Martin Gustafsson (margu424) & Axel Gard (axega554)

# TDDC17 lab2 report

## Part 1

In part 1 we implemented the custom graph search algorithm that works by first taking a node from the frontier. Then checking if that node is the goal, if it is we return the path. We then add the node to the list of explored nodes. We then look at all the nodes connected to the current node, and we go through each one of them and if they are not already explored or in the frontier we add them to the frontier. Here we use the variable defined at the start to find if we should insert at the back or at the front. In the depth first case we insert at the front and in the breadth first case we insert at the back.

## Part 2

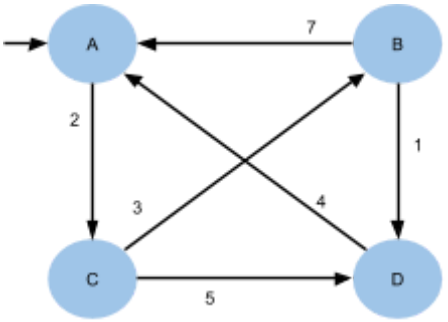
1. States: Every different position of the vacuum cleaner is a state
   actions: move: right, left, up and down
   branching factor: 3 (as there are 4 nodes possible and we must have come from one of them)

2. If the priority queue used on uniform cost search is defaulted to a FIFO queue when priority is the same breadth first search and uniform cost search works the same. If however the queue is not defaulted to a FIFO queue the order of paths explored will be different between breadth first and uniform cost search.

3. Suppose x is the actual cost to a node then we have that h1 <= x and h2 <= x
   a. (h1 + h2) / 2 <= (x+x)/2 = 2x/2 = x
   b. assume h1 = x ⇒ 2h1 = 2x > x ⇒ not admissible
   c. max(h1, h2) = h1 | h2 which are both admissible so max(h1, h2) is admissible

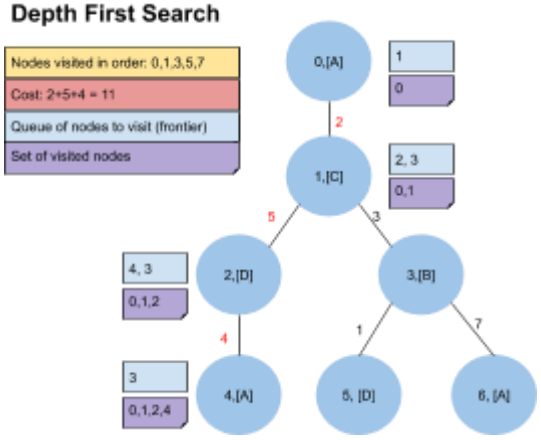
4. heuristic function (h) = $\Delta x + \Delta y$
   cost function (g) = path length (This works as the domain has uniform cost)
   h is admissible as the agent must move at least $\Delta x$ steps to get to the correct x position and at least $\Delta y$ steps to get to the correct y position.

5. Problem: Starting in node A, find a way back to node A in 1 or more moves.
   Depth First Search:
   Breadth First Search:
   Greedy Best First Search:

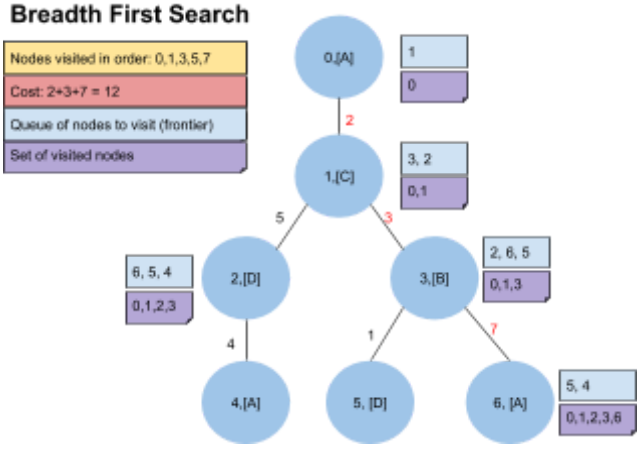


Martin Gustafsson (margu424) & Axel Gard (axega554)

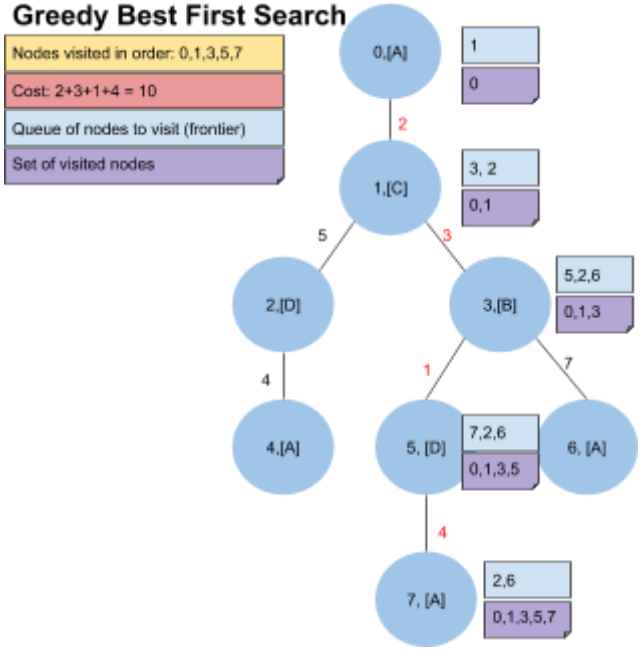
## Depth First Search




## Breadth First Search




## Greedy Best First Search



6. **Breadth First Search:**
   Is Complete because if there exists a solution in any part of the tree it will be found even if the depth is infinite, assuming there is a solution at a finite depth. BFS is not complete if the breath of the tree is infinite.
   Is not Optimal as the first solution to be found will be the one returned. If we return a solution at depth 2 there may be a solution at depth 3 with less cost. Optimal if step costs are equal.
   **Depth first Search:**
   Is not Complete due to being given a tree with an infinite depth even if a solution exists at a finite depth and that solution is not in the first path that is traverst we will keep on going forever.
   Is not Optimal as it will return the first solution it finds.
   **Uniform Cost search:**
   Is Complete as we search all of the state space given.
   Is Optimal as it will always expand the node that gives the least total cost.
   **Depth limited search:**
   Depth limited search is not complete, due to that the algorithm stops when the depth is reached.
   Depth limited search is not optimal as it returns the first find solution.
   **Iterative deepening search:**
   Is Complete because if there exists a solution in any part of the tree it will be found even if the depth is infinite, assuming there is a solution at a finite depth. This search algorithm is not complete if the breath of the tree is infinite.
   Is not Optimal as the first solution to be found will be the one returned. If we return a solution at depth 2 there may be a solution at depth 3 with less cost. Optimal if step costs are equal.
   **Bidirectional search:**
   Is not Complete as the search starting at the root node may continue forever given an infinite depth or infinite breadth.
   Is not Optimal as it returns the first solution found.
   **A* search:**
   Is Complete as we search all of the state space given.
   Is Optimal as it will always expand the node that gives the least total cost.

7. We used Depth first search to search for unknown squares, we did this as we found backtracking to be very expensive so we wanted to avoid that. Because we had uniform cost in that task we feel that Depth first search was the best choice for that task.