Informatics 2. Coursework 1: Search and Games

Nathan Sharp
UNIVERSITY OF EDINBURGH

March 10, 2020

3 Uninformed Search

You will traverse a directed acyclic graph (DAG). Given a start node n_s and a goal node n_g, your task is to implement different uninformed search procedures to reach the goal node from the start node.

1. **Breadth First Search (BFS) and Depth First Search (DFS)**

   (a) **(Manually) perform Breadth First Search with elimination of repeated states. The ordering of the nodes is lexicographic. If you get stuck in a loop, indicate the loop. Full points will only be awarded if you show each step of the algorithm instead of only the end result.**

   The Breadth First Search (BFS) is 'performed' in the diagram below. The number to the left of a node indicates the order in which it is expanded. After reaching the goal node 'L', the algorithm would return the path [A,E,G,L] and the associated cost of 10 (3+4+3). Note the diagram can be read like English; left to right, line by line (breadth first), and the cost is not informative to search order:

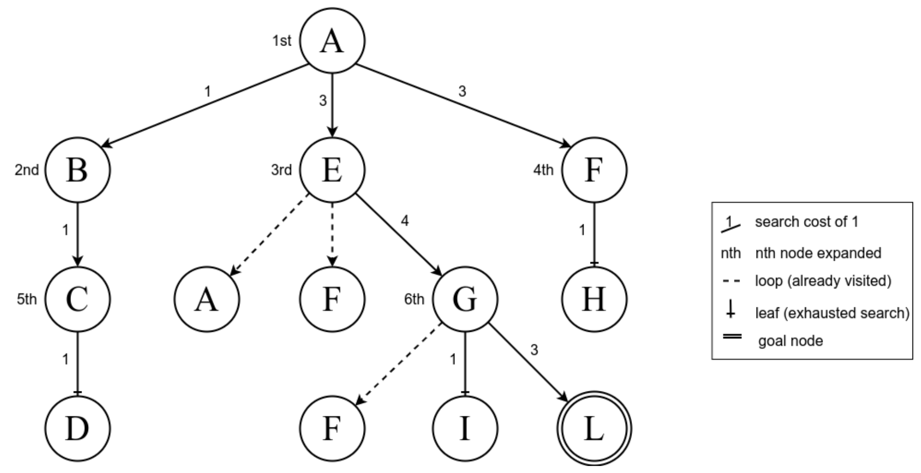

Figure 1: Breadth First Search

(b) **(Manually) perform Depth First Search with elimination of repeated states. The ordering of the nodes is lexicographic. If you get stuck in a loop, indicate the loop. Full points will only be awarded if you show each step of the algorithm instead of only the end result.**

Similarly to BFS above, Depth First Search (DFS) of the given graph is represented graphically below. After reaching the goal node 'L' this algorithm would return the path [A,E,F,H,G,L] and the associated cost of 11 (3+2+1+2+3):

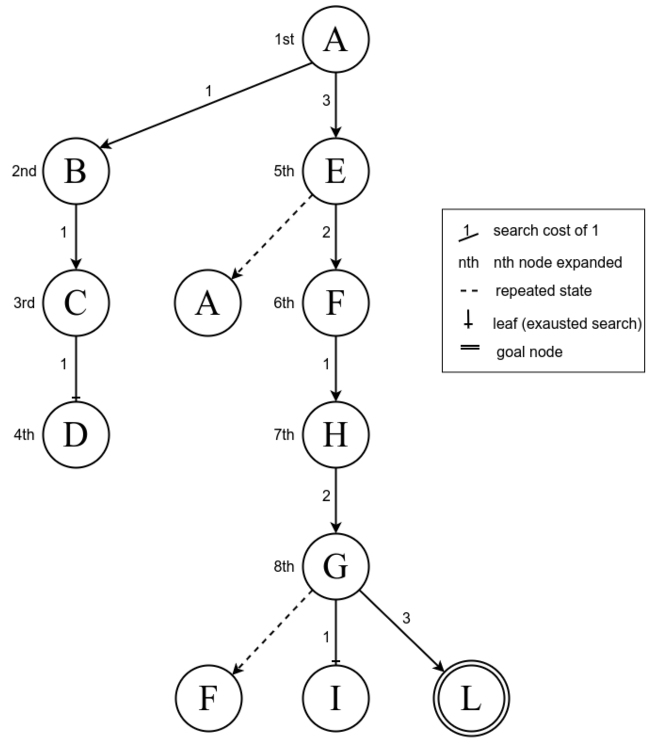

Figure 2: Depth First Search

(c) **Draw a graph that favours BFS. Briefly explain why you choose this graph.**

