CSC384 Sample Questions: Search and CSP

Winter 2020

1 Search

1. It would seem that iterative deepening search should have a higher asymptotic time complexity than breadth-first search because every time the depth-bound is increased it must start its search from scratch. However this is not true given the algorithms presented in class. Why?

2. If $h()$ is admissible and s is the start node, how is $h(s)$ related to the cost of the solution found by A* search?

3. What happens if we use a heuristic $h()$ in A* search that does not have the guarantee that $h(n) \leq h^*(n)$ for all states n?

4. How do depth-first, breadth-first and depth-first with iterative deepening search compare in terms of their asymptotic time complexity? In terms of their asymptotic space complexity? Please indicate any assumptions you are using (i.e. justifications from the textbook, from other sources, etc).

5. Prove that if $h(n) = h^*(n)$ for all n, then whenever A^* expands a node n, n must lie on an optimal path to a goal.

6. Let h be an admissible function and let $f(n) = w * g(n) + (1 - w) * h(n)$ for $0 <= w <= 1$. Will A* find an optimal solution when $w = 1$?, $w = 1/2$?, $w = 3/4$?

7. Consider the problem of finding a path in the grid shown below from the position s to the position g. The robot can move on the grid horizontally and vertically, one square at a time (each step has a cost of one). No step may be made into a forbidden shaded area. The search space and the shaded areas are illustrated in Figure 1.

   - On the grid, number the nodes in the order in which they are removed from the frontier in a depth-first search from s to g, given that the order of the operators you will test is: up, left, right, then down. Assume there is a cycle check.
   - Number the nodes in order in which they are taken off the frontier for an A* search for the same graph. Manhattan distance should be used as the heuristic function. That is, $h(n)$ for any node n is the Manhattan distance from n to g. The Manhattan distance between two points is the distance in the x-direction plus the distance in the y-direction. It corresponds to the distance traveled along city streets arranged in a grid. For example, the Manhattan distance between g and s is 4. What is the path that is found by the A* search?

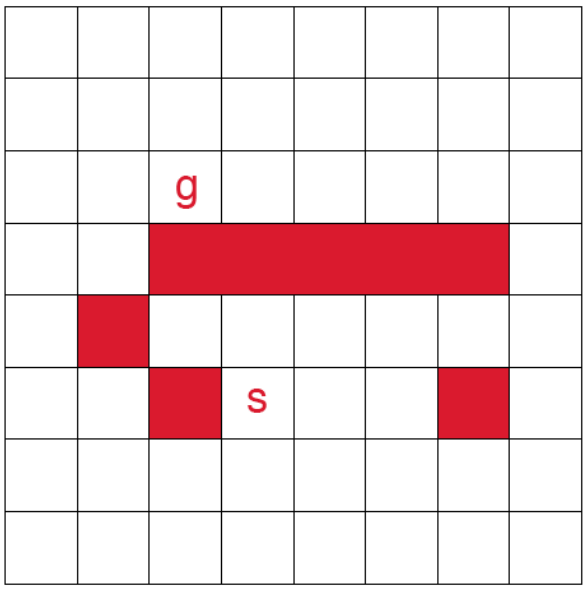


Figure 1: Search Space

- Suppose that the graph extended infinitely in all directions. That is, there is no boundary, but s, g, and the forbidden area are in the same relative positions to each other. Which search methods would no longer find a path? Would A*, or would depth-first search? Which would be the better method, and why?

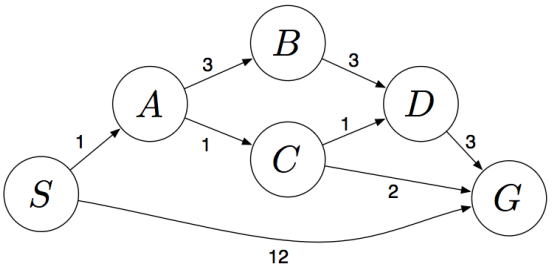


Figure 2: Search Problem

8. Consider the graph above and the heuristics below.

state	h1	h2
S	5	4
A	3	2
B	6	6
C	2	1
D	3	3
G	0	0

- Is h1 admissible? Why or why not?
- Is h1 consistent? Why or why not?
- Is h2 admissible? Why or why not?
- Is h2 consistent? Why or why not?

2 Constraint Satisfaction Problems

1. Consider the map shown below. There are five regions, and the following information is available about what they can represent.

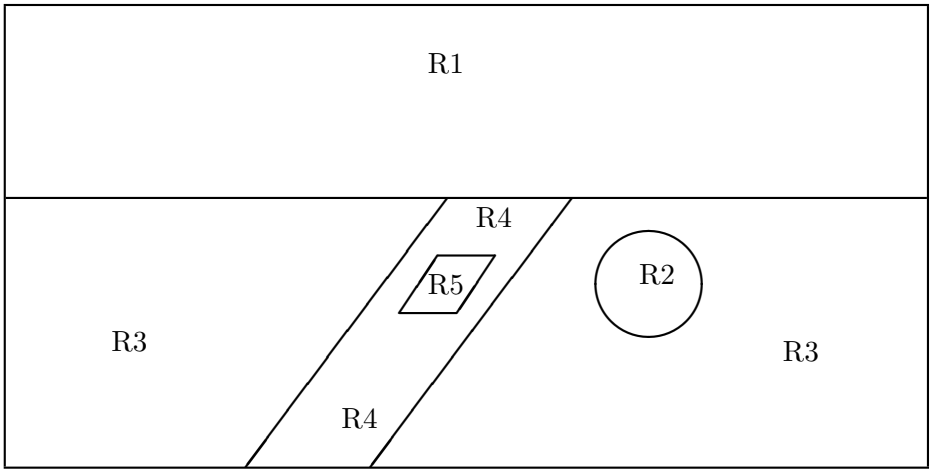
   (a) R1 can only be the sky or grass,

   (b) R2 can only be trees or a road,

   (c) R3 can only be grass or trees,

   (d) R4 can only be road or grass,

   (e) R5 can only be a car.



   Furthermore, we have the following knowledge about the real world, and the map:

   (a) A car cannot be next to grass.

   (b) No two neighboring regions can be the same.

   (c) Only one region is a road.

   The problem is to find what each region in the figure represents.

   (a) Represent the above problem as a CSP. You must specify the meaning of each variable assignment. (That is, in our representation, how can one read off a solution to the problem from an assignment of values to the variables).

   (b) Apply forward checking to solve the problem. Use the heuristic of always assigning next the variable with fewest remaining values (you can break ties as you choose). You *do not* need to show the updated current domains, only show the nodes of the search tree visited by forward checking and the assignments made at those nodes.

   (c) How many solutions are there, and what are they?

2. Consider a CSP with the following variables and constraints:

   - Variables A, B, C, D, E with all variables having the domain $1, 2, 3, 4$

- Constraints:
  - $E - A$ is even.
  - $C \neq D$
  - $C > E$
  - $C \neq A$
  - $B > D$
  - $D > E$
  - $B > C$

Draw these variables and constraints as a constraint graph. Then, perform GAC-Enforce on the graph. Write the domains for each variable that exist after you've completed GAC-Enforce.