

Dr. Hashim Yasin





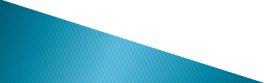
**Visited (Open) List:**

🞂 The *set of all leaf nodes available for expansion* at any given point is called the open list, (may be referred as **frontier**). 🞂 In general, a state is said to be visited if it has ever shown up in search a node.

🞂 The intuition is that ***we have visited*** them, but ***we have not generated its descendants***.

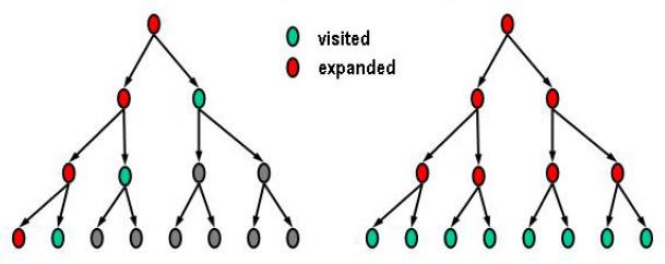
**Expanded (Closed, Explored) List:**

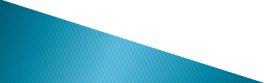
🞂 *Algorithms that forget their history are doomed to repeat it.* 🞂 The way to avoid exploring redundant paths is to remember where one has been.

🞂 To do this, we design **explored set** (also known as the **closed list**), which remembers every expanded node. 

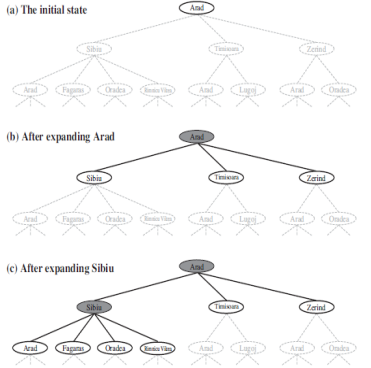
Dr. Hashim Yasin 2



**Depth First Search Breadth First Search**

****Dr. Hashim Yasin 3



Partial search trees for finding a route from Arad to 

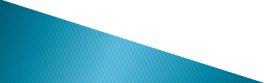
Bucharest.

❑ Nodes that have been **visited but not yet**

**expanded** are outlined in **bold**;

❑ Nodes that have been **expanded are shaded**;

❑ Nodes that have **not been visited** are shown in faint dashed lines.

Dr. Hashim Yasin4







🞂 **Uninformed search** also called **blind search**

🞂 The strategies have **no additional information** about the states beyond that provided in the problem definition. 🞂 ***Use the information only provided*** in the problem definition.

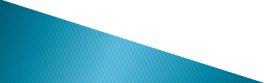
🞂 Breadth-first search

🞂 Depth-first search

◦ Depth-limited search

◦ Iterative deepening search

🞂 Uniform cost search

🞂 Bidirectional search 

Dr. Hashim Yasin 6







**When we need the final path, we have to continue till the goal state is expanded.**

Dr. Hashim Yasin 8





| **Expanded** |
| --- |
| **S** |
| **A,S** |
| **B,A,S** |
| **C,B,A,S** |
| **D,C,B,A,S** |
| **G,D,C,B,A,S** |

**When we need the final path, we have to continue till the goal state is expanded.**

**Tree Traversal = S, A, B, C, D, G**

**The Final path = S, B, G**

Dr. Hashim Yasin 9



Dr. Hashim Yasin 10



**Completeness**: Yes, (if �� is finite)

**Time complexity**: Imagine ***searching a uniform tree*** where every state has �� successors.

◦ The root of the search tree generates �� nodes at the first level, each of which generates �� more nodes,

◦ for a total of ��2at the second level, yielding ��3 nodes at the third level, and so on. Now suppose that the solution is at depth ��



🞂 If the algorithm were to apply the **goal test** to nodes when selected for **expansion**, rather than when visited,

◦ the whole *layer of nodes at depth* �� *would be expanded before the goal was detected* and

◦ the time complexity would be ��(����+��).

Dr. Hashim Yasin 11



**Space complexity:** For breadth-first graph search in particular, every node generated remains in memory. 🞂 There will be ��(����−��) nodes in the explored set 🞂 ��(����) nodes in the frontier,

**Optimality**: Yes, if the cost = 1 per step

*The* ***memory requirement is*** *a bigger problem for breadth first search than the execution time.*

**Dr. Hashim Yasin 12



Dr. Hashim Yasin 13



🞂 The root node is expanded by **first-in-first-out (FIFO), Queue data structure.**

🞂 **Complete**: find the solution eventually

🞂 **Optimal**: if the step cost is 1

**Disadvantages:**

🞂 The branching factor of a node is large,

🞂 The **space complexity and time complexity** are enormous for even small instances (e.g., chess)

