Search Problems in AI

Assignment 2

Michael Li - 192008938
Daniel Liu - 184007283
Robert Kulesa - 185009892

CS440 Fall 2021
Professor Boularias
Rutgers University - New Brunswick
November 12, 2021

Problem 1

The sequence of nodes expanded by A* search, given the tree, and straight line distance heuristics, is in the following order:

0. $(city, f(city), g(city), h(city))$
1. $(\text{Lugoj}, 244, 0, 244)$
2. $(\text{Mehadia}, 311, 70, 241)$
3. $(\text{Lugoj}, 384, 140, 244)$
4. $(\text{Drobeta}, 387, 145, 242)$
5. $(\text{Craiova}, 425, 265, 160)$
6. $(\text{Timisoara}, 440, 111, 329)$
7. $(\text{Lugoj}, 446, 222, 244)$
8. $(\text{Mehadia}, 451, 210, 241)$
9. $(\text{Mehadia}, 461, 220, 241)$
10. $(\text{Pitesti}, 503, 403, 100)$
11. $(\text{Bucharest}, 504, 504, 0)$

Problem 2

(a) BFS: $1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11$
DLS: $1 \to 2 \to 4 \to 8 \to 9 \to 5 \to 10 \to 11$
IDS:
1
$1 \to 2 \to 3$
$1 \to 2 \to 3 \to 4 \to 5 \to 3 \to 6 \to 7$
$1 \to 2 \to 4 \to 8 \to 9 \to 5 \to 10 \to 11$

(b) Bidirectional search would be effective in solving this problem. Bidirectionally finding a path between state 1 and state 11 reduces the number of nodes that needs to be explored.

Forward 1
Backward 11

Forward 2
Backward 5

The branching factor is 2 in the forward direction and 1 in the reverse direction.

Problem 3

(a) True

(b) True

(c) True

(d) False

(e) False

(f) True

(g) True

(h) True

(i) True

Problem 4

Advantage: While the BFS algorithm has a space complexity of $\mathcal{O}(b^d)$, where b is the tree branching factor and d is the depth of the tree, the iterative deepening search algorithm (IDDFS) has a space complexity of $\mathcal{O}(bd)$. In short, IDDFS saves on space while still being complete.

Disadvantage: IDDFS is slightly slower than BFS, since while nodes at the bottom of the search tree are only expanded once, the number of expansions for each node increments at each higher level in the tree.

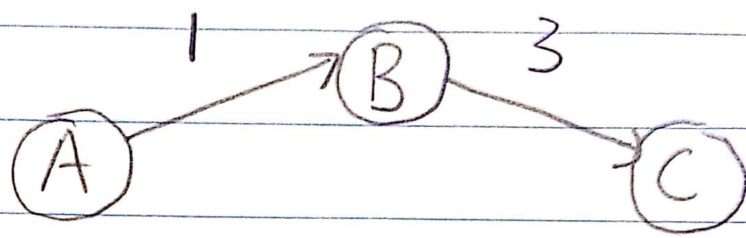
Problem 5

a)
A heuristic is considered consistent iff:
$h(n) \leq c(n, n+1) + h(n+1)$
where h() is the heuristic and n is the goal state
Proving by induction with the $n-1$ node in the shortest path to n (base

step):
$h(n-1) \leq c(n-1, n) + h(n)$
since n is the goal state, $h(n) = h^*(n)$, where $h^*(n)$ is the true minimal cost to the goal state
$h(n) \leq c(n-1, n) + h^*(n)$
given $c(n-1) + h^*(n) = h^*(n-1)$
$h(n-1) \leq h^*(n-1)$
which is the definition of admissibility

Inductive Step with $n-2$:
$h(n-2) \leq c(n-2, n-1) + h(n-1)$
drawing from the base case:
$h(n-2) \leq c(n-2, n-1) + h(n-1) \leq c(n-2, n-1) + h^*(n-1)$
$h(n-2) \leq c(n-2, n-1) + h^*(n-1)$
$h(n-2) \leq h^*(n-2)$
By inductive reasoning, consistency implies admissibility

b)



