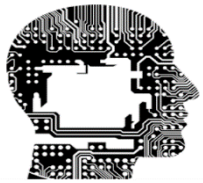
Université Constantine 2
جامعة قسنطينة 2



Foundation of Artificial Intelligence

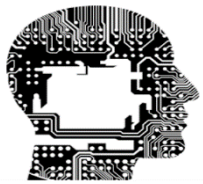
Chapitre 02

Dr. NECIBI Khaled

Faculté des nouvelles technologies

Khaled.necibi@univ-constantine2.dz

Université Constantine 2
جامعة قسنطينة 2



Foundation of Artificial Intelligence

- Résolution de Problèmes via des Stratégies de Recherche -

Dr. NECIBI Khaled

Faculté des nouvelles technologies

Khaled.necibi@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	Matser 01	SDIA

Objectif du cours

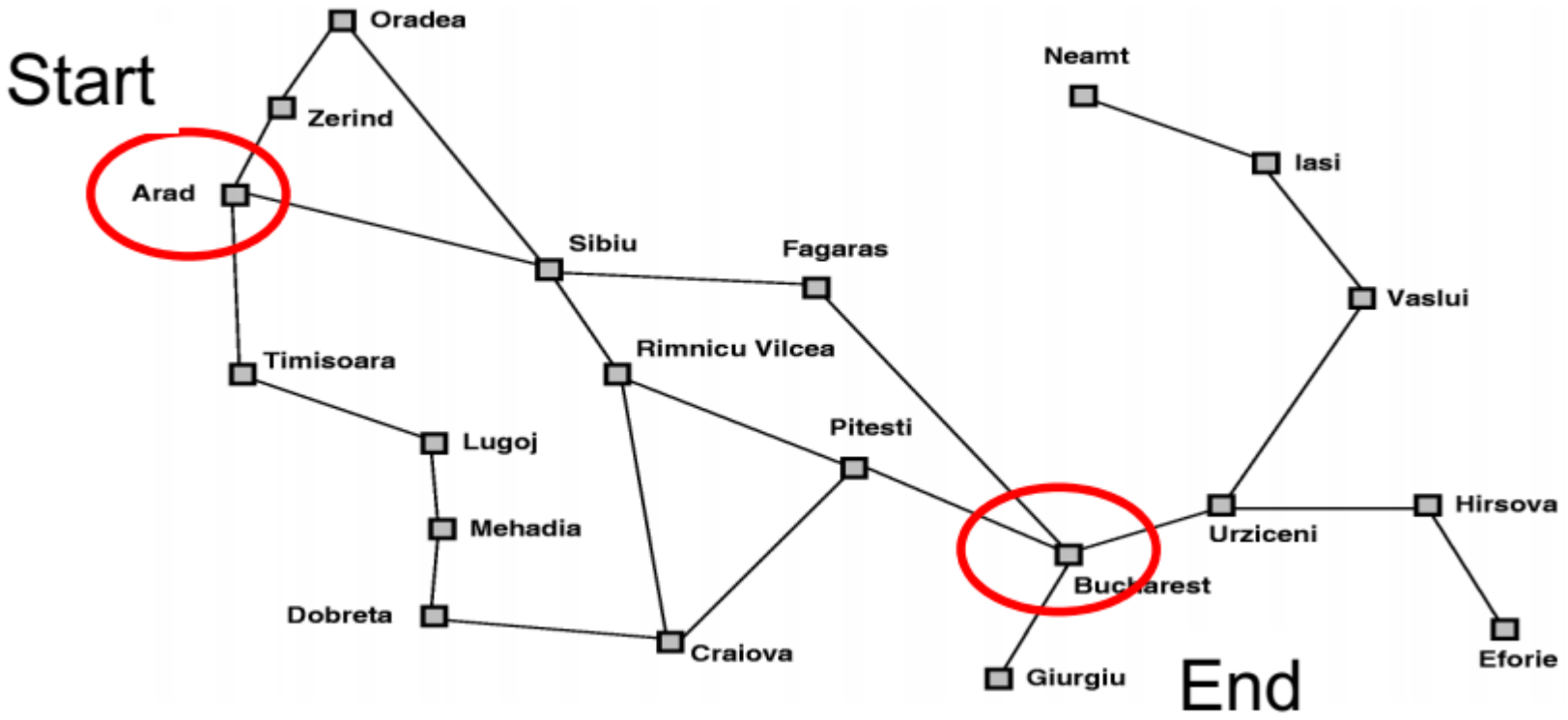
- Apprendre comment formuler un problème de recherche
- Apprendre à maîtriser les algorithmes utilisés pour résoudre des problèmes en utilisant différentes techniques de la recherche
- Apprendre à évaluer les performances d'un algorithme de recherche

