EC-350 AI and Decision Support Systems

Week 4

Solving Problems by Searching

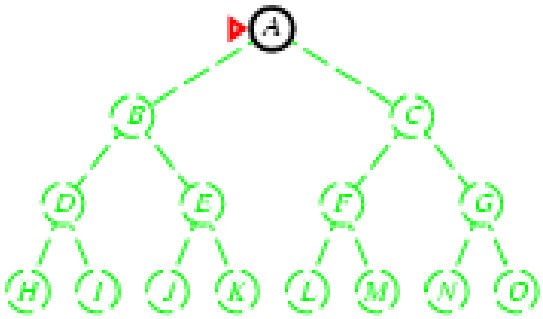
Dr. Arslan Shaukat

NUST

Acknowledgement: Lecture slides material from Stuart Russell

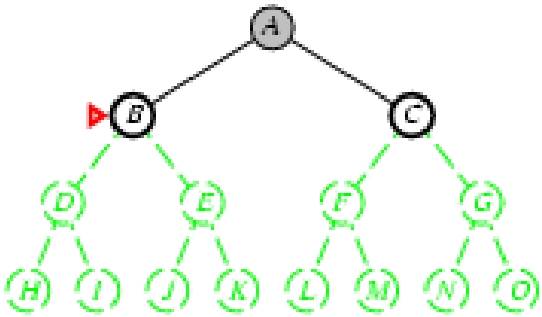
Depth-first Search

- Expand deepest unexpanded node
- Implementation:
  - *Put the expanded node at the front of the fringe*
  - *use fringe as a Stack, LIFO*
  - *Fringe = A*



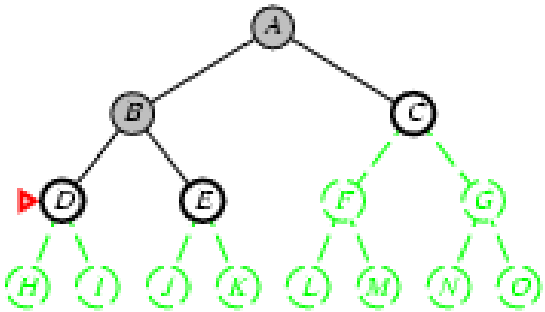
Depth-first Search

- Expand deepest unexpanded node
- Check if A is goal, it is not, expand A
- Fringe = B,C



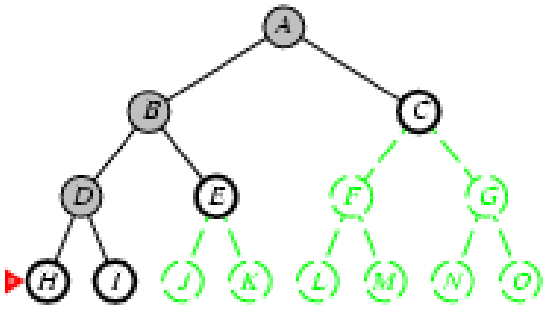
Depth-first Search

- Check if B is goal, it is not, expand B
- Fringe = D,E,C



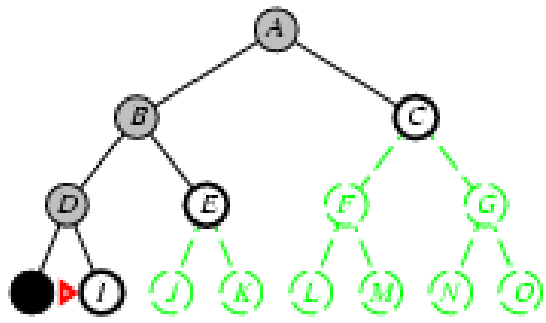
Depth-first Search

- Check if D is goal, it is not, expand D
- Fringe = H, I, E, C



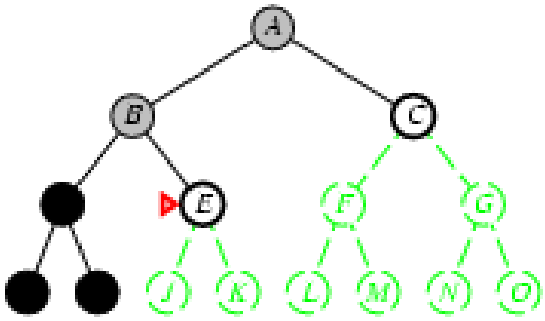
Depth-first Search

- Both H and I has not successors so remove them if it is not goal
- Fringe = E, C



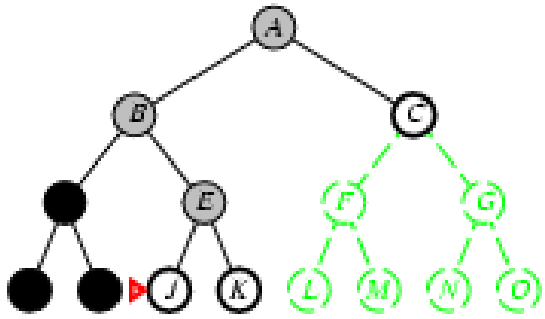
Depth-first Search

- Check if E is goal, it is not, expand E
- Fringe = J, K, C



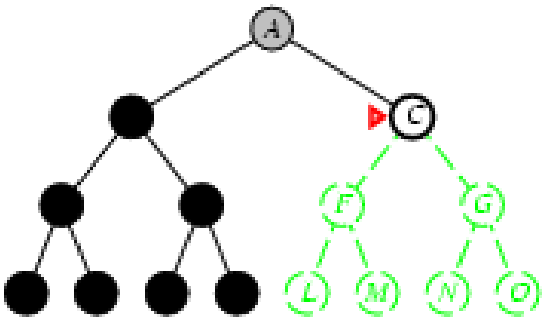
Depth-first Search

- J and K do not have successors neither is any of them goal so remove them from fringe one by one
- Fringe = C



