323.25 Project 4: Dijkstra’s Algorithm for Single Source Shortest Path

Student: Jingshi Liu

Due Date: 3/10/2022

Algorithm Steps for the Project:

```
Step 0: open inFile, SSSfile, deBugFile
        numNodes ← get from inFile
        Allocate and initialize all members in the DijktraSSS class accordingly
Step 1: loadCostMatrix (inFile)
        sourceNode ← 1
Step 2: setBestAry (sourceNode)
        setParentAry (sourceNode)
        setToDoAry (sourceNode)
Step 3: minNode ← findMinNode (...)
        ToDoAry[minNode] ← 0
        debugPrint (...)
Step 4: for(childNode = 1; childNode <= numNodes; childNode++)
            if (todoArray[childNode] == 1 and bestArray[childNode] > computeCost(childNode)
                parentArray[childNode] = minNode
                bestArray[childNode] = newCost
                debuggingPrint()
Step 5: Repeat step 3 and 4 until todoArray contains no 1’s
Step 6: printShortestPath()
Step 7:sourceNode ++
Step 8: Repeat Step 2 to Step 7 while sourceNode <= numNodes
```

Source Code:

```
//
//  main.cpp
//  320Project 4
//
//  Created by Jingshi Liu on 3/7/22.
//

#include <iostream>
#include <fstream>
using namespace std;


class DijkstraSSS{
public:
    int numNodes;
    int sourceNode;
    int minNode;
    int currentNode;
    int newCost;
    int** costMatrix;
    int* parentArray;
    int* todoArray;
    int* bestArray;

    DijkstraSSS(int num_nodes){
        numNodes = num_nodes;
        // Allocate rows of costMatrix
        costMatrix = new int*[numNodes+1];
        // Allocate memory for 3 arrays below
        parentArray = new int[numNodes+1];
        todoArray = new int[numNodes+1];
        bestArray = new int[numNodes+1];
        for(int i = 0; i  < numNodes+1; i++){
            //Current pointer in 1D array now pointing to newly
allocated 1D array, this makes costMatrix a 2D array
            costMatrix[i] = new int[numNodes+1];
            // Initialize arrays below
            todoArray[i] = 1;
            bestArray[i] = 99999;
            for(int j = 0; j < numNodes+1; j++){
                costMatrix[i][j] = 99999;
            }
            // Change the diagonal index in the matrix to 0
            costMatrix[i][i] = 0;
        }
    }
    void loadCostMatrix(ifstream& inFile){
        int startingVertex, endingVertex, cost;
        while(inFile >> startingVertex && inFile >> endingVertex &&
inFile >> cost){
            costMatrix[startingVertex][endingVertex] = cost;
        }
    }

    void setBestArray(){
        for(int i = 1; i < numNodes+1; i++)
            bestArray[i] = costMatrix[sourceNode][i];
        bestArray[0] = 99999;
    }

    void setParentArray(){
        for(int i = 0; i < numNodes+1; i++)
            parentArray[i] = sourceNode;
    }

    void setToDoArray(){
        for(int i = 1; i < numNodes+1; i++)
            todoArray[i] = 1;
        todoArray[sourceNode] = 0;
        todoArray[0] = 0;
    }

    int findMinNode(){
        // Min cost by default is 99999 bc best bestArray[0] was
initialized to 99999
        int minNode = 0;
        for(int i = 1; i < numNodes+1; i++){
            if(todoArray[i] == 1 && bestArray[i] < bestArray[minNode])
                minNode = i;
        }
        todoArray[minNode] = 0;
        this->minNode = minNode;
        return minNode;
    }

    int computeCost(int node){
        this->newCost = bestArray[minNode] + costMatrix[minNode]
[node];
        return this->newCost;
    }

    bool checkToDoArray(){
        for(int i = 1; i < numNodes+1; i++)
            if(todoArray[i] == 1)
                return true;
        return false;
    }

    void debuggingPrint(ostream& debuggingFile){
        debuggingFile << "Source Node       : "<<sourceNode;
        debuggingFile << "\nParent Array    : ";
        for(int i = 1; i < numNodes+1; i++)
            debuggingFile << parentArray[i]<< " ";
        debuggingFile << "\nTo Do Array     : ";
        for(int i = 1; i < numNodes+1; i++)
            debuggingFile << todoArray[i] << " ";
        debuggingFile << "\nBest Cost Array : ";
        for(int i = 1; i < numNodes+1; i++)
            debuggingFile << bestArray[i] << " ";

        debuggingFile <<
"\n\n\n**************************************\n\n\n";
    }

    void printShortestPath(ostream& sssFile){
        int current_node = 1, cost = 0;
        string shortestPath = "";
        sssFile <<
"==================================================\nSource node =
"<<sourceNode<<"\n\n";

        for(int i = 1; i < numNodes+1; i++){
            current_node = i;
            shortestPath = to_string(current_node);
            cost = 0;
            while(current_node != sourceNode){
                shortestPath = to_string(parentArray[current_node]) +
" -> " + shortestPath;
                cost += costMatrix[parentArray[current_node]]
[current_node];
                current_node = parentArray[current_node];
            }
            sssFile << shortestPath << "   Cost = " << cost <<"\n";

        }
    }
};



int main(int argc, const char * argv[]) {
    ifstream inFile;
    ofstream sssFile, debuggingFile;
    DijkstraSSS* dijkstra;
    inFile.open(argv[1]);
    sssFile.open(argv[2]);
    debuggingFile.open(argv[3]);

    int numNodes = 0;
    inFile >> numNodes;

    dijkstra = new DijkstraSSS(numNodes);
    // Load Cost Matrix
    dijkstra->loadCostMatrix(inFile);

    for(int source_node = 1; source_node <= dijkstra->numNodes;
source_node++){

        dijkstra->sourceNode = source_node;
        dijkstra->setBestArray();
        dijkstra->setParentArray();
        dijkstra->setToDoArray();

        while(dijkstra->checkToDoArray()){
            dijkstra->findMinNode();
            for(int i = 1; i <= dijkstra->numNodes; i++){
                if(dijkstra->todoArray[i] == 1 && dijkstra-
>bestArray[i] > dijkstra->computeCost(i)){
                    dijkstra->parentArray[i] = dijkstra->minNode;
                    dijkstra->bestArray[i] = dijkstra->newCost;
                    dijkstra->debuggingPrint(debuggingFile);
                }
            }
        }
        dijkstra->printShortestPath(sssFile);

    }
    inFile.close();
    sssFile.close();
    debuggingFile.close();


    return 0;
}
```

