. Défis
 5. Évaluation des algorithmes de recherche des solutions

Introduction: IA et résolution des problèmes

Humain	Machine
Basé sur des règles et des stratégies	Une grande capacité de calcul et de mémorisation

- Pour certains problèmes, le nombre des possibilités de résolution est très grand
- La machine ne peut pas supporter un nombre gigantesque de possibilités
- On a intérêt de doter la machine par les règles et les stratégies de l'Homme

L'IA s'attaque aux problèmes que l'Homme ne sait pas résoudre facilement

Résolution des problèmes: principe

Considérer qu'il existe des méthodes générales permettant de résoudre n'importe quel type de problème

- L'algorithme doit donc être **neutre** sur le domaine concerné
- Les connaissances de description du problème et de sa résolution doivent être clairement **séparés de l'algorithme**
- Malgré l'ambition unificatrice visée , il existe **différentes démarches** pour aborder la résolution d'un problème

Résolution des problèmes: General Problem Solver

Un solveur général de problème a été construit
en 1959 par Newel et Simon

- **But** : modéliser l'activité humaine de résolution de problèmes
 - **Idée** : essayez de réduire les différences entre la situation courante et le but final
-
- l'influence de ce système est très grande sur les recherches en IA
 - le solveur était très ambitieux mais en pratique il s'est limité à la résolution d'énigmes



Résolution des problèmes: Types de problèmes

Satisfaction des contraintes

Contrainte: relation logique entre des variables

- On connaît l'état de départ
- Peu importe le chemin
- On cherche un état final respectant les contraintes

Exemples:

- Sudoku
- Enigme
- Séquençage ADN
- Rendu monnaie

Planification

- On connaît l'état de départ
- On connaît l'état final
- On cherche un chemin selon certains objectifs

Exemples:

- Emploi du temps
- Ordonnancement
- Transport
- Solitaire

Résolution des problèmes: Description d'un problème

La résolution d'un problème peut être décrite comme suit:



Il peut y avoir plusieurs états finaux satisfaisants et permettant de considérer le problème résolu

Résolution des problèmes: Principe et représentation graphique

Pour résoudre un problème il suffit de:

- Considérer l'état initial
- Considérer les opérations qui permettent de changer l'état du monde
- Tenter ces opérations de manière systématique
- Tester si on arrive à un état final satisfaisant

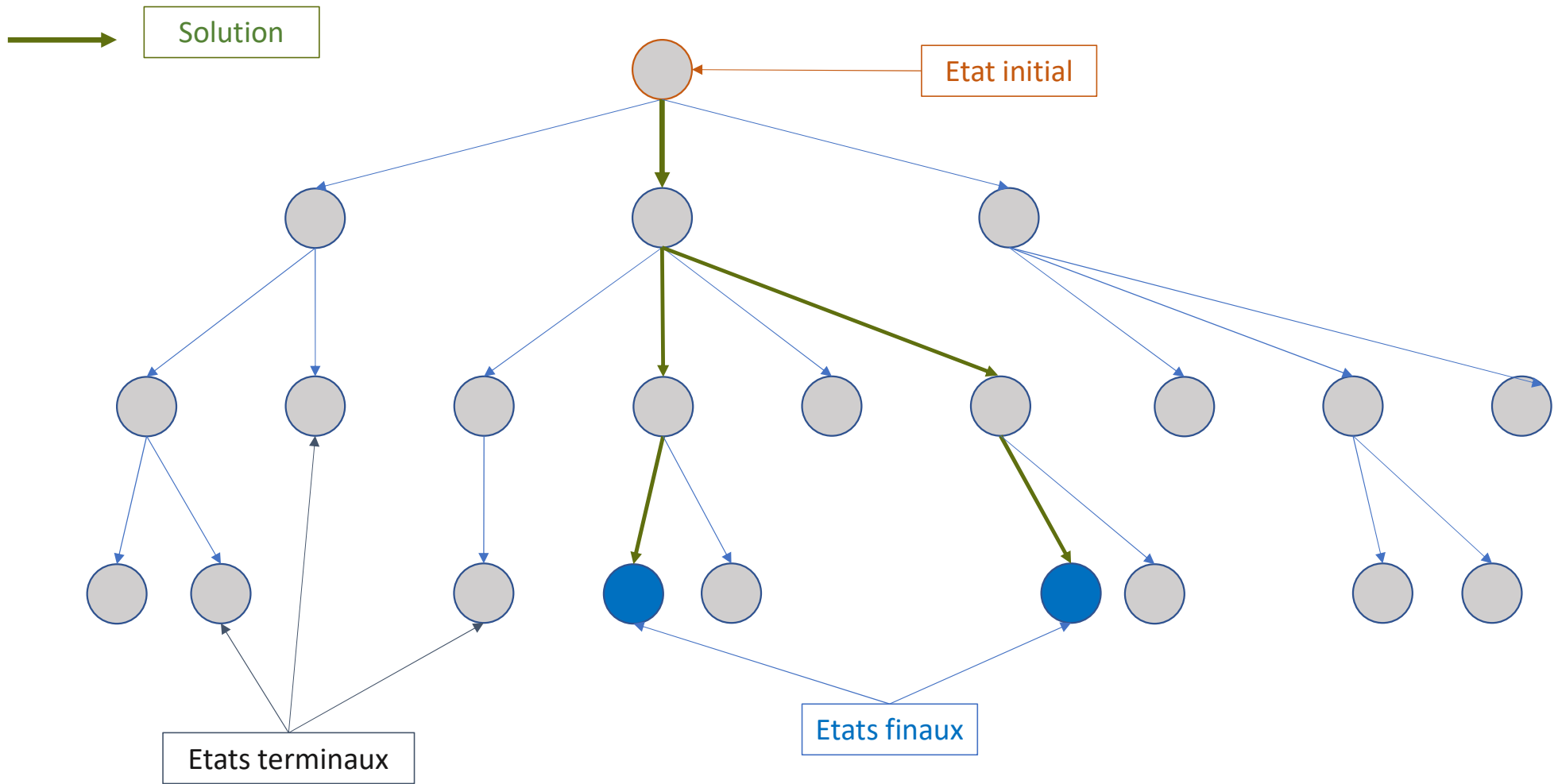
Le système de résolution est modélisé en une structure graphique comportant:

- Un ensemble de structures de données organisées en un graphe ou arbre
- Un ensemble d'opérations caractérisées par leurs conditions d'application et leurs actions
- Une structure de contrôle pour la stratégie de résolution

Un espace de résolution du problème est un graphe d'états

La solution sera un chemin d'un nœud départ vers un nœud but

Résolution des problèmes: Principe et représentation graphique



Représentation et résolution d'un problème: Notations

Un problème est représenté par (I,O,B) avec:

I: ensemble des situations initiales

O: ensembles des opérations qui permettent de changer d'un état à un autre

B: ensemble des situations buts

Résolution d'un problème:

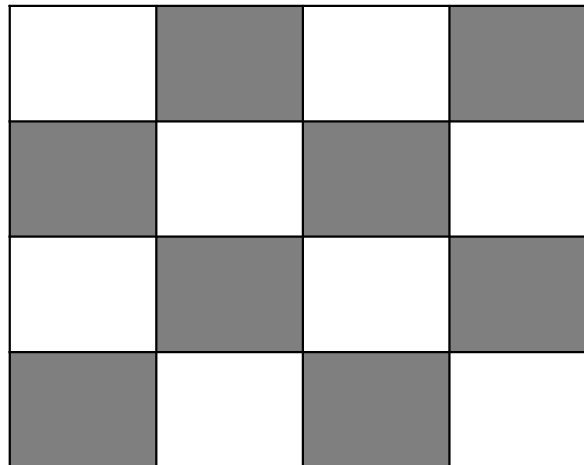
Séquence finie d'opérations permettant d'aller de I à B

Exercice 1: Problème des 4 reines

Il s'agit de placer 4 reines sur un échiquier comportant 4 lignes et 4 colonnes, de manière à ce qu'aucune reine ne soit en prise.

On rappelle que 2 reines sont en prise si elles se trouvent sur une même diagonale, une même ligne ou une même colonne de l'échiquier.

1. Définir le triplet (I,O,B) de ce problème
2. Présenter le graphe de résolution de ce problème

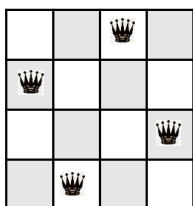
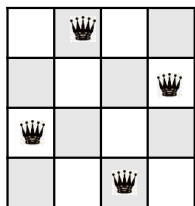
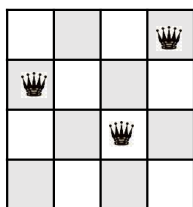
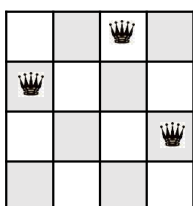
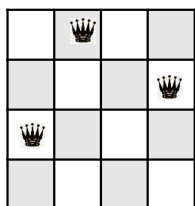
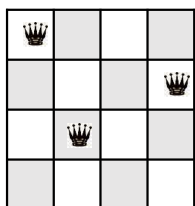
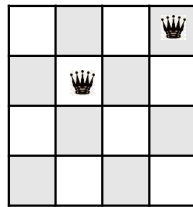
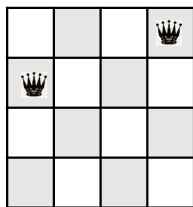
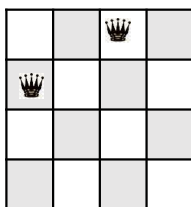
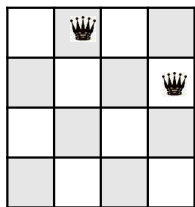
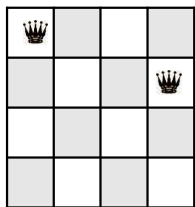
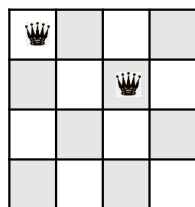
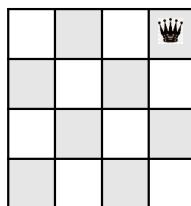
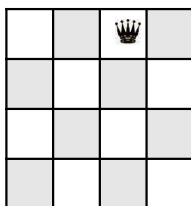
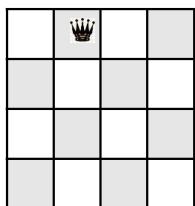
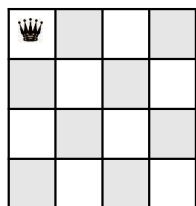
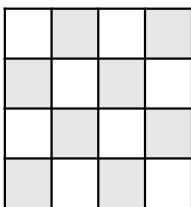


Exercice 1: Problème des 4 reines, solution

I : échiquier vide

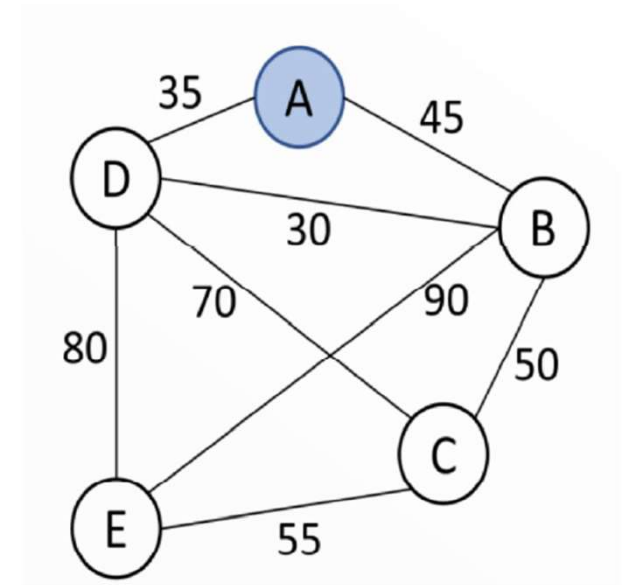
B: les 4 reines sont placées de façon qu'aucune d'entre elle ne menace l'autre