$h(A) = 4$

$h(B) = 2$

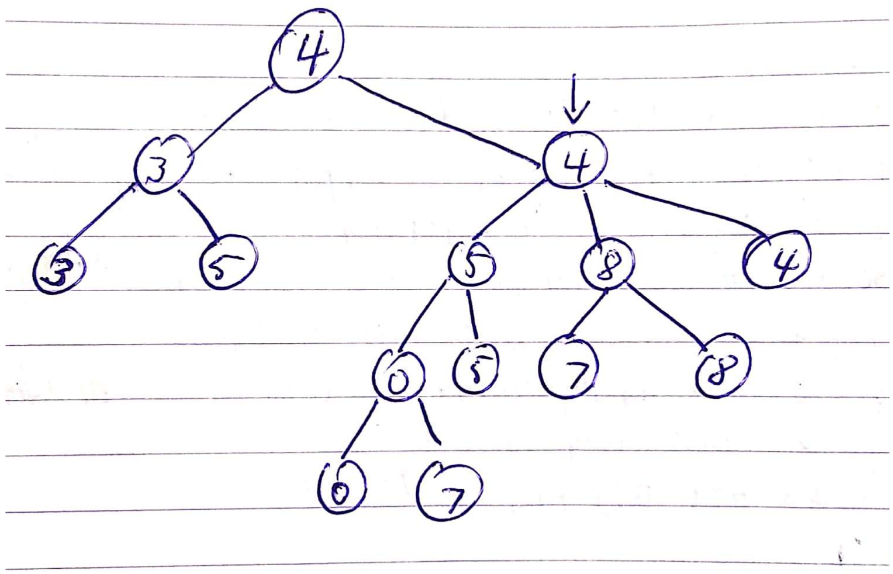
$h(C) = 0$

Problem 6

In solving Constraint Satisfaction Problems, we choose the most constrained variable (MCV) in order to reduce the size of the next sub-problem. In other words, choosing the MCV maximally reduces the size of the next branch among any of the other variables.
When assigning a value for this variable, we want to leave the remaining variables as unconstrained as possible in order to not accidentally eliminate any valid solutions. Thus, we want to assign the MCV the least constraining value (LCV).

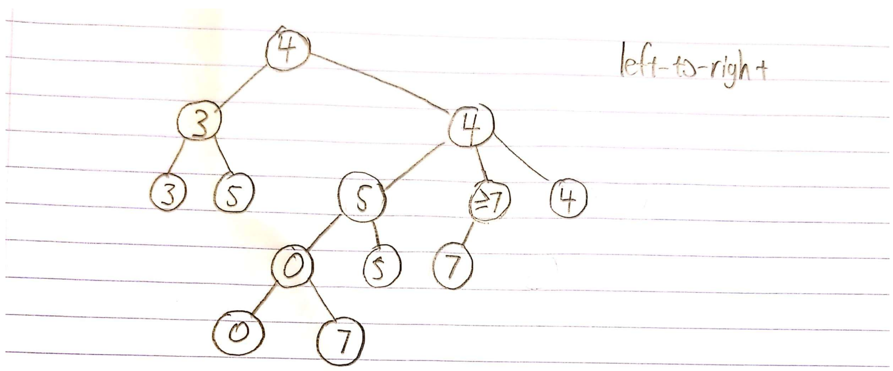
Problem 7

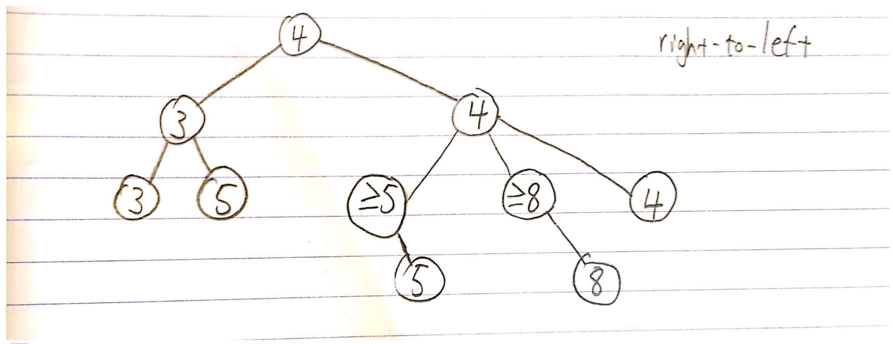
(a)



The optimal choice for the max player would be the right node.

