Uninformed Search Strategies

• Uninformed strategies use only information available in the problem definition
  – Breadth-first search
  – Lowest-cost-first search
  – Depth-first search
  – Iterative deepening search
    • Depth-bounded search
Search Strategies

• A strategy is defined by picking the order of node expansion

• Strategies are evaluated along the following dimensions:
  
  completeness – does it always find a solution if one exists?
  
  time complexity – number of nodes generated/expanded
  
  space complexity – maximum number of nodes in memory
  
  optimality – does it always find a least-cost solution
Search Strategies

• Time and space complexity are measured in terms of

  b – maximum branching factor of the search tree
  d – depth of the least-cost solution
  m – maximum depth of the state space (may be infinite)
Properties of Breadth-First Search

• Complete?? Yes (if b is finite)
• Time?? $1 + b + b^2 + b^3 + \ldots + b^d + b(b^d - 1)$
  $= O( b^{d+1} )$, ie, exp. in d
• Space?? $O( b^{d+1} )$ (keep every node in memory)
• Optimal?? Yes (if cost = 1 per step); not optimal in general
Properties of Depth-First Search

- Complete??
- Time??
- Space??
- Optimal??
Properties of Depth-first Search

• Complete?? No: fails in infinite-depth spaces, spaces with loops
  
  Modify to avoid repeated states along path
  ⇒ complete in finite spaces

• Time?? $O(b^m)$: terrible if $m$ is much larger than $d$
  but if solutions are dense, may be much faster than breadth-first

• Space?? $O(bm)$, i.e., linear space!

• Optimal?? No.
Why might we NOT use

- BFS
- DFS
Uninformed Search Strategies

• Uninformed strategies use only information available in the problem definition
  – Breadth-first search
  – Lowest-cost-first search
  – Depth-first search
  – Iterative deepening search
    • Depth-bounded search
Lowest-cost Search

• Expand least-cost unexpanded node

• Implementation:
  – Frontier = queue ordered by path cost
  – Equivalent to breadth-first if step costs all equal

• Complete?? Yes, if step cost ≥ε

• Time?? # of nodes with g ≤ cost of optimal solution, O(b ⌈C*/ε⌉) where C* is cost of optimal solution

• Space?? # of nodes with g ≤ cost of optimal solution, O(b ⌈C*/ε⌉)

• Optimal?? Yes – nodes expanded in increasing order of g(n)
Uninformed Search Strategies

- Uninformed strategies use only information available in the problem definition
  - Breadth-first search
  - Lowest-cost-first search
  - Depth-first search
  - Iterative deepening search
    - Depth-bounded search
Properties of Iterative Deepening Search

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

Can be modified to explore uniform-cost tree
Properties of Iterative Deepening Search

• Doesn't it repeat a lot of stuff?

• Isn't this inefficient?
Properties of Iterative Deepening Search

• Doesn't it repeat a lot of stuff? A little bit.

• Isn't this inefficient? Actually, NO!
Properties of Iterative Deepening Search

- Numerical comparison for $b=10$ and $d=5$, solution at far right:

  - $N(\text{BFS}) = 10 + 100 + 1,000 + 10,000 + 100,000 + 999,990 = 1,111,100$

  - $N(\text{IDS}) = 50 + 400 + 3,000 + 20,000 + 100,000 = 123,450$
What order will the nodes be visited?

- Breadth First
- Depth First
- Lowest-Cost
- Iterative Deepening
What were your answers?

- **Breadth-First**
  a, b, c, d, e, e, f, f, z

- **Depth-first**
  a, b, e, z

- **Lowest Cost**
  a, c, d, b, f, e, e, f, z

- **Iterative Deepening**
  a, a,b,c,d, a,b,e,c,e,f,d,f, a,b,e,z
PA #1

• Lilypad
  – Help Gorf get from the upper left hand corner of the pond to his "home"
PA #1

• READ the specific requirements
  – Should use Python, Java, or get a language approved by me first.
  – Must have a search() function that takes in a 25 character String representing the board
  – Should return the optimal path
  – Must follow naming conventions listed in assignment. (File named HW1, method named search)