355 2020   sliding tile   asn2   due Thu Oct 8 23:59 Edmonton time

You can work on this assignment with anyone or anything, but you must acknowledge all people/things who give you any info. You must write all your answers in your own words: do not copy anyone/anything else. We might ask you later to explain your answers: if you cannot, we might deduct some or all marks, and of course report any suspected plagiarism to the appropriate authorities.

Your **key** is the 4th digit of your student id, mod 5. So, your key is in {0, . . . , 4}. E.g. if your student id is `***7**`, then your key is 2.

1. [1 mark] **If you don't answer this question, we won't be able to mark the rest of your assignment.** Give the names of all persons you consulted on this assignment, and explain who each is (e.g. classmate, friend, family, etc.). Other than the class webnotes, lectures and github repo, list any resource (book, url, video, etc.) you consulted.

2. [1 mark] How are you coping this semester, compared to pre-covid? Harder/easier to learn? Taking more/less of your time? More/less stressed? Suggested changes?

3. [3 marks] The webnotes shows the top 4 levels of a sliding tile search space tree. Below, from left, are positions P_0, . . . , P_4. Your position is P_k, where k is your key. Starting from your position, by hand draw the top 4 levels of the search space tree.

```
1 * 4      4 8 5      2 * 7      * 1 2      * 5 2
7 8 2      * 3 2      1 5 6      7 3 5      4 8 3
5 6 3      1 7 6      4 8 3      8 6 4      7 1 6
```

4. [3 marks] (i) For your position above, give the number of inversions. Show your work. (ii) If your position is unsolvable, explain why. If your position is solvable, give the minimum number of moves needed to solve your puzzle, and explain how you know your answer is correct.

5. [3 marks] (i) In `simple/stile/stile_search.py`, why is line 123 `elif nbr not in Parent:` and not `else:` ? Explain briefly. (ii) On average, over all solvable 3x3 sliding tile puzzles, how many moves are needed to solve the puzzle? Explain briefly.

6. [4 marks] **(You might find `stile_search.py` and `play_stile.py` helpful here.)** A **greedy** algorithm is one with an easily computed rule that is locally optimal. Greedy algorithms are usually fast, but they don't always find a best solution. Here is a greedy sliding tile algorithm: at each step, make a move which most decreases (or, if all moves increase, least increases) the number of inversions. For your position P_n above, does this greedy algorithm find a best (least number of moves) solution? Explain. If yes, give the sequence of positions in the greedy solution, with the number of inversions of each position. If no, give a position in your solution sequence from which the greedy algorithm does not find a best solution: give the greedy solution, a better solution, and the number of inversions in each position in your answer.