CSE422
Assignment (1)

Name: Subhi Bhuiyan
ID: 17201116

Ans 1: (a) • Initial state: Choose a random piece.
Assuming the piece is a straight one

• Succesor function: Add any piece type for the open peg.
We can add the curved pieces in either orientation.
The fork should be added in both orientation and connected at either hole.

• Goal test: Use all the pieces to create a single connected track with no overlapping tracks and no open pegs nor holes

• Step cost: one per piece

(b) Depth first search.
I think DFS is the appropriate choice as it outperforms BFS in the case of memory-limitation.
DFS takes a linear amount of space with respect to the depth of the tree and consumes very less memory space.
So DFS is suitable because of the big state space.

(c) A fork is used to create a split of two tracks and the only way to recombine these two tracks would require a fork.
So for the problem to be solvable, a fork is required in order to have an ending to the track.

(d) Maximum possible numbe for open pegs is 3.
Assuming each piece is unique, which gives us:

Space:

$(12 + (2 \times 16)) + (2 \times 2) + (2 \times 2 \times 2)$ choices per peg.

= 56 total choices on each peg.

~~Therefore, 32 pieces which leads to 32 depth~~

~~$\therefore$ upper bound = $\frac{16^{32}}{12! * 10! * 2! * 2!}$~~

Multiplying $56 * 3 = 168$ for each of the 3 open pegs.

Upper bound = $\frac{(168)^3}{(12! * 16! * 2! * 2!)}$

the factorials account for the permutation that used the pieces twice

Ans 2: (a) If the constraints for TSP are relaxed such that, each city can be visited more than once and the cost for repeated edges are not counted, then we can get a solution for Minimum Spanning Trees problem.

(b) MST dominates straight-line distance because the shortest distance from any two cities is their straight-line distance.

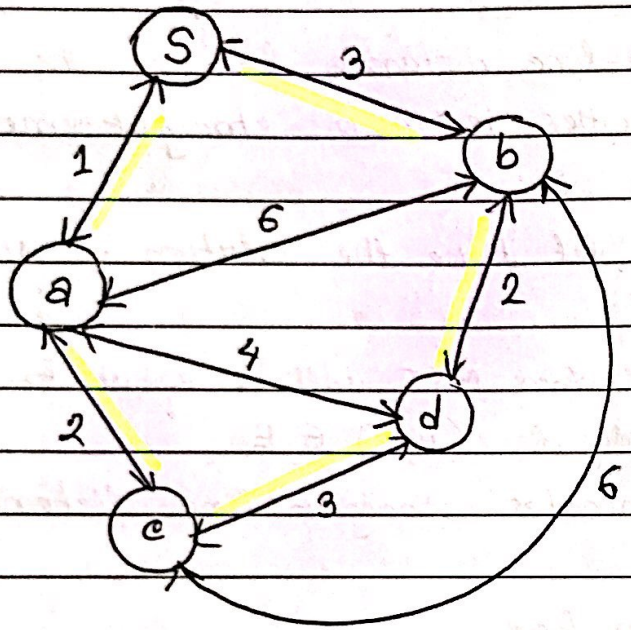
For a given $G_{MST}(V,E)$, let it be the solution of MST problem given the instance G.
The cost from u to v for MST will be equal to the straight-line distance only if $(u,v) \in E$.
Thus MST heuristic dominates straight-line distance.

(c) A simple generator can be:

```
Generator (Integer Nums){
    POINTS = nil
    N=0
    while (N < Nums){
        (x,y)= (Random (0,1) , Random (0,1))
        if ((x,y) ∉ POINTS){
            POINTS ← (x,y)
            N = N+1
        }
    }
    return POINTS
}
```

(d) Any MST Algorithm is accepted.

