cmput 355 2025 practice questions 2 (with answers)

learning outcomes (LO) For game/puzzle algorithm, you will learn to

1. analyze algorithm runtime and space usage with mathematical reasoning,
2. construct logical proofs to demonstrate algorithm correctness and behaviour,
3. explain the behaviour of algorithms, starting with a precise description of the rules,
4. propose potential improvements of algorithms,
5. implement algorithms using python3.

1. [LO 3] What is the maze puzzle (MP), a.k.a. maze traversal problem? What is the sliding tile puzzle (STP)? Explain how the solving the MP is similar to solving the STP.

   Answer. MP: given a maze, start position, and target value, find a path from start to target. STP: given a start ST position and target position, find a sequence of tile moves that goes from start to target. Both problems can be solved by traversing the associated graph, e.g. with BFS or DFS.

2. [LO 3] Why does the 355 github repo STP solver use breadth first search rather than depth first search?

   Answer. With BFS, the path found in the game graph is a shortest path.

3. [LO 2,3] a) Recall: for an $r \times c$ STP, the *state space graph*, also called the *STP graph*, is a graph where each node corresponds to a ________ and two nodes are adjacent if and only if

____ .

Starting from the position below left, draw the next two levels of the STP graph. In your diagram, circle each node.

```
3 5 2                3 5 1
4 _ 1                4 _ 2
```

b) Repeat a) for position above right: draw your diagram isomorphic to that for a).

Answer. a) sliding tile position. you can get from one position to the other with one sliding tile move. Below, I leave it to you to draw the edges and circle the nodes.

```
                  3 5 2
                  4 _ 1

3 5 2             3 _ 2             3 5 2
_ 4 1             4 5 1             4 1 _

_ 5 2       _ 3 2       3 2 _       3 5 _
3 4 1       4 5 1       4 5 1       4 1 2
```

```
b)                3 5 1
                  4 _ 2

3 5 1             3 _ 1             3 5 1
_ 4 2             4 5 2             4 2 _

_ 5 1       _ 3 1       3 1 _       3 5 _
3 4 2       4 5 2       4 5 2       4 2 1
```

4. [LO 1,3] Find the average degree in the a) 3×3 b) 2×5 c) 4×4 STP graph. Show your work.

   Answer. a) 9 tile locations: 4 corners, 4 sides, 1 center. when the empty space is in those locations, the number of possible moves is 2, 3, 4 respectively, so the average degree is
   $(4 * 2 + 4 * 3 + 1 * 4)/9 = 24/9 = 8/3$.
   b) $(4 * 2 + 6 * 3)/10 = 26/10 = 13/5$
   c) $(4 * 2 + 8 * 3 + 4 * 4)/16 = 48/16 = 3$

5. [LO 1,3] Assume that the runtime for an unsolvable 3×3 STP is 2.3s. Estimate the runtime for an unsolvable a) 2×5 STP b) 4×4 STP. Show your work.

   Answer. For BFS, wc runtime is roughly proportional to number of graph edges. Define e_{33}, e_{25}, e_{44} as number of edges in each graph respectively. Define n_{33}, n_{25}, n_{44} as number of nodes in each graph respectively. Here, these equal $9!, 10!, 16!$ respectively. Sum, over all nodes, of node degree is two times the number of edges, so number of edges is $0.5 \times$ sum, over all nodes, of degree.

   Thus e_{33}, e_{25}, e_{44} equal $9! * 0.5 * 8/3$, $10! * 0.5 * 13/5$, $16! * 0.5 * 3$ respectively.

   Answer will be 2.3s $\times$ (e_{25}/e_{33})

   $= 2.3\text{s} \times (10! * 0.5 * 13/5)/(9! * 0.5 * 8/3)$

   $= 2.3\text{s} \times (10 * 13/5)/(8/3)$

   $= 2.3\text{s} \times (10 * 13/5 * 3/8)$

   $= 2.3\text{s} \times 39/4$

   $= 22.45\text{s}$.

