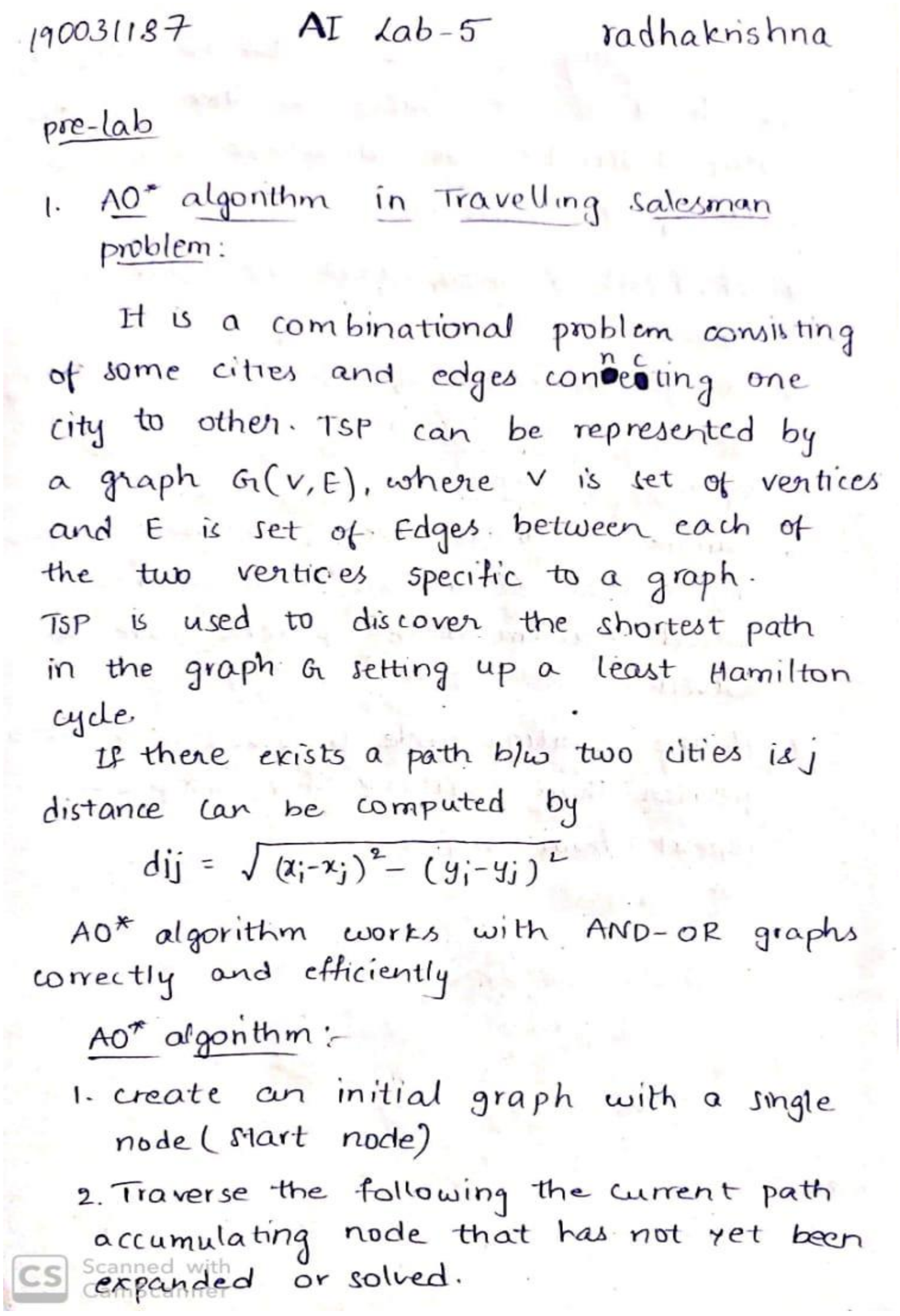
AI Lab-5

PRELAB

1. Describe how AO* algorithm can be used for implementing Travelling Salesman Problem.
   (Travelling Salesman Problem - given a number of cities and the distances between each couple of cities, the aim is to find the smallest possible route that goes to each city exactly once and returns to the origin city i.e. find a least cost Hamiltonian cycle.)



