Ai Lab 1 Report

1)Methods

```java
public void run() {
    queue.add(source);
    while(!queue.isEmpty()) {
        Node current = queue.poll();
        explored.add(current);
        // we have found the destination node
        if(current == destination)
            continue;
        // consider the adjacent nodes
        for(Edge edge : current.getAdjacencyList()) {
            Node child = edge.getTarget();
            double cost = edge.getWeight();
            double tempG = current.getG() + cost;
            double tempF = tempG + heuristic(current, destination);
            // if we have not considered the child and the f(x) is lower
            if(explored.contains(child) && tempF <= child.getF()) {
                continue;
            }
            // else if we have not visited OR the f(x) score is higher
            else if(!queue.contains(child) || tempF > child.getF()) {
                // this is how we can track the shortest path (predecessor)
                child.setParent(current);
                child.setG(tempG);
                child.setF(tempF);
                // we have it in the queue but now we have a higher value
                // instead of update - we remove and reinsert again
                if(queue.contains(child))
                    queue.remove(child);
                queue.add(child);
            }
```

*I implemented a graph for this task.
*So basically this is my Algorithm class.It checkes for the longest path.
*As we can see instead of break the loop (Following picture)

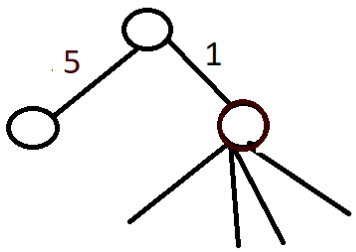
```java
// we have found the destination node
if(current == destination)
    continue;
```

We continue.And the reason is
I implemented start and finish nodes which are basically imaginary. Breaking the loop here is not very smart and the reason is all -1 costs have this end node.Whenever(God knows) it finds the first value(End node) the algorithm stops.So we continue for checking all the trees.

2)Heuristics

I did not use any heuristics and Im gonna explain why I did not.

Lets consider this case;



$$f(x)=g(x)+\underline{h(x)}$$

A.x

x=NumberOfChild

To keep it simple lets say A=1 so
Whenever leftnode weights equal to right weight+x our A* algorithms turn into Dijkstra.And to be honest I dont think this heuristics change too many things because of these things the value of h(x) is zero for all cases.

3)Experiments

Implementation and heuristic is constant

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_small.dag
Total distance : 0.9093387992810442

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_small_sparse.dag
Total distance : 1.5383073446920403

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_medium.dag
Total distance : 17.29052966569518

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_medium_sparse.dag
Total distance : 15.384494209493596

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_large.dag
Total distance : 29.49139233183735

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_large_sparse.dag
Total distance : 20.619404334254586

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_xlarge.dag
Total distance : 39.979429568194575

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
C:\Users\kaann\IdeaProjects\lastai2\test_xlarge_sparse.dag
Total distance : 29.103757495818282

Process finished with exit code 0
```

4)Conclusion Testing

Well I was thinking there will be some delay on the test_xlarge files.(I have never created that much objects before) Surprisingly it finished instantly.Nothing happened.

5)General Conclusion

To sum it up I created 2 array 1 is for costs and other one is neighbours from input.

```java
//Get weights
for (int i = 0; i <numberofnodes;i++) {
    cost[i]=Double.parseDouble(myReader.nextLine());
}
```

```java
//Get neighboors
for (int i = 0; i <numberofnodes ; i++) {
    String input = myReader.nextLine();    // get the entire line after the prompt
    numbers = input.split( regex: " ");
    for (int j = 0; j <numbers.length ; j++) {
        neighboors1[i][j]=Integer.parseInt(numbers[j]);
    }
}
```

Created start and end node, start only connects 0, end connects all -1.

```java
Node start =new Node( name: "S", x: 0, y: 0);
Node finish =new Node( name: "F", x: 0, y: 0);
```

```java
for (int i = 0; i <arr.length; i++) {
    for (int j = 0; j < neighboors1[i].length; j++) {

        if(neighboors1[i][j]==0)
        {
            break;
        }
        else{
            if (neighboors1[i][j]==-1)
            {
                arr[i].addNeighbor(new Edge(finish, weight: 0));
            }
            else {
                constant = (neighboors1[i][j]);
                System.out.println("Arr[" + i + "]" + ".addNeighbor(new Edge(arr[" + constant + "]," + cost[constant] + ")");
                arr[i].addNeighbor(new Edge(arr[constant], cost[constant]));
            }
        }
    }
```

Too see a couple of queries I printed out.So we create our graphs this way.

```
Arr[437].addNeighbor(new Edge(arr[521],0.7398154841337315)
Arr[437].addNeighbor(new Edge(arr[523],0.5215394415371927)
Arr[437].addNeighbor(new Edge(arr[526],0.541442755974465)
Arr[437].addNeighbor(new Edge(arr[528],0.4976606268323548)
Arr[437].addNeighbor(new Edge(arr[530],0.8682154552171609)
```

And the algorithm solves the rests.