Sorting and Searching Assignment 2

Yacob Ben Youb en Youssef Ben Youb
500672040 - 500775494

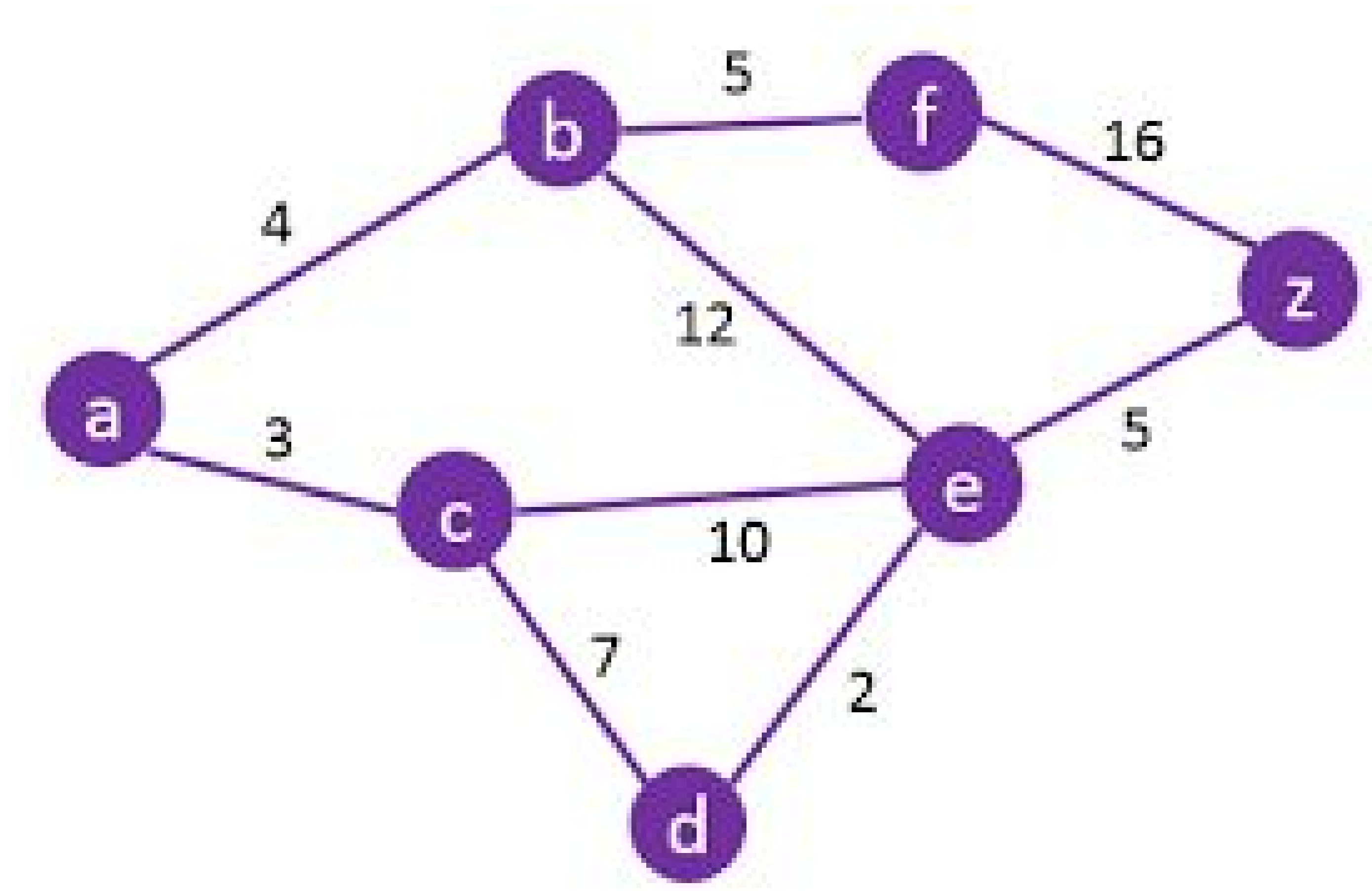


Table of contents:

Introduction

This report will research the differences between 2 algorithms and determining which is the best for which use. The algorithms being the Dijkstra algorithm and the Floyd-Warshall algorithm. We will be comparing the results and efficiency of these 2 different algorithms in different scenarios.

