Oscar Petersson, Johan Levinsson

TDDC17 Lab2

Högskoleingenjörsutbildning i datateknik, 180 hp

Laboration report - September 27, 2016
Artificial Intelligence
TDDC17, Linköpings universitet

Assistant:
Josef Fagerström
IDA

Q1.

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

A1. Actions: Moving (4 directions), Cleaning dirt.
States: Positions in the grid
Branching factor: 4 - maximal possible numbers of branches from a node.

Q2.

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

A2. The implementation (FIFO queue vs priority queue). They behave in the same way though when the cost is constant (e.g. the domain given in the question).

Q3.

Suppose that h1 and h2 are admissible heuristics (used in for example A). Which of the following are also admissible?*
a) $(h1 + h2)/2$
b) $2h1$
c) $\max(h1, h2)$

A3. (a) and (c)

Q4.

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

A4. (h): $\sqrt{\Delta x \Delta y}$ where $\Delta\{x, y\}$ is the absolute distance from current position to target in the axis (euclidian distance) is one possible heuristic which will always underestimate the distance. The manhattan/taxicab distance $|\Delta x| + |\Delta y|$ is another possible heuristic which in the given domain never will overestimate and most often underestimate. Both are admissable in the given domain.
(g): Any non-descending function. The depth of the node in the current search tree is a sensible option.

Q5.

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.

A5. Drawing included on page 3. Memory usage is implicitly expressed in the number of nodes mapped in the search tree.

Q6.

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

A6.

Algorithm	Complete	Optimal
Breadth-First	Yes	Yes
Depth-First	No	No
Iterative-Deepening	Yes	Yes
Bidirectional	BFS	No
Greedy Best First	Graph	No
A*	Yes	Yes
Uniform Cost Search	Yes	Yes

BFS is complete if the branching factor is finite and there is a solution, and it is optimal if the cost is a non-decreasing function.

DFS is incomplete as it can get stuck in infinite branches.

Iterative Deepening Search is complete if the branching factor is finite, and it is optimal if the cost is a non-decreasing function.

Bidirectional Search is implementation (DFS or BFS) dependent regarding completeness. It is not optimal since the path found needs to be verified optimal.

For Greedy BFS, tree search is incomplete as it can get stuck in dead-end branches, but graph search is complete thanks to keeping track of explored noted.

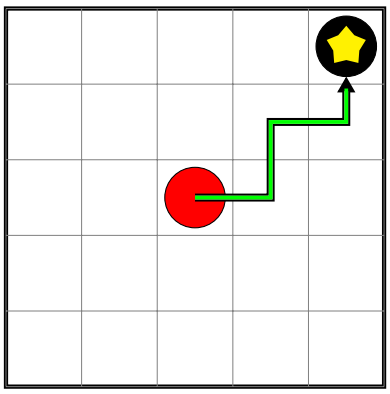
A* is complete. It is optimal if the heuristic function is admissable.

Uniform Cost Search is a general form of BFS.

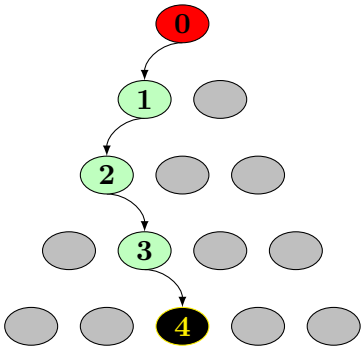
Q7.

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.

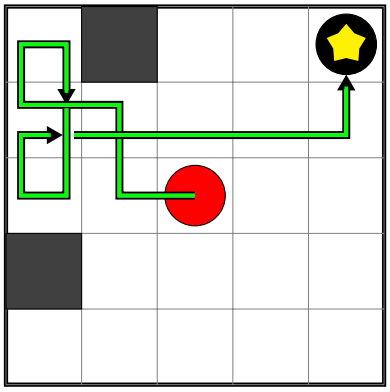
A7. We did use search (BFS), even though either one could do the work in the given domain. The objective would be to find that something doesn't exist in the domain, i.e. accessible unexplored nodes.



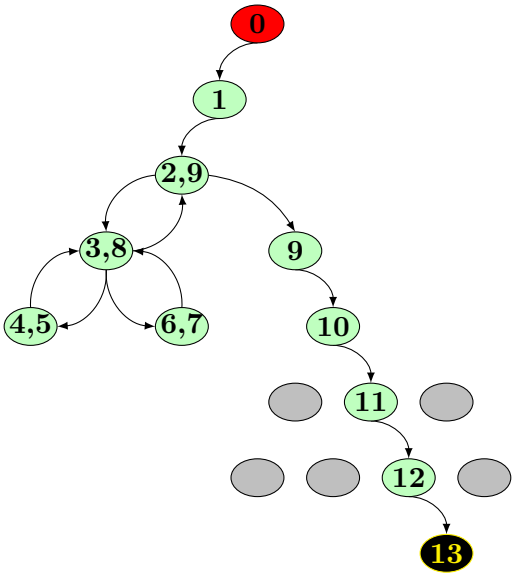
(a) A Search: search through maze*



(b) A Search: search tree, 15 nodes*



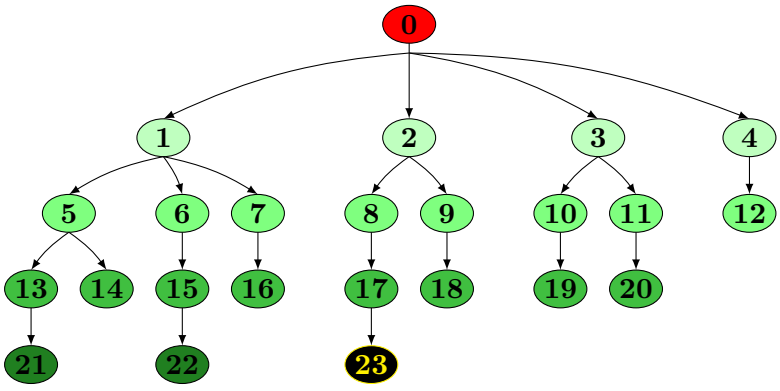
(c) DFS: search through maze



(d) DFS: search tree, 16 nodes

	19	10	18	23
20	11	3	9	17
12	4	0	2	8
13	5	1	7	16
21	14	6	15	22

(e) BFS: search tree, 24 nodes



(f) BFS: search tree