Chapter 4

Uninformed Search Strategies

Contents

[Short type Questions & Answers 3](#_Toc14943448)

[What is meant by search algorithm optimality? 3](#_Toc14943449)

[What are the advantages of breadth-first search (BFS) over depth-first search (DFS)? 3](#_Toc14943450)

[What is the advantage of DFS over BFS? 3](#_Toc14943451)

[What is meant by search algorithm completeness? 3](#_Toc14943452)

[What is backtracking search? 3](#_Toc14943453)

[Define informed search strategy. 3](#_Toc14943454)

[.What do you mean by Best First Search approach? 4](#_Toc14943455)

[Define heuristic function. 4](#_Toc14943456)

[What do you mean by depth limited search? 4](#_Toc14943457)

[What are the problems arises when knowledge of the states or actions is incomplete? 4](#_Toc14943458)

[What are the steps to evaluate an algorithm’s performance? 5](#_Toc14943459)

[What are the four components in problem? 6](#_Toc14943460)

[What is called as a uniformed search? 6](#_Toc14943461)

[What is called informed search? 6](#_Toc14943462)

[Give the complexity of a breath-first search. 6](#_Toc14943463)

[What is iterative deepening search? 7](#_Toc14943464)

[Q What is AI Technique? 7](#_Toc14943465)

[Show how breadth-first-search and depth-first-search can be implemented using some appropriate pseudo-code. If you use another search algorithm as a sub-routine then show this algorithm in detail as well. 7](#_Toc14943466)

[Q Describe any data structures that are used in implementing the BFS & DFS searches and describe the way in which they are used. 8](#_Toc14943467)

[Q What is the worst-case time and space complexity of the above BFS and DFS algorithms. 8](#_Toc14943468)

[Q Describe the terms complete and optimal with regards to evaluating search strategies? Are either depth-first-search or breadth-first-search complete? Are either of them optimal? 9](#_Toc14943469)

[Q Give an example of a road navigation problem for which uniform cost search fails if an algorithm does the goal check on node generation as opposed to goal expansion. 9](#_Toc14943470)

[Descriptive Question & Answer 9](#_Toc14943471)

[Q What is the fewest possible number of nodes, in terms of b, d, and m, that will be generated for each of the following? Assume that all nodes have exactly b children. If the strategy can have both goal-test-at-generation-time and goal-test-at-expansion variants, answer for both. 10](#_Toc14943472)

[a) depth-first search with backtracking 10](#_Toc14943473)

[b) depth-first expansion 10](#_Toc14943474)

[c) breadth-first 10](#_Toc14943475)

[d) depth-first iterative deepening 10](#_Toc14943476)

[Show the working of Uniform-cost Search Algorithm also discuss its various terminologies 13](#_Toc14943477)

[Example: 14](#_Toc14943478)

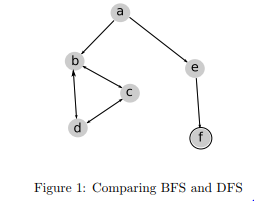
[Q define uninformed search strategies and list its types. Explain one of its type with example 15](#_Toc14943479)

[Q How to solve Route finding problem using DFS. 21](#_Toc14943480)

[Explain Iterative deepening depth-first Search with example 30](#_Toc14943481)

[Example: 31](#_Toc14943482)

[Consider the search problem represented in Figure 1, where a is the start node and f is the goal node. Would you prefer DFS or BFS for this problem? Why? 32](#_Toc14943483)

[ 32](#_Toc14943484)

[MCQs & answers 33](#_Toc14943485)

The chapter consist of Short type Questions & Answers , Descriptive Question & Answer and MCQs & answers

# Short type Questions & Answers

## What is meant by search algorithm optimality?

Answer: If a search algorithm is optimal, then when it finds a solution it finds the best solution.

