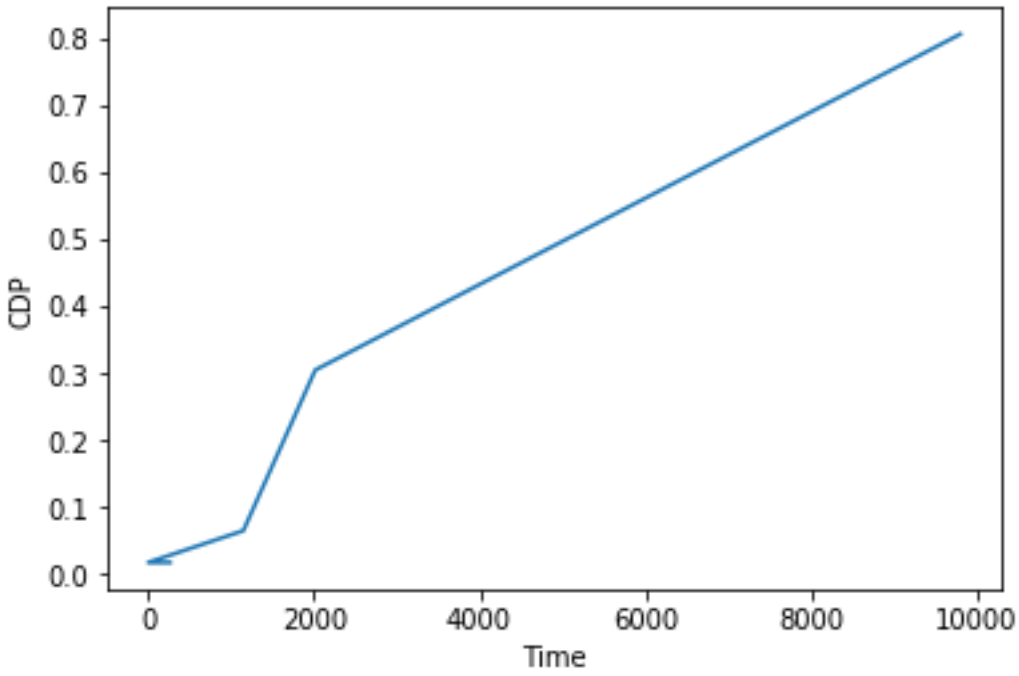
In my project, I generated the list of corner points from the heat map.

I divided the heat map into 4 parts.

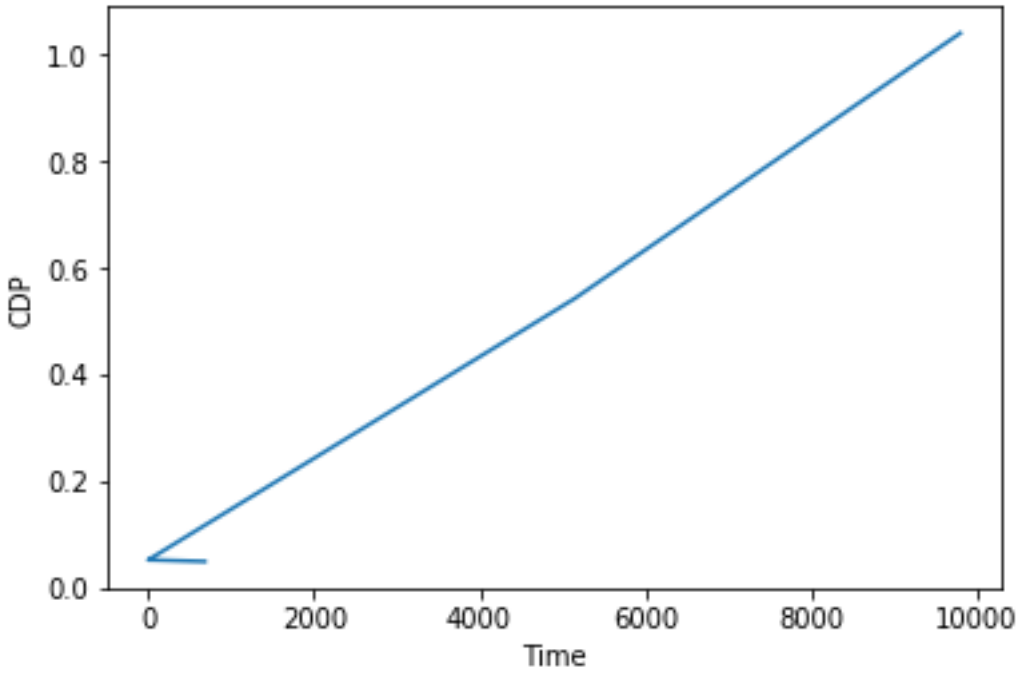
From that, I got the four corner points of the heat map. But , if we traverse taking those points, then it does not result in an optimal solution as it traverses 80% of the heat map so I have suggested a different algorithm which takes into consideration these facts.