Classes

This is a description of the classes that have been added or modified in the project.

TileworldUtil

This class contains methods to read from images and turn the data from these images into a readable format for the other classes so that they can research a shortest path from one point to another in these images. It also saves the paths found by the other classes into a directory of choice. The directory for the saving of images needs to be changed for every individual computer, otherwise the application will fail to run.

Dijkstra

This is one of the classes that contains an algorithm for finding the shortest path from one point to another point. It makes use of the images that have been made readable by the TileworldUtil class.

FloydWarshall

This is another one of the classes that contains an algorithm for finding the shortest path from one point to another point. It also makes use of the images that have been made readable by the TileworldUtil class.

TestResult

This class allows the results of the Dijkstra and Floyd Warshall algorithms to be saved in objects. It saves the bitmap value used, the amount of tiles researched, the length of the path found and the cost of that path.

ResultPrinter

This class prints objects from the TestResult class that have been saved in arraylists in an orderly fashion. First it prints all the bitmap values, then all the researched tiles, then the length and then the cost. This allows for easy copy-pasting information into excel files.

Methods()

This chapter will cover Methods that have been implemented which were not part of the original project.

IncrementCounter(int)

This method is used in the Dijkstra and FloydWarshall classes.

This method requires an arraylist to be defined in the class. The arraylist used in this method is called counter.

```
public void incrementCounter(int countInt){
  if (!counter.contains(countInt)){
     counter.add(countInt);
  }
}
```

In the images every vertex is defined by an int. This int signifies their location and is kind of like an ID. Seeing as every vertex has its own ID it can be used to keep track of how many have been looked at. By using this method it is possible to add the ID of the vertex to an arraylist if it is not already there.

This method is called in the for loops from the Dijkstra and FloydWarshall classes to keep track of the researched nodes.

PrintStringList(ArrayList<TestResult>)

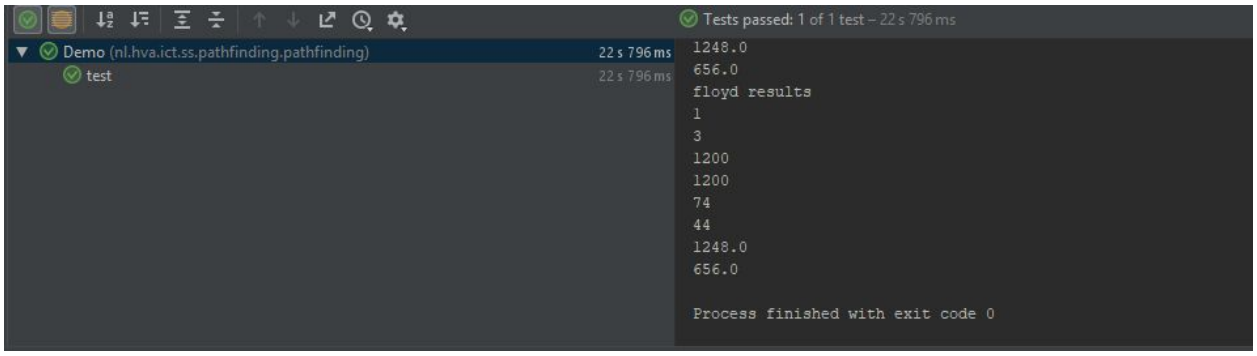
This is the only method in the ResultPrinter class. It prints every bitmap value for every TestResult object from an arraylist, followed by the amount of researched tiles, then the length and finally the cost.

