**Operating Systems**

**Project Report**

**Introduction**:

We have implemented A\* search algorithm using threads to approximate the shortest path in real-life situations, like- in maps, games where there can be many hindrances. Many games and web based maps use this algorithm to find the shortest path very efficiently (approximation). Dijkstra is a special case of A\* Search Algorithm, where h= 0 for all nodes / cells.

**Objective**:

To get the shortest path from the algorithm quickly using parallelism techniques (Pthreads).

**Methodology:**

In A\* search Algorithm we visit all 8 neighbors (East, West, South, North, North East, North West, South East, and South West) of a cell one by one but by using Pthreads method we will visit them all at once using 8 threads that will run in parallel by which we can get the shortest path much quicker than the algorithm without using threads.

**Explanation of algorithm:**

Consider a square grid having many obstacles and we are given a starting cell and a target cell. We want to reach the target cell (if possible) from the starting cell as quickly as possible. Here A\* Search Algorithm comes to the rescue. What A\* Search Algorithm does is that at each step it picks the node according to a value-‘**f**’ which is a parameter equal to the sum of two other parameters – ‘**g**’ and ‘**h**’. At each step it picks the node/cell having the lowest ‘**f**’, and process that node/cell.

We define ‘**g**’ and ‘**h**’ as simply as possible below

**g** = the movement cost to move from the starting point to a given square on the grid, following the path generated to get there.

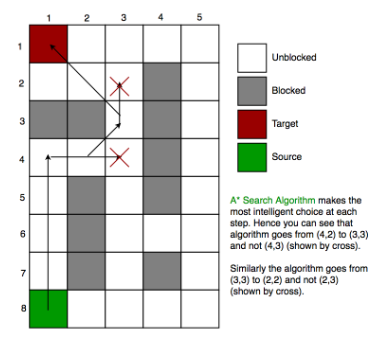
**h** = the estimated movement cost to move from that given square on the grid to the final destination. This is often referred to as the heuristic, which is nothing but a kind of smart guess. We really don’t know the actual distance until we find the path, because all sorts of things can be in the way (walls, water, etc.).

**Method Used to Calculate Heuristic Value:**

We have used Euclidean distance method to calculate the heuristic value because it works when we are allowed to move in any direction



**This is how A\* Search Algorithm works**



**Algorithm of A\* Search Algorithm:**

1. Initialize the open list

2. Initialize the closed list

put the starting node on the open

list (you can leave its **f** at zero)

3. while the open list is not empty

a) find the node with the least **f** on

the open list, call it "q"

b) pop q off the open list

c) generate q's 8 successors and set their

parents to q

d) for each successor

i) if successor is the goal, stop search

ii) else, compute both **g** and **h** for successor

successor.**g** = q.**g** + distance between

successor and q

successor.**h** = distance from goal to

successor

successor.**f** = successor.**g** + successor.**h**

iii) if a node with the same position as

successor is in the OPEN list which has a

lower **f** than successor, skip this successor

iV) if a node with the same position as

successor is in the CLOSED list which has

a lower **f** than successor, skip this successor

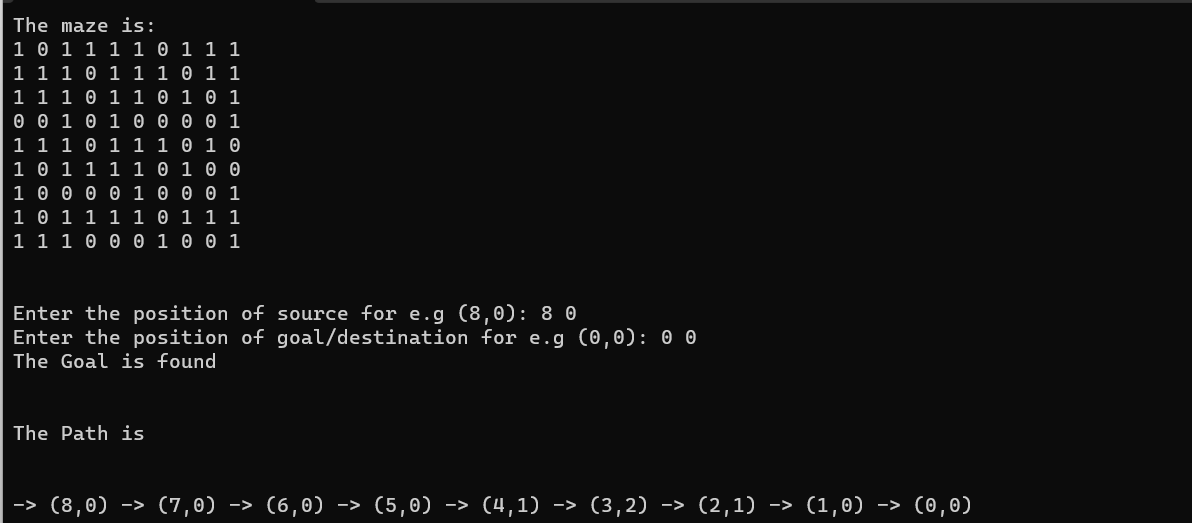
otherwise, add the node to the open list

end (for loop)

e) push q on the closed list

end (while loop)

**Output:**



**Topics Used:**

1. Pthreads to achieve parallelism
2. Mutex Locks for mutual exclusion

**Platform:**

Dev C++

**Language Used:**

C++

**Group Members:**

1. Shiv [K213434]
2. Rohit Raja [K214616]
3. Lov Kumar [K214821]