O: opération de placer une reine



Exercice 2: Problème du voyageur de commerce

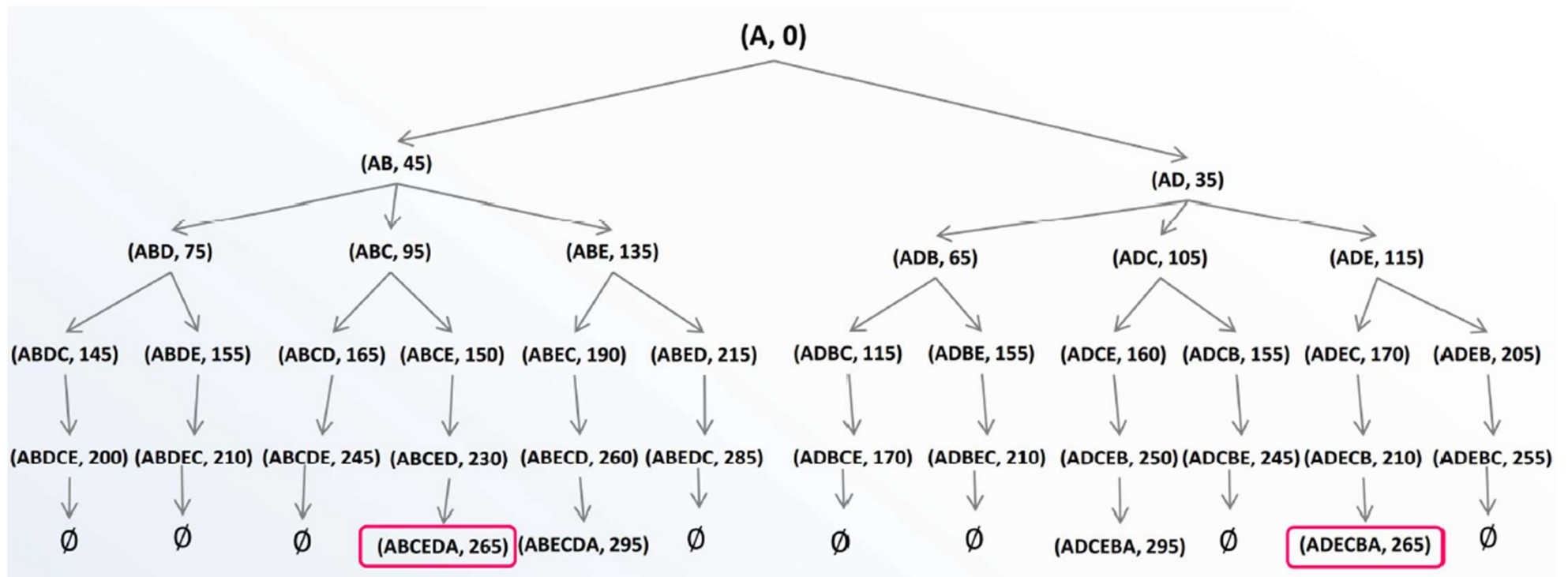
- Le problème de voyageur de commerce consiste à partir d'une ville A et revenir à la même ville en passant par certains nombres de villes B, C, D et E sans repasser 2 fois par la même ville
- on considère la carte des villes indiquant les distances entre les différentes villes à visiter,
- L'objectif est de trouver un chemin partant et revenant à la même ville A



1. Définir le triplet(I,O,B) de ce problème
2. Présenter le graphe de résolution de ce problème
3. Trouver le trajet qu'il doit effectuer en minimisant la distance parcourue

Solution

- I: Initialement, le voyageur se trouve à la ville A
 $I = \{A\}$
- Le but consiste à visiter toutes les villes en revenant à la ville A
 $B = \{A, x_1, x_2, x_3, x_4, A\}$ tels que x_1, x_2, x_3, x_4 représentent les villes B, C, D et E
- L'opération O est de passer d'une ville à une autre



Chemin: (ABCEDA) ou bien (ADECBA)

Coût: 265

Exercice 3: Problème de la traversée de la rivière

- Un fermier avec son loup son mouton et son chou veulent traverser la rivière
 - les 4 se trouvent sur la rive gauche de la rivière
 - le bateau peut transporter le fermier et un des 3 autres d'une rive à l'autre
 - si le loup reste seul avec le mouton il va le manger
 - si le mouton reste seul avec le chou il va le manger
1. Définir le triplet(I,O,B) de ce problème
 2. Présenter le graphe de résolution de ce problème
 3. Déterminer la séquence des traversées pour que les quarts soient sur la rive droite



Solution

- On représente les entités ainsi:
F -> Fermier , M -> Mouton, L -> Loup, C-> Chou
- On distingue deux rives: D,G
- Sur chaque rive, il peut y avoir:
 - Au max les 4 entités: $D=\{F,M,L,C\}$ ou $G=\{F,M,L,C\}$
 - Au mini aucune entités: $D=\{\}$ ou $G=\{\}$

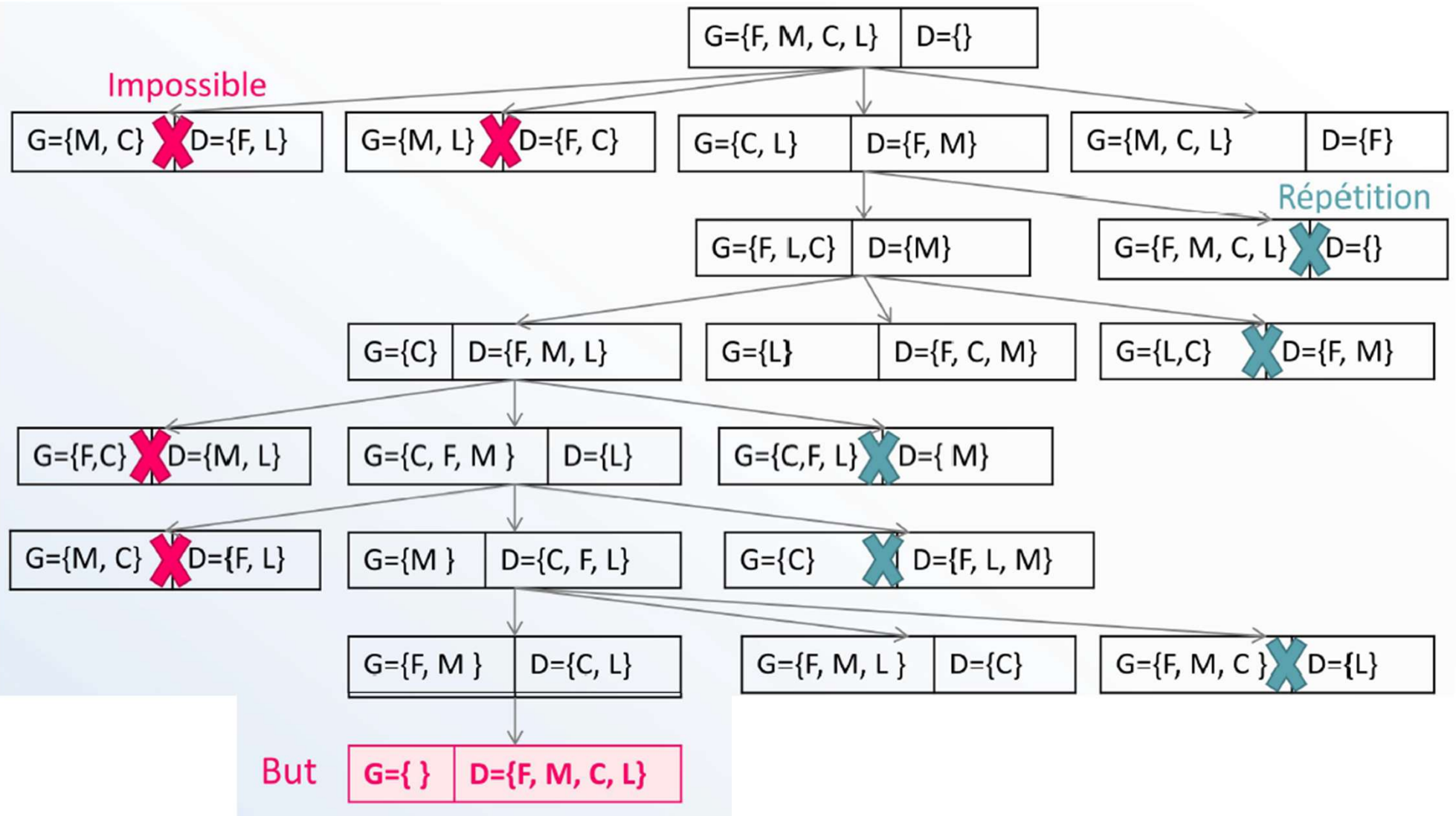
- Initialement, les quatre entités se trouvent sur la rive gauche

$$I = \{G = \{F, M, L, C\}, D = \{\} \}$$

- A la fin, les quatre entités se trouvent sur la rive droite

$$I = \{G = \{\}, D = \{F, M, L, C\} \}$$

- $O = \{O_1, O_2, O_3, O_4\}$
- O_i : le fermier amène l'un des trois entités d'une rive à une autre



