Artificial Intelligence
CSE 5/7320

Eduardo Blanco

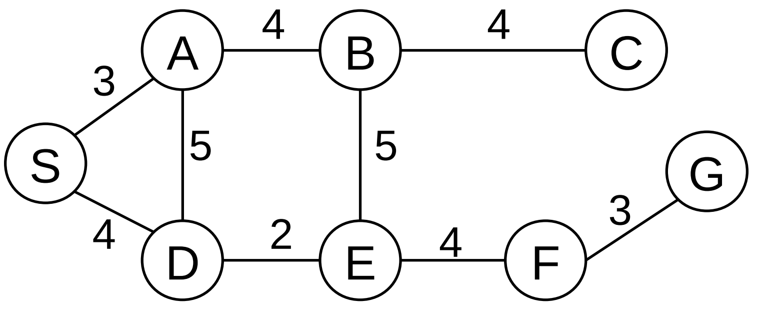
January 29, 2013

Search

Search

- We have seen how to define problems
  - Initial state, actions and transition model implicitly define the state space
    - Directed graph
      - nodes are states
      - edges are actions
  - For serious problems you cannot afford to build the whole graph

Search example

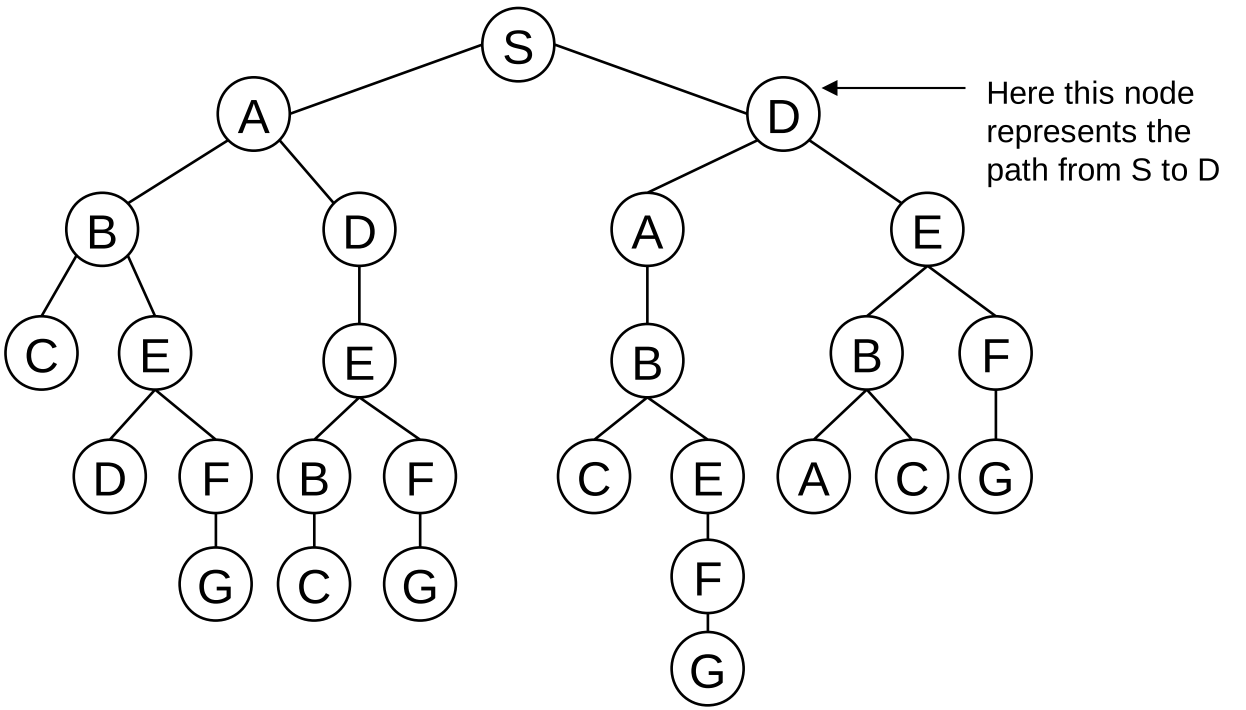


This is a highway map with cities and distances between them. The problem is to find a path (any path) from S to G (from initial/start to goal)

The problem of finding the shortest (optimal path) will be considered later.

