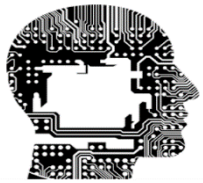
Université Constantine 2
جامعة قسنطينة 2

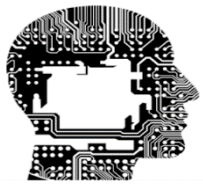


Systèmes Intelligents

Chapitre 02 (Partie 02)

Dr. NECIBI Khaled
Faculté des nouvelles technologies
Khaled.necibi@univ-constantine2.dz

Université Constantine 2
جامعة قسنطينة 2



Systèmes Intelligents

- 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	Licence 3	Science de l'informatique SCI

Objectif du cours

- Comprendre les stratégies de base de la recherche non informé (suite…)
- Maîtriser les stratégies de recherche non informée (suite…)

© Dr. NECIBI Khaled

Recherche en largeur d'abord BFS

- BFS : Breadth First Search
  - Il s'agit d'une simple stratégie dans la quelle le nœud racine est développé dans un premier temps, ensuite tout les nœuds successeurs seront développés
  - En général, tout les nœuds sont développés selon une certaine profondeur dans l'arbre de recherche
  - La recherche en largeur d'abord est une instance de l'algorithme de recherche dans un graphe

© Dr. NECIBI Khaled

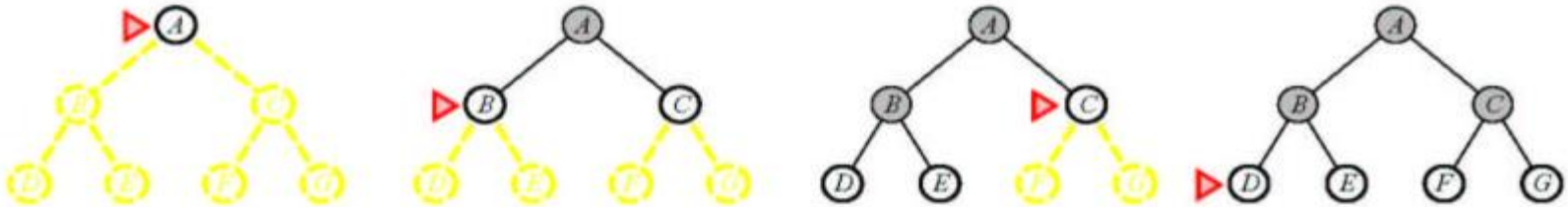
Recherche en largeur d'abord BFS

- BFS : l'idée de base
  - Les nœuds qui appartiennent à une profondeur i sont développés avant les nœuds qui appartiennent à une profondeur i+1
  - Cette stratégie peut être implémentée en faisant appel à l'algorithme de recherche dans un arbre (Tree-Search voir la partie 01 du cours)
  - L'algorithme Tree-Search prend comme paramètres une fille d'attente de type First-In-First-Out (FIFO) ainsi que le problème à résoudre

© Dr. NECIBI Khaled

Recherche en largeur d'abord BFS

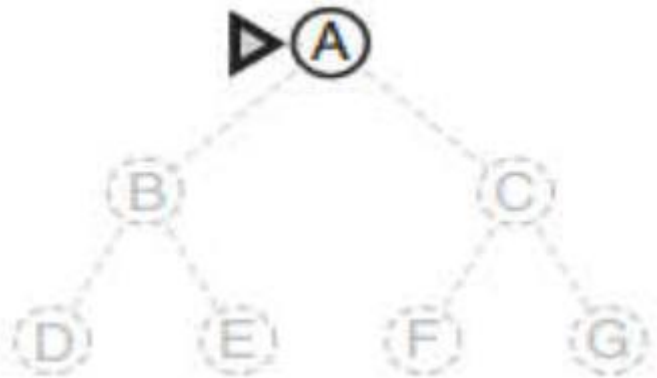
- BFS : l'idée de base
  - Les nœuds peu profond sont développés avant les nœuds les plus profond



© Dr. NECIBI Khaled

Recherche en largeur d'abord BFS

- BFS : Exemple d'exécution
  - La région non visitée est colorée en gris
  - Les nœuds dans le niveau 2 (profondeur = 2) n'ont pas de successeurs et D est l'état final (état but)

