Question 1a

• Assume Pacman is in an NxN maze with no interior walls.

• (1a) What is the branching factor of the successor function (i.e., the maximum number of states produced by the successor function) if Pacman is in the corner? Explain your answer.
  
  • Pacman can move in a maximum of four directions if not obstructed (branching factor 4)
  
  • But in a corner, two of those directions are blocked, yielding **branching factor of 2**.
Question 1b

• Assume Pacman is in an NxN maze with no interior walls.

• (1b) What is the branching factor of the successor function if Pacman is in the middle? Explain your answer.
  • Pacman can move in a maximum of four directions if not obstructed (branching factor 4).
Question 1c

- Assume Pacman is in an NxN maze with no interior walls.
- (1c) What is the maximum possible depth of the search space? Explain your answer.
  - Tricky. DFS or A* can have unbounded search depth if there is no path/cycle-checking (and for DFS, the heuristic is misleading, i.e., not monotone/consistent).
  - Alternatively, if you assume there is path or cycle checking, each node can be visited at most once, yielding a search depth of $O(N^2)$.
  - We accepted either answer.
Question 2

You’ve been asked to analyze space requirements for Pacman’s OPEN list when using A* with a monotone heuristic and starting from a fixed location. Your game has been engineered such that transitions all have non-zero positive costs, $c$, such that $c_{\text{min}} \leq c \leq c_{\text{max}}$. Assume there is no cycle checking.

Let $ctotal$ represent the optimal cost solution and assume the maximum number of states produced by the successor function is 4. Each node in the search space is comprised of the current state as well as the path taken to get to that state. As such each node on the OPEN list corresponds to a different path that is being explored.
Question 2a

• Define the maximum number of paths on the OPEN list, \( n \), at the time the solution is found. Your answer should be in terms of the parameters defined above. Explain your answer.

  • Worst-case space complexity of A* = the worst-case space complexity of UCS: \( O\left(b\left(\frac{c^*}{\epsilon} + 1\right)\right) \)
    • \( \epsilon \) is a lower bound on move cost
    • \( C^* \) = cost of the optimal solution
  
• Rewrite with problem params: \( O\left(\frac{c_{total}}{4c_{min}} + 1\right) \)

McIlraith & Allin, CSC384, University of Toronto, Winter 2018
Question 2b

• Conversely, if you only had space for $n$ paths on the OPEN list, what is the bound on the total cost of the solution that you can guarantee you will find? Again, your answer should be in terms of the parameters defined above. Explain your answer.

  • From previous part, $\frac{ctotal}{cmin} + 1 \leq n$
  • Solve for $ctotal$, yielding: $ctotal \leq cmin \left(\log_4 n - 1\right)$
Question 3a

• Assume that you have a heuristic function $h(n)$ that is monotone. For each evaluation function given below, will running A* with that evaluation function yield an optimal solution? If the answer is no, provide an upper bound for the ratio of the cost of the returned solution to the cost of the optimal solution.

  • (3a) $f(n) = g(n) + h(n)$
    • Standard $f(n)$ for A*. Thus, optimal
Question 3b

• Assume that you have a heuristic function $h(n)$ that is monotone. For each evaluation function given below, will running A* with that evaluation function yield an optimal solution? If the answer is no, provide an upper bound for the ratio cost of returned solution / cost of optimal solution.

  • (3b) $f(n) = g(n) + 3h(n)$
    • Greedier, non-optimal search
    • Worst-case ratio is 3
Question 3c

• Assume that you have a heuristic function $h(n)$ that is monotone. For each evaluation function given below, will running A* with that evaluation function yield an optimal solution? If the answer is no, provide an upper bound for the ratio of the cost of the returned solution to the cost of the optimal solution.

• (3c) $f(n) = 3g(n) + h(n)$
  • More conservative than A*. Optimal
  • Note: slower than regular A* if you have a good heuristic, faster if you have a bad one
Consider two monotone heuristics $h_1$ and $h_2$. $h_1$ dominates $h_2$, but $h_1$ takes much longer to compute than $h_2$.

(4a) Factors that favor $h_1$ over $h_2$:

- $h_1$ uses less space (expands weakly fewer nodes)
- May be faster to find solution, depending on
  - How much more accurate it is
  - How much slower it is
  - Other problem details like:
    - Speed of the successor function (slower is better)
    - Branching factor (higher is better)
    - Solution depth (higher is better)

(4b) The opposite of 4a.