## What are the advantages of breadth-first search (BFS) over depth-first search (DFS)?

Answer: BFS is complete and optimal, while DFS is not guaranteed to halt when there are loops. •

## What is the advantage of DFS over BFS?

Answer: If m is the maximum path length and b is the branching factor, the space complexity for DFS is mb while for BFS it is b m

## What is meant by search algorithm completeness?

Answer: If an algorithm is complete, it means that if at least one solution exists then the algorithm is guaranteed find a solution in a finite amount of time.

## What is backtracking search?

A variant of depth-first search called backtracking search uses still less memory. Only one successor is generated at a time rather than all successors. Each partially expanded node remembers which successor to generate next.

## Define informed search strategy.

Informed search strategy is one that uses problem specific

knowledge beyond the definition of the problem itself that can find solutions more efficiently that an uniformed strategy.

## .What do you mean by Best First Search approach?

Best first search is an instance of the general TREE SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f (n).

## Define heuristic function.

Best first search typically use a heuristic function h (n) that estimates the cost of the solution from n.

h(n) = estimated cost of the cheapest path from node n to a goal node.

## 

## What do you mean by depth limited search?

## 

The problem of unbounded trees can be alleviated by supplying depth-first with a predetermined depth limit l. That is, nodes at depth l are treated as if they have no successors. This approach is called depth-limited search.

## What are the problems arises when knowledge of the states or actions is incomplete?

1.                                                 Sensor less problems

2.                                                 Contingency problems

3.                                                 Exploration problems

## What are the steps to evaluate an algorithm’s performance?

1.                                                 Completeness

2.                                                 Optimality

3.                                                 Time Complexity

4.                                                 Space Complexity

19.            Give examples for real world problems.

i)                                                 The route finding

ii)                                                  Touring

iii)                                                 Traveling sales person

iv)                                                Robot navigation

## What are the four components in problem?

i)                                                 Initial state

ii)                                                  Actions

iii)                                                 Goal state

iv)                                                 Path cost

## What is called as a uniformed search?

This term has no information about the number of steps or path cost current to goal state. They can distinguish a goal state from a non-goal state. Also known as blind search.

## What is called informed search?

It is one that uses problem-specific knowledge beyond the definition of the problem itself and can find solutions more efficiently than an uninformed strategy.

## 

## Give the complexity of a breath-first search.

The time complexity is O(b d), where, d is the depth and b is number at each level.

## What is iterative deepening search?

It is an abstract mathematical description. That maps any given percept sequence to an action

## Q What is AI Technique?

In the real world, the knowledge has some unwelcomed properties −

* Its volume is huge, next to unimaginable.
* It is not well-organized or well-formatted.
* It keeps changing constantly.

AI Technique is a manner to organize and use the knowledge efficiently in such a way that −

* It should be perceivable by the people who provide it.
* It should be easily modifiable to correct errors.
* It should be useful in many situations though it is incomplete or inaccurate.

AI techniques elevate the speed of execution of the complex program it is equipped with.

## Show how breadth-first-search and depth-first-search can be implemented using some appropriate pseudo-code. If you use another search algorithm as a sub-routine then show this algorithm in detail as well.

* Breadth-First Search Implementation

Function BREADTH-FIRST-SEARCH(*problem*) returns a solution or failure

Return GENERAL-SEARCH(*problem*,ENQUEUE-AT-END)

* Depth-First Search Implementation

Function DEPTH-FIRST-SEARCH(*problem*) returns a solution or failure

Return GENERAL-SEARCH(*problem*,ENQUEUE-AT-FRONT)

* Both searches call the GENERAL-SEARCH algorithm

Function GENERAL-SEARCH(problem, QUEUING-FN) returns a solution or failure

nodes = MAKE-QUEUE(MAKE-NODE(INITIAL-STATE[problem]))

Loop do

If nodes is empty then return failure

node = REMOVE-FRONT(nodes)

If GOAL-TEST[problem] applied to STATE(node) succeeds then return node

nodes = QUEUING-FN(nodes,EXPAND(node,OPERATORS[problem]))

End

End Function

## Q Describe any data structures that are used in implementing the BFS & DFS searches and describe the way in which they are used.

