Uninformed Search (Ch. 3-3.4)

Breadth first search

BFS can be implemented by using a simple FIFO (first in, first out) queue to track the fringe/frontier/unexplored nodes

Metrics for BFS:

Complete (i.e. guaranteed to find solution if exists)

Non-optimal (unless uniform path cost)

Time complexity = $O(b^d)$

Space complexity = $O(b^d)$

Uniform-cost search

UCS is..

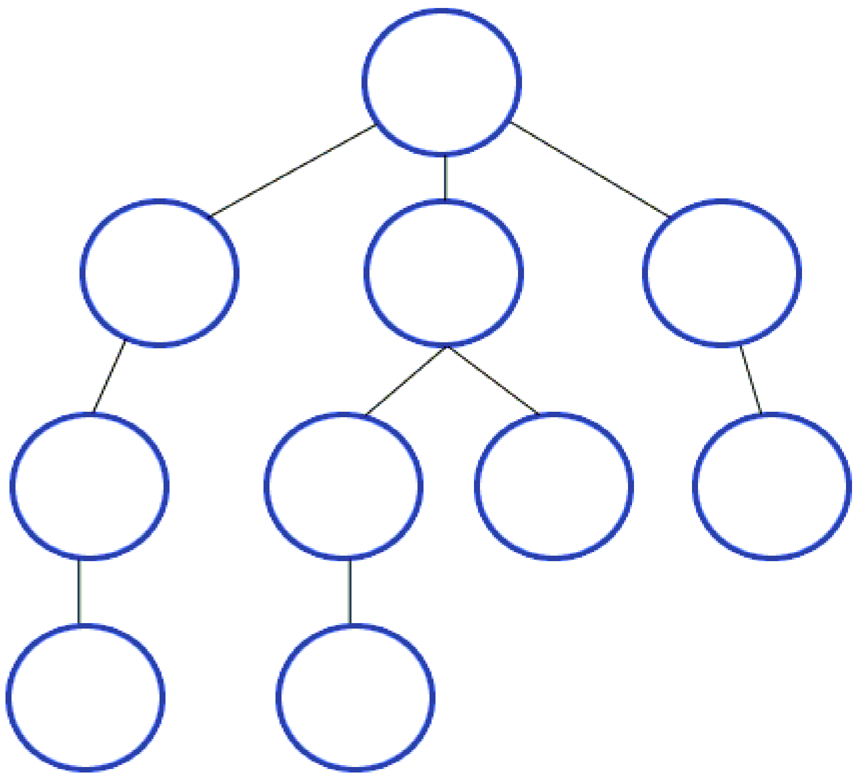
1. Complete (if costs strictly greater than 0)
2. Optimal

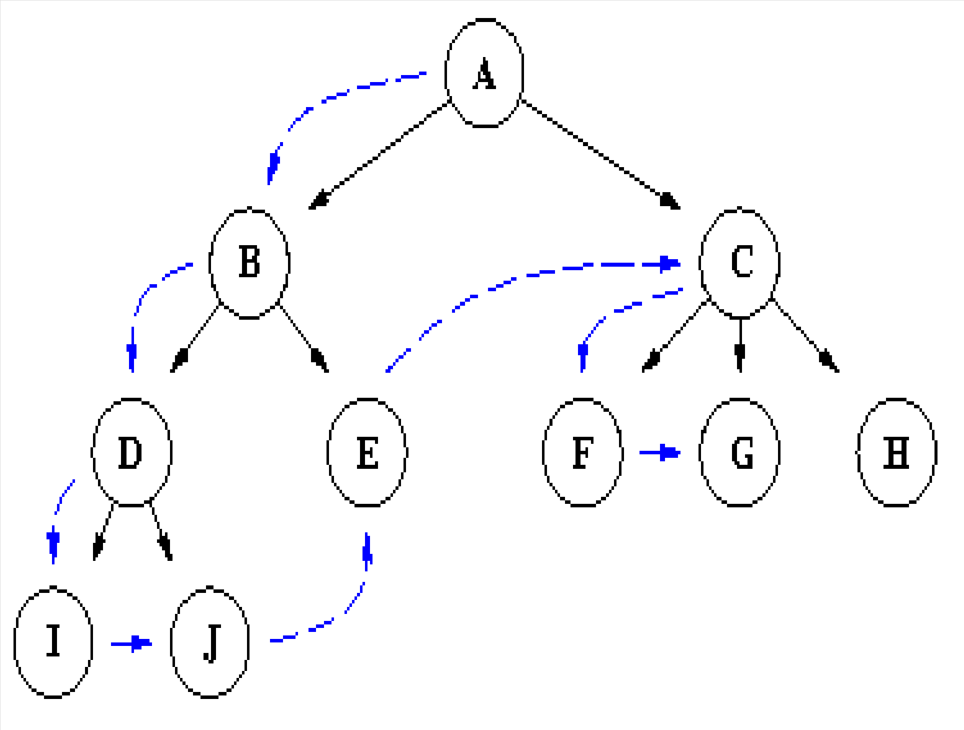
However....