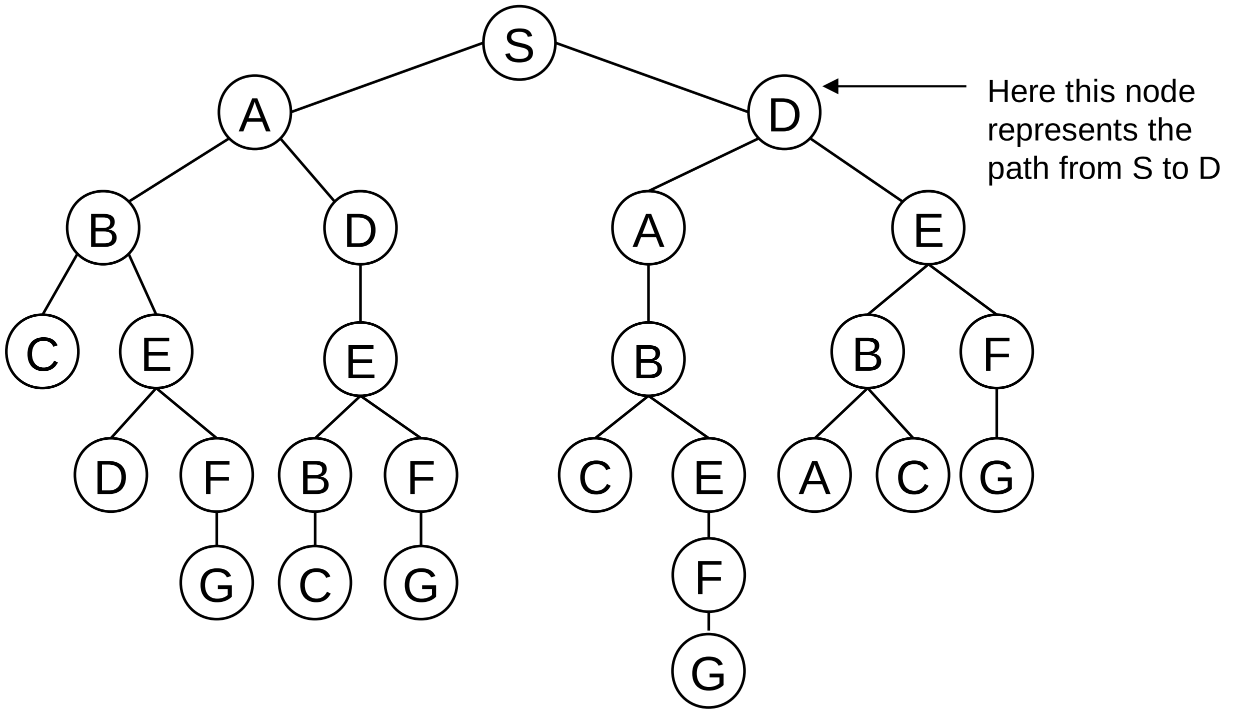
Search Tree

We build a search tree that is our search space.
A search tree is not a graph.
A search tree is constructed such that paths are not redundant.



Nodes are paths in the graph, and branches connect paths.

Search Tree



- Terminology
  - Expanding a node: consider actions available from the node
  - Frontier: leaf nodes available for expansion
- Search strategy decides which node to expand

Search Algorithms

- We will a few search algorithms:
  - Uninformed (aka blind): do not use any information about the problem
  - Informed: use an utility function to estimate how good a node is
    - Is it closer to the goal?
- Measuring performance of search algorithms:
  - Completeness
    - guaranteed to find a solution?
  - Optimality
    - guaranteed to find optimal solution?
  - Time Complexity
    - how long does it take?
  - Space Complexity
    - how much memory does it need?

Search Algorithms

- Time and space complexity are measured in terms of:
  - *b*, branching factor: maximum number of successors of any node
  - *d, depth of least-cost solution*
  - *m*, maximum depth of state space
    - (could be infinity)

Search Algorithms

- Basic search
  - Blind/Uninformed search
    - Breadth-first
    - Uniform-cost
    - Depth-first
    - Depth-limited
    - Iterative Deepening
  - Heuristic/Informed search
    - Hill climbing
    - Beam search
    - Best-first search
    - A* search

Breadth-First Search

- Steps:
  - Expand root
  - Expand all successors of the root
  - Expand the successors of the successors of the root,
  - And so on

- [Example on previous search tree]

- [Example on 8-puzzle]

Breadth-First Search

- Complete?
- Optimal?
- Time?
- Space?

