Page :
Date : / /

Tutorial 1

1 a) 8 Puzzle Problem:

→ Initial State : 9 states with 8 tiles from 1 to 8 & one empty tile randomly placed.

→ Goal State : Tiles from 1 to 8 are arranged in row major order & last is the empty tile.

→ Actions : They include swapping the empty file with either top, left, right or bottom tiles adjacent to it.

→ Path cost : This functions associates a cost with every swap there is to execute in the actions.

→ Transition model : This is any intermediate state after initial state which is not a goal state.

b) Travelling Salesman Problem:

→ Initial State: It is the set of cities representing destinations that the salesman has to visit. They are all unvisited.

→ Transition model: It is the state where few destinations are visited.

→ Actions: Any algorithm we apply, will decide actions like BFS that uses queue data structure and explores nodes level by level.

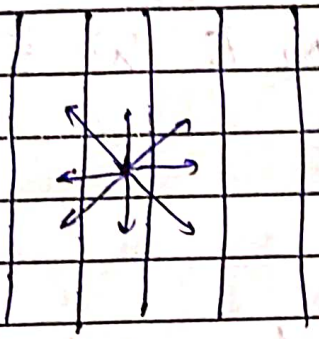
→ Goal state: It is the state where all destinations are visited.

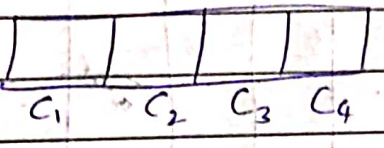
→ Path cost: The distance travelled in total from one node to another sets a cost of the path.

Page :
Date : / /

2. Performance measures are
i) Completeness
ii) Optimality
iii) Time Complexity
iv) Space Complexity

For color map problem, we need 4 colors; C_1, C_2, C_3, C_4





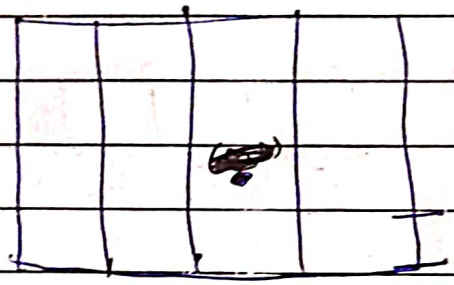
At pixel, b
fill color C_1,
store adjacent pixels in queue
if colored, leave else color with the next color
To choose next color, consider a 3×3 matrix & map positions as given below:

C_2	C_3	C_2
C_0	C_1	C_4
C_2	C_3	C_2

This algorithm ends when queue gets emptied.

It is complete as all the pixels get colored.
It is optimal as pixels are not visited again.
Time complexity is $O(n)$ where n is the number of pixels.
Space complexity is the size of the queue which is $O(n)$.

3.



N E W S

Initial state: robot at coordinates x, y where

$x \in [0, n)$ n is number of columns
$y \in [0, m)$ m is number of rows

Goal state: x, y at $(n-1, m-1)$ to exit.

Actions: Turn N and step forward & loop until a block or exit is found.
If exit, stop else backtrack and turn E, then S, then S.