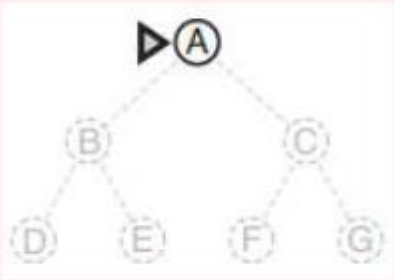


© Dr. NECIBI Khaled

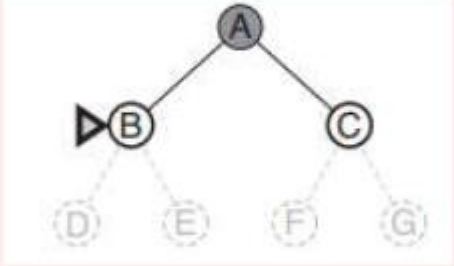
Recherche en largeur d'abord BFS

- BFS : Exemple d'exécution

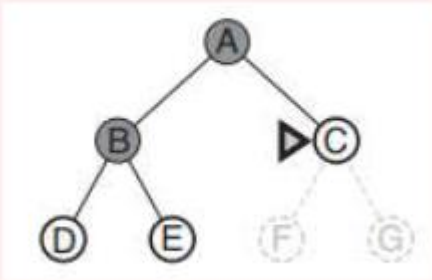
Frontière = [A]



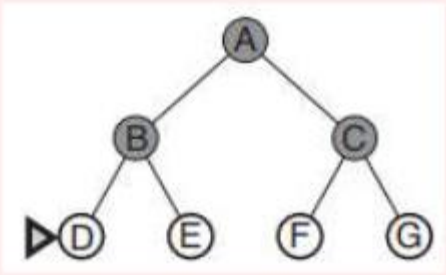
Frontière = [B, C]



Frontière = [C, D, E]



Frontière = [D, E, F, G]



© Dr. NECIBI Khaled

Recherche en largeur d'abord BFS

- Évaluation de performances de la stratégie BFS
  - Complétude : oui si le facteur de branchement b (voir la 1ère partie une du cours) est finit
  - Optimalité : l'état objectif le moins profond (Shallowest goal) n'est pas nécessairement l'optimum. Il est optimal si tout les actions ont le même coût
  - Complexité temporelle : au pire des cas, la stratégie BFS développe chaque nœud (à l'exception du nœud qui correspond à l'état final), donc le temps nécessaire pour exécuter la stratégie BFS est le suivant :

$$1 + b + b^2 + b^3 + \ldots + b^d + (b^{d+1} - b) = O(b^{d+1})$$

  - Complexité spatiale : la stratégie BFS retient chaque nœud dans la mémoire ! L'espace mémoire représente un grand problème

© Dr. NECIBI Khaled

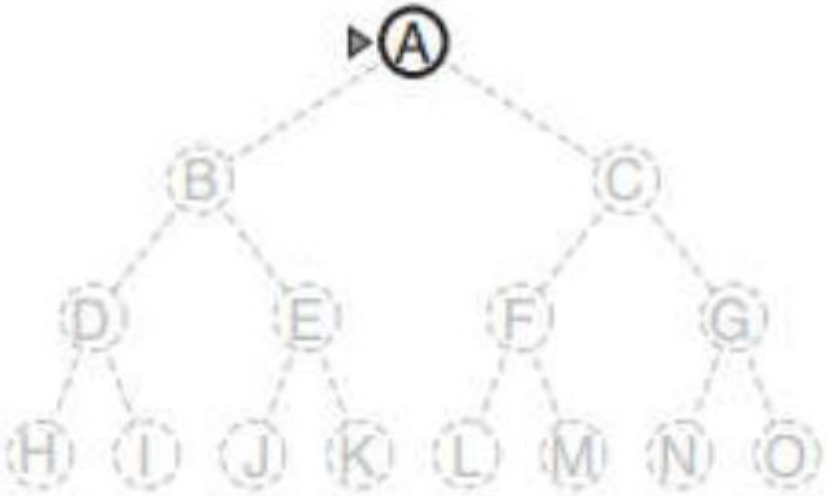
Recherche en Profondeur d'abord DFS

- DFS : Depth First Search
  - L'idée de base : cette stratégie développe le nœud le plus profond dans la frontière courante de l'arbre de recherche
  - Vu que les nœuds sont développés, ils seront supprimés de la frontière pour que la recherche « revient » au nœud le moins profond qui a toujours des nœuds successeurs (qui, à leur tour, ne sont pas encore développés)
  - La stratégie DFS est implémentée en utilisant une fille d'attente de type Last-In First-Out LIFO
  - Une autre alternative est d'implémenter DFS en utilisant une fonction récursive qui fait un appel à elle-même sur chacun de ses nœuds fils