- La solution du problème est alors:
 1. Le fermier conduit le mouton à la rive droite
 2. Le fermier revient tout seul à la rive gauche
 3. Le fermier conduit le loup sur la rive droite
 4. Le fermier ramène le mouton sur la rive gauche
 5. Le fermier conduit le chou vers la rive droite
 6. Le fermier ramène le mouton sur la rive droite

Augmentation de solution partielle

[Generate and Test]

- ☐ On commence la résolution en utilisant un opérateur valide pour l'état initial
- ☐ On continue à le faire à partir de l'état résultant
- ☐ Retour en arrière en cas des états terminaux ou des états déjà explorés
- ☐ Si plus aucun choix d'opérateur n'est possible alors il n'y a pas de solution

Application récursive

[Divide and Solve]

- ☐ On décompose le problème en sous-problèmes jusqu'à arriver à des sous-problèmes « triviaux »
- ☐ On résout ces problèmes par la méthode précédente
- ☐ On remonte l'arbre de décomposition jusqu'à obtenir la solution au problème complet

Complexité en temps

- Combien de temps prend l'algorithme pour trouver la solution ?

Complexité en espace

- Combien de mémoire est utilisé lors de la recherche d'une solution ?

Complétude

- Est-ce que l'algorithme trouve toujours une solution s'il y en a une ?

Optimalité

- Est-ce que l'algorithme renvoie toujours des solutions optimales ?

Résolution d'un problème par recherche aveugle

Chapitre 4

1. Méthodes non informées
2. Recherche en largeur
3. Recherche en profondeur
4. Recherche en profondeur limitée
5. Recherche par approfondissement itératif
6. Recherche par cout uniforme
7. Recherche bidirectionnelle
8. Exercices

Méthodes non-informée

- La recherche non informée ne contient aucune connaissance du domaine tel que la proximité l'emplacement de l'objectif.
- la recherche non informée applique une stratégie sans aucune information sur :
 - l'espace de recherche
 - les opérateurs d'état initial
 - le test de l'objectif
- la recherche non informée examine chaque état de l'arbre et elle génère les états successeurs des états déjà explorés jusqu'à ce qu'elle atteigne l'état but
- il s'agit d'une recherche aveugle

Méthode non informée: algorithme général

Fonction recherche_générale(problème, stratégie)

Input: un problème (représenté sous la forme d'un arbre ou graphe), une stratégie

Output: solution ou échec

1. Initialiser l'arbre pour le graphe de recherche par l'état initial du problème
2. **Boucle**
3. **Si** (il n'y a pas de candidat pour l'expansion) **alors**
4. retourner **échec**
5. **Sinon**
6. Choisir un nœud (un état) pour l'expansion en fonction de la stratégie
7. **Si** (le nœud contient un état final) **alors**
8. Retourner solution
9. **sinon**
10. étendre le nœud
11. ajouter les nœuds fils dans l'arbre ou le graphe
12. **Fin**

Méthodes non informée: algorithmes dérivés

Méthodes aveugles (non
informées)

Recherche en largeur

Recherche en profondeur

Recherche en profondeur limitée

Recherche par approfondissement itératif

Recherche en coût uniforme

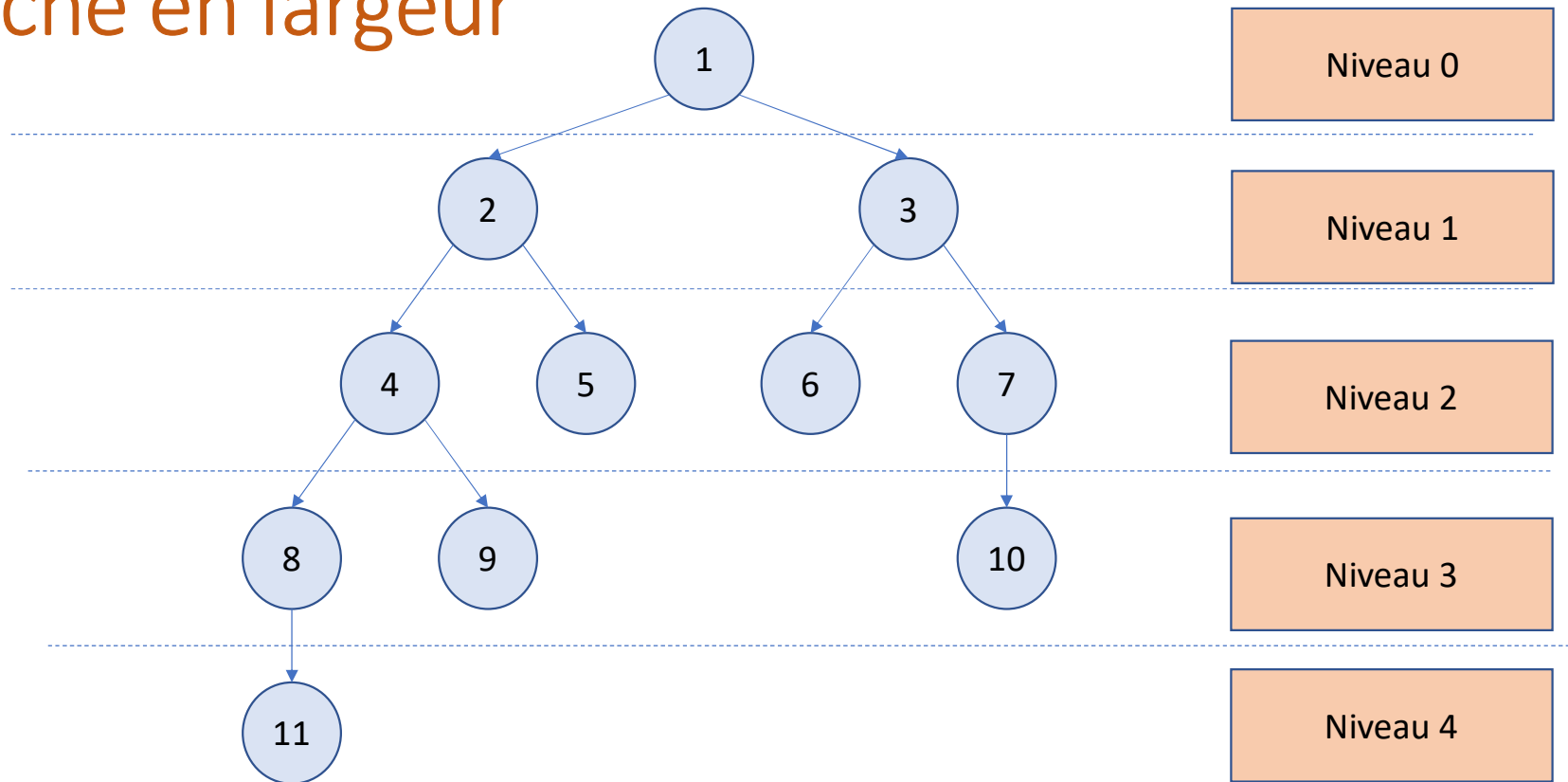
- La recherche non informée est une classe d'algorithmes de recherche à usage général qui fonctionne de force brute
- les méthodes de recherche diffèrent les unes des autres selon l'ordre dans lequel les nœuds sont explorés : différentes stratégies

