Maximilian Bragazzi Ihrén maxbr431@student.liu.se

TDDC17 – Lab 2 Part 2

In the vacuum cleaner domain part 1, what were the states and actions? What is the branching factor?

Each square in the grid is a state. Actions are: Suck, Move up, move down, move left, move right. Branching factor is 4.

What is the difference between Breadth First Search and Uniform Cost Search in a domain where the cost of each action is 1?

BFS would stop as soon as it reaches the goal. UCS would continue searching for the most cost effective path.

Suppose that h1 and h2 are admissible heuristics (used in for example A*). Which of the following are also admissible?

An admissible heuristic should never overestimate the cost to the goal. Given this...

a) (h1 + h2) / 2 is admissible

b) 2 * h1 is **not** admissible

c) max(h1, h2) is admissible

If one would use A* to search for a path to one specific square in the vacuum domain, what would the heuristic (h) be? The cost function (g)? Is it an admissable heuristic?

The heuristic h could be the Manhattan distance from the current position to the goal.

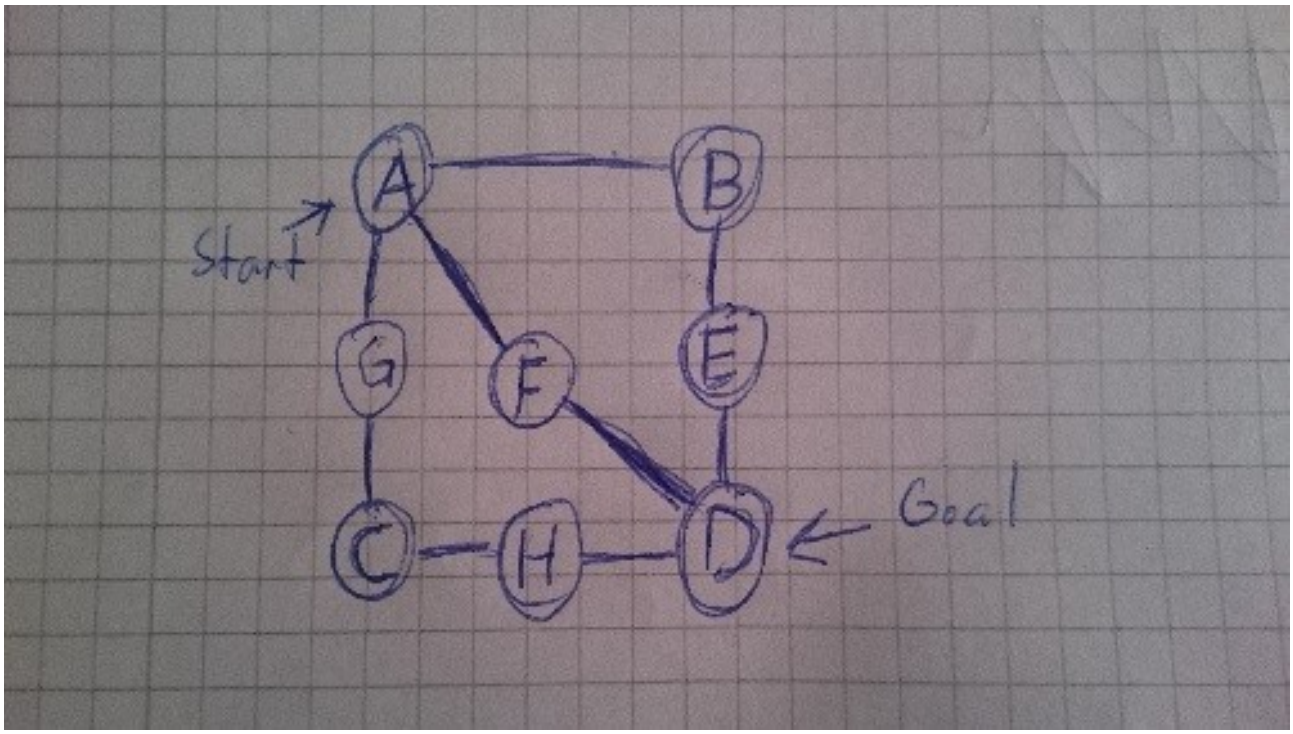
The cost g could be the total movement count from the start position to the current position.

A heuristic that uses Manhattan distance will be admissible, since Manhattan distance is admissible by default.