Proceeding with the same key as for the exercises above with the added '...' to represent a finitely large or infinite number of nodes in the search tree. The most convincing case favouring BFS is that in which the hidden, '...' part has infinitely many nodes as DFS would never reach the goal state. BFS is preferred here as it reaches the goal state easily.

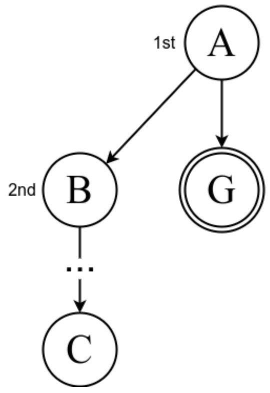

Figure 3: A graph that favours Breadth First Search

(d) **Draw a graph that favours DFS. Briefly explain why you choose this graph.**
In the following diagram the goal state is in a deep position following the expansion of a child node which is high in the expansion order. DFS can go more directly to the goal state, whereas BFS has to fully expand all children of any node before deepening the search.

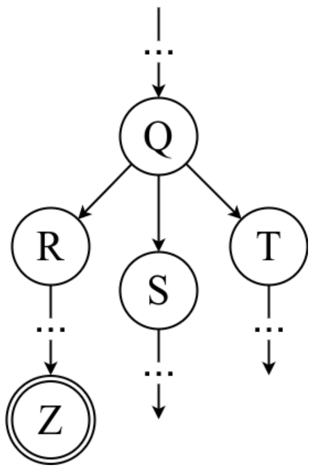

Figure 4: A graph that favours Depth First Search

(e) **Compare BFS and DFS. What are the biggest differences? (Be concise. Character Limit: 750 characters.)**

- The Fundamental difference is the order in which the frontier is expanded. BFS uses a FIFO queue (first in-first out) to expand the frontier so the first node added is the first to be expanded while DFS uses a LIFO stack (last in-first out) to expand the frontier so that the last (most recent) node added is expanded first.
- The running time of the algorithms is $O(b^d)$ for BFS and $O(b^m)$ for DFS where b is the branching factor, d is the depth of the solution and m is the maximum graph depth. The algorithms have a complimentary nature with the worst case run-time of DFS being the best case of BFS and vice versa.

- The space complexity of BFS is $O(b^d)$ makes it highly impractical for large tasks. This is compared to DFS which has $O(bd)$ space complexity as expanded nodes can be removed from memory once all their children have been explored.
- BFS checks if a node is a goal node when its generated whereas DFS checks when its expanded.
- DFS might find a non-optimal deep solution, or no solution at all if the graph has infinite depth, whereas although not always optimal, BFS is always guaranteed to find the shallowest solution (assuming an finite branching factor), this is particularly good if the path costs are uniform as this will correspond to an optimal solution.

2. Iterative Deepening Search

(a) **What is the optimal depth for the graph in Figure 2?**

Optimal depth is 3, as calculated by our manual BFS implementation which always returns an optimally deep solution.

(b) **Compare Depth First Search and Iterative Deepening Search. When would you use Iterative Deepening Search over Depth First Search? (Be concise. Character Limit: 500 characters.**

IDS is an iterative variant of DFS which has an progressively increasing depth limit. It is generally preferable in the case where the the search space is large/unknown, has infinite depth or a case where an optimal solution is highly preferable. Furthermore, in the case that the depth of the shallowest solution is less that the maximum depth of the tree, IDS has better space and time complexity than DFS.