3&4. Time complexity = space complexity = $O(b^{1+C^*/\min(\text{path cost})})$, where C* cost of optimal solution (much worse than BFS)

Depth first search

DFS is same as BFS except with a FILO (or LIFO) instead of a FIFO queue





Depth first search

Metrics:

1. Might not terminate (not complete) (e.g. in vacuum world, if first expand is action L)
2. Non-optimal (just... no)
3. Time complexity = $O(b^m)$
4. Space complexity = $O(b*m)$

Only way this is better than BFS is the space complexity...

Depth limited search

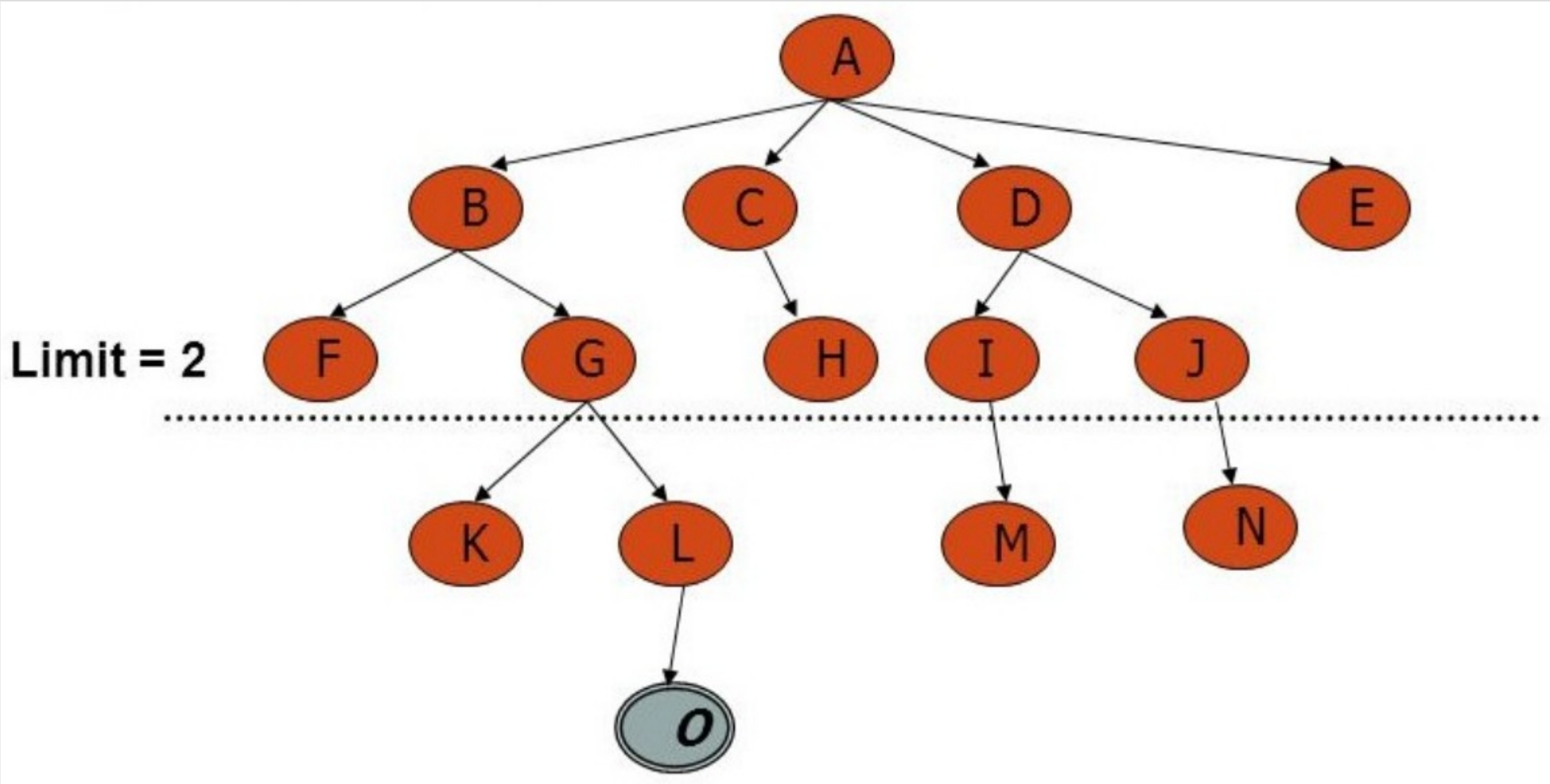
DFS by itself is not great, but it has two (very) useful modifications