Recherche en largeur

Recherche en largeur

- **Stratégie**: L'algorithme de recherche en largeur commence la recherche à partir d'une racine de l'arbre et développe tous les nœuds successeurs au niveau actuel avant de passer aux nœuds du niveau suivant.
- **File d'attente** : recherche en largeur et implémentée à l'aide de la structure de données de la file d'attente fifo

Recherche en largeur



Recherche en largeur: exemple Problème de navigation d'un robot

Il s'agit de déterminer le meilleur chemin qui permet d'atteindre un lieu donnée,

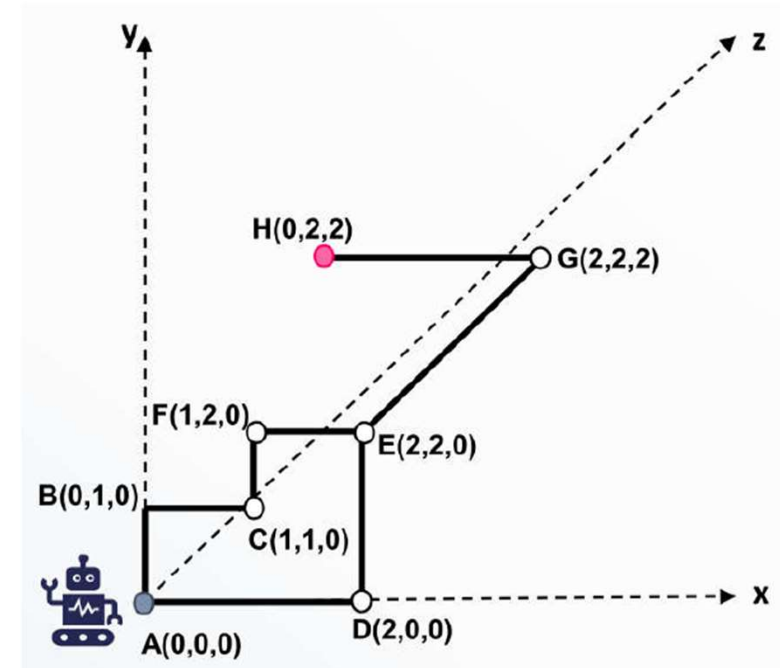
Dans notre cas, le robot veut se rendre de A à H

1. Représenter le graphe d'états de ce problème en étiquetant le passage d'un point à un autre par la distance entre ces deux points

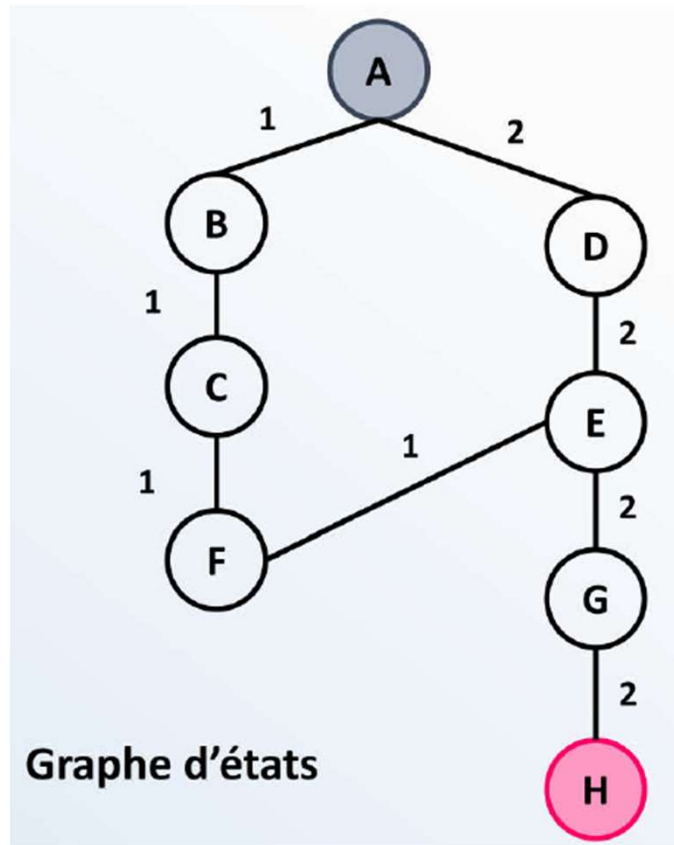
Nous proposons d'utiliser la distance euclidienne:

$$d(A, b) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$$

2. Appliquer la méthode de recherche en largeur pour atteindre le point H

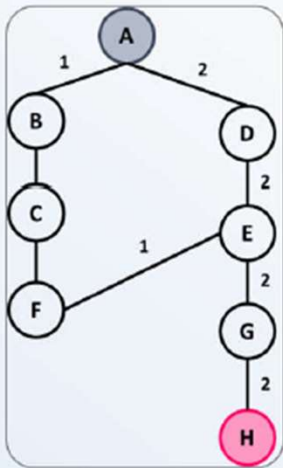


Graphe d'état

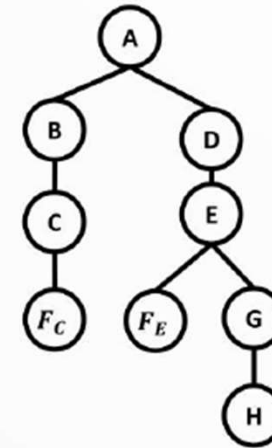


Recherche en largeur

- **Stratégie** : étendre le nœud le **moins profond**
- **Implémentation** : insertion des successeurs à **la fin** de la file d'attente



Nœud étendu	Nœuds à étendre
	{A}
A	{B, D}
B	{D, C}
D	{C, E}
C	{E, F _C }
E	{F _C , F _E , G}
F _C	{F _E , G}
F _E	{G}
G	{H}
H	→ But