Résolution de Problèmes par Recherche

- Résoudre un problème particulier → Besoin de définir les éléments d'un problème donné et proposer les éventuelles solutions correspondantes
  - Définir le problème d'une manière précise
  - Isoler et définir le processus de la représentation de connaissance
    - Une tâche nécessaire pour la résolution d'un problème
  - Choisir et appliquer les meilleures techniques pour la résolution de ce problème
- Tâche = Recherche
  - Recherche : processus nécessaire afin de résoudre une grande variété de problèmes
  - Recherche : méthode qui peut être utilisée par les ordinateurs pour examiner un grand espace de problèmes afin de trouver un but ou une solution
  - Challenge : Comment trouver un but aussi rapide que possible sans pour autant utiliser un grand nombre de ressources ?

Résolution de Problèmes par Recherche

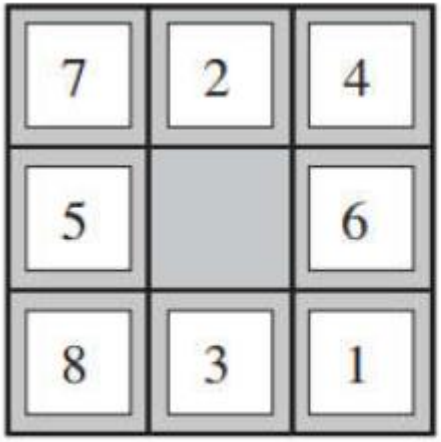
- Résoudre un problème particulier → Besoin de définir les éléments d'un problème donné et proposer les éventuelles solutions correspondantes



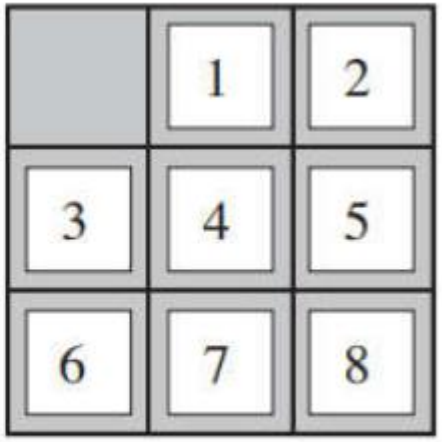
Résolution de Problèmes par Recherche

- Exemples de problèmes
  - Les approches de résolution de problèmes ont été appliquées à de grande variété de tâches environnementales dont on peut citer : Des problèmes réels (monde réel) et problèmes de jeu
  - Problèmes du monde réel
    - Navigation de Robot
    - Recherche de route (plus court chemin)
    - Recherche Internet
    - . . .