Dr. Hashim Yasin 14







❑ Pick first element of Q

❑ Add path extensions **in front** to Q

| 1 | **Q**  (S) |
| --- | --- |
| 2 | **(A S) (B S)** |
| 3 | **(C A S) (D A S)** (B S) |
| 4 | (D A S) (B S) |
| 5 | **(C D A S) (G D A S)** (B S) |
| 6 | **(G D A S)** (B S) |

❑ **Blue Color represents added paths**

****

❑ Path has been shown in **reversed order**, node state is the first entry. ❑ **Tree Traversal = S, A, C, D, C, G** 

❑ **The Final path = S, A, D, G**

Dr. Hashim Yasin 16



❑ Pick first element of Q

❑ Add path extensions **in front** to Q

| 1 | **Q**  (S) | **Visited**  S | **Expanded** S |
| --- | --- | --- | --- |
| 2 | **(A S) (B S)** | A, B, S | A, S |
| 3 | **(C A S) (D A S)** (B S) | C, D, B, A, S | C, A, S |
| 4 | (D A S) (B S) | C, D, B, A, S | D, C, A, S |
| 5 | **(G D A S)** (B S) | G, C, D, B, A, S | G, D, C, A, S |

❑ **Blue Color represents added paths**

****

❑ Path has been shown in **reversed order**, node state is the first entry. ❑ **Tree Traversal = S, A, C, D, G**

****❑ **The Final path = S, A, D, G**

Dr. Hashim Yasin 17



**Numbers indicate order pulled off of Q (expanded)**

**Dark blue fill = Visited and Expanded** 

**Light gray fill = Visited**

Dr. Hashim Yasin 18

**DFS Example 2**

****Dr. Hashim Yasin 19



🞂 Depth-first search always expands the ***deepest node*** 🞂 Implemented with a **last-in-first-out (LIFO)** strategy, also known as a ***stack***.

🞂 As an alternative to the ***Graph-Search-style***

implementation, depth-first search is implemented with a **recursive function** that calls itself on each of its children in turn.

Dr. Hashim Yasin 20



**Completeness:**

🞂 The *graph-search version*, which *avoids repeated states and redundant paths*, is **complete** in finite state spaces because it will eventually expand every node.

🞂 The *tree-search version*, on the other hand, it is ***not* complete**.

**Optimality:**

🞂 It doesn't guarantee the best solution.

Dr. Hashim Yasin 21



**Start** 

**Goal**

**A simplified road map of part of Romania.**

Dr. Hashim Yasin 22

Dr. Hashim Yasin 23



It overcomes **time and space** complexities.

**Time:**

🞂 ��(����): terrible if �� (maximum length of the depth) is much larger than the size of the state space.

🞂 *If solutions are dense, may be much faster than breadth-first*.

**Space:**

🞂 ��(����)***,*** i.e., linear space!

🞂 A variant of depth-first search with **backtracking search** uses still less memory ��(��)***.*** 

Dr. Hashim Yasin 24





The depth-first search would require **156 kilobytes** instead of 10 Exabyte at depth *d = 16*, a factor of **7 trillion times less space**.

Dr. Hashim Yasin 25







🞂 The problem of depth first search can be alleviated by supplying depth-first search with a **pre-determined depth limit** ��*.*

�� < ��

🞂 Nodes at depth �� are treated as if they have no successors. **Completeness:**

🞂 It *may be incomplete* if we choose �� < ��**,** that is, the shallowest **goal is beyond the depth limit** ��**.**

**Optimality:**

🞂 The depth-limited search *may be non-optimal* if we choose �� > ��.

Dr. Hashim Yasin 27



**Time complexity:** ��(����)

**Space complexity:** ��(����)

🞂 *Depth-first search can be viewed as a special case of depth-limited search with*

�� = ∞

🞂 Depth limited search can be based on knowledge of the problem.

◦ For the most problems, however, **we will not know a good depth limit until** we have solved the problem.

Dr. Hashim Yasin 28







🞂 No selection of the best depth limit

🞂 It tries all possible depth limits:

◦ first 0, then 1, 2, and so on

🞂 Combines the benefits of depth-first and breadth-first search,

The cutoff value indicates that there is no any solution within the depth limit.

Dr. Hashim Yasin 30



🞂 ℓ = 0:



🞂 ℓ = 1:

Dr. Hashim Yasin 31



🞂 ℓ = 2:

Dr. Hashim Yasin 32



🞂 ℓ = 3:

Dr. Hashim Yasin 33

if �� is finite

**IDS suitable for the problem**

• having a large search space

• and the *depth of the solution is not known*

**Dr. Hashim Yasin 34







❑ Finding a path in the given grid using DFS. ❑ The order of the actions are **up**, **left**, **right**, then **down**. 

❑ Maintain the visited list to avoid looping.

❑ Number the square according to the

traversal.

Dr. Hashim Yasin 36





Dr. Hashim Yasin 37



🞂 **Artificial Intelligence,** A Modern Approach **Stuart J. Russell and Peter Norvig**

◦ **Chapter 3.**

****Dr. Hashim Yasin 38