1. For the first path, I considered the value of i to be 0 and the value of j from 0 to 50. If there was no non-zero probability point, I incremented the value of i by 1. I stored all the value of non-zero probability points in a list. Then, I considered the first, second, last and second-last points from the list and formed a rectangle. Then, I traversed the points in this rectangle and calculated the cumulative probability of this path.
2. For the second path, I considered the value of i to be 99 and the value of j from 0 to 50. If there was no non-zero probability point, I decremented the value of i by 1. I stored all the value of non-zero probability points in a list. Then, I considered the first, second, last and second-last points from the list and formed a rectangle. Then, I traversed the points in this rectangle and calculated the cumulative probability of this path.
3. For the third path, I considered the value of j to be 0 and the value of i from 50 to 100. If there was no non-zero probability point, I incremented the value of j by 1. I stored all the value of non-zero probability points in a list. Then, I considered the first, second, last and second-last points from the list and formed a rectangle. Then, I traversed the points in this rectangle and calculated the cumulative probability of this path.
4. For the fourth path, I considered the value of j to be 99 and the value of i from 50 to 100. If there was no non-zero probability point, I decremented the value of j by 1. I stored all the value of non-zero probability points in a list. Then, I considered the first, second, last and second-last points from the list and formed a rectangle. Then, I traversed the points in this rectangle and calculated the cumulative probability of this path.
5. In the end, I traversed all the points which were not there in these four lists and calculated the cumulative probability of the path.

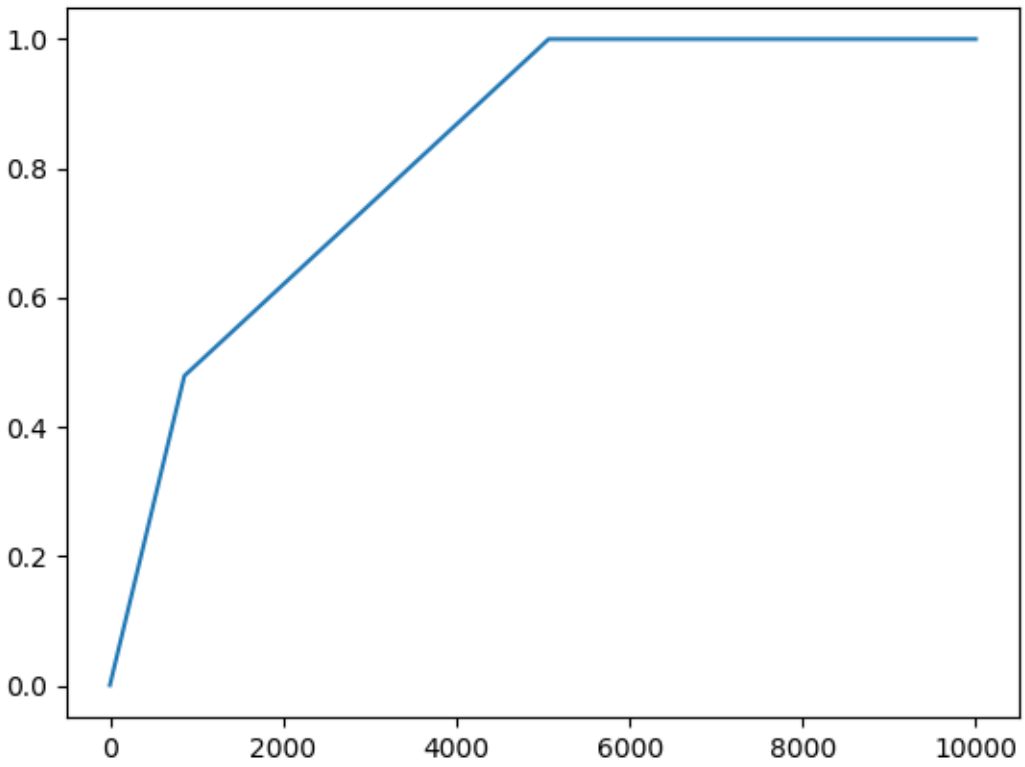
   Following these steps, I calculated the CDP vs time graph for both the heatmaps which were provided to us and compared it with the base algorithm where I traversed all the non-zero probability points first and then all the points with zero probabilities. The results with the graphs have been discussed below.