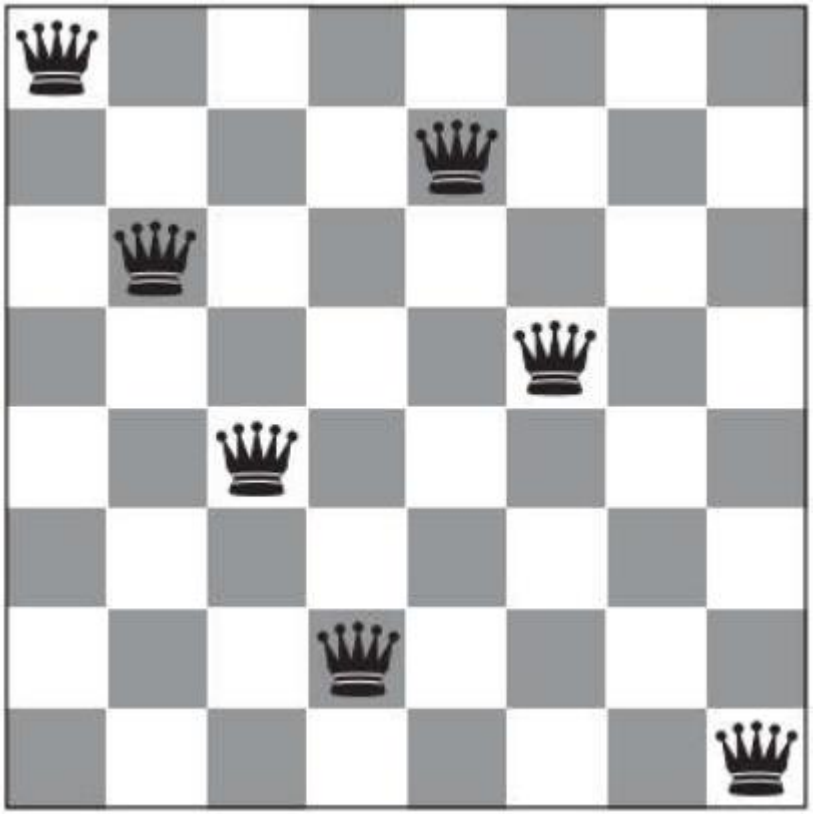
- Exemples de problèmes
  - Problèmes de jeu
  - 8-puzzle
  - 8-queens
  - . . .



Start State



Goal State



Résolution de Problèmes par Recherche

- Deux types de Recherche
  - Recherche non informée ou Recherche Aveugle (Blind Search ou Uninformed Search)
    - Recherche qui peut s'effectuée sans avoir aucune information supplémentaire à propos de la solution obtenue
  - Recherche informée (moyennant des heuristiques)
    - Recherche qui peut s'effectuée en utilisant des heuristiques

- Recherche non informée
  - Pour que la recherche non informée puisse fonctionner, quelques conditions doivent être vérifiées :
    - Cette Recherche doit être complétée : elle doit générer toutes les solutions possibles, sinon une solution optimale pourrait être négligée
    - Elle devrait être capable de trouver la meilleure solution

Résolution de Problèmes par Recherche

- Les questions qui doivent être posées lors de la Recherche non informée
  - Qu'el est la nature du but à achever ?
  - Comment savoir si le but est achevé ?
  - De quels types de connaissances devrions nous avoir besoin ?
  - Quelles sont les actions ou les instructions à exécuter ?
- Un but peut être décrit comme suit:
  - Une tâche à accomplir
  - Une situation à atteindre
  - Un ensemble de propriétés qui doivent être acquis