6. [LO 2,3] Here are two 3×7 STP positions. Some tile numbers are hidden.

   a) At left, what is the change in the number of inversions after move **18 up**? Explain.

   b) At right, what is the change in the number of inversions after move **11 down**? Explain.

```
 ?  ?  ?  ? __  4 15        ?  ?  ?  ?  ?  ?  ?
 7 22 11 24 18  ?  ?        ?  ? 11  3 18  5 26
 ?  ?  ?  ?  ?  ?  ?       15  9 __  ?  ?  ?  ?
```

   Answer. a) The relative order of tile 18 and any hidden position does not change, so the change in the number of inversions is given by the change in inversions from the subsequence `(__ 4 15 7 22 11 24 18)` to subsequence `(18 4 15 7 22 11 24 __)`. Before the change, 18 is out of order with respect to 2 tiles in the subsequence, so after the change it is out of order with all the other tiles, so $6 - 2 = 4$. So the number of inversions goes up by exactly 2.

   b) Before, 11 is inverted with 3 tiles in the subsequence, so after it is inverted with $6 - 3 = 3$ tiles, so no change in the number of inversions.

7. [LO 2,3] For each STP, give the change in the number of inversions when a tile is moved

   a) up b) down c) left d) right.

```
3 8 7       3  8  7  9
2 _ 5       2  _  5 11
1 4 6       1 10  4  6
```

   Answer.

   a) permutation subsequence (8,7,2) changes to (7,2,8), inversions 3 changes to 1: -2.
      permutation subsequence (8,7,9,2) to (7,9,2,8), inversions 4 to 3: -1.
   b) perm'n subs. (5,1,4) to (4,5,1), inversions 2 to 2: 0.
      perm'n subs. (5,11,1,10) to (10,5,11,1), inversions 3 to 4: $+1$.
   c) permutation unchanged: no change in inversions.
   d) permutation unchanged: no change in inversions.

8. [LO 2,3] For any STP position X, let Y be the position obtained from X by exchanging the locations of tiles 1 and 2. For any position P, let $\mathcal{I}(P)$ be the number of inversions. Give all possible values for $\mathcal{I}(X) - \mathcal{I}(Y)$. Justify your answer.

   Answer. $\{+1, -1\}$. In the sorted list of numbers, there are no numbers between 1 and 2, so changing their respective positions will change only the {1,2} inversion: all other inversions will be unchanged. In one of these positions, 1 appears before 2 and {1,2} is not inverted. In the other position, 2 appears before 1 and {1,2} is inverted, so the latter position has exactly one more inversion than the former position.

9. [LO 2,3] For each STP, give the solvability condition and whether it is solvable.

```
1 5 4     1 4 _     4 _     4 3
3 2 _     3 2 5     2 3     1 _
                    1 5     2 5
```

   Answer. From left.
   columns parity odd: solvable iff inversions parity (here 6) is even. solvable.
   columns parity odd: solvable iff inversions parity (here 3) is even. unsolvable.
   columns parity even: solvable iff inversions parity (here 5: odd) differs from blank's row-from-bottom parity (here 2: even). solvable.
   columns parity even: solvable iff inversions parity (here 5: odd) differs from blank's row-from-bottom parity (here 1: odd). unsolvable.

10. [LO 2,3] Claim: for an STP with an odd number of columns, every move leaves the parity of the number of inversions unchanged. Prove the claim.

    Answer. Assume that there are t columns. If the move is slide left or slide right, the number of inversions is unchanged. Assume that the move is to slide a tile up. For the tile p_k at the position above the blank at the start of the move, the permutation subsequence $(p_k, p_{k+1}, \ldots, p_{k+t-1})$ changes to $(p_{k+1}, \ldots, p_{k+t-1}, p_k)$. Exactly $t-1$ pairs, each with p_k, are reversed by this move. In this subsequence, if p_k was in 0 inversions before, it is in $t-1$ now, an increase in $t-1$ inversions; if it was in 1 inversion before, it is in $t-1-1 = t-2$ now, an increase in $t-2-1 = t-3$ inversions, and so on. In each case, the change in the number of inversions is even.

