Homework 1

Simon Zheng
260744353

January 30th, 2018

1 Six-Puzzle

1.1 a)

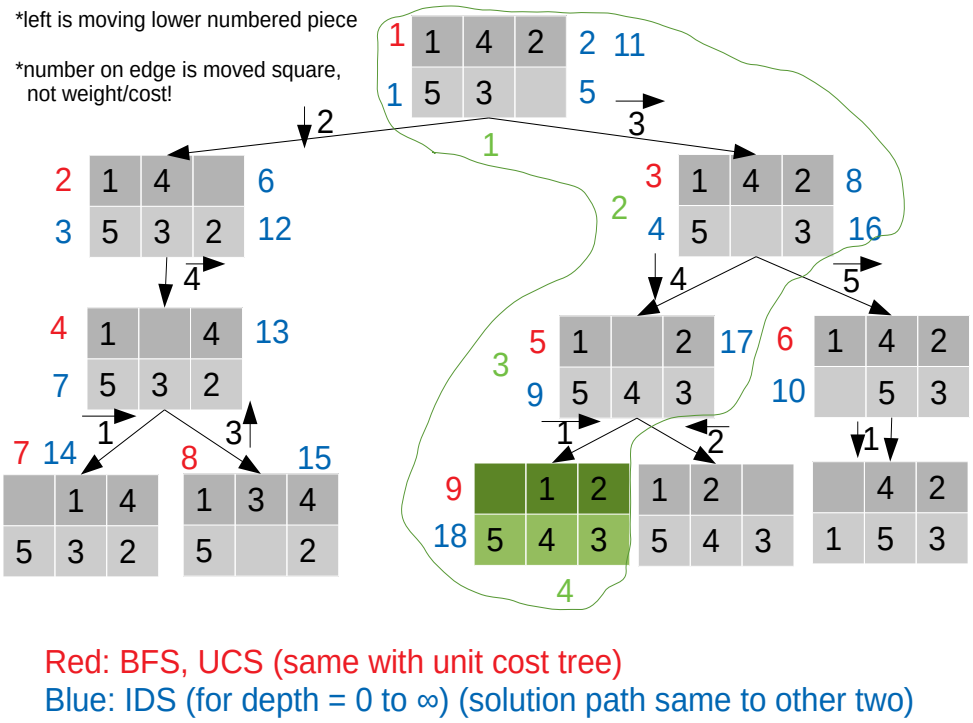


Figure 1: Search tree

1.2 b)

Yes. If we use Manhattan distance, then we have an underestimating (optimistic) heuristic as it is exactly calculating the most direct path from a square's current position to its goal position. In a version of the puzzle where squares can move over each other, it is optimal.

If we add cost (based on square values), we are not changing this fact, as while the cost would increase, Manhattan would still be the same most optimistic path so the smallest (unit) cost.

1.3 c)

In this case, we simply use a sort of "Manhattan distance" except each movement has a cost of its value (multiply the distance for each square as the final cost) for the heuristic.

It is still optimistic for the same reason as previously explained, but with the cost factored in the calculations, it dominates the unit cost version.

1.4 d)

No. Manhattan distance would overestimate every up and down movement. It would still be optimistic for most cases (as there are only two rows so at most one up and down movement but one or two left and right movement) but for the simple case of the distance being one and so only moving once up or down, Manhattan would estimate 1 even though the cost is 0.5.

2 Search algorithms

2.1 a)

A state space which is a tree with a single branch, where each child has a single child (a path) is $O(n)$ for DFS (just go down the tree) but $\sum_{k=1}^{n} = O(n^2)$ for IDS as it keeps restarting at every depth.

2.2 b)

True. If you run UCS on a graph with unit costs (or equal costs), then it will behave like BFS, thus BFS is a special case of UCS with unit costs. The algorithm would not prioritize one child or neighbor over another since the cost is not better or worst for any of them. Instead, the depth would add up equally and thus this UCS would expand the lowest depth nodes first. Thus, the priority queue acts as a normal queue and the algorithm behave as a BFS.