Both searches use a queue. The next node to expand is always taken from the front of the queue. The important aspect is that the search is implemented by adding nodes to the queue in different orders (thus the need for a QUEUING-FN in the above implementations).

Breadth-First Search adds new nodes to the end of the queue.

Depth-First Search adds new nodes to the beginning of the queue.

## Q What is the worst-case time and space complexity of the above BFS and DFS algorithms.

|  |  |  |
| --- | --- | --- |
| Evaluation | Breadth First | Depth First |
| Time | BD | BM |
| Space | BD | BM |

Where B = Branching Factor

D = Depth of Solution

M = Maximum Depth of the Search Tree

## Q Describe the terms complete and optimal with regards to evaluating search strategies? Are either depth-first-search or breadth-first-search complete? Are either of them optimal?

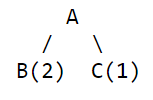
Complete : Is the search guaranteed to find a solution if there is one?

Optimal : Is the search guaranteed to find the optimal (cheapest) solution (however cheapest has been defined for that search problem)?

Breadth-First Search is both optimal and complete

Depth-First Search is neither optimal nor complete

## Q Give an example of a road navigation problem for which uniform cost search fails if an algorithm does the goal check on node generation as opposed to goal expansion.



where A is the start state and is not

a goal. B and C are both goals. B is generated first. B isn't optimal though; you have to fully expand A so that C will be considered.

# Descriptive Question & Answer

## Q What is the fewest possible number of nodes, in terms of b, d, and m, that will be generated for each of the following? Assume that all nodes have exactly b children. If the strategy can have both goal-test-at-generation-time and goal-test-at-expansion variants, answer for both.

## depth-first search with backtracking

## depth-first expansion

## breadth-first

## depth-first iterative deepening

We're looking for the fewest possible number of nodes generated for different algorithms. For all of the cases in this problem, the best case is where the (shallowest) goal node is the leftmost on level d. Answers vary depending on whether you check for goalness at expansion or generation time.

a) DFS with backtracking. This algorithm by nature does not do

expansion, so we check at generation time. d+1 nodes are generated.

(1 at level 0, 1 at level 1, 1 at level 2, ..., 1 at level d.)

b) Depth-first expansion. If the test is made at generation time:

1 + b + b + b + ... + 1 = 1 + (d-1)b + 1 = b(d-1)+2. If the test

is made at expansion time: 1 + b + b + ... + b = bd+1.

c) Breadth-First search: If test is made at generation time:

1 + b + b^2 + b^3 + ... b^(d-1) + 1. If the test is made at

expansion time: 1 + b + b^2 + b^3 + ... b^(d-1) + b.

Don't worry, you did not have to create closed form expressions

for these. :)

e) Iterative Deepening: We never expand nodes at level d, so:

1 +

(1 + b) +

(1 + b + b^2) +

(1 + b + b^2 + b^3) +

...

(1 + b + b^2 + ... + b^(d-1)) +

(1 + b + b^2 + ... + b^(d-1) + 1)

=

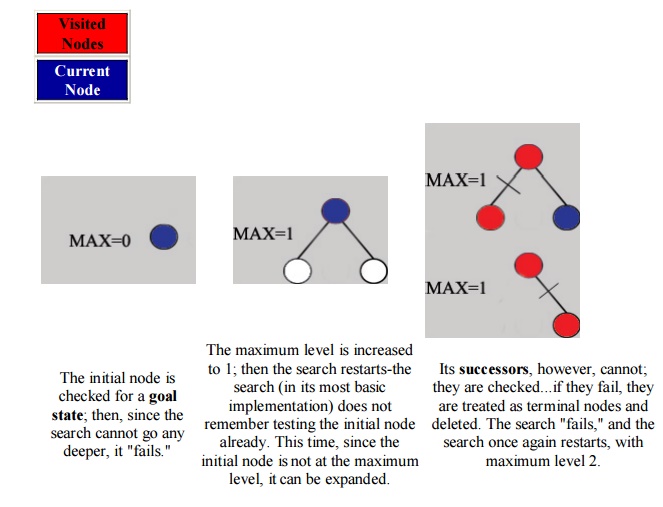
d + (d-1)b + (d-2)b^2 + ... + 2b^(d-1) + 1.

