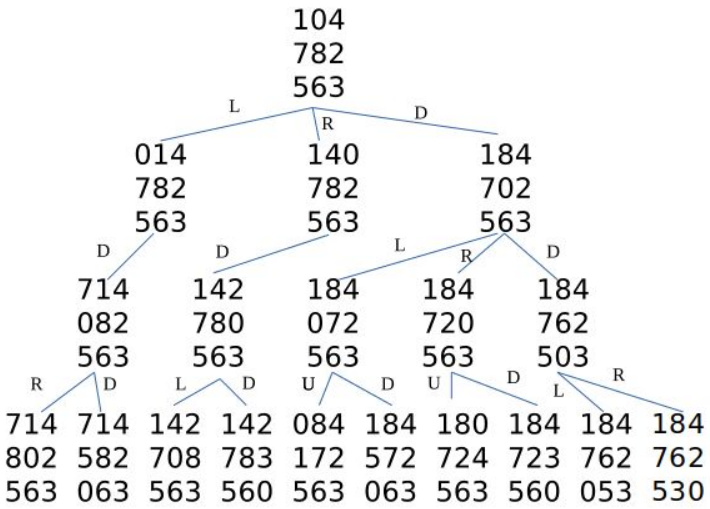
CMPUT 355 Fall 2020: Assignment 2 Solutions

Marked by Sahir (Q3), Ali Naiem Abadi (Q4), Douglas Rebstock (Q1, Q2, Q5, Q6) - address questions on marking to these TAs

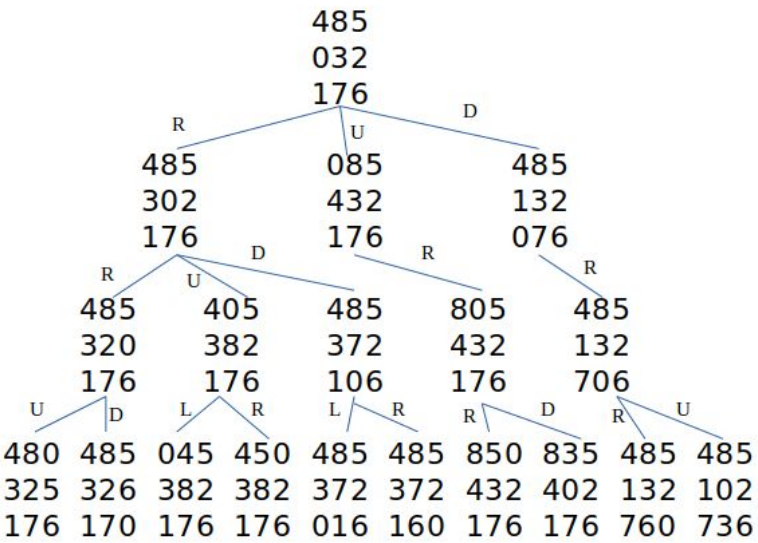
Q1,Q2: 1 mark each for completion

Q3: (3 marks)

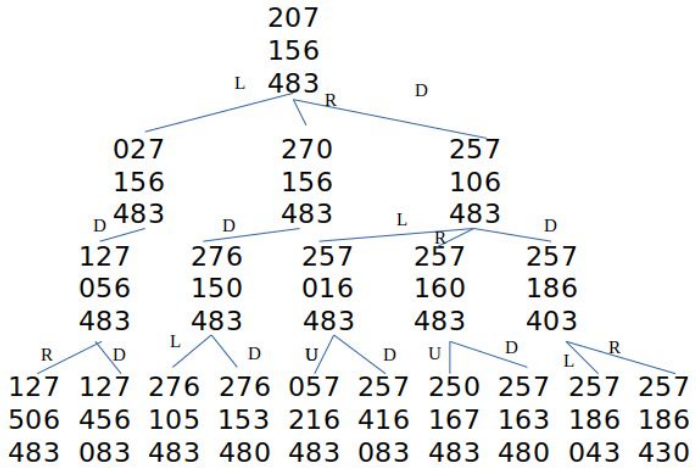
P0:



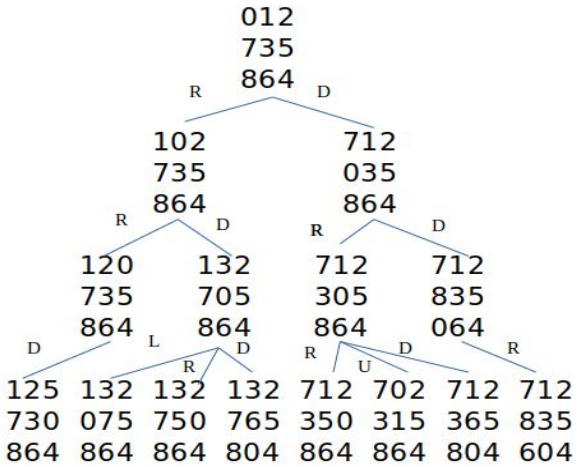
P1:



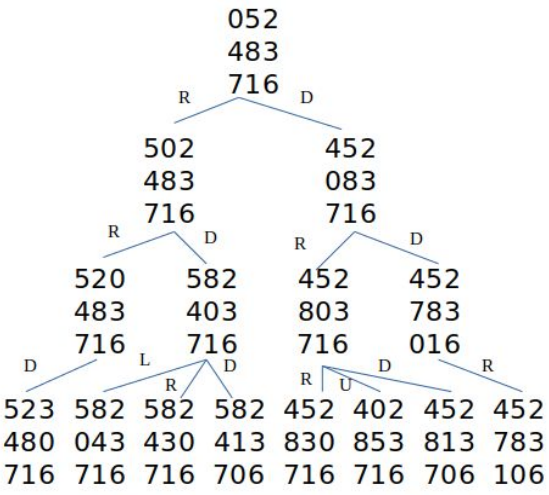
P2:


207 156 483
L R D
027 156 483
270 156 483
257 106 483
D D L R D
127 056 483
276 150 483
257 016 483
257 160 483
257 186 403
R D L D U D U D L R
127 506 483
127 456 083
276 105 483
276 153 480
057 216 483
257 416 083
250 167 483
257 163 480
257 186 043
257 186 430

P3:


012 735 864
R D
102 735 864
712 035 864
R D R D
120 735 864
132 705 864
712 305 864
712 835 064
D L R D R U D R
125 730 864
132 075 864
132 750 864
132 765 804
712 350 864
702 315 864
712 365 804
712 835 604

P4:


052 483 716
R D
502 483 716
452 083 716
R D R D
520 483 716
582 403 716
452 803 716
452 783 016
D L R D R U D R
523 480 716
582 043 716
582 430 716
582 413 706
452 830 716
402 853 716
452 813 706
452 783 106

Q4:

Puzzle : P0
number of inversions: 12
Example work: (0 + 0 + 0 + 0 + 3 + 2 + 2 + 5) = 12

Shortest path: 15
Sample explanation: this is correct since stile_search implements BFS and I used the max search depth of stile_search when ran on my puzzle. BFS iteratively expands all nodes at increasing depths (path lengths) until it finds the solution, so if there were a shorter path, the node would have been expanded earlier (which it cannot since each node is only expanded once)

NOTE: if you say it's unsolvable because of wrong inversions leads to not solvable and you show proper work, you should get partial credit

Puzzle : P1
number of inversions: 16
Is solvable? True
Shortest path: 17

Puzzle : P2
number of inversions: 12
Is solvable? True
Shortest path: 19

Puzzle : P3
number of inversions: 8
Is solvable? True
Shortest path: 18

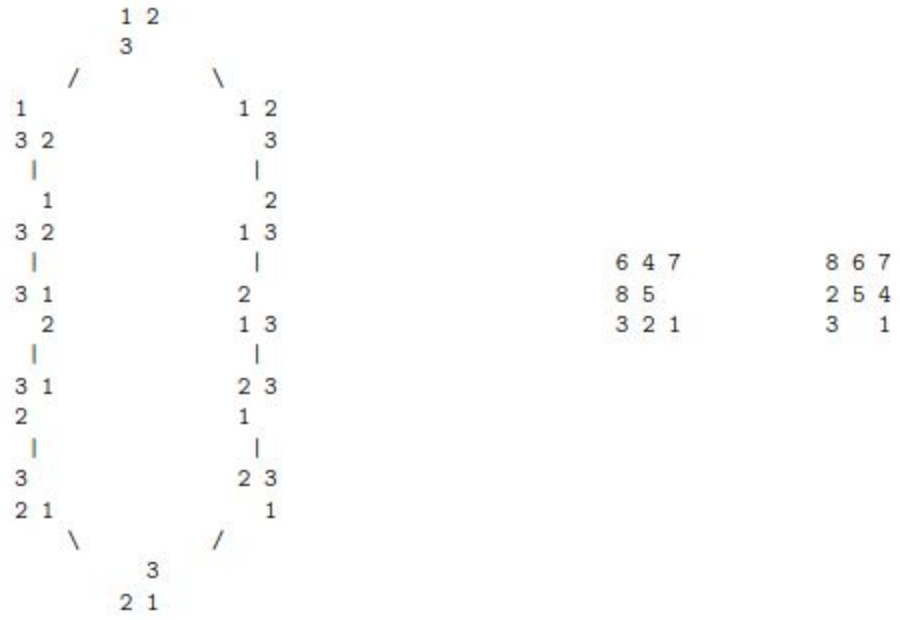
Puzzle : P4
number of inversions: 14
Is solvable? True
Shortest path: 16