Depth limited search runs normal DFS, but if it is at a specified depth limit, you cannot have children (i.e. take another action)

Typically with a little more knowledge, you can create a reasonable limit and makes the algorithm correct

Depth limited search

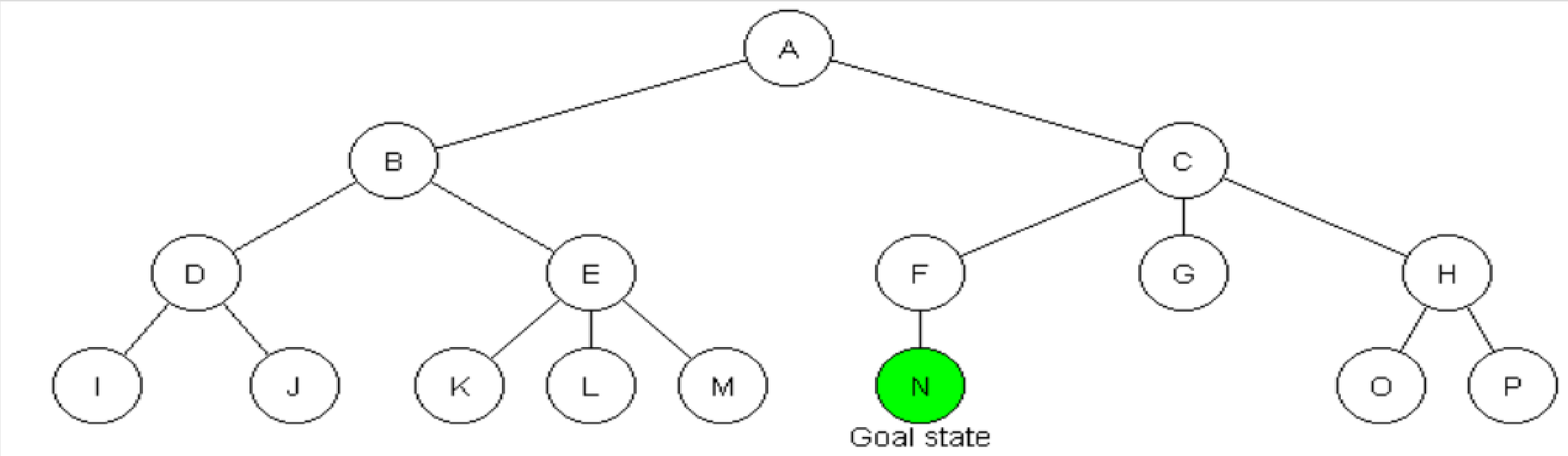
However, if you pick the depth limit before d, you will not find a solution (not correct, but will terminate)