(b)



(c) 

Alpha-beta pruning is a heuristic that depends heavily on where elements are positioned, as it only operates on the best values found so far when exploring the tree. If the algorithm happens to find the 'best' node first then it can 'prune' more nodes from the tree. This is the case in right-to-left pruning, which, due to how the elements were positioned, was able to find the 'best' value more quickly and prune more nodes.

Problem 8

(a) i. Given both heuristics h_1 and h_2 are admissible,

$$h(n) = min\{h_1(n), h_2(n)\}$$

is an admissible heuristic. Since both heuristics never overestimate the true cost of getting to the target state, then choosing the minimum of these two heuristic values for each state is an admissible heuristic, as none of the new values will overestimate the true cost of getting to the target state.

ii. Given both heuristics h_1 and h_2 are consistent:
Since $h_1(n) \leq c(n,n') + h_1(n')$ and $h_2(n) \leq c(n,n') + h_2(n')$,

$$h(n) \leq min\{c(n,n') + h_1(n'), c(n,n') + h_2(n')\}$$
$$h(n) \leq min\{h_1(n'), h_2(n')\} + c(n,n')$$

Substituting $h(n')$ for $min\{h_1(n'), h_2(n')\}$,

$$h(n) \leq h(n') + c(n,n')$$

Thus, $h(n) = min\{h_1(n), h_2(n)\}$ is a consistent heuristic.

(b) i. Given both heuristics h_1 and h_2 are admissible,

$$h(n) = wh_1(n) + (1 - w)h_2(n)$$

is an admissible heuristic. Since both heuristics never overestimate the true cost of getting to the target state, then choosing a value between these two heuristic values for each state is an admissible heuristic, as none of the new values will overestimate the true cost of getting to the target state.

ii. Given both heuristics h_1 and h_2 are consistent:
Since $h_1(n) \leq c(n, n') + h_1(n')$ and $h_2(n) \leq c(n, n') + h_2(n')$,

$$h(n) \leq w\left(c(n, n') + h_1(n')\right) + (1 - w)\left(c(n, n') + h_2(n')\right)$$
$$h(n) \leq wh_1(n') + (1 - w)h_2(n') + wc(n, n') + (1 - w)c(n, n')$$
$$h(n) \leq wh_1(n') + (1 - w)h_2(n') + c(n, n')$$

Substituting $h(n')$ for $wh_1(n) + (1 - w)h_2(n)$,

$$h(n) \leq h(n') + c(n, n')$$

Thus, $h(n) = wh_1(n) + (1 - w)h_2(n)$ is a consistent heuristic.

(c) i. Given both heuristics h_1 and h_2 are admissible,

$$h(n) = max\{h_1(n), h_2(n)\}$$

is an admissible heuristic. Since both heuristics never overestimate the true cost of getting to the target state, then choosing the maximum of these two heuristic values for each state is an admissible heuristic, as none of the new values will overestimate the true cost of getting to the target state.

ii. Given both heuristics h_1 and h_2 are consistent:
Since $h_1(n) \leq c(n, n') + h_1(n')$ and $h_2(n) \leq c(n, n') + h_2(n')$,

$$h(n) \leq max\{c(n, n') + h_1(n'), c(n, n') + h_2(n')\}$$
$$h(n) \leq max\{h_1(n'), h_2(n')\} + c(n, n')$$

Substituting $h(n')$ for $max\{h_1(n'), h_2(n')\}$,

$$h(n) \leq h(n') + c(n, n')$$

Thus, $h(n) = max\{h_1(n), h_2(n)\}$ is a consistent heuristic.

(d) Heuristic C, $h(n) = max\{h_1(n), h_2(n)\}$, should be chosen. For consistent heuristics, higher cost estimation values (without going over) are better, as less states will be explored before reaching the goal state. Thus, a heuristic that chooses the maximum among two consistent heuristics for each state is preferred over the other options.

Problem 9

(a)

(b)

(c)

(d)

(e)

(f)