Q5:

i) (1 mark) This is simply to avoid expanding the same node more than once (as we do not want to do this in BFS). In this implementation, this would cause the program to be less efficient when there is a solution (cycles and redundant paths will be explored) and it will not terminate if there is not a solution. As well, the reporting of the path is affected since the parents are overwritten. Basically, don't replace it with an **else**.

ii) (2 marks) Previously provided solution:

Some assignment questions are routine. This question was challenging. One way to answer this question is to run `stile_search.py`over all9!/2 solvable input states. Here is another way. The 2x2 sliding tile adjacency graph below starts from the goal state and includes every solvable state: the number of moves in a best solution of a state is just its distance to the top goal state.



E.g. if we start from the goal state, a shortest solution makes 0 moves. If our start is either of the two states at the next level, a shortest solution makes 1 move,and so on. So, for the 2x2 puzzle,the average number of moves in a best solution will be (0 + 1 + 1 + 2 + 2 + 3+ 3 + 4 + 4 + 5 + 5 + 6)divided by the number of solvable states, 12, so (0 + 2×1 + 2×2 + 2×3 + 2×4 + 2×5 + 6)/12= 36/12 = 3.

So, how do we find the number of nodes at each level in the corresponding diagram for 3x3?Take our 3x3 goal state, and change it by exchanging the values of the last two tiles, so (1 2 3 4 56 8 7 -). This gives us

exactly the adjacency graph we want (except that we have exchanged the8 and 7, which makes the state unsolvable and so guarantees that the algorithm will run through the complete state).

The output from `stile_search.py`on this unsolvable state shows us the number of nodes at each level of our diagram: 1 2 4 8 16 20 39 62 116 152 286 396 748 10241893 2512 4485 56389529 10878 16993 17110 23952 20224 24047 15578 14560 6274 3910 760 221 2.So, over all 181440 solvable states, the average number of moves in a best 3x3 solution is(0∗1 + 1∗2 + 2∗4 + 3∗8 + 4∗16 + 5∗20 + 6∗39 +. . .+ 30∗221 + 31∗2)/181440≈21.97.

The data above shows that only two sliding tile puzzles — shown above right — have a 31-moveshortest solution. How did I find them? Start from unsolvable 12345687. Modify `stile_search.py` to print out any puzzles at depth 31. Take these unsolvable positions and switch tiles 7,8.

Note: While we intended for the question to be solved in a manner similar to the provided solution, it does fall within the rules we provided to use online sources. However, you must still explain how the source got their answer (explain briefly was included in the question), **directly** cite the source (only listing it in Q1 without indicating that it was the source for Q5 part ii is not sufficient), and write your answer in your own words (explicitly stated on the assignment). Partial or all marks were taken off for part ii due to not following this.

Q6: (4 marks)

The greedy algorithm is not guaranteed to find (will not <u>always find)</u> an optimal solution for the given puzzle configuration (this is true for all P0 to P4).

To prove this, you just need to provide a contradiction. One method would be to work out a path that follows the greedy rule, and show that this path is longer than the optimal solution length for your puzzle.

Another option is to start with the greedy rule, then use the stile_search algorithm from this new point and show that the combined path's length is greater than the optimal solution.

Common mistakes:

- Stating that the algorithm **cannot** find an optimal solution (while this may be true for your puzzle, it is not sufficient to show a single counterexample to prove that it **cannot** find one)
- Only showing that the greedy rule can deviate from the solution reported by stile_search (stile_search provides an optimal solution, but this does not mean that there are not other equally optimal solutions)
- Finding an optimal solution that uses the greedy rule, and concluding that it will always find the optimal solution (cannot prove a positive through example, unless the example is exhaustive of all

possibilities). In the optimal solutions provided, the algorithm chose moves that had equal resulting inversions in which the other moves were not optimal.

- Not showing much / any work / explanation
- Not directly answering the question (implying an answer through shown work is not enough)