11. [LO 2,3] Recall: the STP target position has tiles sorted (row by row) in increasing order followed by the empty space. Define *solvable position*.

    Answer. A position p is solvable if there is a sequence of moves from p to the target position.

12. [LO 2,3] Claim: for two solvable STP positions, there is a sequence of moves from one to the other. Prove the claim.

    Answer. Call the positions P_1 and P_2. P_1 is solvable, so there is a sequence S_1 of moves from P_1 to target. P_2 is solvable, so there is a sequence S_2 of moves from P_2 to target. So the sequence S_1 followed by the reverse of S_2 goes from P_1 to target to P_2. So we have found a sequence of moves from P_1 to P_2.

13. [LO 2,3] Let x, y be two solvable positions in the STP graph. We know that there is a path between then in the STP graph: let $p = (p_0 = x, \ldots, p_t = 7)$ be any such path. Let x' (respectively y') be the position obtained from x (resp. y) by exchanging the locations of tiles 1 and 2. Define $p' = (p'_0, \ldots, p'_t)$ similarly. Explain why p' is a path in the STP graph.

    Answer. Let m be the move that takes $x = p_0$ to p_1. Then the same move m takes $x' = p'_0$ to p'_1. Repeating this argument for each consecutive pair of nodes in p', we see that the same sequence of moves that takes x to y also takes x' to y'.

14. [LO 2,3] Explain why the STP graph has exactly two components.

    Answer. A component is a set of nodes such that, for each pair in the set, there is a path between them, so this follows from the previous question.

15. [LO 2,3] Claim: for two unsolvable STP positions, there is a sequence of moves from one to the other. Prove the claim.

    Answer. This is just another way to ask question 13, so the answer to that question also answers this question.

16. [LO 2,3] Claim: there is no sequence of moves from a solvable STP position to an unsolvable one. Prove the claim.

    Answer. Argue by contradiction: assume there is such a sequence. Then we can get from an unsolvable position to a solvable position and then (because it is solvable) to the target, so we can get from the first position to the target. Thus, by the definition of solvable position (there is some sequence of moves to the target), the first position is solvable. But we assumed it was not solvable, contradiction.

17. [LO 2,3] Using info below from `python3 stp_search2.py` give a solvable STP whose shortest solution has 31 moves. Explain briefly.

```
start 821354670
level 32: 0 nodes, no sol'n found
last psn seen 746253108
```

    Answer. From the info: there is a path from start $s_0 = 821354670$ to last position seen (LPS) $q_0 = 746253108$. We want to relabel the tiles so that one of relabelled positions s_1, q_1 will be the target 123456780. We can't do this with q_0 (blank is not at end of last row), but we can with s_0. Relabel $s_0 =$ as new target 123456780: relabel 8 as 1, 2 as 2, 1 as 3, ... 7 as 8 (below left): this gives the permutation below middle. Under this permutation, relabelling q_0 gives $q_1 = 867254301$ (below right), a solvable STP with shortest solution 31 moves.

```
82135467   (1 2 3 4 5 6 7 8)     8 6 7
12345678   (3 2 4 6 5 7 8 1)     2 5 4
                                 3 _ 1
```

18. [LO 1,2,3] Using info below from `python3 stp_search2.py` give a solvable STP whose shortest solution has 21 moves. Explain briefly.

```
start 231540
level 22: 0 nodes, no sol'n found
last psn seen 540231
```

    Answer. From info: there is a path from start $s_0 = 231540$ to last position seen (LPS) $q_0 = 540231$. Relabel the tiles so that s_1 (relabelled s_0) is target 123450 (we can't do this with q_0 because the blank is in the wrong location), so relabel 2 as 1, 3 as 2, 1 as 3, 5 as 4, 4 as 5 (below left): this gives the permutation below middle. Under this permutation, relabelling $q_0 = 540231$ gives $q_1 = 450123$ (below right), a solvable STP with shortest solution 21 moves.

```
23154   (1 2 3 4 5)    4 5 _
12345   (3 1 2 5 4)    2 3 1
```

19. [LO 2,3] For each STP, give (a) number of inversions (b) whether solvable (c) number of miplaced tiles (d) taxicab score. Justify briefly.

```
4 3 1     4 _ 2     1 4 _    1 5 4
_ 2 5     3 1 5     3 2 5    3 2 _
```

Answer. Check your answers with `play_stile.py`: 5 no 5 6. 5 no 5 8. 3 no 4 7. 6 yes 4 8.

