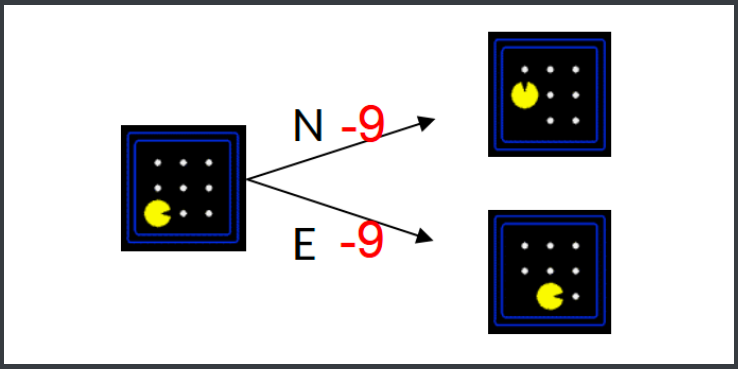
FAI: lecture 3 (search problems)

search problems

- take PacMan as an example:



-> state space *S*

represents the environment and the position of the agent in it

-> an initial state s_0

-> Actions *A(s)* in each state

-> Transition model *Result(s,a)*

-> A goal test *G(s)*

for PacMan, this would be, state *s* has no dots left

-> Action cost *c(s,a,s')*

- a solution: a sequence of actions that reaches a goal state

  an optimal solution: a solution with minimal cost

- a world state v.s. a search state

  for the PacMan problem,

  -> # world states = 120 (agent positions) * 2^{30} (food distribution) * 12^2(ghost distribution) * 4 (agent facing)

-> # pathing states = 120

-> # eat-all-dots states = 120 * 2^{30}

search trees

1. state space graphs (each state occurs only once), too big to build/store
2. search trees

   node expansion

   explored nodes

   Frontier / fringe

psuedo code for on-demand tree search

Note that each node should only be traversed once (no repeat);

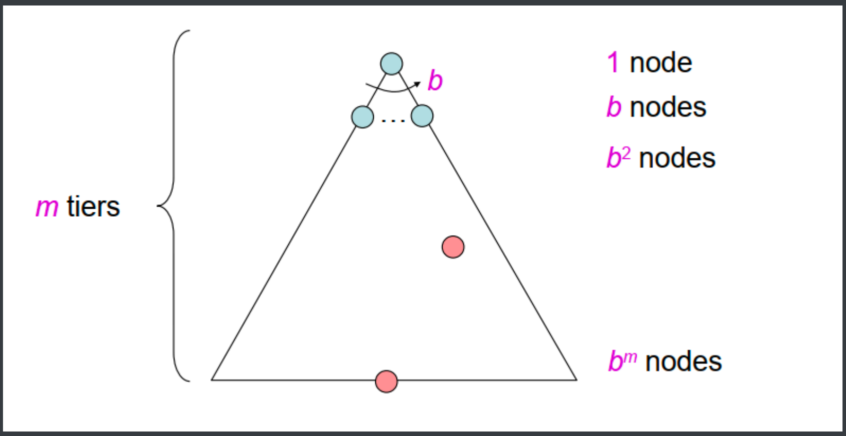
```
function TREE_SEARCH(problem)
  start_node := MAKE_NODE(problem.INITIAL_STATE)
  frontier := set consisting of start_node
  reached := an empty set

  while not IS_EMPTY(frontier) do:
    node := select and remove an element from frontier
    if problem.IS_GOAL(node.STATE) then
      return node
    if node.STATE not in reached then
    add node.STATE to reached
      for child_state in problem.NEIGHBORS(node.STATE) do:
        child_node := MAKE_NODE(child_state, node)
        frontier := INSERT(child_node, frontier)
 return FAILURE
```

uninformed search methods

DFS

- Stack, LIFO
- Properties



b - branching factor

m - maximum depth

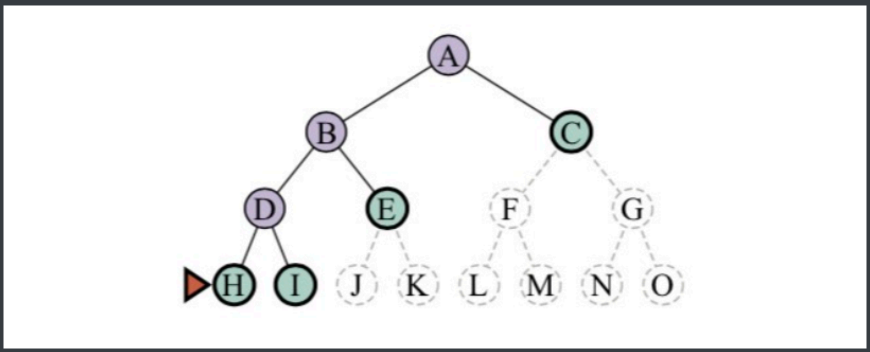
(i) # nodes in entire tree = $1 + b + b^2 + ... + b^m = O(b^m)$

  time complexity (if m is finite) = $O(b^m)$

(ii) space complexity = $O(bm)$

  only stores siblings on path to root

  (stores at most `b` nodes for each layer)



(iii) optimal?