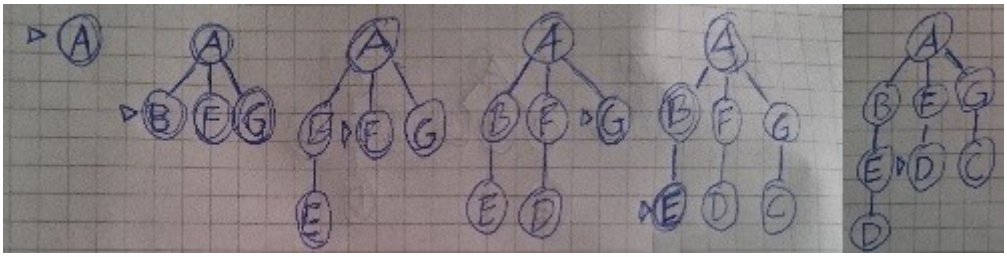
Maximilian Bragazzi Ihrén maxbr431@student.liu.se

Draw and explain. Choose your three favorite search algorithms and apply them to any problem domain (it might be a good idea to use a domain where you can identify a good heuristic function). Draw the search tree for them, and explain how they proceed in the searching. Also include the memory usage. You can attach a hand-made drawing.

Given the following graph:



Breadth-First Search

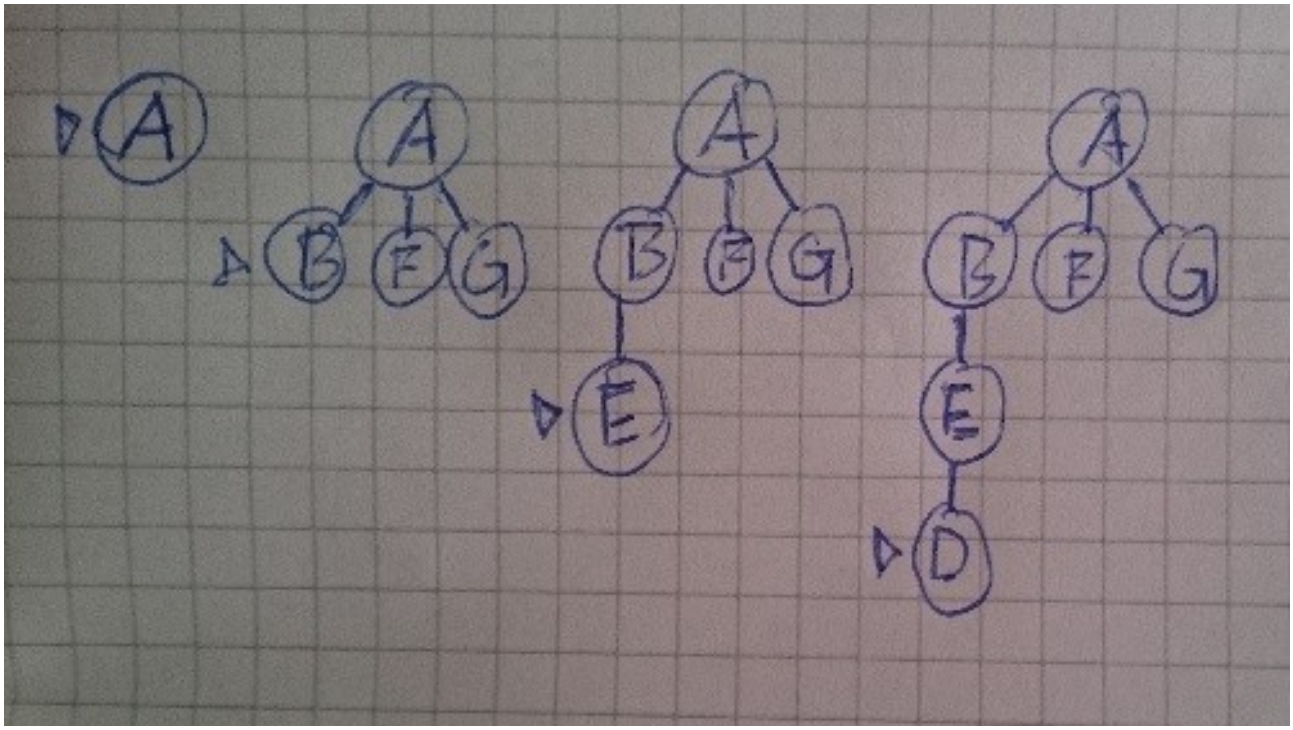


BFS would move through one entire depth layer before moving on to the next.

