Assignment - 1

1 a) Initial State - To choose a random piece, assuming it is a straight one

Successor function - Addition of ~~another~~ another piece. For a curved piece, add in either orientation. Forks should be added in both orientation and connected at either hole.

Goal test - Single connected track with no open pegs or holes and no overlapping ~~tracks~~ tracks.

Step cost - one per piece

b) DFS is suitable as all solutions have some depth

c) In order for the problem to be solvable there must be even number of ends tracks, and each fork creates two ends, thus only a fork can rejoin.

d) Maximum possible number is 3 for open pegs.

Assuming each piece is unique

which gives us $(12 + (2\times16) + (2\times2) + (2\times2\times2))$ choices

per peg.

$\therefore$ 32 pieces which leads to 32 depth.

$\therefore$ upper bound $\simeq \frac{16^{32}}{12! * 16! \times 2! \times 2!}$

2a) Each cities can be visited more than once if needed, the summation of repeated edges is not counted

b) It determines the shortest distance, thus meaning a straight line.

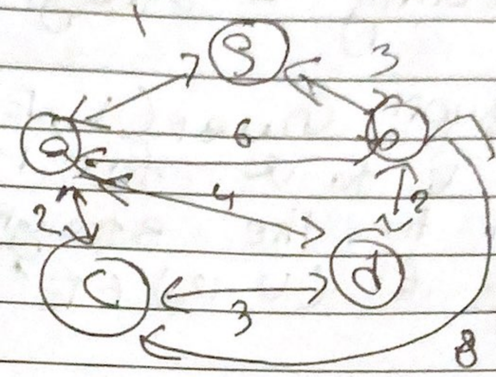
For a given $G_{MST}(V, E)$, the cost from v to u for MST will be equal to the straight line distance only if $(v, u) \in E'$.

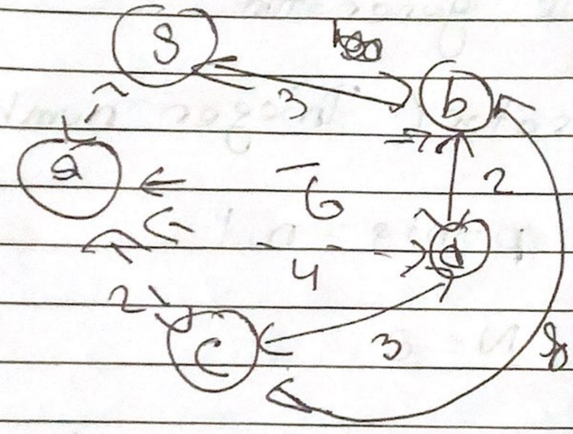
c) A simple generator

```
Generator( Integer numbers) {

    points = nil

    N = 0

    while (N < numbers) {

        (X, Y) (Random (0,1) Random (0,1))
        if ((X, Y) ∉ points ){
            points <- (X,Y)
            N = N+1
        }
    } return points
}
```