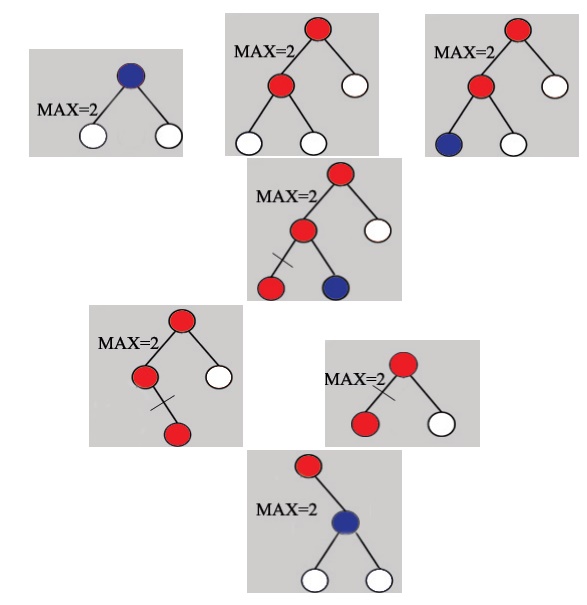
(As always, only one node at level d is generated. Plus, you weren't

required to produce a closed form expression.)

Q "While still an unintelligent algorithm, the iterative deepening search combines the positive elements of breadth-first and depth-first searching to create an algorithm which is often an improvement over each method individually." justify your answer.

An iterative deepening search operates like a depth-first search, except slightly more constrained--there is a maximum depth which defines how many levels deep the algorithm can look for solutions. A node at the maximum level of depth is treated as terminal, even if it would ordinarily have successor nodes. If a search "fails," then the maximum level is increased by one and the process repeats. The value for the maximum depth is initially set at 0 (i.e., only the initial node).





This continues until a solution is found.

An interesting observation is that the nodes in this search are first checked in the same order they would be checked in a breadth-first-search; however, since nodes are deleted as the search progresses, much less memory is used at any given time.

The drawback to the iterative deepening search is clear from the walkthrough--it can be painfully redundant, rechecking every node it has already checked with each new iteration. The algorithm can be enhanced to remember what nodes it has already seen, but this sacrifices most of the memory efficiency that made the algorithm worthwhile in the first place, and nodes at the maximum level for one iteration will still need to be re-accessed and expanded in the following iteration. Still, when memory is at a premium, iterative deepening is preferable to a plain depth-first search when there is danger of looping or the most efficient solution is desired.

## Show the working of Uniform-cost Search Algorithm also discuss its various terminologies

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

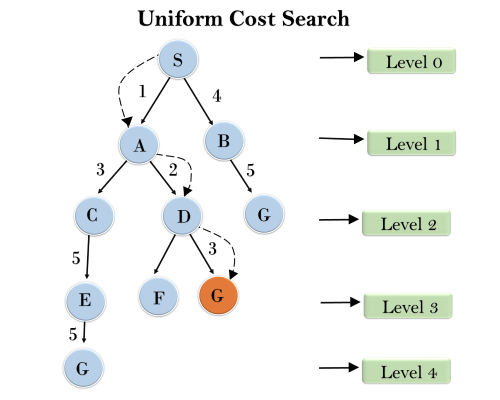
**Advantages:**

* Uniform cost search is optimal because at every state the path with the least cost is chosen.

**Disadvantages:**

* It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

### Example:



**Completeness:**

Uniform-cost search is complete, such as if there is a solution, UCS will find it.

**Time Complexity:**

Let C\* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C\*/ε+1. Here we have taken +1, as we start from state 0 and end to C\*/ε.

Hence, the worst-case time complexity of Uniform-cost search is**O(b1 + [C\*/ε])/**.