20. [LO 1,3] a) Solve this STP. After each move, show the position (you might not need all space given). The first move has been done for you.

```
start   5 4 3   _ 4 3   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _
        _ 2 1   5 2 1   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _

_ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   1 2 3  finished
_ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   _ _ _   4 5 _
```

b) When solving the above STP with breadth-first search, the number of positions encountered is around **(circle ONE ONLY)**

10 50 100 150 200 250 300 350 700 1400 3000 6000 12000

Answer. a) Solution from `stp_search2.py` with input 23.3.

```
543   43   4 3   423   423  42    4 2   412   412    12   1 2   12    123
 21  521   521   5 1   51   513   513   5 3    53   453   453   453   45
```

b) From this practice question set, we know that the maximum length solution of a 2×3 STP is 21 moves. This solution takes 12 moves, so we might guess that the number of nodes searched is about 12/21 the number of nodes in the solvable component, which is (2×3)!/2 = 360. 12/21∗360 is around 200, so that is our guess. (From running the program we see that the search explored 187 nodes.)

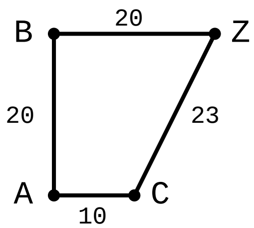
21. [LO 3] For STPs, what algorithm(s) always find(s) a shortest solution? **(circle ALL that apply)**

a) breadth-first search   b) A*-search with taxicab distance heuristic

c) depth-first search   d) A*-search with number-misplaced-tiles heuristic

Answer. a) b) d)

22. [LO 3] Here is a road map and `astar.py` output after node A is done. Show output after next node is done. ERD[x]: est. remaining dist to Z. DSF[x]: dist-so-far from A. ETD[x]: est. total dist A to x to Z.

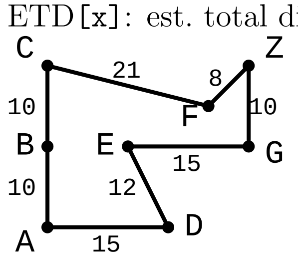


	A	B	C	Z
ERD	28	20	22	0
DSF	0	inf	inf	inf
ETD	0	inf	inf	inf
done?	yes			
DSF	0	___	___	___
ETD	0	___	___	___
done?	yes			

Answer. From running `stile/astar.py`.

DSF	0	20	10	inf
ETD	0	40	32	inf
done?	yes		yes	

23. [LO 3] Here is a road map and `astar.py` output after node D is done: show output after next node is done. ERD[x]: est. remaining dist to Z. DSF[x]: dist-so-far from A. ETD[x]: est. total dist A to x to Z.



	A	B	C	D	E	F	G	Z
ERD	28	26	24	22	18	7	10	0
DSF	0	10	20	15	inf	inf	inf	inf
ETD	0	36	44	37	inf	inf	inf	inf
done?	yes	yes		yes				
DSF	0	10	___	15	___	___	___	___
ETD	0	36	___	37	___	___	___	___
done?	yes	yes	___	yes	___	___	___	___

Answer. From running `stile/astar.py`.

DSF	0	10	20	15	27	inf	inf	inf
ETD	0	36	44	37	45	inf	inf	inf
done?	yes	yes	yes	yes				

24. [LO 1,3] We ran `stile/15puzzle.py -p 15 14 13 12 10 9 8 11 7 6 4 2 5 1 3` three times, once for each schedule A,B,C. (schedule A places tiles {1,2,3,4} first, schedule B places tile {1} first, etc). For each run, in the solution found, guess the total moves made and nodes searched.