d) Any MST algo is accepted

~~A - sta~~ A* algo





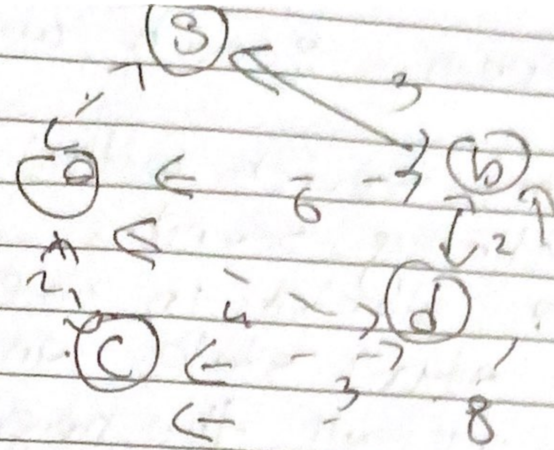
$S \to a$

$F(a) = G(a) + H(a)$

$= 1 + 3 + 2 + 3 = 9$

$S \to b$

$F(b) = 9$



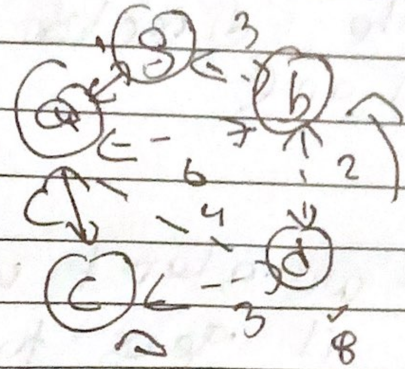
s -> b

F(b) = 9

s -> a -> c -> d

~~s c~~

F(d) = 9



s -> a -> c - d

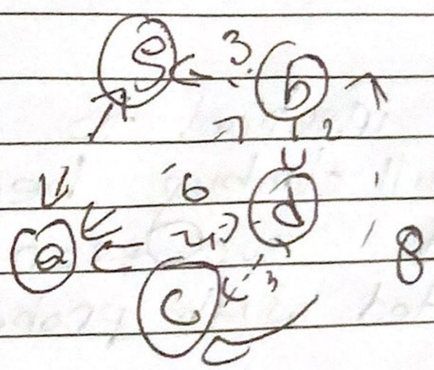
F(d) = 9

s -> b - d -> c

F(c) = 11

s -> b - a

cannot construct

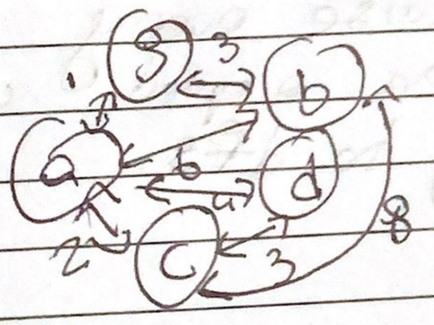


s -> a - c - d - b

F(b) = 8

s -> b -> d - c

F(c) = 11



s -> a -> c - d - b - s

Good

F(b) = G(b) + H(b)

= 1+2+3+3+0 = 11

3) a) Local beam search with k=1

Local beam search with k=1 is hill climbing search. 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 one

b) BFS. If we start with one state, it goes to the other neighbors and so on.

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

d) Random walk because every successor would be accepted with 1, thus leading to dfs.

e) Random walk, if population is 1, then two selected parents will be the same individual, crossover yields an exact copy of the individual, then there is a small chance of mutation.

4) Having 32 pieces, while removing any piece, we can reconnect it anywhere else. For closed loops, moving one joint will make others or all move. To reduce the set of actions, we must ensure resulting connection is ready to be restricted.

By using simulated annealing which samples the next state, evaluates it and takes the next according to $e^{delta(v)}$ (delta (v) := difference in ~~steps~~ state values).

$v = (a \times \text{no. of } \cancel{gps} \text{ gaps}) + (b \times \text{no of misconnected pieces}) + (c \times \text{sum of sizes of gap})$

(a, b, c) adjusted values by learning

5a) Hill climbing approach

Input - generate the minimum cost

local variable : current e-node

Neighbor e-node

current ← make node

loop do

Neighbor ← highest valued successor

If value [neighbor] ≤ value [current]

Then return state [current]

current ← neighbor

A*

cheapest solution
estimated cost

returns ~~to~~ a state
with less cost

moves to the next step
by adding the cost to
heuristic value

hill climbing

repeatedly moves
in the direction of
increasing cost.

returns when
reaching a locally
maximized state

The next state is
the node with the
highest valued successor

b) A*

A* gives a solution to
travelling sales man problem
as the least cost in which
the person would travel

returns the state with
less cost

moves to the next step ~~state~~
by adding cost to heuristic

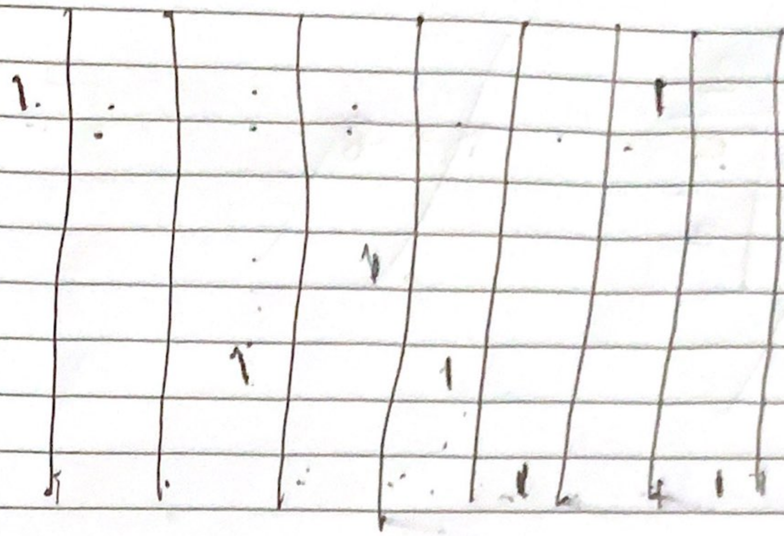
Genetic

randomly selects
the individuals each
at once from the
population ~~base~~ based on
fitness and generates
a single state by
combining two parent states