2.3 c)

True. A Best-First Search which uses a heuristic function where it expands the deepest node first behaves like DFS. The heuristic would evaluate the negative depth (from the root node) such that a larger depth has an inferior cost.

2.4 d)

True. We know that an A* Search evaluates which nodes to expand based on a function which calculates the cost and uses a heuristic. As an uninformed search, UCS would simply ignore the heuristic part of the evaluation function. Thus, what we are left with is the cost function, which would simply be a UCS.

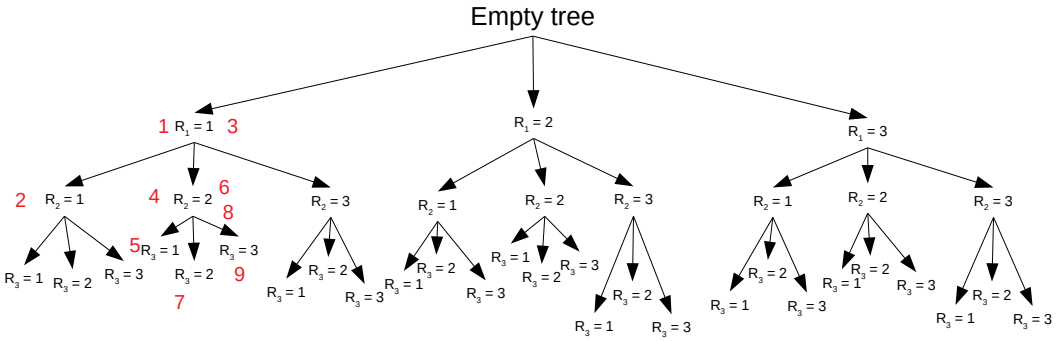
3 Optimization

4 Constraint satisfaction

4.1 a)

- Variables: rooks $\{R_1, R_2, ..., R_k\}$ (one rook per row which is the index, value is column)
- Domain: $\{1, 2, ..., n\}$, $k \in \mathbf{N}$
- Constraints: $k \leq n^2$ or $k \leq R_i \neq R_j$ (no two rooks can be on the same column)

4.2 b)



	1	2	3	4	5	6	7	8	9
R_1	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{1}
R_2	{1,2,3}	{1}	{1,2,3}	{2}	{2}	{2}	{2}	{2}	{2}
R_3	{1,2,3}	{1,2,3}	{1,2,3}	{1,2,3}	{1}	{1,2,3}	{2}	{1,2,3}	{3}

Figure 2: Search tree without forward checking. The backtracking algorithm would only generate the nodes marked with red numbers.

4.3 c)

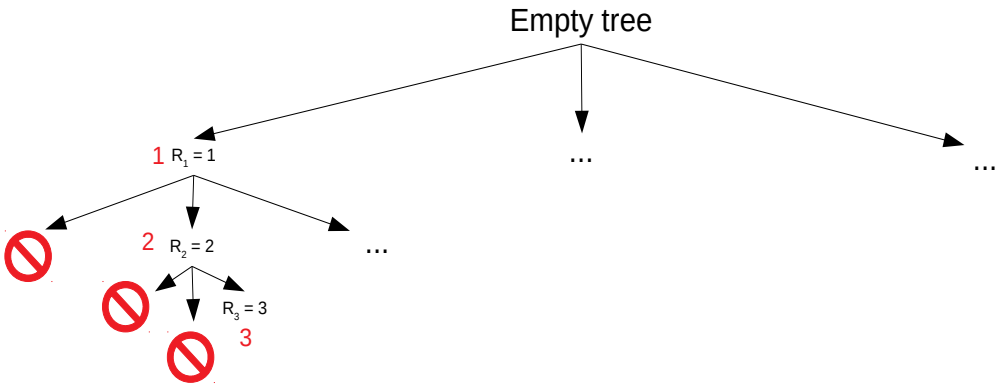


Figure 3: Search tree with forward checking.