It is used in the Demo test

```
public  void printStringList(ArrayList<TestResult> resultList) {
  for (TestResult test : resultList) {
     System.out.println(test.getBitmapName());
  }
  for (TestResult test : resultList) {
     System.out.println(test.getTiles());
  }
  for (TestResult test : resultList) {
     System.out.println(test.getLength());
  }
  for (TestResult test : resultList) {
     System.out.println(test.getCost());
  }
}
```

Output

This chapter will show a picture of an output from the application. This proves that the application works and gives an idea as to what the application looks like.


Tests passed: 1 of 1 test – 22 s 796 ms
Demo (nl.hva.ict.ss.pathfinding.pathfinding) 22 s 796 ms
test 22 s 796 ms

```
1248.0
656.0
floyd results
1
3
1200
1200
74
44
1248.0
656.0

Process finished with exit code 0
```

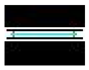
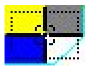
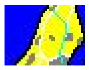
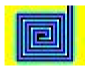
Results and conclusions.

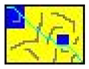
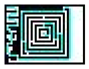
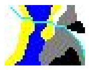
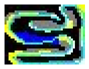
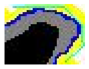
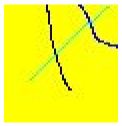
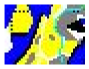
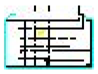
In this chapter the results from the tests are shown in tables and graphs and a conclusion is drawn from these results to determine what algorithm is best to use. Besides the images that were provided along with the project a few extra images were added.

There have been images in which there was not a possible solution. These did not cause an error, however, they have caused for empty spaces in the result tables and graphs. Below are the results of images in which it was possible to find a path for the algorithms.

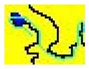
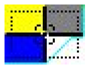
Result Tables

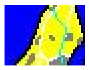
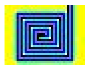
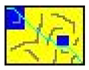
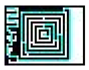
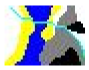
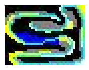
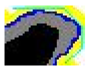
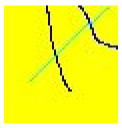
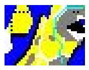
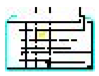
Dijkstra:

Bitmap	Researched tiles	Length	Dijkstra	Solution
1	1060	74	1248	
3	1194	44	656	
4	960	108	1588	
5	203	32	348	
7	1015	54	982	
8	676	27	380	
9	1056	79	910	
11	1159	52	600	
13	1037	224	2292	
14	1173	22	314	
15	1199	34	376	

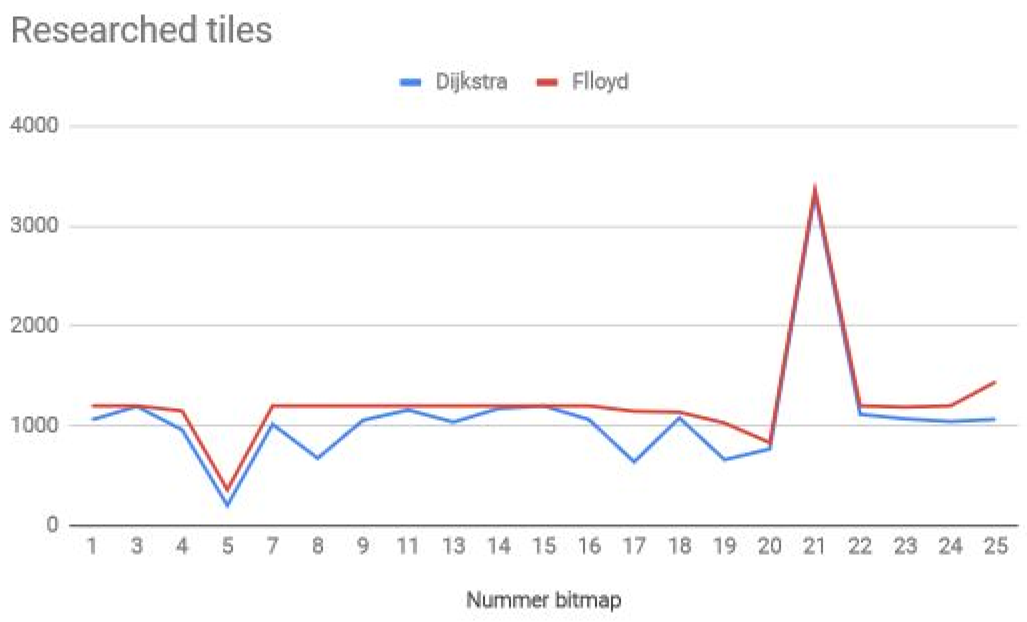
16	1063	38	760	
17	639	178	1928	
18	1079	41	690	
19	661	187	2486	
20	768	53	792	
21	3315	39	762	
22	1115	18	268	
23	1070	35	604	
24	1043	45	684	
25	1065	99	1028	