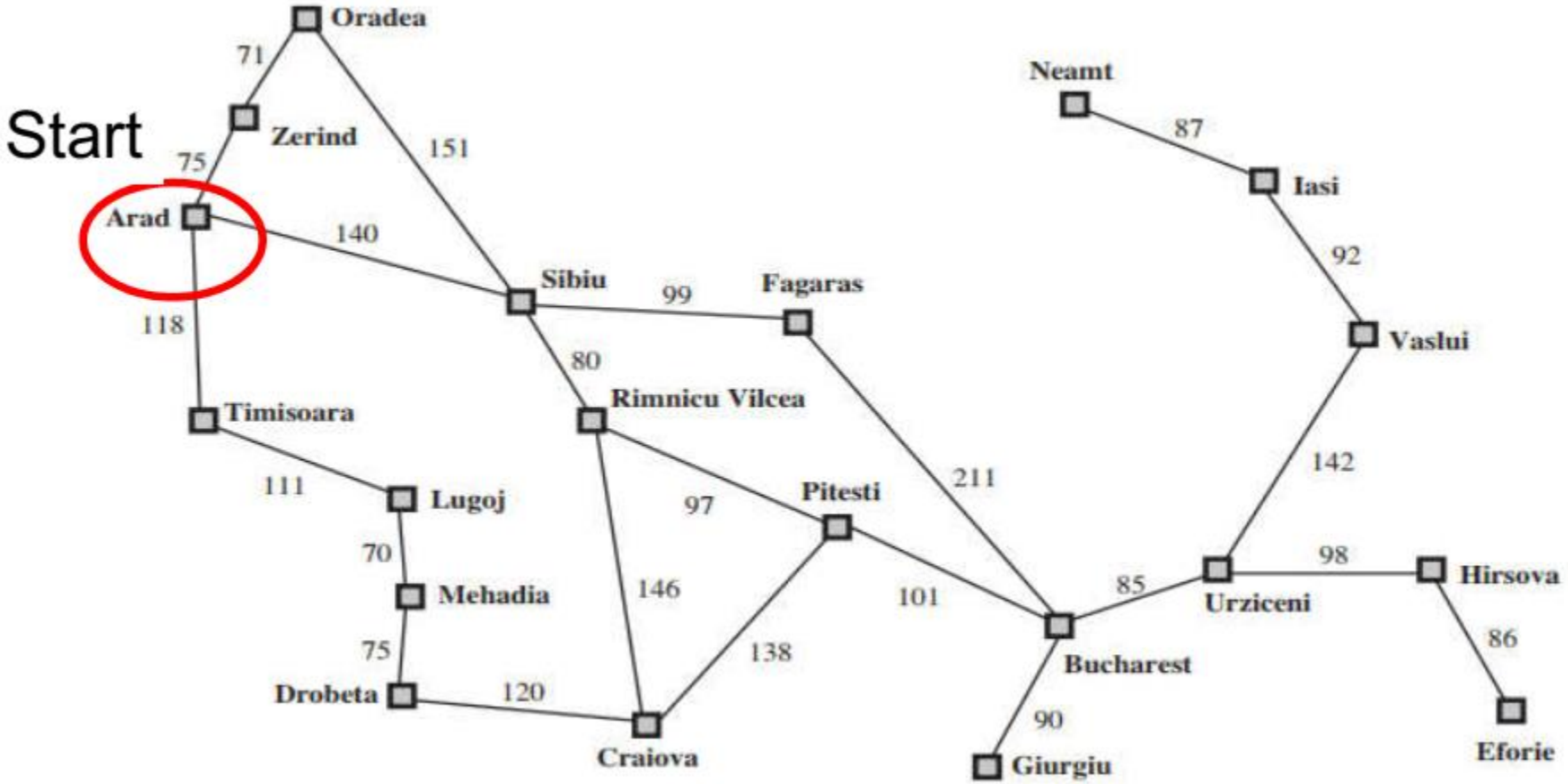
Résolution de Problèmes par Recherche

- Les concepts de base de la recherche non informée
  - Etat : configuration, situation, point…
  - Formulation du problème : selon le but à atteindre, il faut décider quelles sont les actions et les états à prendre en considération
  - Solution : Séquence d'actions qui vont aider à atteindre l'objectif ou le chemin, partant d'un état initial vers l'état final (l'état qui représente le but final)
  - Recherche : processus qui consiste à prendre en entré le problème considéré et retourne une solution à ce problème
  - Exécution : effectuer les actions recommandées par la solution proposée

Résolution de Problèmes par Recherche

- Recherche non-informée
  - Un problème peut être définit formellement par :
  - Un état initial
  - Une fonction successeur (successor function)
  - Test d'objectif (goal test) : pour déterminer si un état donné est un état final ou non
  - Une fonction coût (cost function): attribuer un coût numérique pour chaque chemin
  - Dans ce qui suit nous donnerons plus de détails sur la formulation de problèmes