© Dr. NECIBI Khaled

Recherche en Profondeur d'abord DFS

- DFS : Exemple d'exécution
  - La région non visitée est colorée en gris
  - Les nœuds visités dans la frontière et qui n'ont pas de successeurs sont retirés de la mémoire
  - Les nœuds dans le niveau 3 (profondeur = 3) n'ont pas de successeurs et M est l'état final (état but)

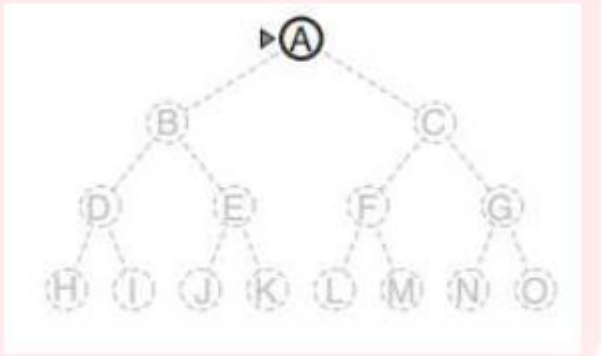


© Dr. NECIBI Khaled

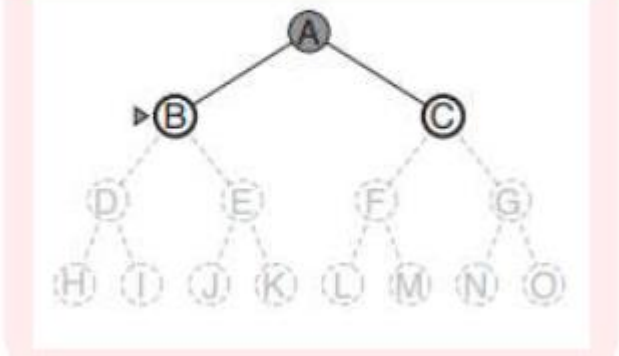
Recherche en Profondeur d'abord DFS

- DFS : Exemple d'exécution

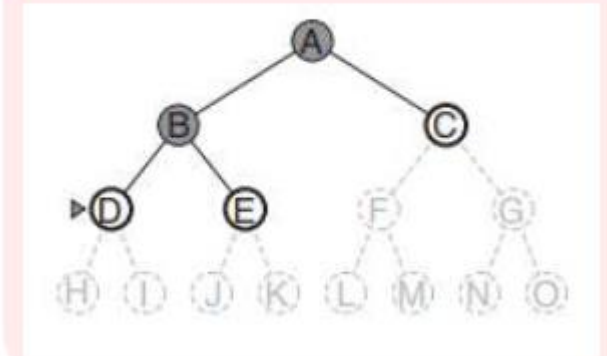
Frontière = [A]



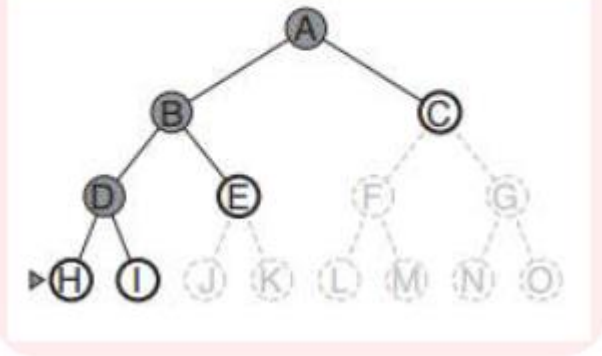
Frontière = [B, C]



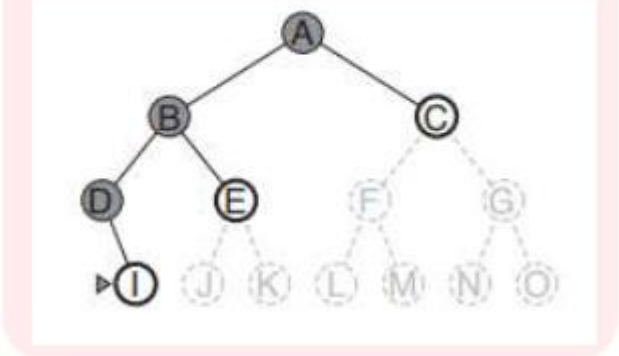
Frontière = [D, E, C]