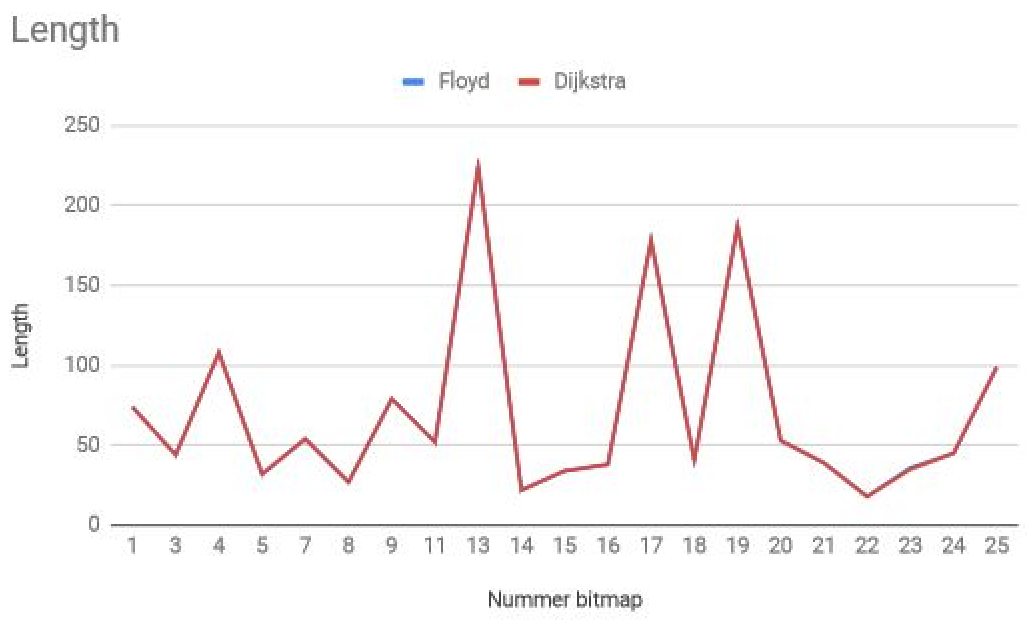
Floyd:

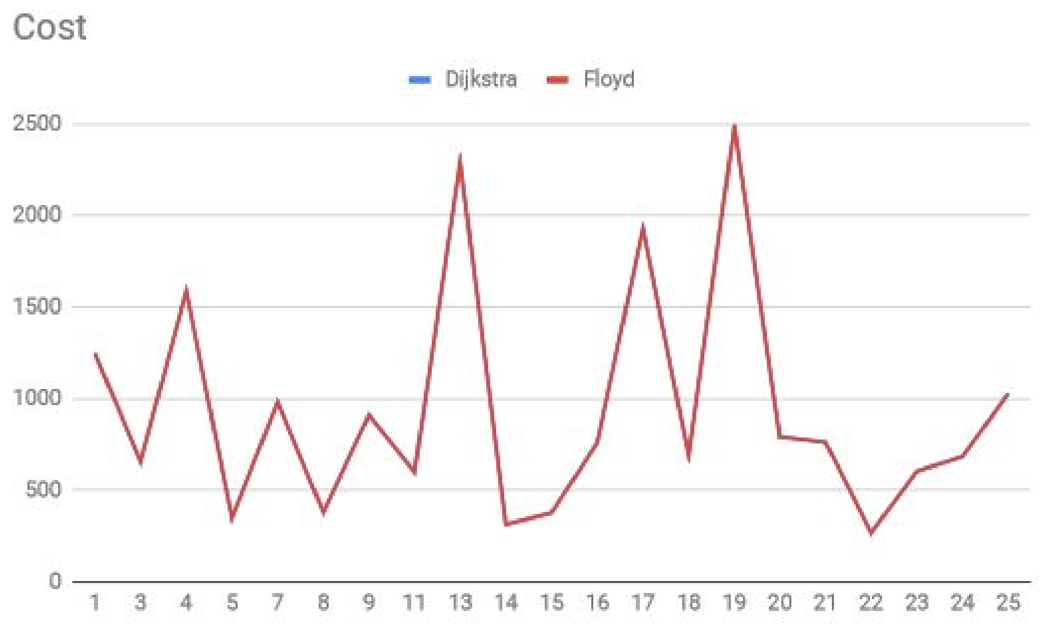
Bitmap	Researched tiles	Length	Cost	Solution
1	1200	74	1248	
3	1200	44	656	
4	1149	108	1588	
5	360	32	348	
7	1200	54	982	

8	1200	27	380	
9	1200	79	910	
11	1200	52	600	
13	1200	224	2292	
14	1200	22	314	
15	1200	34	376	
16	1200	38	760	
17	1146	178	1928	
18	1137	41	690	
19	1028	187	2486	
20	832	53	792	
21	3363	39	762	
22	1200	18	268	
23	1188	36	604	
24	1200	45	684	
25	1440	99	1028	

Result Graphs







When it comes to finding the shortest path, both Dijkstra and Floyd found paths that are of equal length and cost. In the graphs one of the lines is not visible because it overlaps with the other one. Often the only difference between the results in these algorithms is at what moment the line makes a turn in the path. Dijkstra makes turns as late as possible, while Floyd chooses to make them as early as possible.

But there some differences between them. In our tests the most relevant was the amount of researched tiles. Floyd either has the same amount of tiles researched as dijkstra or more. This was also noticeable in the runtime: Floyd took a significant amount of time longer than Dijkstra with certain images. Not only does Floyd research more tiles, it also performs a more checks on the tiles it checks than Dijkstra.

As both algorithms found the same lengths and costs for the shortest path for all the images, it makes the Dijkstra algorithm seem very reliable while also taking less time. In our tests Dijkstra came out as the clear winner. One case in which Floyd has a clear advantage over Dijkstra, is when introducing variables with a negative cost. Dijkstra is not able to calculate a path with negative costs, while Floyd can easily do this. This does not fall within our test circumstances, because it requires us to change the cost of traversing a tile to negative, but it can be beneficial in other scenarios.

Although the results don't show Floyd to be more accurate, they don’t show it to be less accurate either. It is likely that Floyd is more accurate and these images did not serve well to show the difference. After all Floyd has a longer runtime and researches more or an equal amount of tiles as the Dijkstra algorithm. This means that Floyd probably researches more paths than Dijkstra(or it is poorly optimized).

Conclusion

Judging from the results the Dijkstra algorithm appears to be as reliable as the Floyd algorithm as long as there are no negative costs for paths.. However, the Dijkstra algorithm has proven itself to be quite reliable by finding the shortest path in 25 different images.

Concluding: It seems that Dijkstra is generally the better algorithm for use based on the results. If it is very crucial to truly find the shortest path with a hundred percent certainty in every case, it might be worth it to use the Floyd algorithm at the cost of longer calculation time.