190031187 AI Lab-5 radhakrishna

pre-lab

1. AO* algorithm in Travelling Salesman problem:

It is a combinational problem consisting of some cities and edges connecting one city to other. TSP can be represented by a graph G(V, E), where V is set of vertices and E is set of Edges between each of the two vertices specific to a graph. TSP is used to discover the shortest path in the graph G setting up a least Hamilton cycle.

If there exists a path b/w two cities i, j distance can be computed by

$$dij = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$$

AO* algorithm works with AND-OR graphs correctly and efficiently

AO* algorithm :-

1. create an initial graph with a single node (start node)
2. Traverse the following the current path accumulating node that has not yet been expanded or solved.

3. slect any of these nodes and explore it. If it has no successors then call this value - FUTILITY else calculate $f'(n)$ for each of the successors.

4. If $f'(n) = 0$ then mark the node as solved.

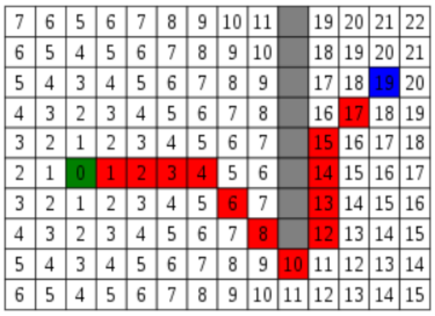
5. change the value of $f'(n)$ for the newly created node to reflect its succersors by back propogation,

6. Whenever possible use of the most promisi -ng routes, If a node is marked as solved then mark the parent node as solved,

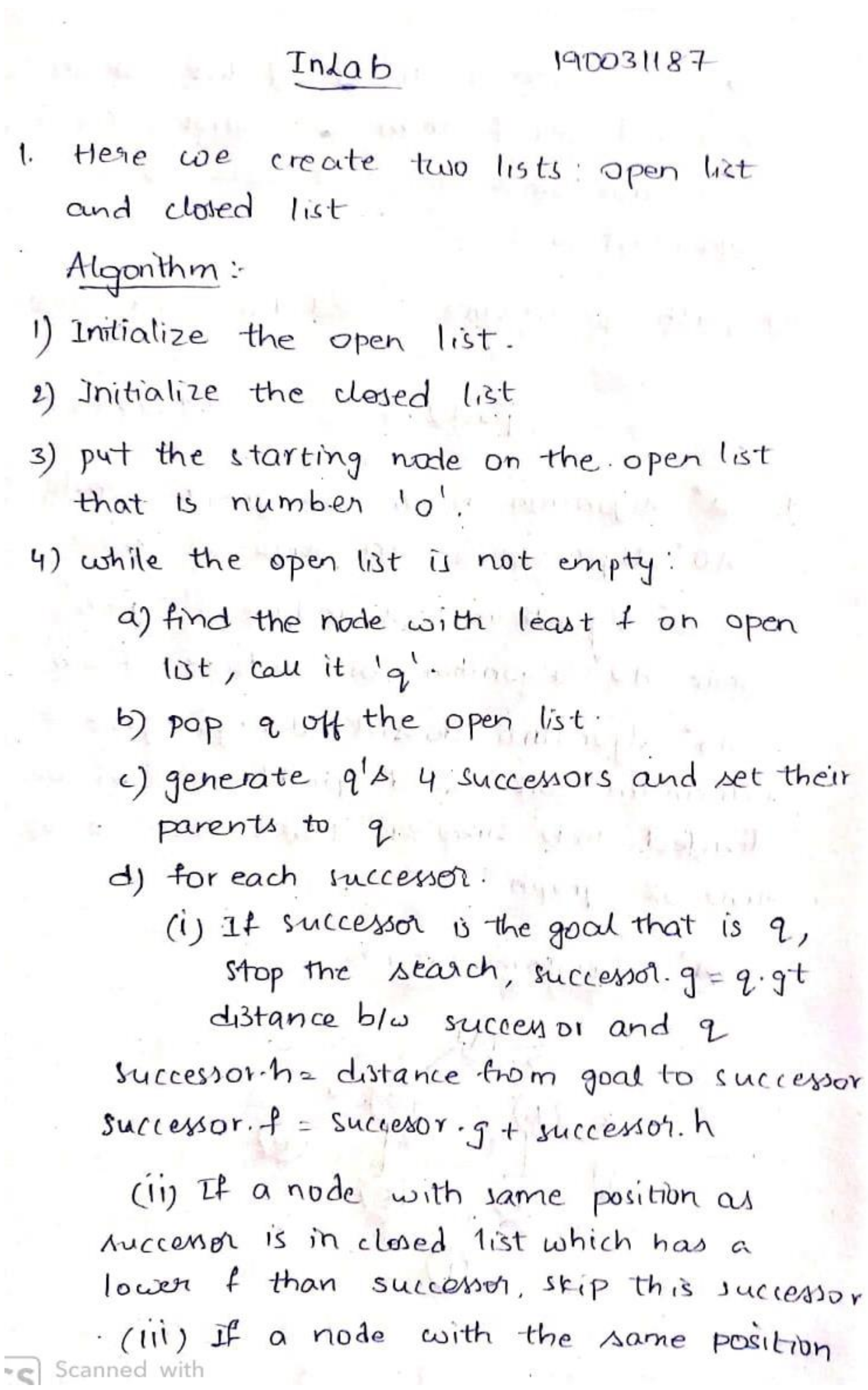
7. If the starting node is solved or value is greater than FUTILITY then stop else repeat from step-2