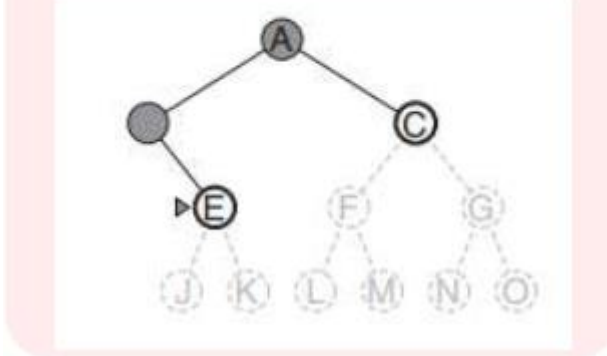
Frontière = [H, I, E, C]



Frontière = [I, E, C]



Frontière = [E, C]

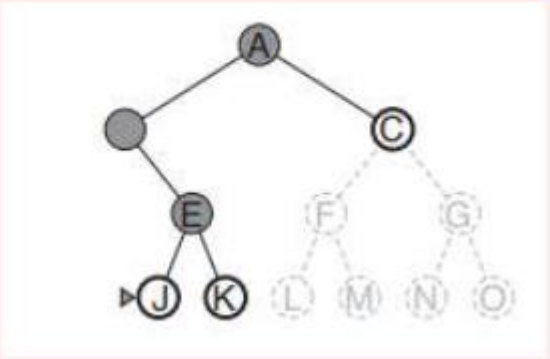


© Dr. NECIBI Khaled

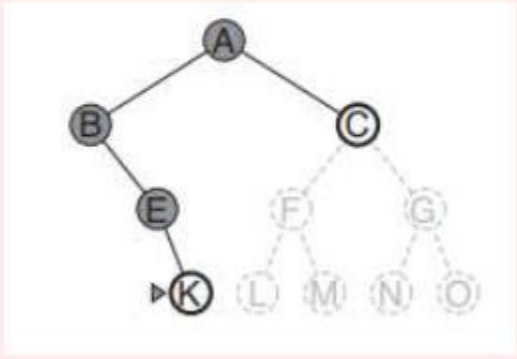
Recherche en Profondeur d'abord DFS

- DFS : Exemple d'exécution

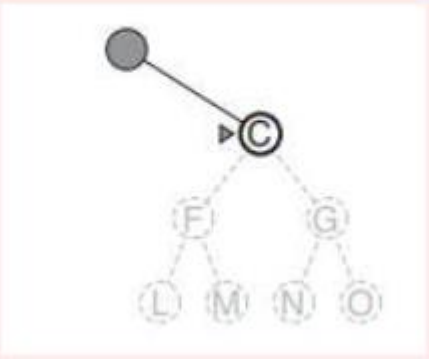
Frontière = [J, K, C]



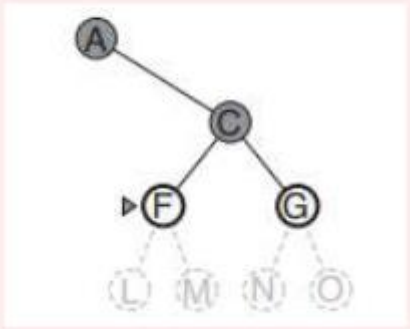
Frontière = [K, C]



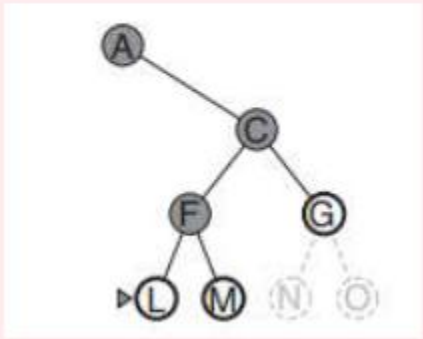
Frontière = [C]



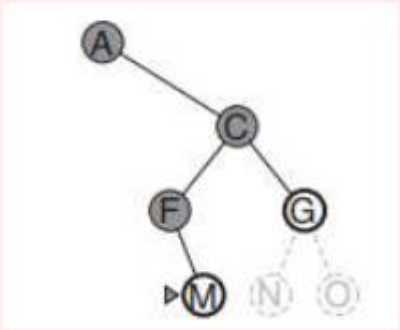
Frontière = [F, G]



Frontière = [L, M, G]



Frontière = [M, G]