4 Informed Search: The A-Start Algorithm

1. **Heuristics**

(a) **Why is the straight line distance a valid heuristic? Name the criteria and why they apply in this case.**

Any valid heuristic needs to be admissible and consistent.

To be admissible the heuristic must not overestimate the cost to reach the goal. As the straight line distance is by definition the shortest path between to points it is impossible for it to overestimate the distance cost.

To be consistent (graphs search only) it must satisfy the condition that the estimated cost $h(n)$ from a node n to the goal is less than or equal to the estimated cost $h(n^{'})$; the cost from successor node $n^{'}$ to the goal node plus the cost $c(n, n^{'})$ of going from the node to this successor. This applies geometrically to the straight line distance heuristic by the triangle inequality which states that two sides of a triangle must be greater than or equal to in length of the final side; $h(n) \leq c(n, n^{'}) + h(n^{'})$ with our notation. This demonstrates that the straight line distance heuristic is consistent.

(b) **Describe (1 sentence) three more problems that can be solved with A-Star search and name at least one valid heuristic for each.**

   i. Shortest driving path between two points with the heuristic of great-circle distance.

   ii. The next move for Solving a Rubik's cube with the heuristic of misplaced edges.

   iii. Traveling salesman problem with the heuristic of the total weight of minimum weight spanning tree on a subgraph of unvisited nodes and the current node.

2. Best-First and A-Star Search Comparison

(a) **What are the differences between the Best-First and A-Star search strategies?**

They are similar search strategies which expand frontiers by selecting a node that minimises some evaluation function. For each node best first search considers only the estimated distance to the goal node, $f(n) = h(n)$, whereas A-star adds this to the path cost from the root node to the current node, represented by $g(n)$ for a composite evaluation function of: $f(n) = h(n)+g(n)$. The outcome of this difference is that sometimes best first search fails to find a solution such as in the case where the path to the goal node requires choosing a node with a higher estimated cost at the current step. It is also more likely to find a non-optimal solution and get stuck in loops than A-star search which is both complete and optimal. However A-star has greater memory requirements.

(b) **How would you change your A-Star implementation to be a search instead? (You do not need to implement these changes, simply state them here.**

Assuming this question is referring to best first search, by the equations stated above it could be very easily implemented by changing the evaluation function to remove the path cost part, $g(n)$ in the above question.

5 Games: Connect Four with a Twist

In this section you will implement minimax search with alpha-beta pruning (see Chapter 5 of RN) for the two-player game Connect Four with a slight twist. The board size is 4 columns and 4 rows.

1. Familiarise yourself with the game and its rules.

Sir yes Sir.

2. **Given that you can rotate, place a piece and then rotate again, how many possible actions can a player perform during one term?**

This question is very unclear. I will choose to interpret the number of possible 'actions' as the number of possible *rotation* · *insertion* · *rotation* sequences of events such that two distinct action sequences which result in the same board position are counted separately. In this case you have four possible rotations (original, left, right, flip) and 16 possible insertion positions (four on each of the four sides) to give a total of $4 \cdot 16 \cdot 4 = 256$ possible actions.

3. **What are the main challenges for implementing Quadrio over Connect Four with a Twist for alpha-beta pruning? Would such an implementation be practical? Give reasons for you decision. (Be concise. Character Limit for this question: 1000 characters.)**

   The main challenge would be far greater complexity, specifically in the case of the number of possible actions which results in a branching factor $b = 256$ compared to connect Four with a twists branching factor of $b = 8$. The number of turns which corresponds to the depth of the tree in the limit is the same as the standard game, 7 in the minimum and 16 in the maximum. Assuming worst case depth and optimal move ordering (which is incredibly generous) we would have $\sqrt{256}^{16}$ nodes to search (best case alpha-beta pruning). This is a highly impractical implementation as this would take several years to run even on a state of the art computer. Assuming that you could search a node in a single cycle of a quad-core 3GHz processor, this would still take 48.71 years to run. We can conclude alpha-beta pruning alone is not a suitable search strategy for this game as the search tree is to big.