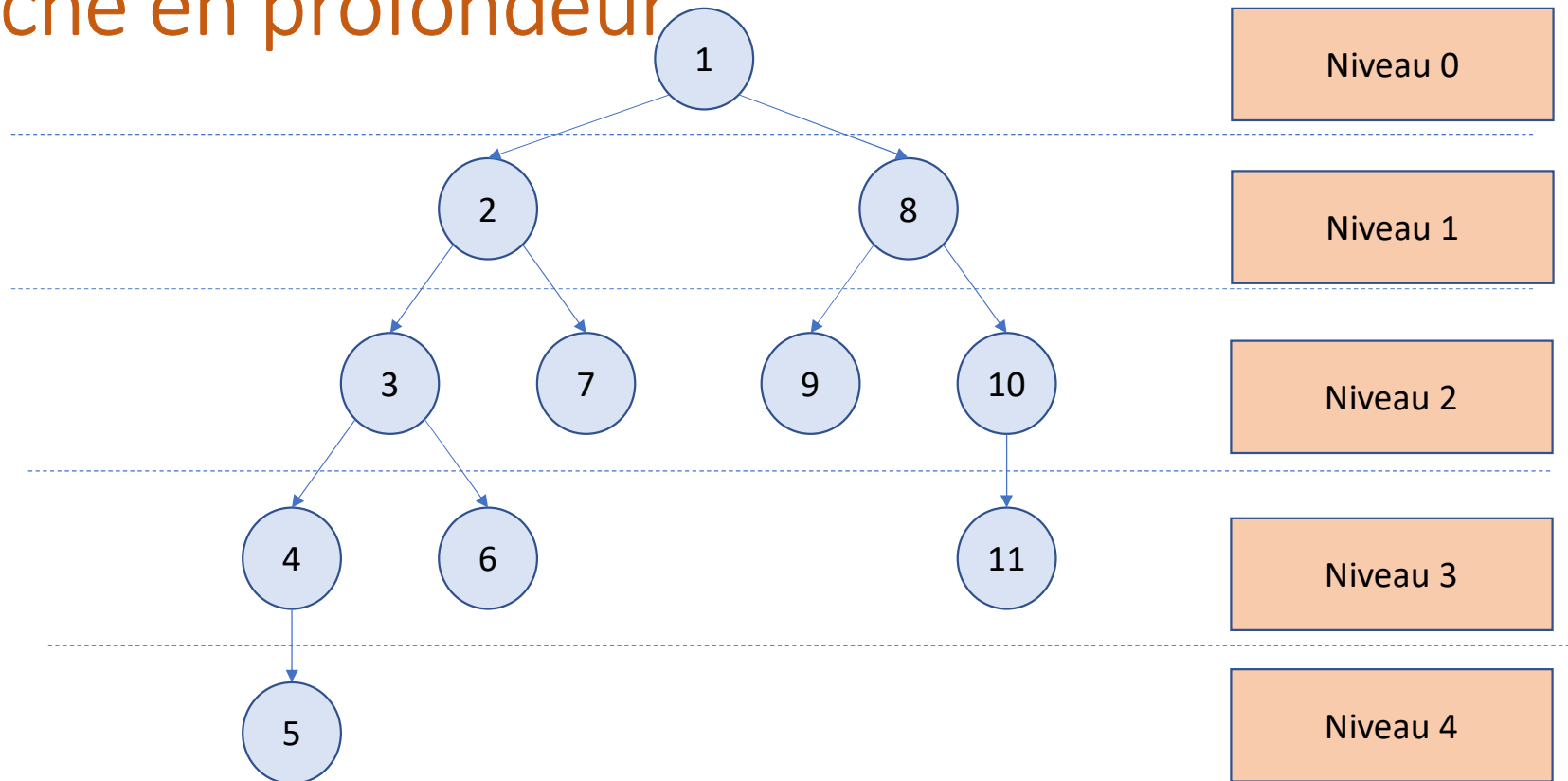


- **Chemin** : A D E G H
- **Nœuds testés** : 9
- **Nœuds étendus** : 6

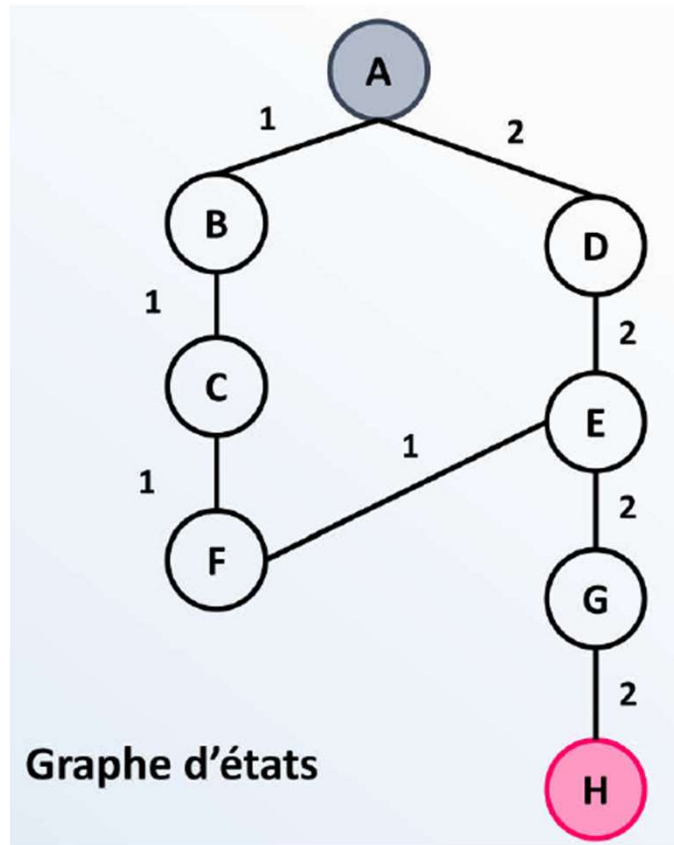
Recherche en profondeur

- **Stratégie**: la recherche en profondeur part du noeud racine et suit chaque chemin jusqu'à son nœud de profondeur la plus élevée avant de passer au chemin suivant: explorez l'arbre ou le graphe en allant vers le degré de profondeur croissant
- **En cas d'échec**:
 - on revient en arrière: le père
 - en recherche une autre branche à explorer
- **File d'attente** : recherche en largeur et implémentée à l'aide de la structure de données de la pile d'attente LIFO