Here, we are using A* algorithm with MST as its heuristic function



$S \to a$: $F(a) = G(a) + H(a) = 1 + (3+2+3) = 9$

$S \to b$: $f(b) = G(b) + H(b) = 3 + (1+2+3) = 9$

$S \to a \to c$: $f(c) = G(c) + H(c) = (1+2) + (2+3) = 8$

~~$S \to b$~~:

$S \to a \to b$: cannot construct a connected tree

$S \to a \to d$: cannot construct a connected tree

$S \to a \to c \to d$: $F(d) = G(d) + H(d) = (1+2+3) + (3) = 9$

~~$S \to a \to c$~~ $S \to b \to c$: cannot construct a connected tree

$S \to b \to a$: cannot construct a connected tree

$S \to a \to c \to d \to b$: $F(b) = G(b) + H(b) = (1+2+3+2) + 0 = 8$

$S \to b \to d \to c$: $F(c) = G(c) + H(c) = (3+2+3) + (2+1) = 11$

$S \to a \to c \to d \to b \to S$: GOAL

$F(b) = G(b) + H(b) = (1+2+3+2+3) + 0 = 11$

Ans 3: (a) Local beam search with $k=1$

→ Hill Climbing Search

In this, search is started in one random start state and looks at all the neighboring states and continues the search in the direction of the best neighbour.

(b) Local beam search with one initial state and no limit on the number of states retained.

→ Breadth First Search

We start with one state and look at all the neighbours, and then continue to look at their neighbors and so on.

(c) Simulated annealing with $T=0$ at all times

The search is identical to first-choice hill climbing because every downward successor would be rejected with probability 1.

(d) Simulated annealing with $T=\infty$ at all times

The search would be identical to a random walk because every successor would be accepted with probability 1.

In this case, a random walk is equivalent to Depth-first Search

(e) Genentic Algorithm with population size $N=1$

This statement follows the Random walk. If the population size is 1, then the two selected parents will be the same individual. Crossover yields an exact copy of the individual; then there's a small chance of mutation.

Thus, the algorithm executes a random walk in the space of individuals. With only one individual, we cannot do selection or crossover.

Ans 4: Simulated Annealing is very similar to Hill climbing, where we use the greedy approach for choosing the best move. We pick a neighbor randomly even if it is the worst solution and the move is accepted only if the situation is improved. Otherwise, the algorithm accepts the move with the same probability less than 1.

Having 32 pieces in total, we can remove any piece and then rejoin it anywhere else. For closed loops, moving one joint will make others or all of it to move.
To reduce the set of actions, we must ensure resulting connection is ready to be restarted.

By using Simulated Annealing, which samples the next state, evaluates it and takes the next according to
~~$e^{delta(u)}$ (delta(u)) = d~~

$e^{delta(u)}$ [delta(u) → difference in state values]

u = (a x no. of gaps) + (b x no. of misconnected pieces) + (c x sum of gap sizes)

(a, b, c) adjusted values by learning.

Ans 5: (a) The TSP can be solved by using the MST heuristic. When a partial tour has already been constructed the MST heuristic is used for estimating the cost of completing a tour.

Hill Climbing Approach:

The Hill climbing approach is a loop which repeatedly moves in the direction of increasing value and terminates when it reaches the peak value. The Hill climbing method returns to a state that is locally maximized.

function Hillclimbing returns locally maximized state
Input: A problem to generate the minimum cost
Local variables: current, a_node

Neighbor a_node
current ← make node [initial state] (problem)
loop do
Neighbor ← highest valued successor
if value [Neighbor] ≤ value [current] then
return state [current]
current ← Neighbor

A* algorithm A* gives a solution to travelling sales man problem as the least cost in which the person should travel. The evaluation function is $f(n) = g(n) + h(n)$, where $f(n)$ is the cheapest solution estimated cost.

	Hill Climbing Method	A* Algorithm
1.	The method repeatedly moves in the direction of increasing cost.	The evaluation function is the cheapest solution estimated cost.
2.	The algorithm returns when it reaches a locally maximized state	The algorithm returns to a state with less cost.
3.	The next state is the node with the highest valued successor	It moves to the next state by adding the cost to the heuristic value.