Résolution de Problèmes par Recherche



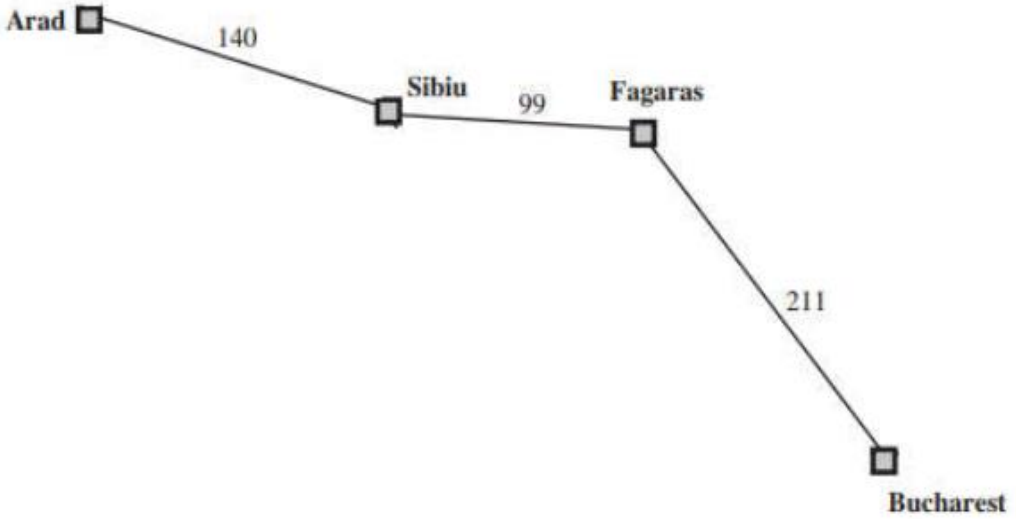
Résolution de Problèmes par Recherche

- Recherche non-informée : Formulation du problème
  - Fonction successeur (successor function)
  - Description d'actions possibles (ou actions applicables)
  - Etant donné un état S, La fonction Actions(S) renvoie un ensemble d'actions qui peuvent être exécutées une fois arrivé à l'état S
  - Exemple : à partir de l'état Dans(Arad) les actions applicables possibles sont {Aller-à(Sibiu), Aller-à(Timisoara), Aller-à(Zerind)}
  - fsuccessor(x) = {(<action, nouvel état>)}
  - Description de ce que fait chaque action (modèle de transition ou Transition Model), exemple : la fonction Résultat(S,A) qui renvoie l'état résultant après avoir effectué l'action A à l'état S
  - Résultat(Dans(Arad), Aller-à(Zerind))=Dans(Zerind)

- Recherche non-informée : Formulation du problème
  - Espace d'états :
    - Avec l'état initial, la fonction successeur permet de définir un espace d'états qui représente un ensemble de tous les états accessibles à partir de l'état initial (en suivant n'importe quelle séquence d'actions)
  - L'espace d'états forme un graphe dans lequel chaque nœud représente un état et chaque lien (arc) entre les nœuds représente une action
  - Un chemin dans l'espace d'états est une séquence d'états connectées par une séquence d'actions

- Recherche non-informée : Formulation du problème
  - Test d'objectif (goal test)
    - Permet de déterminer si un état donné est un état final (état où l'objectif est atteint)
  - Fonction coût de chemin (cost function)
    - Permet d'attribuer un coût numérique pour chaque chemin
    - Un agent intelligent choisit une fonction coût qui doit reflétée ses propres mesures de performances
  - Abstraction
    - À ne pas prendre en considération les détails physiques liés à la représentation du problème
    - Exemple : Dans la carte de la Romanie, lors de la formulation du problème, il ne faut pas prendre en considération l'état des routes, les feux rouges, les éventuelles influences du climat au déplacement,…etc.

- Recherche non-informée : Formulation du problème
  - Solution
  - Une solution à un problème donné est une séquence d'actions qui mène de l'état initial vers l'état où l'objectif est atteint
  - La qualité de la solution est mesurée par la fonction coût du chemin
  - Une solution optimale doit avoir le coût le plus bas par rapport à tout les autres solutions