**Space Complexity:**

The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is **O(b1 + [C\*/ε])**.

**Optimal:**

Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

## Q define uninformed search strategies and list its types. Explain one of its type with example

•                      *Uninformed*strategiesuse only the information available in the problem definition

–  Also known as blind searching

–  *Uninformed search methods*:

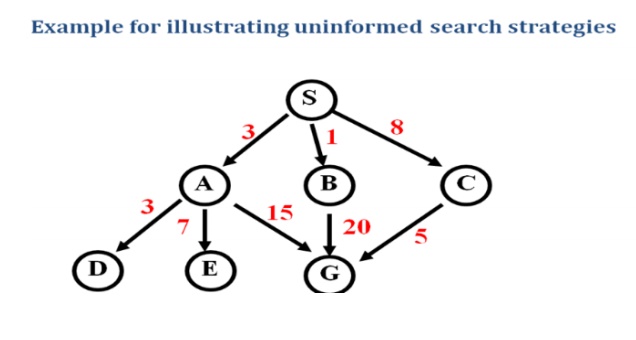
•                                                                                          Breadth-first search

•                                                                                          Uniform-cost search

•                                                                                          Depth-first search

•                                                                                          Depth-limited search

•                                                                                          Iterative deepening search



BREADTH-

FIRSTSEARCH

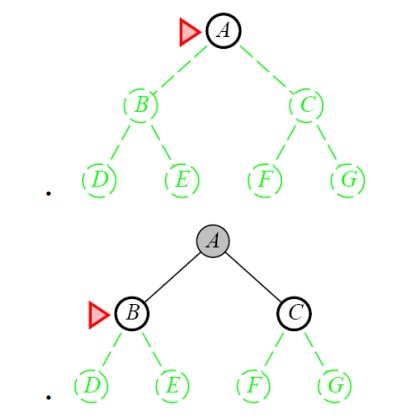
Definition:

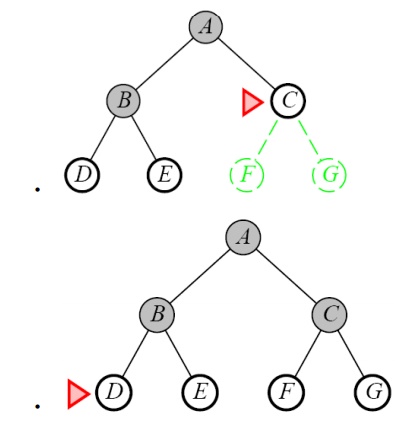
The root node is expanded first, and then all the nodes generated by the node are expanded.

•                   Expand the *shallowest* unexpanded node

•                   Place all new successors at the end of a FIFO queue

Implementation:





Properties of Breadth-First Search

•                   Complete

–  Yes if b (max branching factor) is finite

•                   Time

–  1 + b + b2 + … + bd + b(bd-1) = O(bd+1)

–  exponential in d

•                   Space

–  O(bd+1)

–  Keeps every node in memory

– This is the big problem; an agent that generates nodes at 10 MB/sec will produce

860 MB in 24 hours

•                   Optimal

–  Yes (if cost is 1 per step); not optimal in general

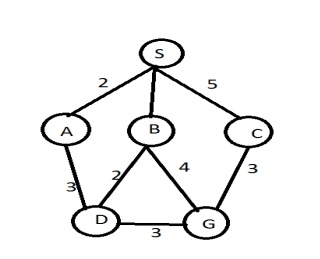
Lessons from Breadth First Search

The memory requirements are a bigger problem for breadth-first search than is execution time

•                   Exponential-complexity search problems cannot be solved by uniformed methods for any but the smallest instances