© Dr. NECIBI Khaled

Recherche en Profondeur d'abord DFS

- Évaluation de performances de la stratégie DFS
  - Complétude : cet algorithme serait incomplet en la présence d'une profondeur non limitée qui ne contient aucune solution
  - Optimalité : cet algorithme ne fournie pas toujours de solutions optimales
  - Complexité temporelle : dans le pire des cas, l'algorithme génère $O(b^m)$ nœuds dans le graphe de recherche (où b est le facteur de branchement, m est la profondeur maximale)
  - Complexité spatiale : l'algorithme DFS nécessite peu d'espace mémoire par rapport à l'algorithme BFS
    - Il n'a besoin de sauvegarder qu'un seul chemin à partir du nœud racine (root node) jusqu'au nœud feuille (leaf node)
    - Le stockage nécessaire est (b^{m+1})

© Dr. NECIBI Khaled

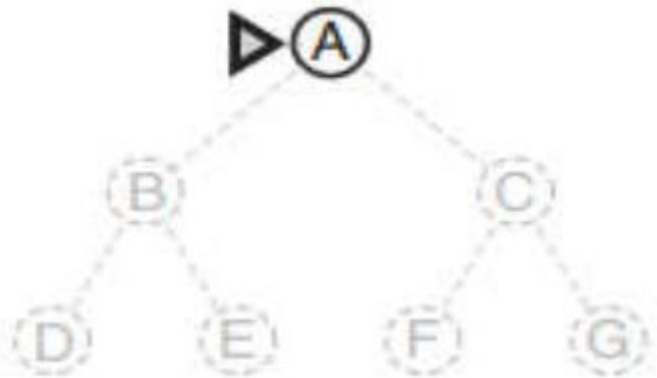
Recherche en Profondeur Limitée DLS

- DLS : Depth Limited Search
  - L'idée de base : il s'agit de la même stratégie adoptée par l'algorithme DFS mais avec la contrainte limite de profondeur
    - i.e. La recherche est non permise au-delà de la limite de profondeur imposée
  - Cette stratégie donne de meilleur performances si on connait la profondeur de la solution
  - La complétude de cette stratégie est garantie
  - Si la solution (ou l'état final) se trouve en dessous de la limite de la profondeur imposée, la recherche ne peut trouver l'état objectif final (dans ce cas l'algorithme de recherche est incomplet)
  - Sinon on utilise la stratégie de recherche itérative en profondeur (Iterative Deeping Search IDS)

© Dr. NECIBI Khaled

Recherche en Profondeur Limitée DLS

- DLS : Exemple d'exécution
  - La région non visitée est colorée en gris
  - Les nœuds dans le niveau 2 (profondeur = 2) n'ont pas de successeurs et G est l'état final (état but)