Breadth-First Search

- Complete?
  - Yes (if *b* is finite)
- Optimal?
  - Yes (if cost = 1 per step, i.e., all actions cost the same)
- Time?
  - $b + b^2 + b^3 + \ldots + b^d = O(b^d)$ [d is the depth of the solution]
- Space?
  - $O(b^d)$ (keeps every node in memory)

Depth-First Search

- Steps:
  - Expand the deepest node in the current frontier of the search tree
- [Example on previous search tree]
- [Example on 9-puzzle]

Depth-First Search

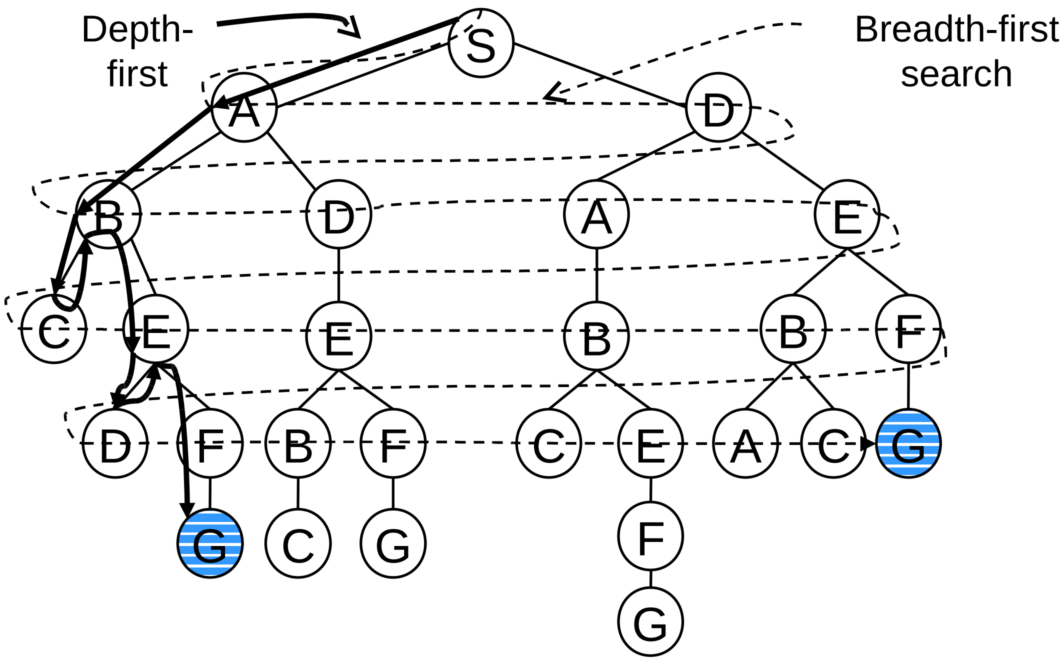
- Complete?
- Optimal?
- Time?
- Space?

Depth-First Search

- Complete?
  - No, fails in infinite-depth spaces, spaces with loops
    - Avoid repeated states (tree vs. graph search)
  - Complete in finite spaces
- Optimal?
  - No
- Time?
  - $O(b^m)$ [m is maximum depth of any node]
- Space?
  - $O(bm)$ [the only advantage over BFS]

- Note that *m* may be much larger than *d*

BFS vs. DFS

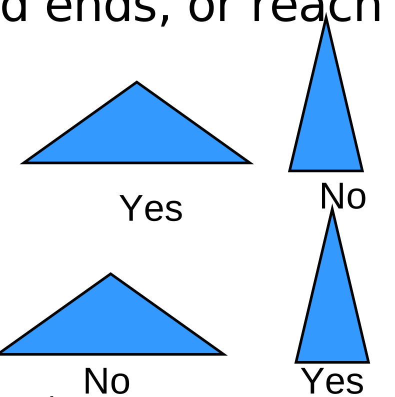


Depth-first is recommended when all paths reach dead ends, or reach the goal in reasonable number of steps

d—depth of tree is small

Breadth first search is better for trees that are deep.

Not good for large *b*.



Uniform-cost search

- BFS expands the shallowest unexpanded node
  - It is optimal if all actions cost the same
- What if actions have different costs?
  - (very) similar idea: expand the node with the lowest path cost
  - *g(n)* is the path cost of node *n*
    - Cost from initial state until *n*
- If all actions cost the same (each edge costs 1), BFS and uniform-cost search are exactly the same