INLAB

1. Implement A* algorithm to find the shortest path from 0 to 19.



7	6	5	6	7	8	9	10	11	▨	19	20	21	22
6	5	4	5	6	7	8	9	10	▨	18	19	20	21
5	4	3	4	5	6	7	8	9	▨	17	18	19	20
4	3	2	3	4	5	6	7	8	▨	16	17	18	19
3	2	1	2	3	4	5	6	7	▨	15	16	17	18
2	1	0	1	2	3	4	5	6	▨	14	15	16	17
3	2	1	2	3	4	5	6	7	▨	13	14	15	16
4	3	2	3	4	5	6	7	8	▨	12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15



Inlab 190031187

1. Here we create two lists: open list and closed list

Algorithm :-

1) Initialize the open list.

2) Initialize the closed list

3) put the starting node on the open list that is number '0'.

4) while the open list is not empty:

   a) find the node with least f on open list, call it 'q'.

   b) pop q off the open list.

   c) generate q's 4 successors and set their parents to q

   d) for each successor.

      (i) If successor is the goal that is q, stop the search, successor.g = q.g + distance b/w successor and q

   successor.h = distance from goal to successor

   successor.f = successor.g + successor.h

   (ii) If a node with same position as successor is in closed list which has a lower f than successor, skip this successor

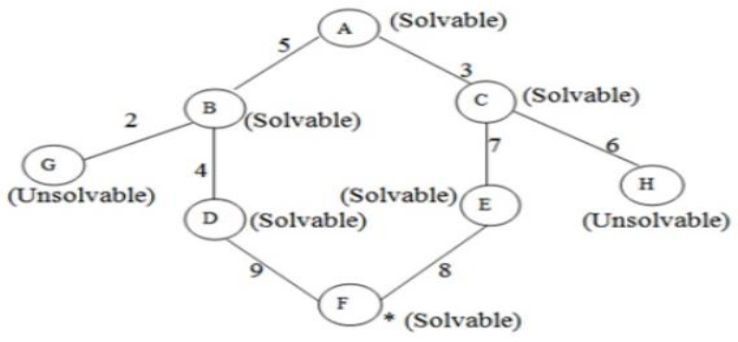
   (iii) If a node with the same position

as successor is in closed list which
has a lower f than successor, skip this
successor otherwise, add note to the
open list end

e) push q on the closed list end(whileloop)