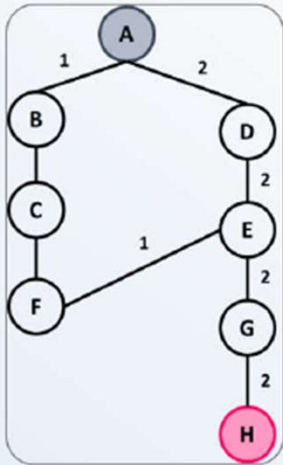
Recherche en profondeur



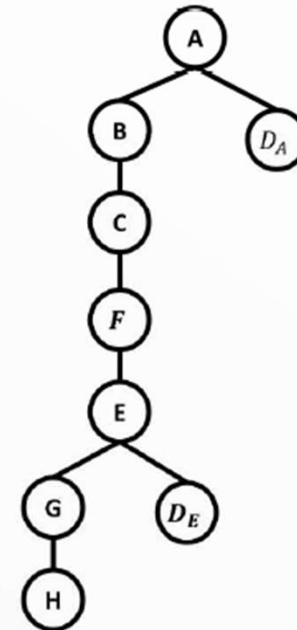
Graphe d'état



- **Stratégie** : étendre le nœud le **plus profond**
- **Implémentation** : insertion des successeurs **en tête** de la file d'attente



Nœud étendu	Nœuds à étendre
A	{A}
B	{B, D}
C	{C, D}
F	{F, D}
E	{E, D _A }
G	{G, D _E , D _A }
H	{H, D _E , D _A }
	→ But



- **Chemin** : A B C F E G H
- **Nœuds testés** : 7
- **Nœuds étendus** : 6

Méthodes non informée: algorithmes dérivés

Méthodes aveugles (non
informées)

Recherche en largeur

Recherche en profondeur

Recherche en profondeur limitée

Recherche par approfondissement itératif

Recherche en coût uniforme

- La recherche non informée est une classe d'algorithme de recherche à usage général qui fonctionne de force brute
- les méthodes de recherche diffèrent les unes des autres selon l'ordre dans lequel les nœuds sont explorés : différentes stratégies

Recherche en coût uniforme

- La **recherche à coût uniforme** est utilisée pour parcourir un arbre (ou un graphe) pondéré.

- **Objectif:**

L'objectif principal de la recherche à coût uniforme est de trouver un chemin vers le nœud d'objectif qui a le coût cumulé le plus bas.

- **Stratégie :**

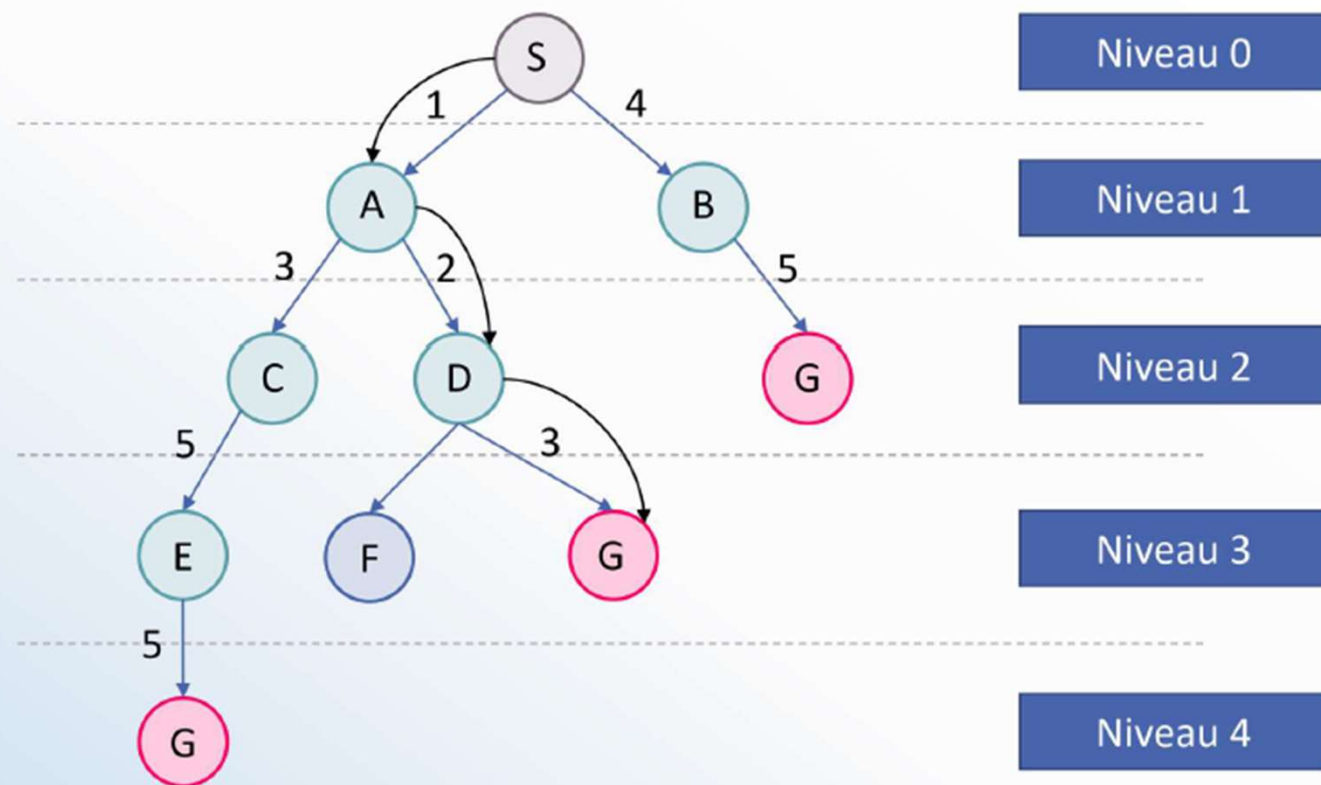
Étendre le nœud de cout le plus faible

- **File d'attente :**

Cette recherche est mise en œuvre par la file d'attente prioritaire.
Ordonnancement de la file d'attente dans **l'ordre croissant des coûts** de chemin

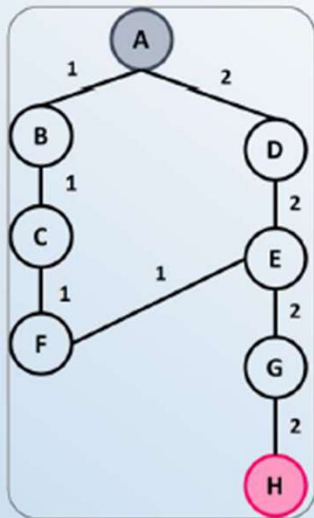
Recherche en coût uniforme

- La recherche en coût uniforme développe les nœuds en fonction de leurs coûts de chemin à partir du nœud racine.
- Elle donne la priorité maximale au coût cumulé le plus bas.
- La recherche de coût uniforme est équivalente à l'algorithme de **recherche en largeur** si le coût de chemin de toutes les connexions est **le même**.