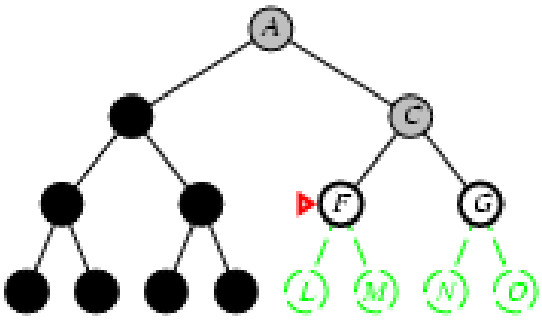
Depth-first Search

- Check if C is goal, it is not, expand C
- Fringe = F, G



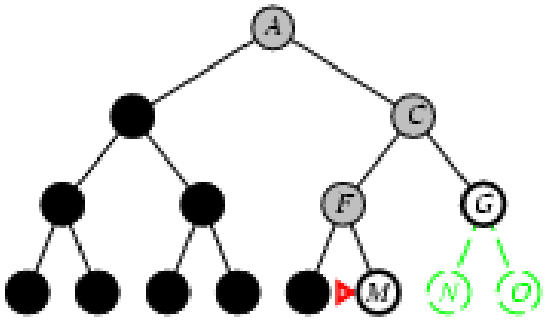
Depth-first Search

- Check if F is goal, it is not, expand F
- Fringe = L, M, G



Depth-first Search

- Both L, M are not goal, neither they have any successor
- Remove L, M from fringe
- Fringe = G
- Check if G is goal, it is
- The search terminates



Properties of Depth-first Search

- Complete? No: fails in infinite-depth spaces, spaces with loops
  - *Complete in Finite state space*
  - *Modify to avoid repeated states along path*
- Time? $O(b^m)$: terrible if m is much larger than d
- Space? $O(bm)$
- Optimal? No

Depth-Limited Search

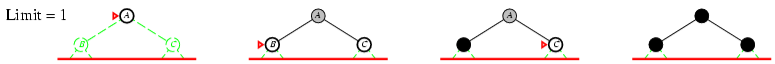
- Depth-first search with depth limit l, i.e., nodes at depth l have no successors
- Introduces an additional source of incompleteness if we choose $l < d$
- DFS can be viewed as a special case of depth-limited search with $l = \infty$.

Iterative Deepening DFS

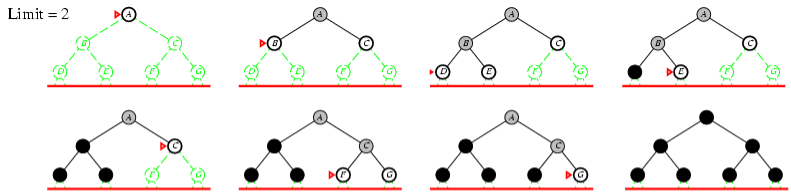
- General search strategy used with DFS
- We look for the solution at first level and then gradually increase the depth limit
- First 0, then 1, then 2 and so on
- Combines the benefits of BFS and DFS

Iterative Deepening Search $l = 0$

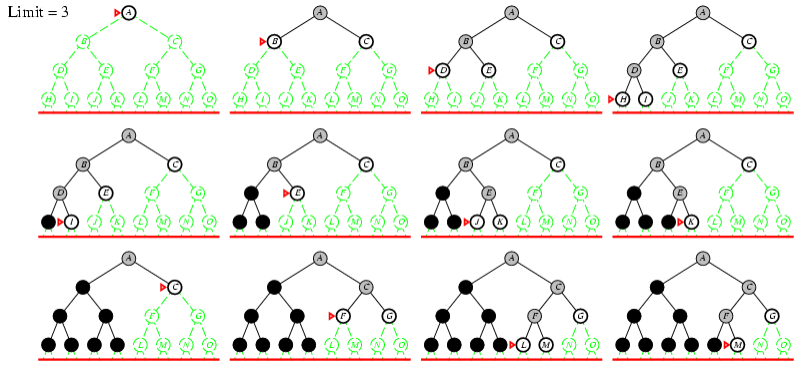
Iterative Deepening Search $l = 1$



Iterative Deepening Search *l* =2



Iterative Deepening Search *l* =3



Properties of Iterative Deepening Search

- Complete? Yes
- Time? $(d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- Space? $O(bd)$
- Optimal? Yes, if step cost = 1

Iterative Deepening Search

- No of nodes generated in a Breadth-first search to depth d with branching factor b:
  $N_{BFS} = b^1 + b^2 + \ldots + b^d$
- No of nodes generated in an iterative deepening search:
  $N_{IDS} = (d)b^1 + (d-1)b^2 + \ldots + (1)b^d$
- For $b = 10$, $d = 5$,
  - $N_{BFS} = 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 = 111{,}100$
  - $N_{IDS} = 50 + 400 + 3{,}000 + 20{,}000 + 100{,}000 = 123{,}450$
- In general IDS is the preferred uninformed search method when there is a large search space and the depth of the solution is not known

Example

- Find path from S to E
- Use BFS
- DFS
- UCS