Postlab

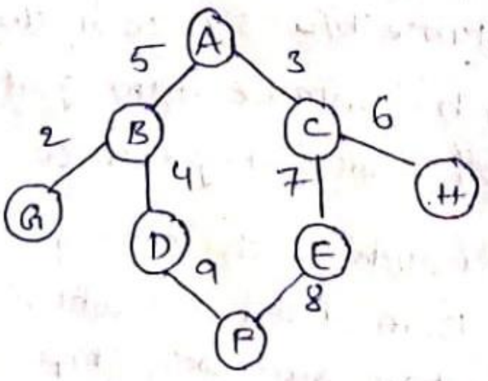
1. For the given graph implement compare A* and AO* algorithm by implementation and check which one gives the optimal solution?



post Lab 190031187

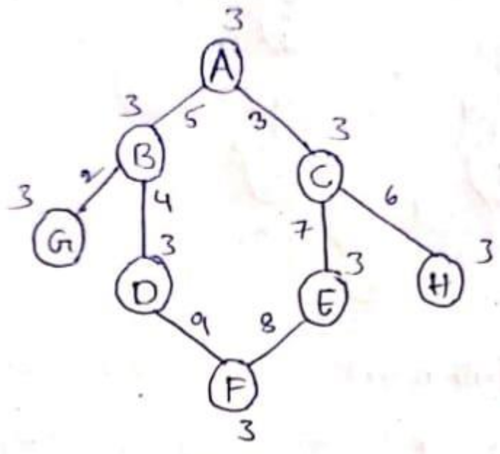
1. A* Algorithm is an OR graph while AO* Algorithm is an AND-OR graph
→ A* algorithm cost include $f' = g' + h'$ while AO* algorithm cost include $f' = h'$
AO* algorithm doesn't always give minimum cost and problems can be divided into simpler tasks because of AND-OR graph

A* algorithm search



→ Numbers written on edges represent ditance b/w nodes

→ Let us assume heuristic value for each node is 3.



In this our start node is A.

From A we can go to A → B or A → C

$f(A \to B) = (3+5) + 3 = 11$

$f(A \to C) = (3+3) + 3 = 9$

Since $f(C) < f(B)$ it decides to go to C

path: A → C

From C we can go to E or H

$f(H) = (3+3+3) + 6+3 = 18$

$f(E) = (3+3+3) + 7+3 = 19$

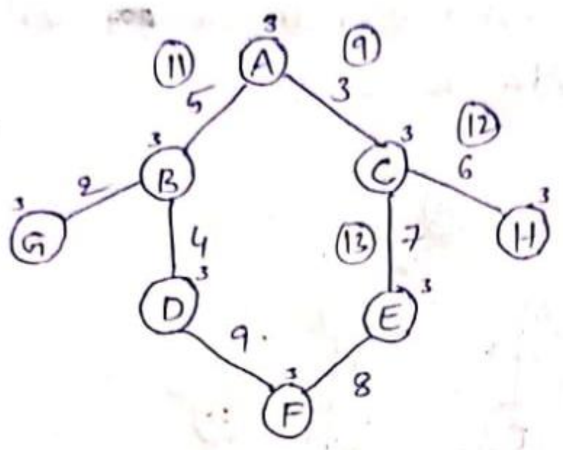
Since $f(E) < f(H)$ it decides to go to H

path: A → C → H

from 'H' there are no nodes at all and path ends here

$\boxed{\text{Final path: } A \to C \to H}$

AO* algorithm search



$f(n) = g(n) + h(n)$

$f(A \to C) = 9$

$f(A \to B) = 11$

path : $A \to C$

$f(C \to H) = 12$

$f(C \to E) = 13$

path : $A \to C \to H$