cmput 355 2024 homework 2

1. What is the maze traversal problem? Explain how solving a sliding tile puzzle (STP) is similar to maze traversal.

2. At right, unscramble this code (see `hexgo/stone_board.py`) so that it prints nodes in (a) dfs preorder (b) dfs postorder. Write line numbers only: indent properly. We have written the first line number for you.

```
print(p)                  #(1)        (6)  ___  ___  ___  ___  ___
self.dfs(nbr, seen)       #(2)             ___  ___  ___  ___  ___  ___
seen[p] = True            #(3)             ___  ___  ___  ___  ___  ___
if not seen[p]:           #(4)             ___  ___  ___  ___  ___  ___
for nbr in self.nbrs[p]: #(5)              ___  ___  ___  ___  ___  ___
def dfs(self, p, seen):   #(6)             ___  ___  ___  ___  ___  ___
```

3.
```
for sort_nbrs in (True, False):
  brd.bfs_demo(2, sort_nbrs)
  brd.dfs_demo(2, sort_nbrs, True)
  brd.dfs_demo(2, sort_nbrs, False)
```

   In the above code (from `hexgo/hex.py`), explain each of the following:

   a) brd? b) `brd.dfs_demo`? c) 2? d) `sort_nbrs`? e) line 3 versus line 4?

   f) Show code output for the 2×3 hex board with these neighbor sets:

```
-4 {0, 1, 2}  0 {1, 3, -4, -1}    3 {0, 1, 4, -1, -2}
-3 {2, 5}     1 {0, 2, 3, 4, -4}  4 {1, 2, 3, 5, -2}
-2 {3, 4, 5}  2 {1, 4, 5, -4, -3} 5 {2, 4, -3, -2}
-1 {0, 3}
```

   g) Show code output for the 3×4 go board with these neighbor sets:

```
0 {1, 4}     4 {0, 8, 5}      8 {9, 4}
1 {0, 2, 5}  5 {1, 4, 6, 9}   9 {8, 10, 5}
2 {1, 3, 6}  6 {2, 10, 5, 7} 10 {9, 11, 6}
3 {2, 7}     7 {11, 3, 6}    11 {10, 7}
```

4. For each STP, give (a) the number of inversions and (b) whether the STP is solvable. Justify briefly.

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

5. 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.

6. For this STP, give A) the number of inversions and B) the taxicab value. Also, from 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.

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

7. 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 _
```

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

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

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

9. Prove: if you take a STP position and swap the locations of tiles 1 and 2, the number of inversions changes by exactly 1.

10. 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 24 18  5 26
 ?  ?  ?  ?  ?  ?  ?       15  9 __  ?  ?  ?  ?
```

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

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

13. Run `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, give 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]]                  _____ _____
```

14. 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  (                        )
```

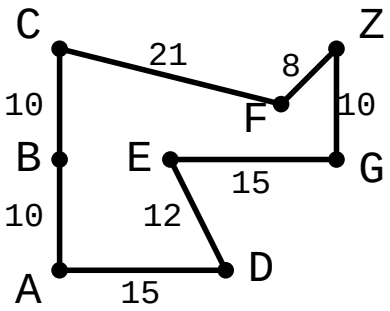
d) Apply permuation c) to each position from b) and show each new position.

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 3×3 STPs with longest solution? Explain carefully.

15. (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 3×3 STP with an even number of inversions is solvable.

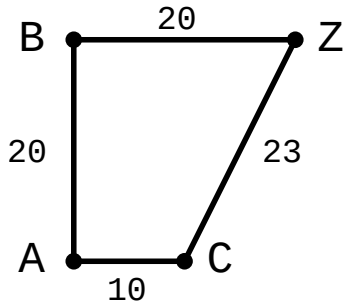
16. Here are two roadmaps labelled with distances. Below are heuristic estimates of distance Z, and A* pseudocode from class. Trace the pseudocode on the graph at left, with start A and finish Z. Each time `current` is assigned a node, give the node and its priority. For the graph at right, this is the answer: A 0, C 32, Z 33



heuristic estimates

BZ	CZ	DZ	EZ	FZ	GZ	ZZ
26	24	22	18	7	10	0

answer: _____



heuristic estimates

BZ	CZ	ZZ
20	22	0

answer: A 0, C 32, Z 33

```
fringe = PQ()
fringe.add(start, 0)
parent, cost, done = {}, {}, []
parent[start], cost[start] = None, 0
while not fringe.empty():
  current = fringe.remove() # min priority
  done.add(current)
  if current == target: break
  for next in nbrs(current):
    if next not in done:
      new_cost = cost[current] + wt(current, next)
      if next not in cost or new_cost < cost[next]:
        cost[next] = new_cost
        priority = new_cost + heuristic(next, target)
        fringe.add(next, priority)
        parent[next] = current
```