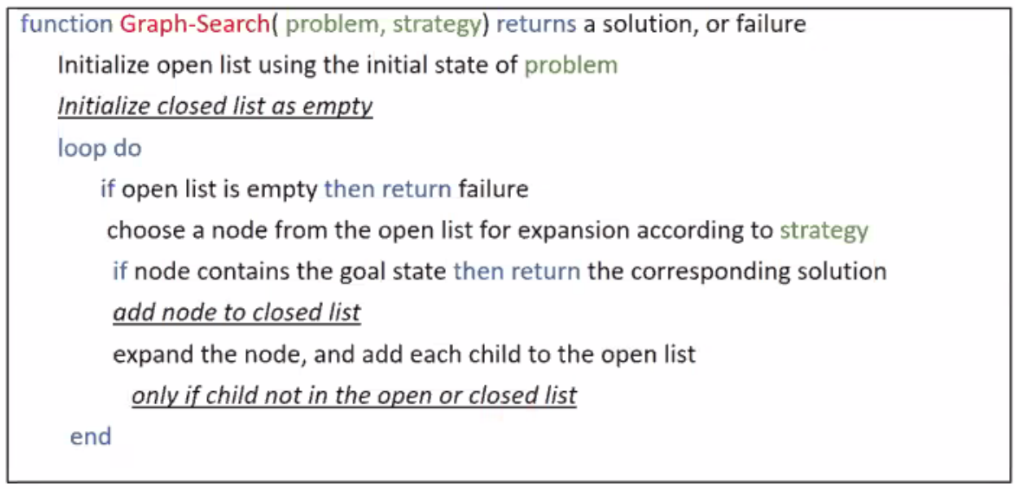
August 27th, Lecture 3,

Tree Search

* Start state is root
* Maintain a frontier/open list of leaf nodes to expand
* Try to expand as few tree nodes as possible

Graph Search

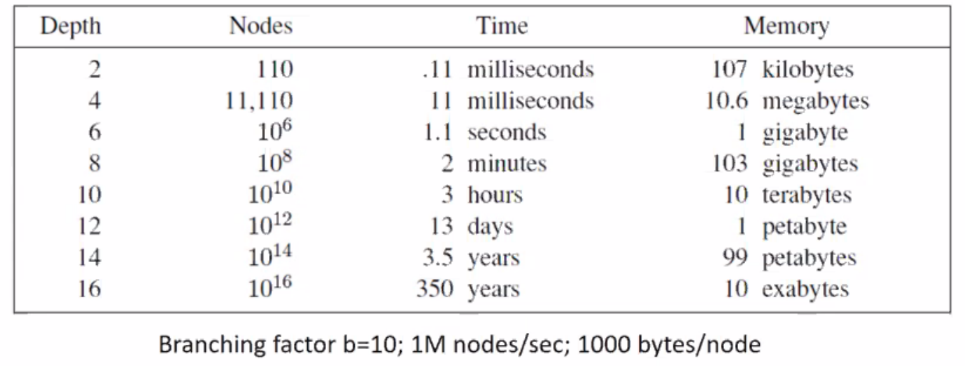
* Never expands state twice
* Tree search + closed list (remembers expanded nodes)
* Implement “closed list” as a set

Uninformed (blind) search methods

Breadth-first search

* Expand shallowest node first
* Frontier is implemented as First-In-First-Out queue
* Pick leftmost open node, open all open nodes of a tree before opening node’s node’s
* Finds shortest path to a solution

Evaluating search algorithms:

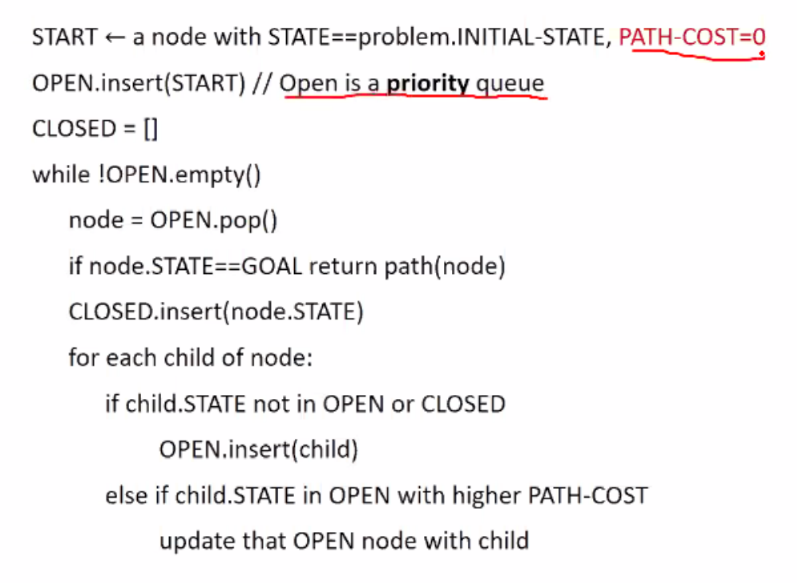
* Completeness: Can we find a solution if it exists?
* Optimality: Can we find the least-cost solution?
* Time complexity: Number of nodes expanded
* Space complexity: Max number of nodes in memory
* Useful quantities:
  + B – branching factor of search tree
  + M – maximum depth of the state space
  + D – depth of shallowest solution
* Breadth-first search
  + Time?
    - Let d be depth of shallwest solution
    - O(bd)
  + Space?
    - At depth d, there are only O(bd) nodes
  + Complete?
    - Yes (if d is finite!)
  + Optimal?
    - If step costs are equal (if cost increases as you go down – NO)

Depth-first search

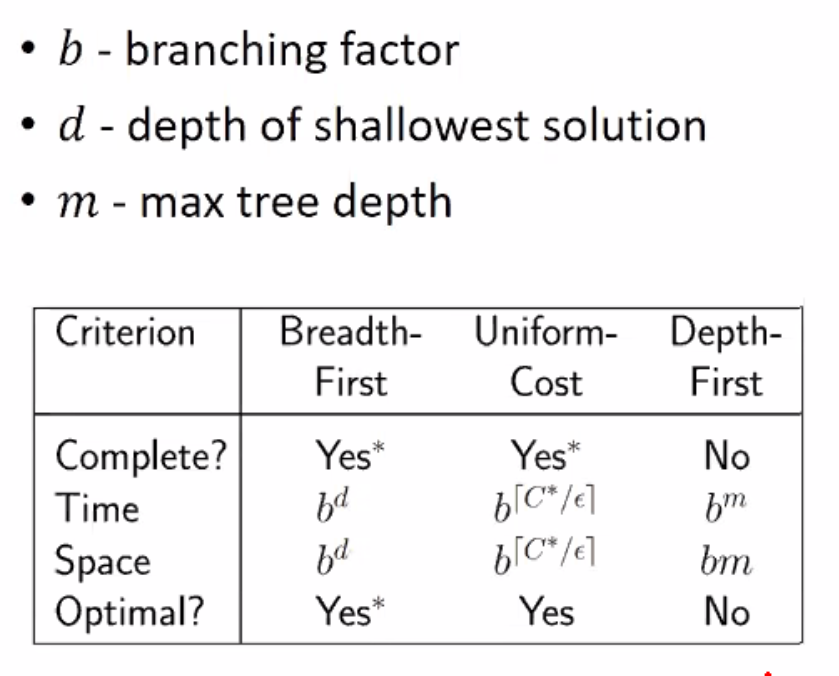
* Open leftmost node first
* Open is last in first out, new successors go in front
* Complete:
  + If m is finite and no cycles exist
* Optimal:
  + Ignores cost
* Time:
  + If m is finite, O(bm)
  + Terrifle if m >> d
* Space:
  + Keeps only siblings along path to root, leading to O(bm) – linear space

Iterative deepending:

* DFS’s space advantage with BFS’s time / shallow solution advantages
* Run DFSs with incrementing depth limit if no solution found

Uniform cost search

* Expand last-code node first
* Frontier implemented as a PRIORITY queue
* Goal test when node selected for expansion, not generated
* Explores in all directions
* No goal-oriented expansion
* Optimal:
  + Expands nodes in order of increasing g values
* Complete:
  + Yes, if cost of every action is at least epslilon > 0
* Time?
  + Process all nodes with g < cost of optimal solution
  + If C\* is optimal cost, effective depth is C\* / epison
  + Hence, takes O(bC\*/eps)
* Space?
  + Same as time, roughly O(bC\*/eps)

In summary: