HW #1Posted

- Two weeks to complete
- Present results in a research paper format
  - Abstract / Introduction / Experimental results
  - Include tables / charts
- Attach your source code

HW #1 State Spaces

- Grid-based movement
  - Pathfinding or DNA sequence alignment
- Generic $n$-ary tree
- Sliding-tile puzzle

DFS

- Expand nodes and put them on a stack
  - Pop top node and expand
- depth-first expansion v. generation
- Solution quality -- delay discussion
DFS - Space Complexity

- Assume we know the solution is at depth $d$
- How much memory required?
  - $O(d) \text{ v. } O(b \cdot d)$

Implementation Complexity

- What do you keep in memory for BFS?
  - Have to keep copy of each state
- What do you keep in memory for DFS?
  - Only have to keep a single copy of the state
  - Apply & undo moves to state

DFS - Time Complexity

- Worst-case $O(b^d)$
- Best-case $O(d)$
- Assumptions
  - Duplicate detection?
  - Infinite graph?
- What if we don’t know the solution depth?
  - (Complete?)

DFID

- Run DFS with successive depth bounds:
  - $1, 2, 3...d$
- Solution Quality?
- Space Complexity?
DFID

• Time complexity?
• BFS work at depth d: $b^d \cdot b/(b-1)$

$$\frac{b^d}{b-1} + \frac{b^{d-1}}{b-1} + \frac{b^{d-2}}{b-1} + \ldots + \frac{b}{b-1} + \frac{b}{b-1}$$

$$\frac{b}{b-1}(b^d + b^{d-1} + b^{d-2} + \ldots + b + 1)$$

$$\frac{b}{b-1}b^d \cdot \frac{b}{b-1} = b^d \cdot \left(\frac{b}{b-1}\right)^2$$

Optimality - Time

• DFID is a time-optimal brute-force algorithm
• Proof by contradiction
  • $b^d$ nodes at the level of the solution
  • If algorithm A uses less than $b^d$ expansions it most not expand some node at level d
  • Create new problem -- swap goal and the unexpanded node -- then A won't find the solution

Optimality - Space

• $O(d)$ space
• How much space must an algorithm use?
• If it has $b^d$ time, must have $\log(b^d)$ space
  • $d \log(b)$ -- assume $b$ is constant
• If algorithm is a FSM and runs $K$ steps, each step must have unique state
  • At least a counter to distinguish btwn states

State Spaces

• Formally define:
  • Map-based pathfinding
Really Optimal?

- What about problems with cycles?
- Pathfinding in a grid?
  - BFS - $O(r^2)$
  - DFID - $O(4^r)$
- Removing short cycles?
- Difference between explicit/implicit (mark nodes)

Best-First Algorithms

- Class of algorithms with non-uniform edge costs
- Generally have:
  - Open list priority queue/sorted
  - Closed list

Definitions

- Node is expanded when taken off queue
- Node is generated when neighbor(s) expanded
- $g$-cost is the cost from the start to the current node
- $c(a, b)$ is the edge cost between $a$ and $b$
  - Sometimes also designates optimal path cost

Reconstructing paths?

- BFS?
- DFS?
Best-First Algorithms

- Best-First Algorithm Pseudo-Code
  1. Put start on OPEN
  2. While(OPEN is not empty)
     1. Pop best node \( n \) from OPEN
     2. if \( n == \text{goal} \) return path\( (n, \text{goal}) \)
     3. for each child of \( n \) // generate children
        1. put/update value on OPEN/CLOSED
  3. return NO PATH

Uniform-Cost Search

- Dijkstra’s algorithm -- sort by g-cost
  - Assume no negative weight cycles

Dijkstra’s Completeness

- Complete? / Will it find a solution?
  - Finite Graph: yes, optimal
  - Infinite Graph
    - Must have finite path to goal
    - Cost must be finite
    - Cannot have \( \leq 0 \) loops
    - Edge costs at least epsilon

Proof: High Level View
Dijkstra’s Optimality Proof (1)

• Step 1 by induction:
  • A node on the optimal path is always on the open list with its optimal cost from the start:
    • step 0; trivial
    • assume step n; step n+1 (expand new node; either on or off optimal path)
  • If we ever run out of OPEN nodes, there can’t be a path to the goal
  • There is always a path to the goal on OPEN

Dijkstra’s Optimality (2)

• Every node with finite cost will eventually be expanded
  • Minimum edge cost, so cost of best path will always be increasing
  • Increase without bound
  • every time we expand a node, we replace it with children which have higher cost
  • When we expand the goal we terminate
    • If it has cost K, all other nodes in OPEN must have cost ≥ K
    • Lengthening the path can only increase the cost
    • Optimal path must be ≥ K, found path cost K, optimal