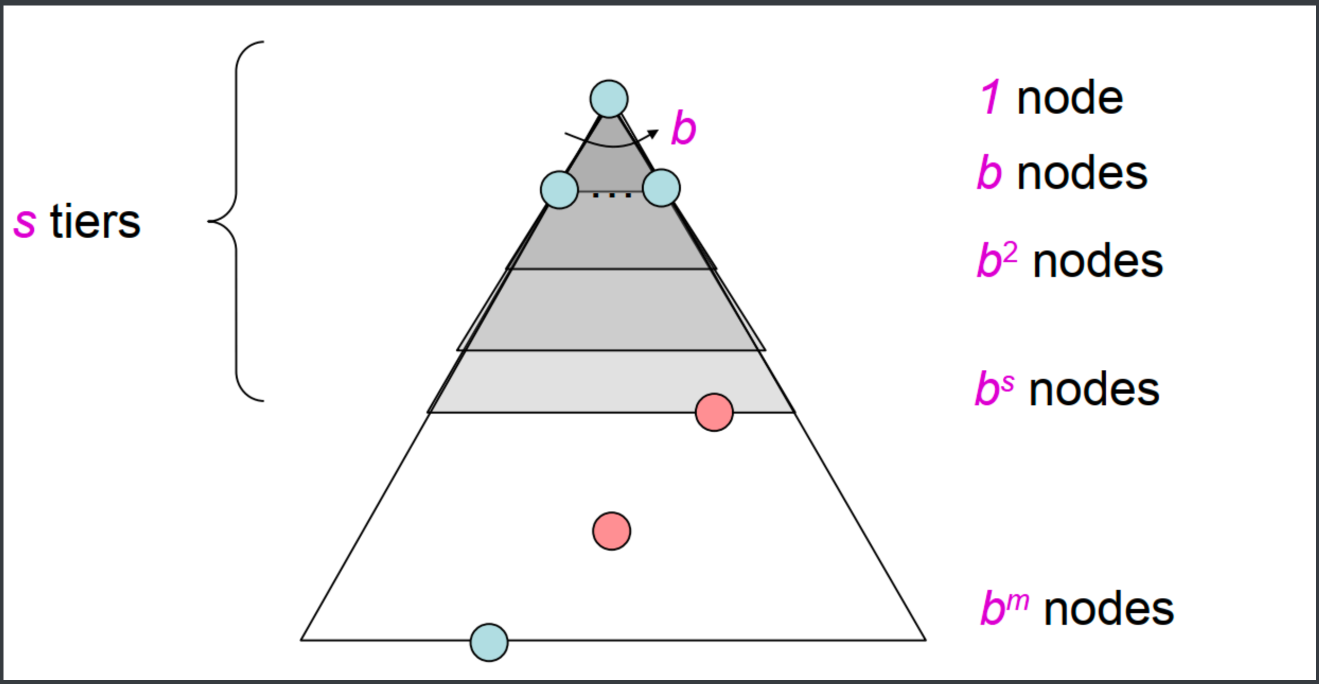
⛔, the "leftmost" solution, relardless of depth or cost;

(iv) complete?

not necessarily, only if m is finite

BFS

- Queue, FIFO
- properties



(i) time complexity

Searched s tiers, $O(b^S)$

(ii) space complexity

$O(b^S)$

(iii) optimal?

Shallowest solution, if costs are equal then yes

(iv) complete?

yes (s must be finite if a solution ever exists)

BFS v.s. DFS

- BFS - space bottleneck, solution containing the fewest arcs;
- DFS - for restricted space, solutions are long
- Combining DFS & BFS:

iterative deepening

Run a DFS with depth limit 1. If no solution…
Run a DFS with depth limit 2. If no solution…
Run a DFS with depth limit 3. …..

(i) time complexity: $O(b^S)$, the same solution found as BFS

(ii) space complexity: same as DFS

(iii) complete? as BFS

(iv) optimal? as BFS (only if costs are equal)

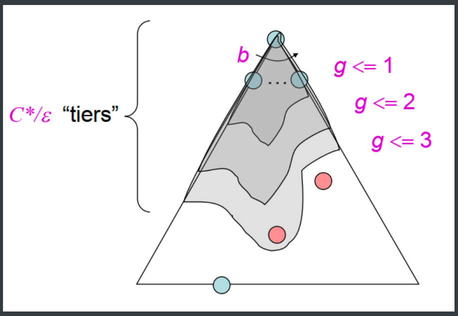
uniform cost search (ups)

- priority queue
- expands all nodes with cost less than the cheapest solution
- properties

time complexity

solution costs C*

arcs cost at least $$\epsilon$$
"effective depth" is roughly C*/e



space complexity

complete?

optimal?

Yes