# CS 225 Final report

<u>Summary of our final datasets:</u>

Our datasets are two dimensional grids with walls. Each grid has a starting point and an ending point. The goal of our algorithm is to find the most optimal path from the starting to the ending point. We have five separate datasets of varying magnitude.

| Size of grid | Number of grids |
|---|---|
| Small (5x5) to (10x10) | 73 |
| Medium (10x10) to (50x50) | 24 |
| Large (50x50) to (200x200) | 12 |
| Super (200x200) to (2000x2000) | 122 |
| Extreme (2000x2000) to (20000x20000) | 5 |

<u>The output and correctness of the algorithm</u>

As for the A star search algorithm we implement, we decided to use two main functions.

```
void converttogrid(std::string infile);
```

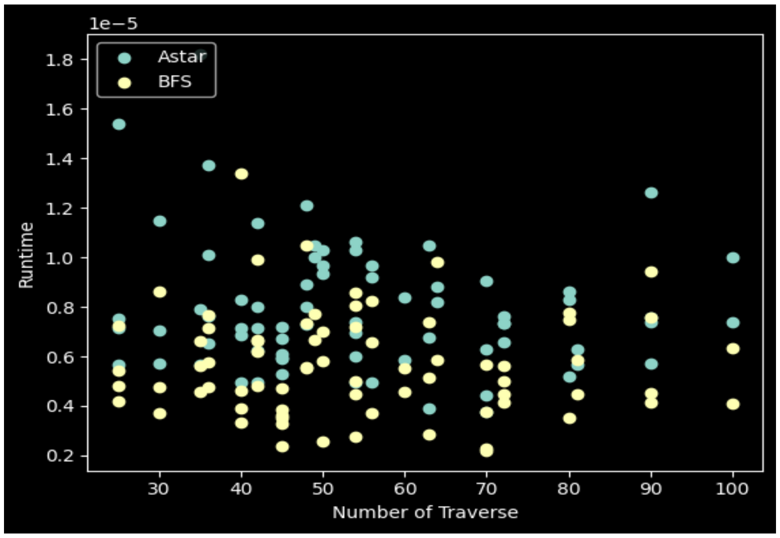
This is the first function we used to convert the datasets to the grid that we can traverse. It takes in a file name as the parameter and produces a 2D vector grid. The value at (x, y) coordinate can be accessed by grid[x][y]. We tested this function by comparing the result grid produced by the function with the file it reads. We make sure the value at any position (x,y) in the grid is the same as that in the file. This proves the correctness of the function. We also tested if the function recorded the start and end point correctly by checking the value at the start (x,y) and end (x,y). All in all, our test cases test if the function can successfully get all the information in the file, which is sufficient to prove the correctness of the function. In the end, the function can pass all the test cases, which shows that our function is correct.

```
vector<pair<int, int>> aStar(pair<int, int> start, pair<int, int> target,
vector<vector<int>> grid, Heuristic h);
```

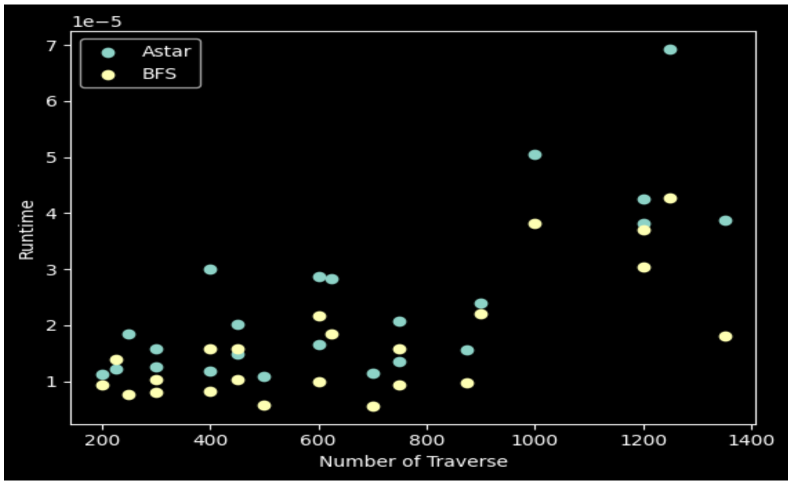
This is the second function we implemented, which is the A star search main function. It takes in the coordinates of the starting and end points, the grid we get from the file, and the heuristic function as parameters. It will output a path from the starting point to the end point. The size of the output vector - 1 should be the shortest path from the starting point to the end point. The reason for minus one is because we include the starting point and end point at the same time. In order to test if the function can successfully find the shortest path. We implement a BFS to find the shortest path at the same time. Then, we compare the length of the path they find. This will work since BFS is guaranteed to find the shortest path. We test the function using all the dataset we have from size 5-10 to 2000-20000. This can ensure the A star searching function can find the shortest path. Also, we generate a runtime diagram to see if the A star function is faster than BFS. The time diagram will be shown in the next section. In general, our test cases test if the A star search function can find the shortest path and if it is faster than BFS. In the end, the A star function can pass all the dataset, which shows the correctness of the function.

<u>Benchmark of the algorithm and a proposed Big O</u>

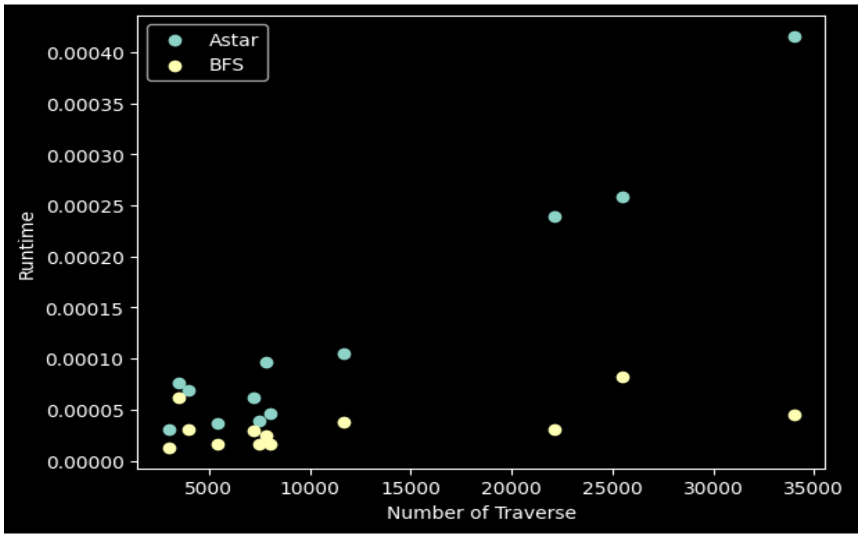
We tested our A star Search Algorithm using datasets with different sizes from 5-10, 10-50, 50-200, 200-2000, and 2000-20000. We also generated the runtime diagram for each group of datasets. The below should be the runtime for datasets with size 5-10. As we can see from the diagram, the runtime of A star and BFS is very close. BFS is slightly faster than A star, only a few milliseconds.



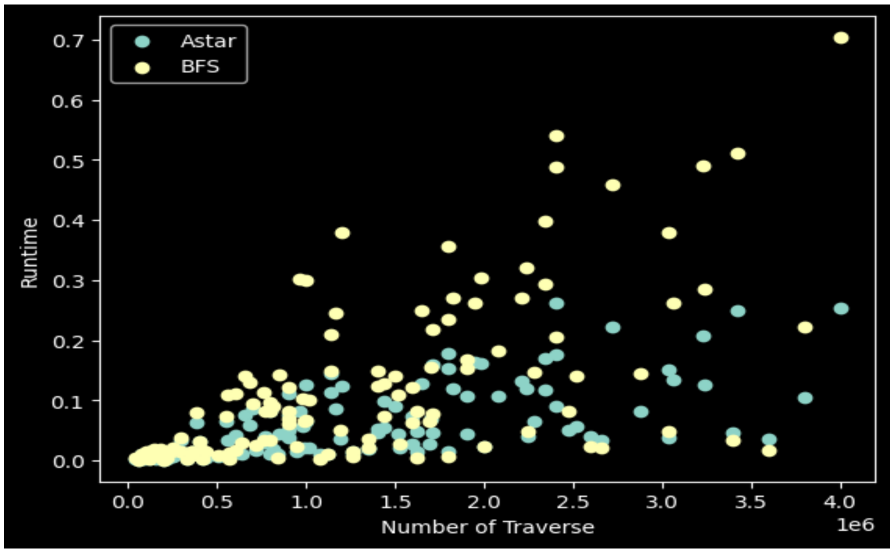
Here is the runtime diagram for the datasets with size 10-50. As you can see from the time diagram, the runtime between BFS and A star is still very close and BFS is slightly faster than A star.



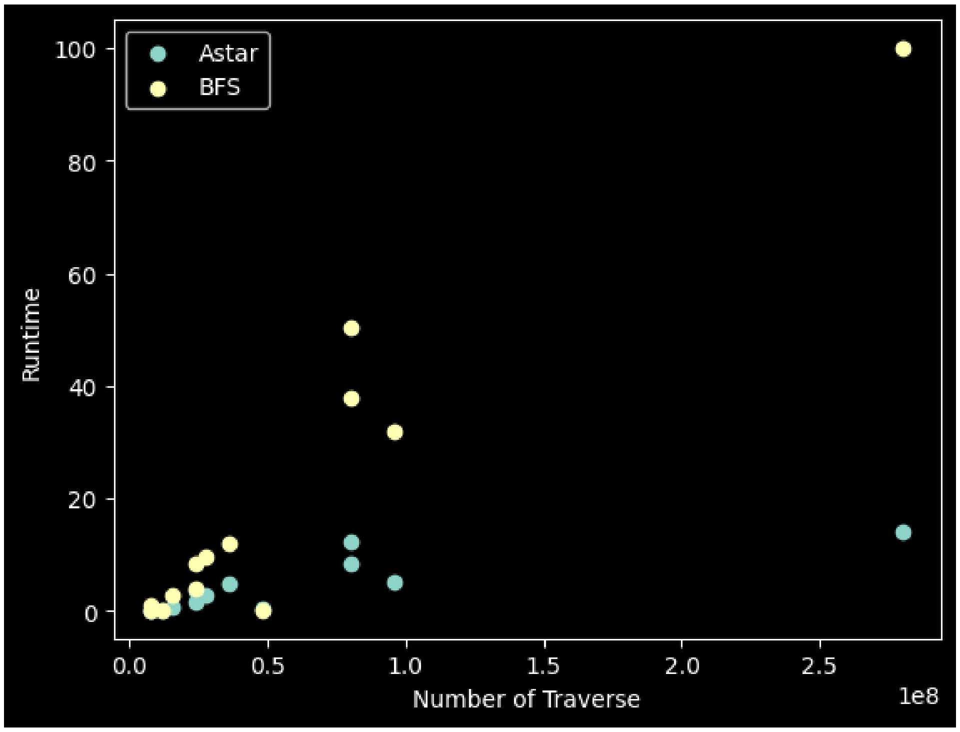
Then, we test on the dataset with size from 50 to 200 with max number of traverses to be 35000. In this case, BFS is still faster than A star. Therefore, we can conclude that for relatively small datasets, the BFS is slightly faster than A star. But BFS is only faster than A star for a few milliseconds, which will not be cared a lot in real applications.



After that, we test the dataset with sizes from 200 to 2000, which can be considered as relatively large datasets. As you can see from the time diagram, at this time, A star is generally faster than BFS. For the 2000 * 2000 dataset, the runtime of A star is twice less than BFS. This shows that for large datasets, the A star works better than BFS, which matches and assumptions and ideas of A star.



Finally, we test on the datasets with size from 2000-20000, which is too large to upload to GitHub. That is why we upload the datasets as several zip files. As you can see from the graph, when the dataset is large enough, A star is much faster than BFS.



Based on all the runtime diagrams, we found that as the size of the grid or graph increases, the A star will be much better than BFS. This shows that our implementation is correct and time efficient.

Besides, how about the big O for A star and BFS. For the grid with size M*N, in the worst case, BFS needs to traverse all the points in the grid, which is O(M*N). As for A star, in the worst case, it still needs to traverse all the point, which is also O(M*N). However, due to the idea of A star and the use of the heuristic function, in large dataset, A star usually does not traverse that many points and traverses much less points than BFS.