- Recherche non-informée : Exemple de problème
  - Problème de voyageur de commerce (problème du touriste ou Touring problem)
  - Étant donné un ensemble de cités n, le problème du voyageur de commerce consiste à visiter toutes les cités au moins une seul fois, tout en démarrant et finissant le voyage dans la même cité
  - Espace d'états
    - L'espace d'états peut être spécifié par la cité courante, ainsi que l'ensemble de cités déjà visitées
  - L'état initial
    - N'importe quel état peut être un état initial
  - La fonction successeur
    - Génère la cité suivante à visiter et ce par rapport à l'état courant
  - Test d'objectif
    - Arrivé à la cité destination + toutes les cités sont visitées

- Recherche non-informée : Exemple de problème
  - Problème de voyageur de commerce (problème du touriste ou Touring Problem)
  - Étant donné un ensemble de cités n, le problème du voyageur de commerce consiste à visiter toutes les cités au moins une seul fois, tout en démarrant et finissant le voyage dans la même cité
  - Fonction coût de chemin
    - La somme de tous les coûts numériques de chaque chemin traversé

Résolution de Problèmes par Recherche

- Recherche non-informée : Recherche de solutions
  - Pour résoudre un problème, on doit définir une stratégie de recherche qui va permettre l'exploration efficace de l'espace d'états
  - Comment représenter l'espace d'état ?
    - → Arbre de recherche
    - → Graphe de recherche
    - Chaque état est représenté par un nœud
  - Opération de développement des états (nœuds successeurs)
    - un nœud est dit « développé » si la fonction successeur de l'état correspondent génère de nouvel états (expansion)
    - Ou bien : exploration de l'espace d'états en générant des successeurs d'états déjà générés
    - On s'arrête lorsqu'on a choisi de développer un nœud qui est un état final (leaf node)

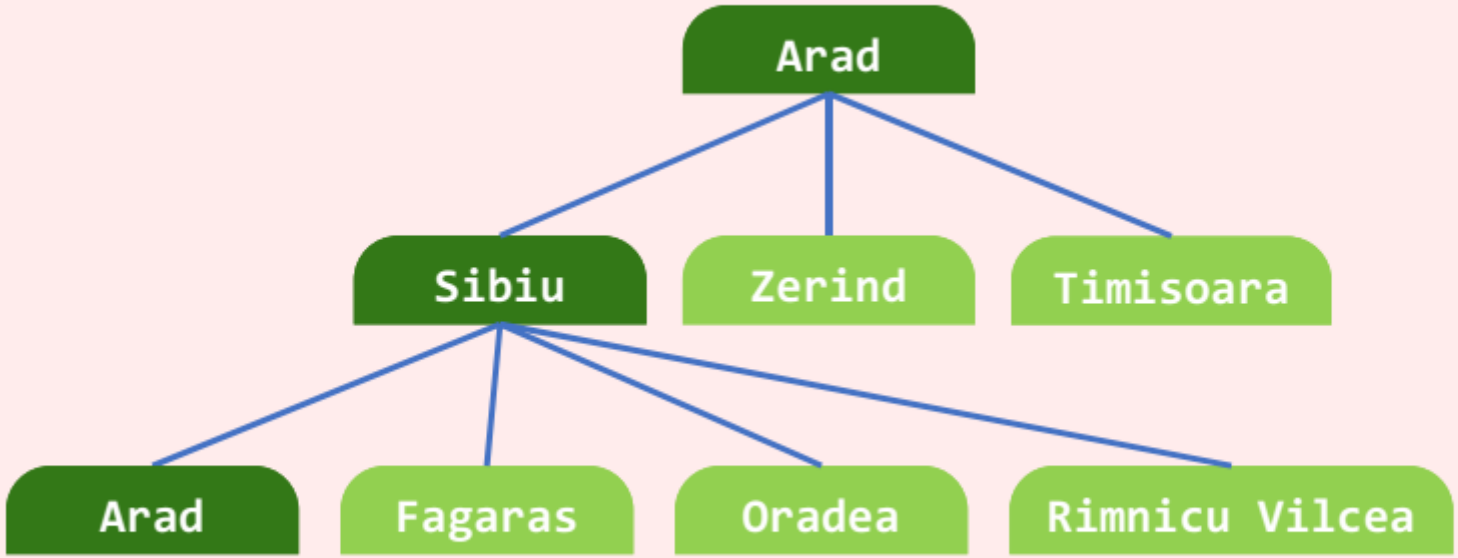
- Recherche non-informée : Description informelle