5 (b):

	Genetic Algorithm	A* Algorithm
1.	randomly selects the individuals from the population based on the fitness and generates a single child state by combining two parent states	A* algorithm gives a solution to TSP as the least cost in which a person could travel
2.	returns the best individual in the population	returns the state with less cost
3.	moves to the next state based on fitness	moves to the next step by adding cost to the heuristic

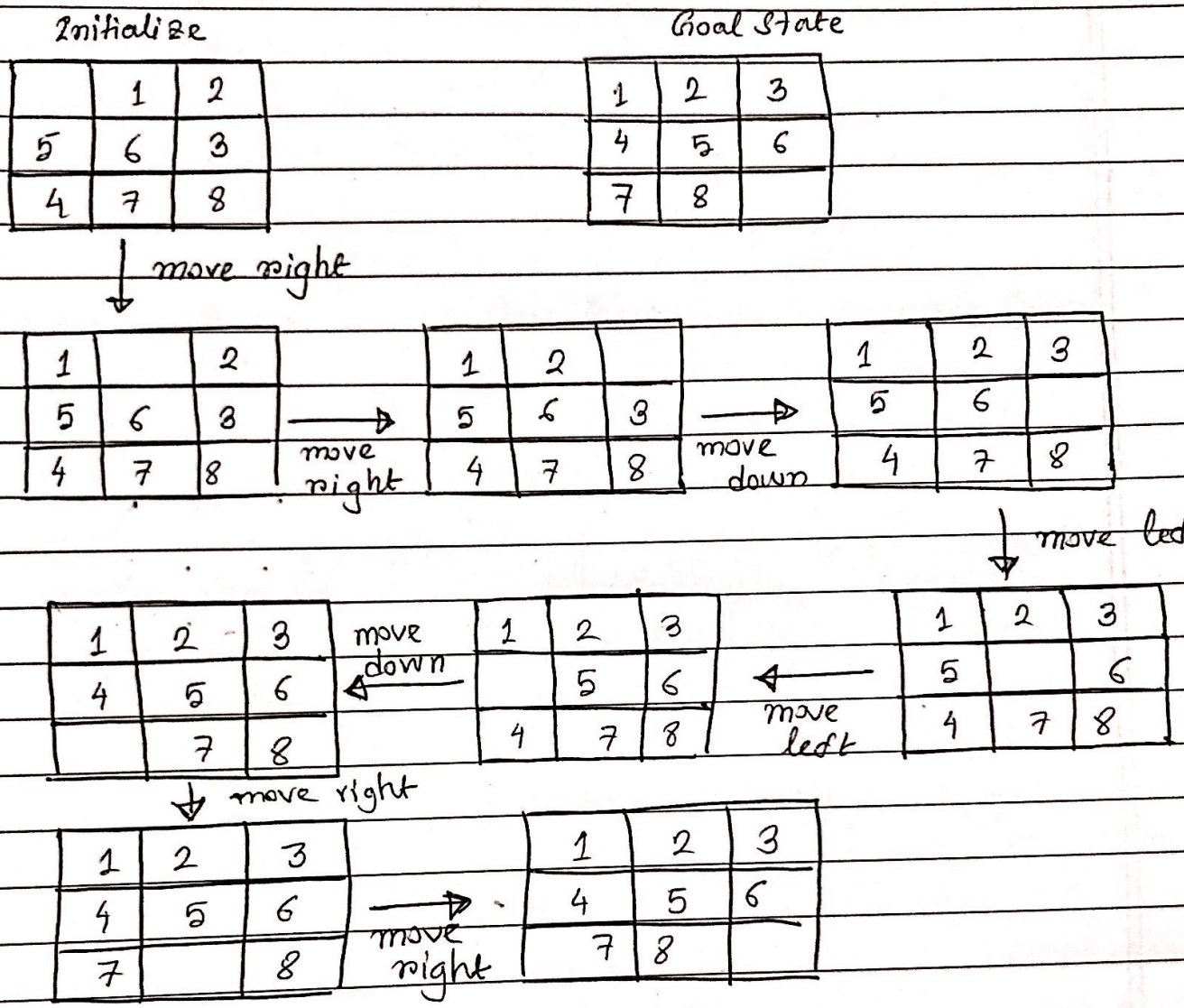
Ans 6: We know that an optimal solution has the lowest path cost among all the solutions.
The optimal solution would be all queens where they're already placed in the correct positions and from there it would only have to move 1 or 2 moves and so on.

	1	2	3	4	5	6	7	8
1					Q			
2							Q	
3	Q							
4			Q					
5								Q
6						Q		
7				Q				
8		Q						

This could be a probable solution for a large number of 8 puzzle and 8 queen instances.

~~For the above state, the number of pairs of queens that are attacking each other, h=17~~

Using Hill climbing to solve the 8-puzzle problem.



Name: Subhi Bhuiyan
ID: 17201116
Section: 03