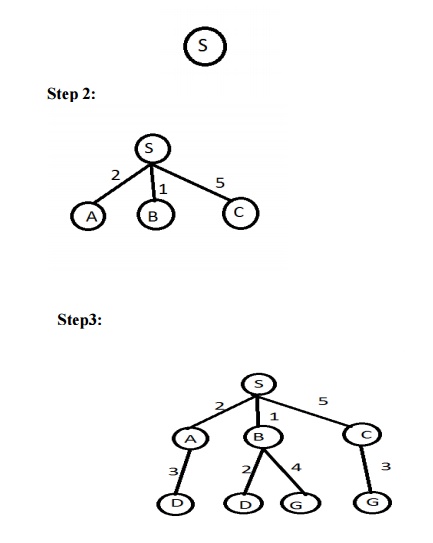
Ex: Route finding problem

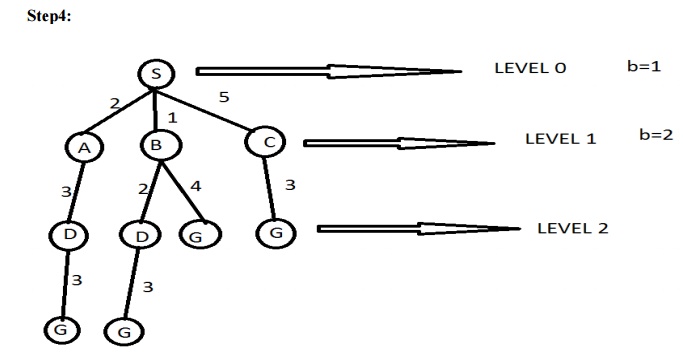
Given:



Task: Find the route from  S to G using BFS.

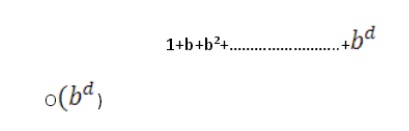
Step1:





Answer : The path in the 2nd depth level that is SBG (or ) SCG.

Time complexity



## Q How to solve Route finding problem using DFS.

DEPTH-FIRST SEARCH OR BACK TRACKING SEARCHING

Definition:

Expand one node to the depth of the tree. If dead end occurs, backtracking is done to the next immediate previous node for the nodes to be expanded

•                      Expand the *deepest* unexpanded node

•                      Unexplored successors are placed on a stack until fully explored

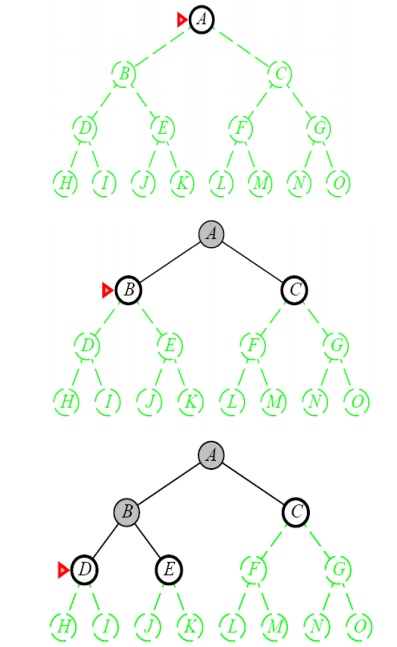
•                      Enqueue nodes on nodes in LIFO (last-in, first-out) order. That is, nodes used as a stack data structure to order nodes.

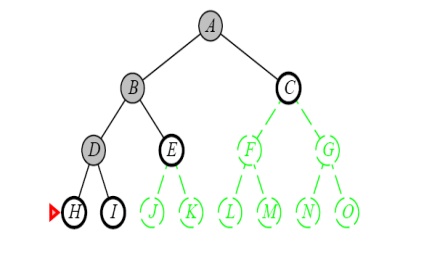
•                      It has modest memory requirement.