/l'algorithme arbre-recherche (Tree search)

```
Fonction arbre-recherche (problème, stratégie) returns solution ou échec {
        initialiser l'arbre de recherche en utilisant l'état initial du
        problème;
        répéter
                s'il n'existe aucun candidats pour le développement alors
                        returns  échec;
                choisir un nœud feuille (leaf node) pour développement selon
                la stratégie;
                si le nœud contient un état final alors returns solution
                correspondante;
                sinon
                développer le nœud et ajouter les nœuds successeurs résultants
                dans l'arbre de recherche
        Fin
}
```

- Recherche non-informée : Description informelle

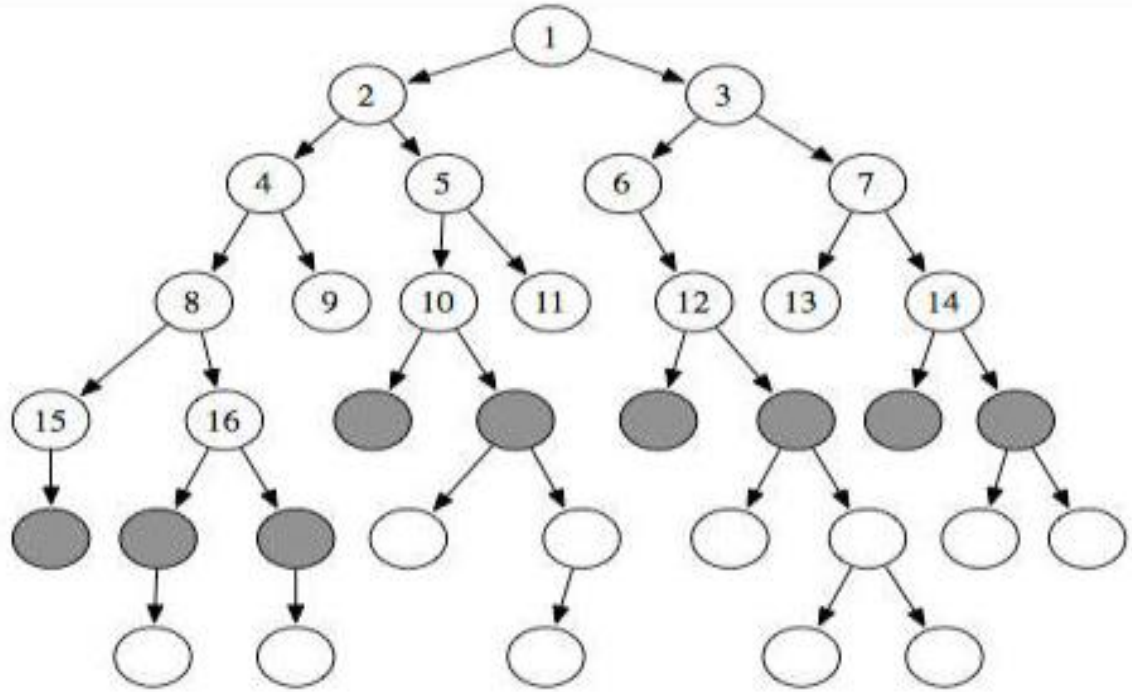
Exemple Arbre de recherche



- Recherche non-informée : Vers une description formelle
  - Représentation d'un nœud avec une structure de donnée:
  - État : Représentation de la configuration physique (une cité, reines pour jeu d'échec…)
  - Nœud parent : le nœud dans l'arbre de recherche qui a généré le nœud actuel
  - Action : l'action appliquée par le nœud parent pour générer le nœud actuel
  - Coût de chemin : le coût du chemin partant de l'état initial jusqu'au nœud actuel
  - Profondeur : le nombre de sauts le long du chemin à partir de l'état initial