Frontier	Explored
A	
B, F, G	A
F, G, E	A, B
G, E, D	A, B, F
E, D, C	A, B, F, G
D, C, D	A, B, F, G, E

Final path: A → F → D

Depth-First Search



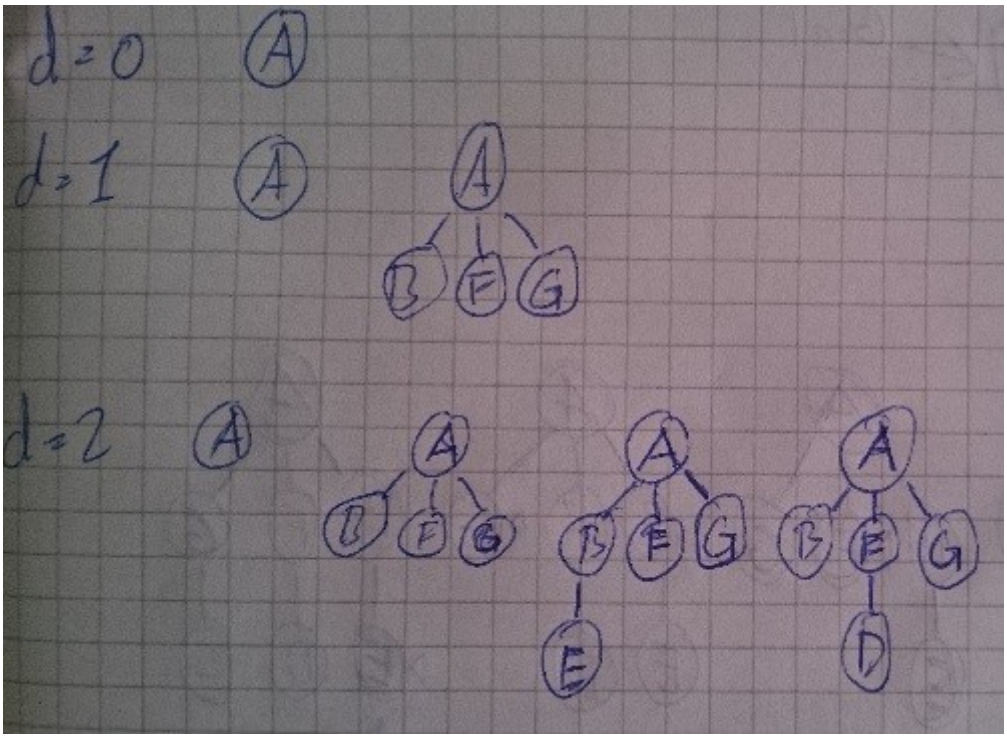
DFS would expand one “branch” fully before moving on to the next.

Frontier	Explored
A	
B, F, G	A
E, F, G	A, B
D, F, G	A, B, E

Final path: A → B → E → D

Maximilian Bragazzi Ihrén maxbr431@student.liu.se

Iterative Deepening Search



IDS would increment the depth it searches on, and search all the branches up to that depth.

Frontier	Explored
A	
	A
A	
B, F, G	A
F, G	A, B
G	A, B, F
	A, B, F, G
A	
B, F, G	A
E, F, G	A, B
F, G	A, B, E
D, G	A, B, E, F

Final path: A → F → D

Maximilian Bragazzi Ihrén maxbr431@student.liu.se

Look at all the offline search algorithms presented in chapter 3 plus A* search. Are they complete? Are they optimal? Explain why!

Breadth First Search

Complete: yes (if there is a way to the goal, it'll find it).

Optimal: yes, if all the path costs are the same (since it searches all of, for example, depth 1 before starting on the next, it will find the shortest path from the computer POV, which would not take path cost into consideration).

Uniform Cost Search

Complete: yes.

Optimal: yes, provided the path cost never decreases as we move forward.

Depth-First Search

Complete: yes in a graph search, no in a tree search (as it can get stuck in a loop if it doesn't remember what nodes it has already searched).

Optimal: no (it will always return the first thing it finds, even if there are cheaper paths).

Depth-Limited Search

Complete: yes, if the limit is chosen well.

Optimal: no (it is still depth search, which will return the first and not necessarily the cheapest path)

Iterative Deepening Search

Complete: yes (like breadth-first, it searches one depth at a time).

Optimal: yes.

Bidirectional Search

Complete: yes, if you use BFS in both directions.

Optimal: yes, if you use BFS in both directions.

A* Search

Complete: yes.

Optimal: yes, if the heuristic is admissible.

Maximilian Bragazzi Ihrén maxbr431@student.liu.se

Assume that you had to go back and do Lab 1/Task 2 once more (if you did not use search already). Remember that the agent did not have perfect knowledge of the environment but had to explore it incrementally. Which of the search algorithms you have learned would be most suited in this situation to guide the agent's execution? What would you search for? Give an example.

I'd say Iterative Deepening Search or Breadth-First Search would be the most suitable, given that you search for an unvisited location. Since it will give you (one of) the closest undiscovered locations at all times, the agent will always visit all the locations in the cheapest cost possible (if you don't count turning as being part of the cost). Once there are no more undiscovered locations to be found, one could switch to A* in order to get the most optimal path home.