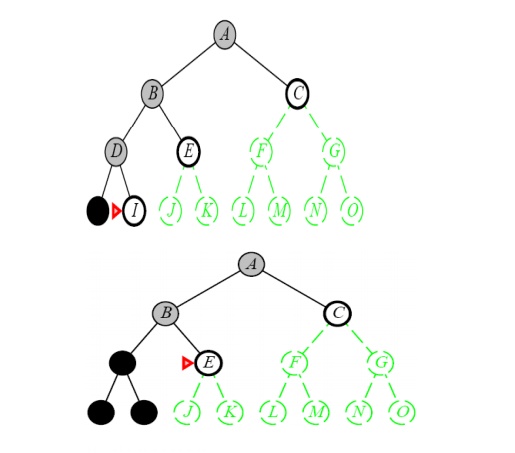
•                      It needs to store only a single path from the root to a leaf node, along with remaining unexpanded sibling nodes for each node on a path

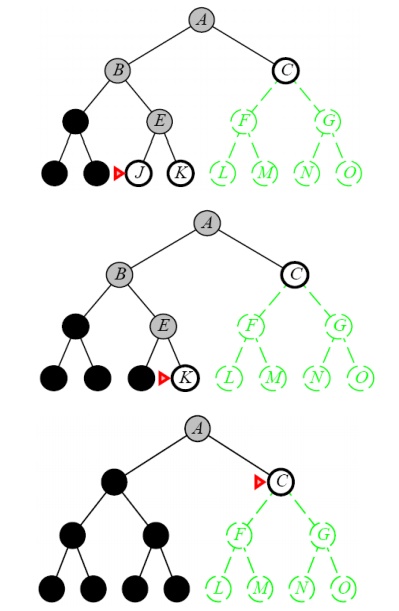
•                      Back track uses less memory.

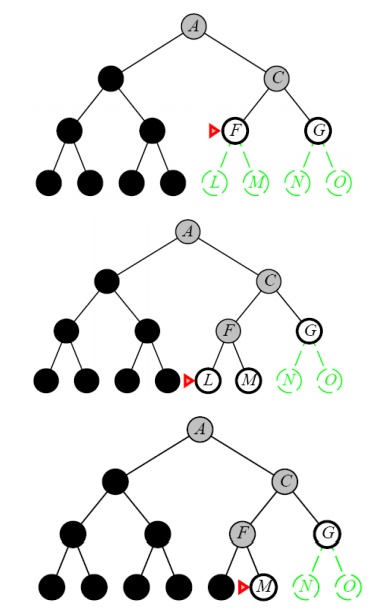
Implementation:











Properties of Depth-First Search

•                   Complete

–  No: fails in infinite-depth spaces, spaces with loops

•                                                                 Modify to avoid repeated spaces along path

–  Yes: in finite spaces

•                   Time

–  O(bm)

–  Not great if m is much larger than d

–  But if the solutions are dense, this may be faster than breadth-first search

•                   Space

–  O(bm)…linear space

•                   Optimal

–  No

•                   When search hits a dead-end, can only back up one level at a time even if the “problem” occurs because of a bad operator choice near the top of the tree. Hence, only does “chronological backtracking”

Advantage:

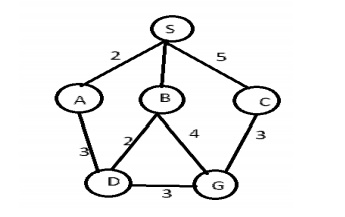
•                   If more than one solution exists or no of levels is high then dfs is best because exploration is done only a small portion of the white space.

Disadvantage:

•                     No guaranteed to find solution.

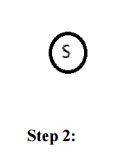
*Example:  Route finding problem*

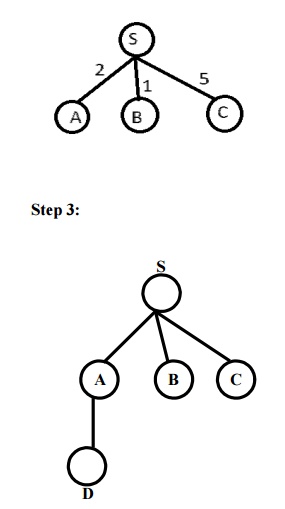
Given problem:

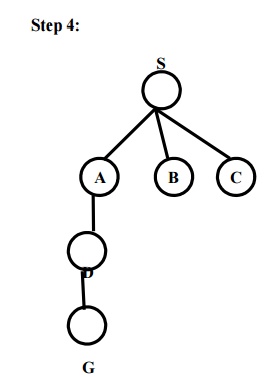


Task: Find a route between A to B

Step 1:







Answer: Path in 3rd level is SADG

## Explain Iterative deepening depth-first Search with example

The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.