- Recherche non-informée : Vers une description formelle
  - Comment procéder avec les nœuds qui ne sont pas explorables ? (les nœuds qui ne sont pas développés)
  - Frontière (Fringe) : c'est un ensemble de nœuds générés qui n'est pas encore développé ou exploré (derniers nœuds visités). Pour représenter une frontière il faut définir :
  - Un ensemble de nœuds : la plus simple façon, mais ça peut être très couteux en terme de calcul
  - File d'attente : la meilleur implémentation d'une frontière

- Recherche non-informée : Vers une description formelle
  - Cor
    expl
  - Fron
    pas
    repr
  - Un e
    très
  - File (



s)

qui n'est

). Pour

eut être

- Recherche non-informée : Vers une description formelle
  - Les opérations applicables sur la fille d'attente :
  - Make-Node(état, parent, action, profondeur, coût) : crée un nœud avec les paramètres en entré
  - Empty?(queue) : retourne True seulement dans le cas où il n y a plus de nœuds dans la fille d'attente
  - First(queue) : retourne le premier nœud dans la fille d'attente
  - Remove-First(queue) : retourne First(queue) et en l'enlevant de la fille d'attente
  - Insert(element, queue) : insérer un nœud dans la fille d'attente et retourner la fille d'attente résultante
  - Insert-All(elements, queue) : insérer un ensemble de nœuds dans la fille d'attente et retourner la fille d'attente résultante

- Recherche non-informée : Vers une description formelle

/la fonction arbre-recherche (Tree search)

```
Fonction arbre-recherche (problème, frontière) returns solution ou échec {
        frontière ← Insert(Make-Node(state[problème], Null, Null, profondeur,
                                                    coût_de_chemin), frontière);
      répéter
            if Empty?(frontière) then returns échec;

            node ← Remove-First(frontière);

            if Goal-Test[problème] appliqué à State[node] réussit then

            returns solution(node);

            frontière ← Insert-all(Expand(node, problème), frontière);
      Fin
}
```

- Recherche non-informée : Vers une description formelle

/la fonction Expand(node, problem)

```
Fonction Expand(node, problème) returns un ensemble de nœuds{
        « successeurs » ← ensemble vide;

                pour chaque <action, résultat> in fsuccessor(State(node))faire
{
                coût_de_chemin ← coût-chemin[node] + coût-pas(State[node]
                                                        action, résultat);
                Profondeur ← Profondeur[node] + 1;

                S ← Make-Node(resultat, node, action, coût_de_chemin,
                                                            Profondeur);
                Ajouter S au « Successeurs »;

                returns « Successeurs »;
}
```

Résolution de Problèmes par Recherche

- Recherche non-informée : Evaluation des performances
  - La performance des algorithmes de résolution de problèmes peut être évaluée selon :
  - Complétude : est-ce que cette stratégie trouve toujours une solution si elle existe ?
  - Optimalité : est-ce que cette stratégie trouve toujours la solution la moins coûteuse ?
  - Complexité temporelle (complexité en temps): comme bien de temps faut-il pour trouver une solution ? → Peut être mesuré par le nombre de nœuds générés
  - Complexité spatiale (complexité en mémoire) : comme bien de mémoire faut-il pour effectuer la recherche ? → Peut être mesuré par le nombre maximum de nœuds stockés en mémoire

- Recherche non-informée : Evaluation des performances
  - La complexité en temps et en mémoire se mesure en termes de :
  - b : le facteur maximum de branchement de l'arbre de recherche : le nombre maximum de fils de nœuds dans l'arbre de recherche
  - d : la profondeur de la solution la moins coûteuse
  - m : la profondeur maximale de l'arbre de recherche

Résolution de Problèmes par Recherche

- Recherche non-informée : Evaluation des performances
  - Les stratégies de recherche non-informées utilisent seulement des informations disponibles dans le problème
  - Il existe plusieurs stratégies de recherche :
    - Recherche en largeur d'abord (Breadth Firt Search; BFS)
    - Recherche en coût uniforme (Uniform Cost Search; UCS)
    - Recherche en profondeur d'abord (Depth First Search; DFS)
    - Recherche en profondeur limitée (Depth Limited Search; DLS)
    - Recherche itérative en profondeur (Iterative Deeping Search; IDS)
    - ...