Depth-limited Search

- The issue of DFS is that it may go too deep
  - and *"get lost"*

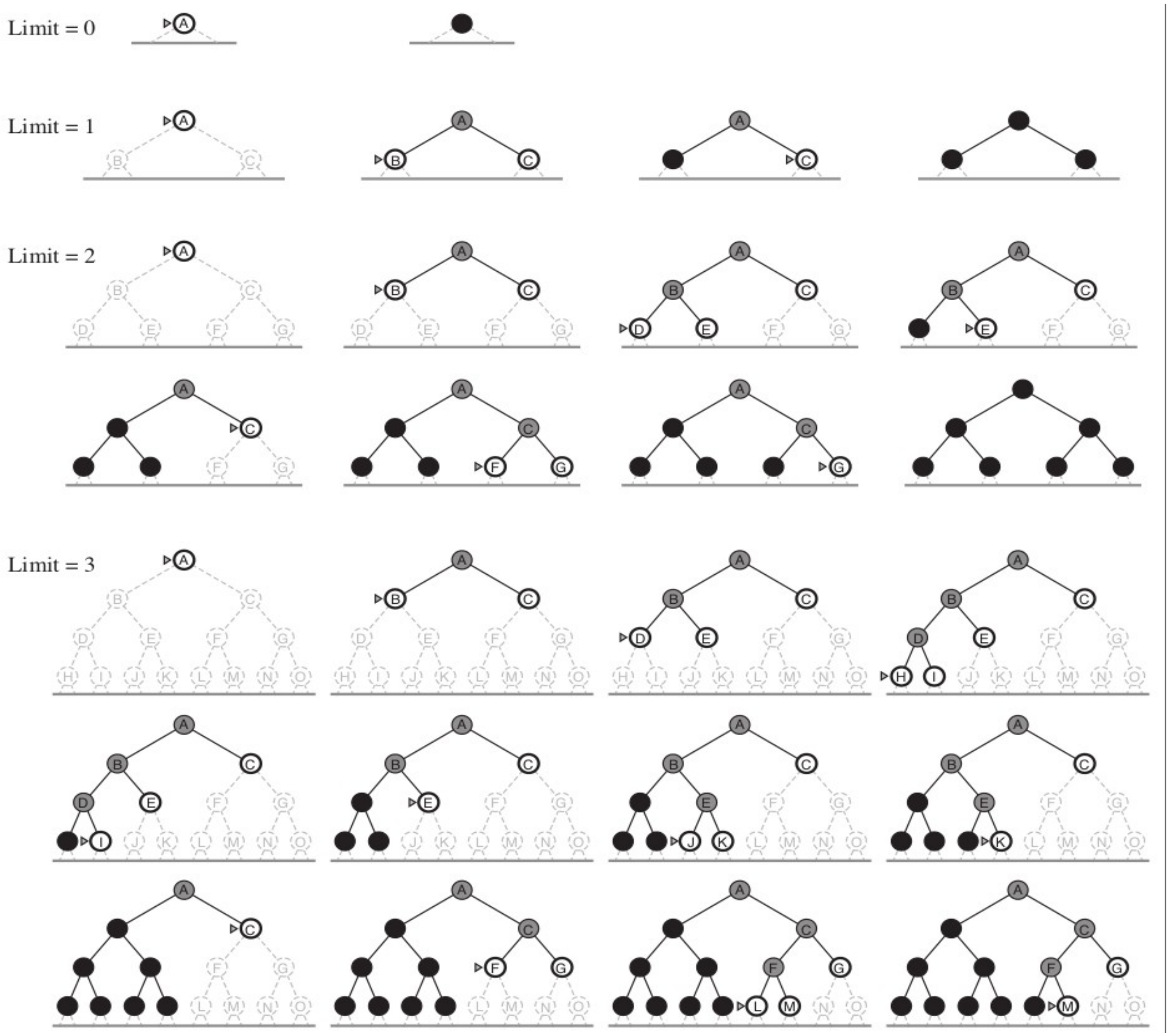
- Easy solution: impose a limit *l* on depth: depth-limited search (DLS)
  - Problem if the solution is actually deeper than *l*
    - We may not find the solution

- DLS is exactly the same than DFS, the only difference is that we stop at a certain depth *l*

Iterative deepening search

- Gradually increase the limit for Depth-limited Search until a goal is found
- [Example with search tree]

Iterative deepening search



Properties of iterative deepening search

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

Iterative deepening search

- Number of nodes generated in a depth-limited search to depth *d* with branching factor *b*:

$$N_{DLS} = b^1 + b^2 + \dots + b^{d-2} + b^{d-1} + b^d$$

- Number of nodes generated in an iterative deepening search to depth *d* with branching factor *b*:

$$N_{IDS} = d\,b^1 + (d-1)b^2 + \dots + 3b^{d-2} + 2b^{d-1} + 1b^d = O(b^d)$$

- For b = 10, d = 5,

  $N_{DLS} = 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 = 111{,}110$

  $N_{IDS} = 50 + 400 + 3{,}000 + 20{,}000 + 100{,}000 = 123{,}450$

- Overhead = (123,450 - 111,110)/111,110 = 11%

IDS is iterative-deepening search
DLS is depth-limited search

Comparing Search Algorithms

Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	O(bm)	O(bl)	O(bd)
Optimal?	Yes	Yes	No	No	Yes