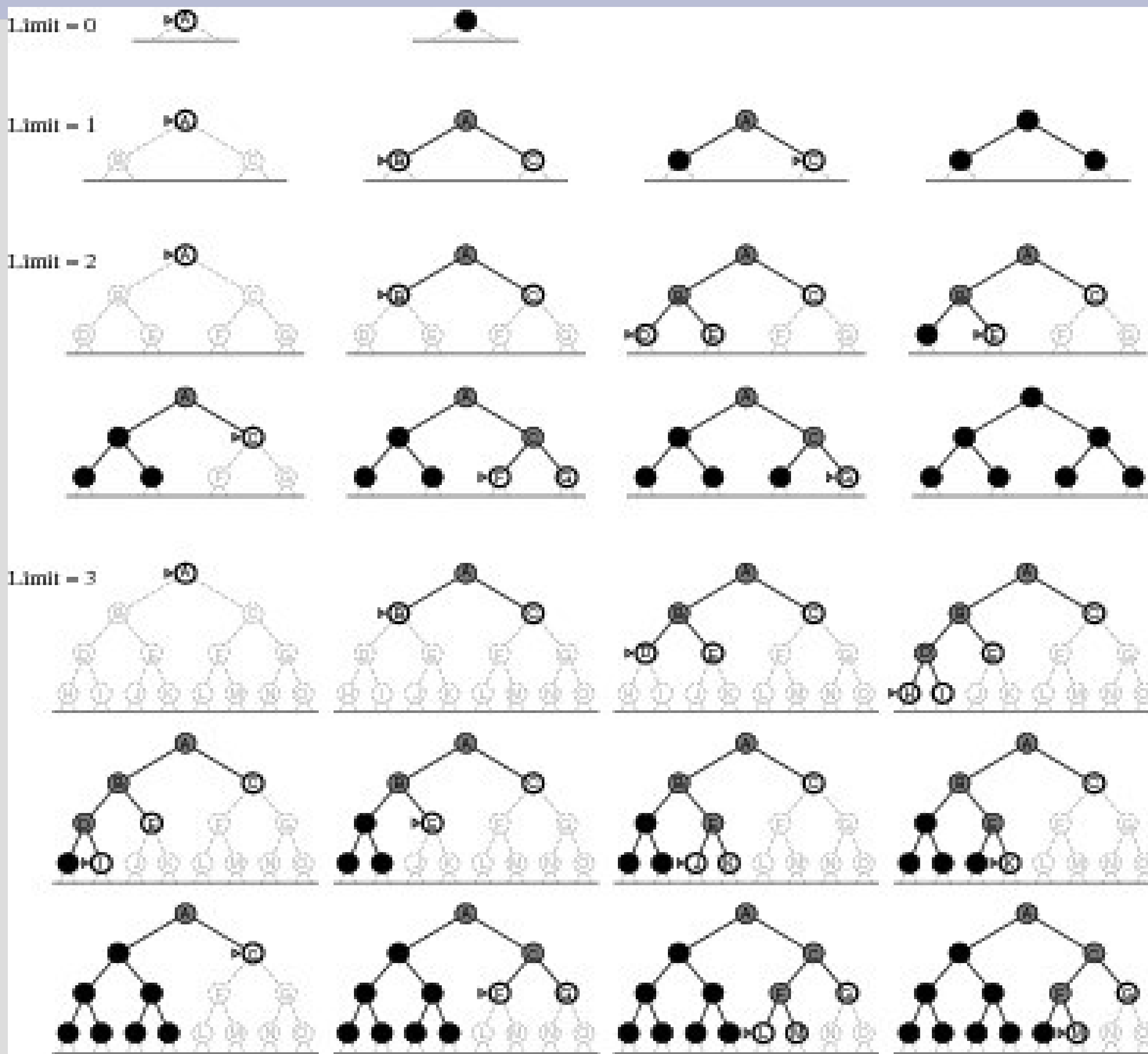
Iterative deepening DFS

Probably the most useful uninformed search is <u>iterative deepening DFS</u>

This search performs depth limited search with maximum depth 1, then maximum depth 2, then 3... until it finds a solution



Iterative deepening DFS



Iterative deepening DFS

The first few states do get re-checked multiple times in IDS, however it is not too many

When you find the solution at depth d, depth 1 is expanded d times (at most b of them)

The second depth are expanded d-1 times (at most b^2 of them)

Thus $(d+1)\cdot 1 + d\cdot b + (d-1)\cdot b^2 + ... + 1\cdot b^d = O(b^d)$

Iterative deepening DFS

Metrics:

1. Complete
2. Non-optimal (unless uniform cost)
3. $O(b^d)$
4. $O(b*d)$

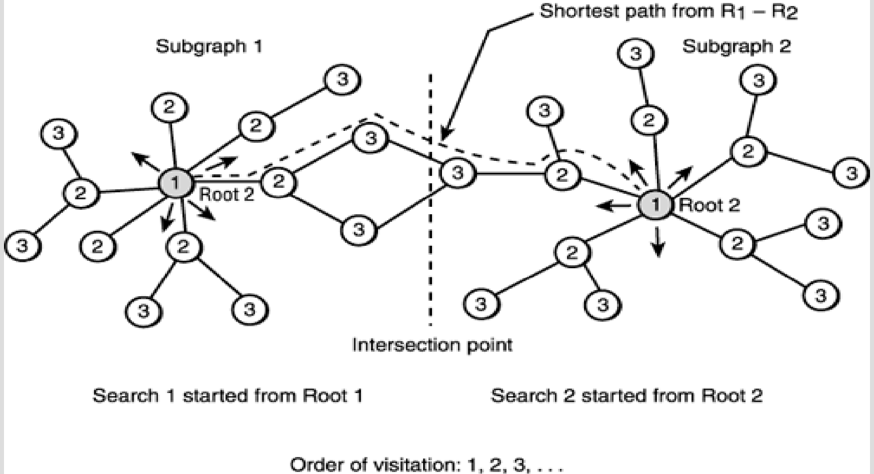
Thus IDS is better in every way than BFS (asymptotically)

One of the best uninformed searches

Bidirectional search

<u>Bidirectional search</u> starts from both the goal and start (using BFS) until the trees meet

This is better as $2*(b^{d/2}) < b^d$
(the space is much worse than IDS, so only applicable to smaller problems)



Bidirectional search

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

Summary of algorithms
Fig. 3.21, p. 91

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening DLS	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{\lfloor 1+C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{\lfloor 1+C^*/\varepsilon \rfloor})$	O(bm)	O(bl)	O(bd)	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Generally the preferred uninformed search strategy

There are a number of footnotes, caveats, and assumptions.
See Fig. 3.21, p. 91.
[a] complete if b is finite
[b] complete if step costs $\geq \varepsilon > 0$
[c] optimal if step costs are all identical
(also if path cost non-decreasing function of depth only)
[d] if both directions use breadth-first search
(also if both directions use uniform-cost search with step costs $\geq \varepsilon > 0$)