This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.

The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

**Advantages:**

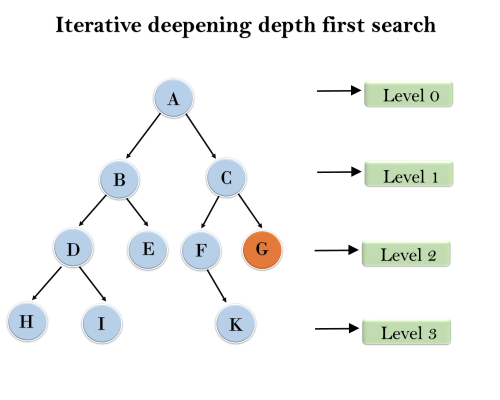
* Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

**Disadvantages:**

* The main drawback of IDDFS is that it repeats all the work of the previous phase.

### Example:

Following tree structure is showing the iterative deepening depth-first search. IDDFS algorithm performs various iterations until it does not find the goal node. The iteration performed by the algorithm is given as:



1'st Iteration-----> A   
2'nd Iteration----> A, B, C  
3'rd Iteration------>A, B, D, E, C, F, G  
4'th Iteration------>A, B, D, H, I, E, C, F, K, G  
In the fourth iteration, the algorithm will find the goal node.

**Completeness:**

This algorithm is complete is ifthe branching factor is finite.

**Time Complexity:**

Let's suppose b is the branching factor and depth is d then the worst-case time complexity is **O(bd)**.

**Space Complexity:**

The space complexity of IDDFS will be **O(bd)**.

**Optimal:**

IDDFS algorithm is optimal if path cost is a non- decreasing function of the depth of the node.

## Consider the search problem represented in Figure 1, where a is the start node and f is the goal node. Would you prefer DFS or BFS for this problem? Why?

# 

: If we were just running vanilla DFS (no pruning or loop checking) then we would prefer BFS, because DFS could get stuck in an infinite loop. Note that DFS is sensitive to the ordering of the nodes. If it explores to the left first it will get stuck in the loop, whereas if it explores to the right first it will find the goal very quickly.

Which sequences of paths are explored by BFS and DFS in this problem? Answer: DFS explores a − − > b − − > d − − > b − − > d and keeps oscillating between the two nodes b and d. BFS first adds a − − > b and a − − > e to the frontier. It expands ab and adds abd and abc to the frontier. Path ae is then expanded, adding aef to the frontier. Path abd is selected and removed from the frontier, and expanded so that abdb and abdc are added to the frontier. Path abc is selected and expanded, adding abcb and abcd to the frontier. Finally, aef is selected and the goal node is reached.

# MCQs & answers

1. Which search strategy is also known as blind search?

(a) Uninformed search

(b) Informed search

(c) Simple reflex search

(d) All of the above

2. Which search is implemented with an empty first-in-first-out queue?

(a) Depth-first search

(b) Breadth-first search

(c) Bidirectional search

(d) None of the above

3. When is the breadth-first search optimal?

(a) When there is less number of nodes

(b) When all step costs are equal

(c) When all step costs are unequal

(d) None of the above

4. How many successors are generated in the backtracking search?

(a) 1 (b) 2

(c) 3 (d) 4

5. Which search algorithm imposes a fixed depth limit on the nodes?

(a) Depth-limited search

(b) Depth-first search

(c) Iterative deepening search

(d) Bidirectional search

6. Which search implements stack operation for searching the states?

(a) Depth-limited search

(b) Depth-first search

(c) Breadth-first search

(d) None of the above

Answers

1. (a) 2. (b) 3. (b) 4. (a) 5. (a) 6. (b)