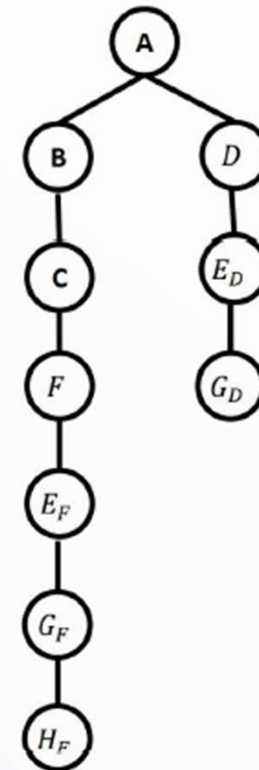


Recherche en coût uniforme

- **Stratégie** : Étendre le nœud de coût le plus faible
- **Implémentation** : insertion des successeurs dans l'ordre croissant des coûts de chemin
- En cas d'égalité: appliquer le principe LIFO



Nœud étendu	Nœuds à étendre
A	{A : 0 }
B	{B : 1 , D : 2 }
C	{D : 2 , F : 3 }
D	{F : 3 , E _D : 4 }
F	{E _F : 4 , E _D : 4 }
E _F	{E _D : 4 , G _F : 6 }
E _D	{G _D : 6 , G _F : 6 }
G _D	{G _F : 6 , H _D : 8 }
G _F	{H _F : 8 , H _D : 8 }
H _F	→ But



Résolution d'un problème par recherche heuristique

Chapitre 4

Définitions

- **Heuristique**: des stratégies de contrôle de recherche
- Les heuristiques permettent d'introduire des informations sur le problème pouvant estimer le chemin vers la solution
- Elle garantissent de trouver une bonne solution dans un délai raisonnable
- C'est un moyen qui n'est pas toujours garanti pour les meilleurs solutions

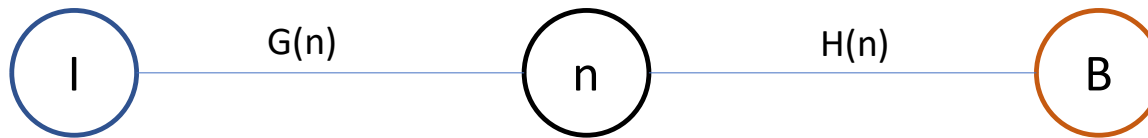
Techniques avec les heuristiques

- Deux heuristiques sont envisageables:
 1. Utiliser une fonction d'ordonnancement des opérateurs à appliquer
 2. Utiliser une fonction d'évaluation pour quantifier la validité d'un nœud par rapport aux autres
- A chaque étape les nœuds sont réordonnés
 - Pas de file ni de pile
 - On parle de recherche ordonnée
- Un moyen d'ordonner dynamiquement la liste des successeurs selon leur **promesse de se rapprocher d'un but**
- Par exemple: à chaque étape on choisit le meilleur nœud à explorer: le nœud ayant le plus de chance de mener au but

Fonctions d'évaluation

- $G(n)$: coût du nœud de départ jusqu'au nœud n : recherche uniforme
- $H(n)$: coût estimé du nœud n jusqu'au but: recherche gourmande
- $F(n)$: coût total estimé du chemin passant par n pour se rendre au but

$F(n) = G(n) + H(n)$: combinaison entre recherche gourmande et uniforme



- Pour un nœud n , on cherche à optimiser la fonction d'évaluation $F(n)$
- On distingue alors:
 - $F^*(n)$: coût idéal du chemin passant par le nœud n et qui arrive au but final
 - $F^*(n) = G^*(n) + H^*(n)$

Avec:

G^* : est le coût du meilleur chemin déjà rencontré de I à n

H^* : est le coût du meilleur chemin qui mène au but de n à B

Recherche gourmande: $F(N)=H(n)$

- **Principe :**

Le principe est d'utiliser la fonction heuristique comme fonction d'évaluation
 $f(n) = h(n)$

- **Algorithme :**

Fonction recherche_gourmande (problème, heuristique)

Input: un problème (représenté sous la forme d'un arbre/graphes), une heuristique

Output: solution ou échec

1. Ajouter l'état initial dans une liste d'attente initialement vide
2. **Tant que** (la liste d'attente n'est pas vide)
3. **Si** (le premier état de la liste est un état final) **alors**
4. la recherche a réussie: retourner **succès**
5. Générer les successeurs, et leurs valeurs heuristiques, du premier état de la liste
6. Trier la liste d'attente par ordre croissant des heuristiques
7. **Fin Tant que**
8. Retourner **échec**

Recherche avec A* : heuristique admissible

- Si A utilise une **fonction admissible** alors A est optimal. Il s'agit dans ce cas de l'algorithme A*
- L'algorithme de recherche A* est **complet et optimal**
 - Si il y a **un nombre fini de successeurs**
 - Et si la **fonction h est admissible**
- Qu'est-ce qu'une **heuristique admissible** ?
 - La valeur $h(n)$ ne doit jamais être supérieure au coût réel du meilleur chemin entre n et un état but

Une fonction heuristique est **admissible** si elle ne surestime jamais le coût réel

$$\forall n, h(n) \leq h^*(n)$$

avec $h^*(n)$ = coût réel depuis n au but

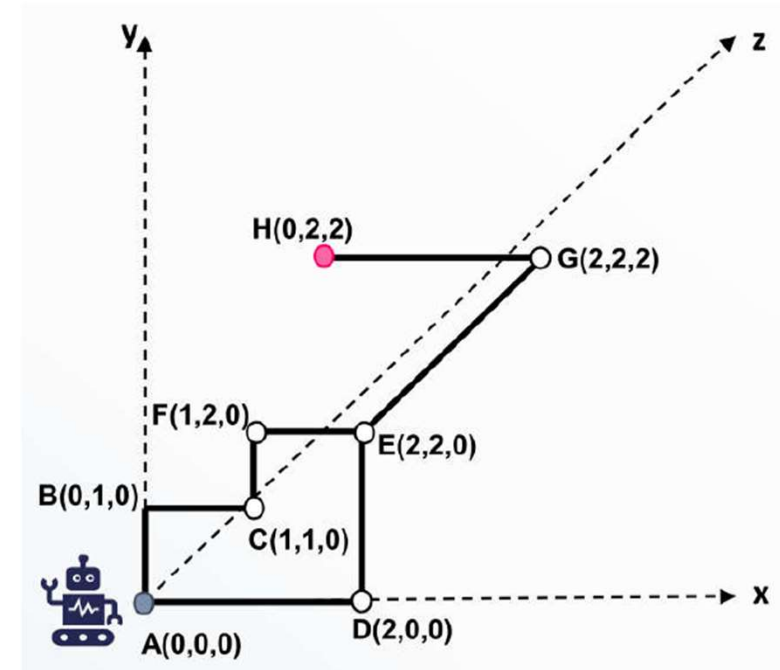
Recherche avec A*

Reprenons le problème de navigation du robot

1. Définir les trois fonctions d'évaluation: $F(n)$, $G(n)$ et $H(n)$
2. Vérifier si l'heuristique est admissible
3. Appliquer la méthode gourmande et l'algorithme de recherche A* en utilisant l'heuristique

Nous proposons d'utiliser la distance euclidienne:

$$d(A, b) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$$



Jeux stratégiques et algorithmes de recherche