This is the CDP vs Time graph for the first heat map.



This is the CDP vs Time graph for the second heat map.



This is the CDP vs time graph for the reference algorithm and then all the nodes with non-zero probabilities.

Comparing the graphs, (CDP vs Time)

For the first heat map, we can see that the performance graph of my proposed algorithm reaches 0.3 CDP in 2000 seconds and then increases linearly.

For the second heat map, we can see that the performance graph of CDP vs time increases linearly and reaches 1 in 9000 seconds.

Comparing it with the base algorithm, we can see that the graph of CDP vs time reaches 1 in less than 6000 seconds but the reason behind that it that it takes all the high probability points into consideration first and then traverses the zero probability nodes. So, that is why it reaches optimal solution path compared to my algorithm but in practicality, it results in a lot of space complexity as it traverses the entire heat map.

Comparing the efficiency of my algorithm and the base algorithm, we must first take into account that the efficiency of our algorithm is calculated by the running time of the algorithm. Obviously, as the base

algorithm traverses all the high-probability points first, it is bound to have a better running time compared to my algorithm as its CDP converges to 1 earlier but if we take the path complexity of the algorithm in to consideration, my algorithm may better the base algorithm as my algorithm divides the heat map into four parts first and traverses each quarter simultaneously rather than traversing each node individually.

	BASE ALGORITHM	**PROPOSED ALGORITHM (HEAT MAP 1)**	**PROPOSED ALGORITHM (HEAT MAP 2)**
RUNNING TIME	**6000 seconds**	**9500 seconds**	**9000 seconds**
PATH COST	**MAX**	**MEDIUM**	**MEDIUM**

EFFICIENCY	**BASE ALGORITHM (HEAT MAP 1)**	**BASE ALGORITHM (HEAT MAP 2)**	**PROPOSED ALGORITHM (HEAT MAP 1)**	**PROPOSED ALGORITHM (HEAT MAP 2)**
T=100s	**99.8**	**99.6**	**80.96**	**80.92**
T=200s	**82.92**	**80.43**	**39.58**	**68.96**
T=300s	**84.57**	**80.34**	**32.84**	**71.94**
T=600s	**59.37**	**62.54**	**20.03**	**43.98**
T=900s	**79.16**	**76.74**	**16.66**	**36.12**

Here, we can see that when we compare the efficiency of both the algorithms, we can notice that the efficiency of my algorithm for both the heatmaps is high since I divide the entire search space in to 4 parts and start from one of the non-zero probability nodes. Then, as and when the search space increases, we can notice that the efficiency of the algorithm goes on to decrease.

So, in the end, we can conclude that the efficiency of this search technique can be found by the performance matrices by which we choose to measure the efficiency of our algorithms. While considering the efficiency, we must also take the physical state of the victim which is the main motivation for our search algorithm.

Traversing all the nodes as in the base algorithm, we may cover all the non-zero probability points with a lesser time complexity, but in my proposed algorithm, as we divide the entire search space into four halves, there is always a probability of finding the missing person early. That was not the case in these heat maps which were given to us, but there is always a high probability of finding the missing person earlier than the base algorithm given another heat map.