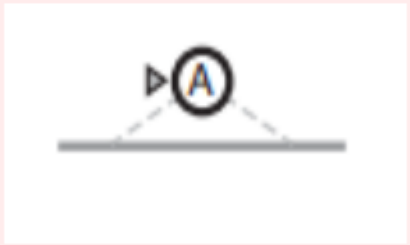


© Dr. NECIBI Khaled

Recherche en Profondeur Limitée DLS

- DLS : Exemple d'exécution

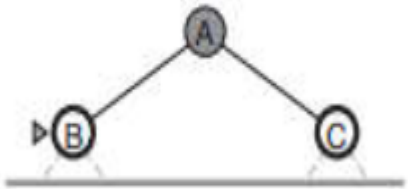
l =0



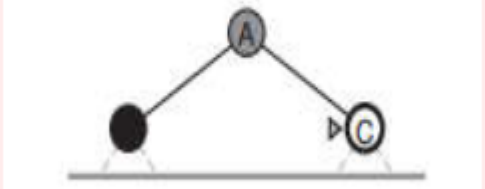
l = 0

l = 1

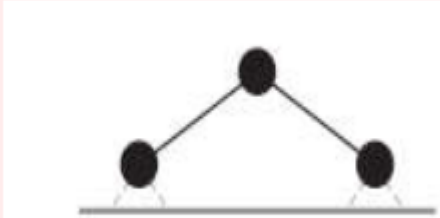
l = 1



l = 1



l = 1

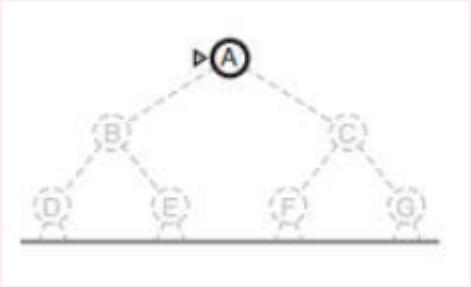


© Dr. NECIBI Khaled

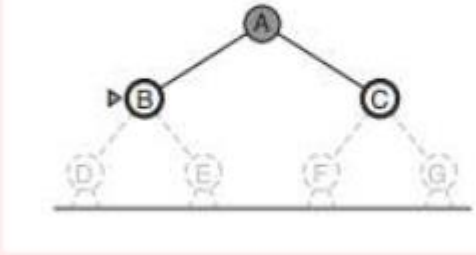
Recherche en Profondeur Limitée DLS

- DLS : Exemple d'exécution

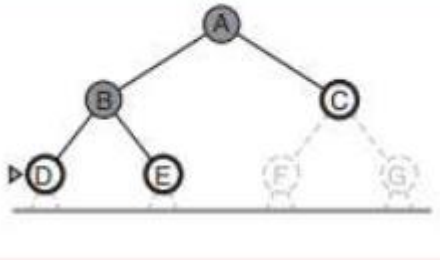
l =2



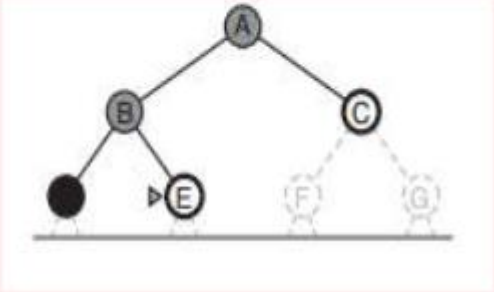
l = 2



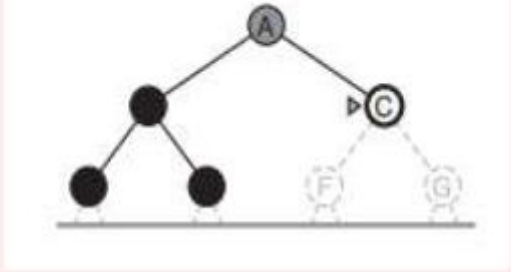
l = 2



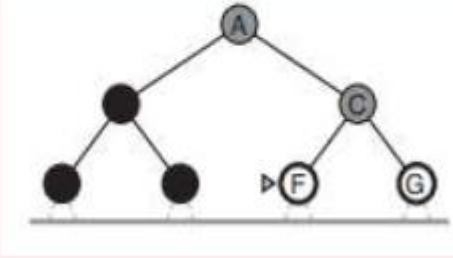
l = 2



l = 2



l = 2

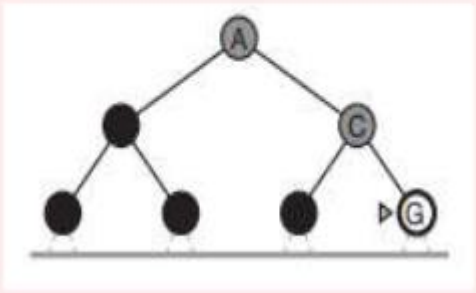


© Dr. NECIBI Khaled

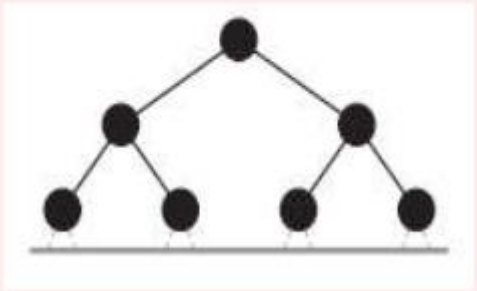
Recherche en Profondeur Limitée DLS

- DLS : Exemple d'exécution

l =2



l = 2



© Dr. NECIBI Khaled

Recherche en Profondeur Limitée DLS

- Évaluation de performances de la stratégie DLS
  - Complétude : Complet si $l \geq d$
  - Optimalité : non optimal
  - Complexité temporelle : $O(b^l)$
  - Complexité spatiale : $O(b^l)$

© Dr. NECIBI Khaled

Recherche Itérative en Profondeur IDS

- IDS : Iterative Deepening Search
  - L'idée de base : La recherche itérative en profondeur IDS répète l'algorithme de recherche en profondeur limité DLS
    - L'algorithme s'arrête quand une solution est trouvée ou bien aucune solution n'est trouvée
  - La stratégie IDS combine les avantages de BFS et DFS :
    - Comme la stratégie DFS l'espace mémoire nécessaire est très modeste $O(b^d)$.
    - Similairement aussi au BFS, l'algorithme IDS se termine quand le facteur de branchement est fini
  - Le nombre total de nœuds générés est :