Hint: each answer is in {82, 90, 120, 6865, 145722, 1765263 }.

	moves	searched
A)[[1,2,3,4], [5,9,13], [6,7,8,10,11,12,14,15]]	_____	_____
B)[[1], [2], [3,4], [5], [6], [7,8], [9,13], [10,14], [11,12,15]	_____	_____
C)[[1,2], [3,4], [5,6,7,8], [9,10,11,12,13,14,15]]	_____	_____

Answer. Schedule A has 3 subtasks: first place tiles 1,2,3,4 (top row); then place tiles 5,9,13 (left column); then finish. Schedule B has 9 subtasks. Schedule C has 4 subtasks. As the number of subtasks increases, we expect the number of moves in the solution to increase (because those subtasks might be taking us out of the way of a shortest solution) and the number of nodes searched to decrease (because once we reach an intermediate target we reset our queue to be just the neighbours of the target). The smaller numbers will be for the total number of moves, the larger for the nodes searched. So we guess as shown below:

	moves	searched
A)[[1,2,3,4], [5,9,13], [6,7,8,10,11,12,14,15]]	__82_	1765263
B)[[1], [2], [3,4], [5], [6], [7,8], [9,13], [10,14], [11,12,15]	_120_	6865
C)[[1,2], [3,4], [5,6,7,8], [9,10,11,12,13,14,15]]	__90_	145722

Extra problems.

25. For this STP, give A) the number of inversions and B) the taxicab value. Also, below this position (the root), draw the next two levels of the search space graph. Also, give C) the number of positions reachable from this position, i.e. the number of nodes in this component of the search space graph.

```
A ____   B ____    C ____     1 3 5 7
                              2 4 6 -
```

26. What is the maximum number of inversions in a STP with these dimensions:
    (a) 3×3 (b) 3×4 (c) 4×4 (d) $r\times c$ Explain briefly.

27. P is the STP below. Q is obtained from P by exchanging the places of the tiles 1 and 2.

    a) Draw the first three levels of a breadth first search of the STP graph starting from P.

    b) Repeat the question for Q.

    c) Which of the positions in your answer to a) are solvable?

    d) Which of the positions in your answer to b) are solvable?

    e) Give the number of positions in the STP graph containing P. Repeat for Q.

    f) Explain why your answers to e) are the same. Explain why the two STP components are isomorphic.

```
_ 3 5
2 4 1
```

28. Prove: the solution position of a STP has 0 inversions.

29. Prove: for a STP with an odd number of columns, the change in the number of inversions after each move is an even number.

30. Prove: every solvable STP with an odd number of columns has an even number of inversions.

31. a) Prove that this STP is unsolvable.

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

b) In the class github repo, execute `stile/stile_search_v2.py < in/33no` . What are the two positions found at level 31 (the deepest level) of the search?

c) Give a permutation of 1 to 8 that maps the puzzle in a) into the STP solution position.

```
number          1   2   3   4   5   6   7   8
permutation   (                               )
```

d) Apply permuation c) to each position from b) and show each new position.

e) Execute `stile_search_v2.py < in/33longa`. How many moves does it take to solve `in/33longa`? Can there be a shorter solution?

f) Are `in/33longa` and `in/33longb` the only two solvable $3{\times}3$ STPs with longest solution? Explain carefully.

32. (a) Run `stile/stile_search_v2.py < in/300`. Is this STP solvable or unsolvable?
(b) Create a new file `in/300no` by exchanging the positions of tiles 1 and 2, and repeat (a).
(c) Explain briefly exactly one of the STPs in (a,b) is solvable.
(d) Using the output data from these two executions of `stile/stile_search_v2.py < in/300`, prove that every $3{\times}3$ STP with an even number of inversions is solvable.

33. The sliding tile solvability formula assumes that the target position has tiles in ascending order and with the blank in the last position in the last row. Modify the formula so that it works as follows: given a starting position and a target position, return `True` if and only if there is some sequence of moves that goes from start to target.