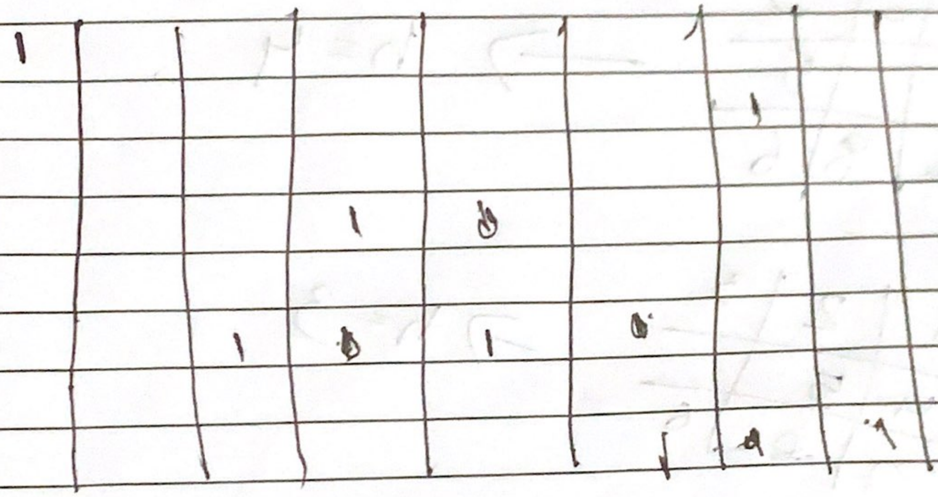
returns the best
individual in the population

moves to next
state based on
fitness

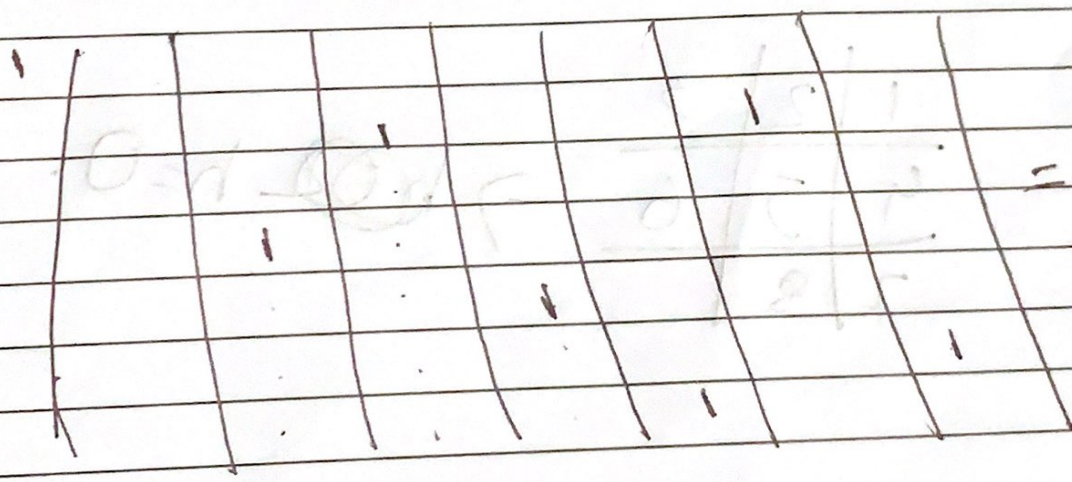
6)



$h = 5$



$h = 4$



$= 1$

n=0

1	2	3
4		5
7	8	6

$\rightarrow h=4$

1	2	3
4	5	6
7	8	

$\rightarrow h=3$

1	2	3
4	5	6
7	8	

$\rightarrow h=0$.