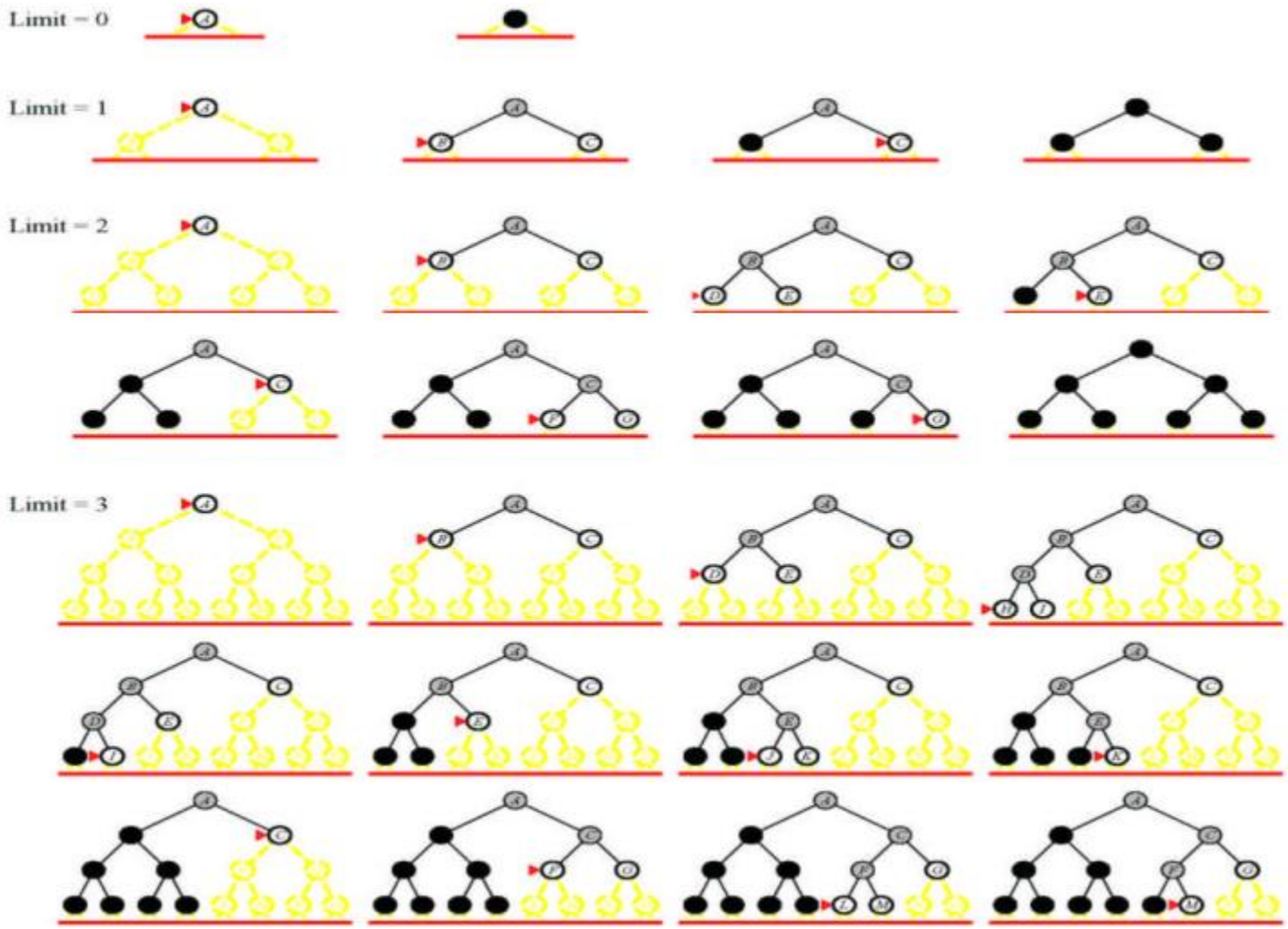
$$N(IDS)=(d)b + (d-1)\ b^2 + \ldots+(1)b^d$$

  - En général, IDS est la méthode préférée de la recherche non informée quand :
    - L'espace de recherche est suffisamment large
    - La profondeur de la solution n'est pas connue d'avance

© Dr. NECIBI Khaled

Recherche Itérative en Profondeur IDS

- IDS : Exemple d'exécution de 04 itérations



© Dr. NECIBI Khaled

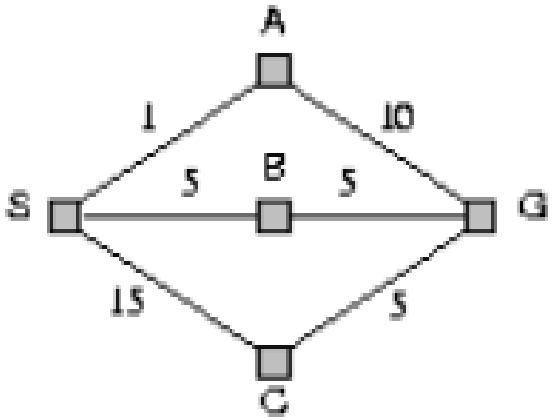
Recherche en Coût Uniforme UCS

- UCS : Uniform Cost Search
  - L'idée de base : Quand les coûts de chaque chemin sont les mêmes, la recherche en largeur d'abord BFS trouve la solution optimale
  - Pour n'importe quelle fonction coût de chemin, la recherche en coût uniforme UCS développe le nœud n avec le chemin le moins coûteux
  - La recherche UCS prends en considération le coût total
  - La recherche UCS est guidée par les coûts de chemin plutôt que par les profondeurs
    - Les nœuds sont ordonnés selon le coût de leur chemins

© Dr. NECIBI Khaled

Recherche en Coût Uniforme UCS

- UCS : Exemple d'exécution
  - Exemple : la fonction coût g(n) rajoute les nœuds dans l'ordre selon le chemin le moins coûteux
  - S : état initial, G : l'état final (l'état but)
  - Question : proposer une solution optimale (le chemin le moins coûteux) pour atteindre l'état objectif G en utilisant la stratégie UCS

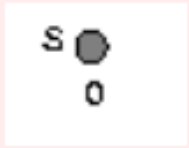


© Dr. NECIBI Khaled

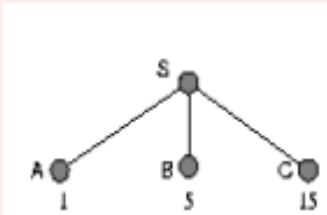
Recherche en Coût Uniforme UCS

- UCS : Exemple d'exécution

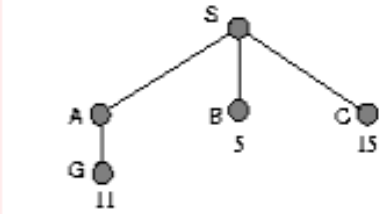
Fringe = [S, 0]



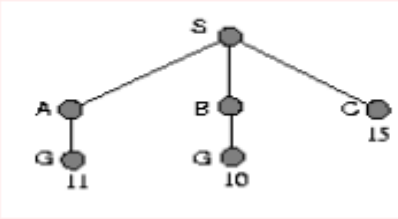
Fring = [(A,1),(B,5),(C,15)]



Fring = [(B,5),(G,11),(C,15)]



Fring = [(G,10),(G,11),(C,15)]



© Dr. NECIBI Khaled

Recherche en Coût Uniforme UCS

- Évaluation de performances de la stratégie UCS
  - Complétude : Complet si le coût de chaque pas est strictement supérieur à 0
  - Optimalité : Optimal car les nœuds sont développés en fonction du coût
  - Complexité temporelle : Nombre de nœuds pour lesquels le coût est inférieur ou égal au coût de la solution optimale
  - Complexité spatiale : Similaire à la complexité temporelle

© Dr. NECIBI Khaled

Recherche non informée : Les états répétés

- Les états répétés
  - Problème : possibilité de perdre du temps en développant des états qui ont été déjà visités et développés
  - La répétition des états se produit quand les actions sont réversibles
  - Un problème qui peut être résolu peut ne pas avoir de solutions si l'algorithme ne peut pas détecter les états répétés
  - Afin de détecter les répétitions, une opération de comparaison est nécessaire
  - Pour la stratégie DFS, les chemins répétés peuvent être ignorés
  - Besoin de garder plus de nœuds dans la mémoire : échange entre le temps et l'espace mémoire
  - Garder une liste pour stocker chaque nœud développé

© Dr. NECIBI Khaled

Recherche non informée : éviter les états répétés

/la fonction arbre-recherche (Tree search)

```
Fonction graphe-recherche (problème, frontière) returns solution ou échec {

        ** closed ← vide;

        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);

                ** if state[node] n'appartient pas à la liste closed then

                        closed ← state[node] ;

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


© Dr. NECIBI Khaled