Output:

<u>Data 1, SSSFile1, and debuggingFile1</u>

Data 1:

```
5
3 5 10
1 2 10
2 3 50
4 3 20
5 1 40
1 4 30
4 5 60
1 3 70
```

SSSFile for data 1:

```
=====================================================
Source node = 1

1   Cost = 0
1 -> 2   Cost = 10
1 -> 4 -> 3   Cost = 50
1 -> 4   Cost = 30
1 -> 4 -> 3 -> 5   Cost = 60
=====================================================
Source node = 2

2 -> 3 -> 5 -> 1   Cost = 100
2   Cost = 0
2 -> 3   Cost = 50
2 -> 3 -> 5 -> 1 -> 4   Cost = 130
2 -> 3 -> 5   Cost = 60
=====================================================
Source node = 3

3 -> 5 -> 1   Cost = 50
3 -> 5 -> 1 -> 2   Cost = 60
3   Cost = 0
3 -> 5 -> 1 -> 4   Cost = 80
3 -> 5   Cost = 10
=====================================================
Source node = 4

4 -> 3 -> 5 -> 1   Cost = 70
4 -> 3 -> 5 -> 1 -> 2   Cost = 80
4 -> 3   Cost = 20
4   Cost = 0
4 -> 3 -> 5   Cost = 30
=====================================================
Source node = 5

5 -> 1   Cost = 40
5 -> 1 -> 2   Cost = 50
5 -> 1 -> 4 -> 3   Cost = 90
5 -> 1 -> 4   Cost = 70
5   Cost = 0
```

debuggingFile2 for data 1:

```
Source Node       : 1
Parent Array    : 1 1 2 1 1
To Do Array     : 0 0 1 1 1
Best Cost Array : 0 10 60 30 99999


*************************************


Source Node       : 1
Parent Array    : 1 1 4 1 1
To Do Array     : 0 0 1 0 1
Best Cost Array : 0 10 50 30 99999


*************************************


Source Node       : 1
Parent Array    : 1 1 4 1 4
To Do Array     : 0 0 1 0 1
Best Cost Array : 0 10 50 30 90


*************************************


Source Node       : 1
Parent Array    : 1 1 4 1 3
To Do Array     : 0 0 0 0 1
Best Cost Array : 0 10 50 30 60


*************************************


Source Node       : 2
Parent Array    : 2 2 2 2 3
To Do Array     : 1 0 0 1 1
Best Cost Array : 99999 0 50 99999 60
```

```
**************************************

Source Node       : 2
Parent Array    : 5 2 2 2 3
To Do Array     : 1 0 0 1 0
Best Cost Array : 100 0 50 99999 60

**************************************

Source Node       : 2
Parent Array    : 5 2 2 1 3
To Do Array     : 0 0 0 1 0
Best Cost Array : 100 0 50 130 60

**************************************

Source Node       : 3
Parent Array    : 5 3 3 3 3
To Do Array     : 1 1 0 1 0
Best Cost Array : 50 99999 0 99999 10

**************************************

Source Node       : 3
Parent Array    : 5 1 3 3 3
To Do Array     : 0 1 0 1 0
Best Cost Array : 50 60 0 99999 10

**************************************

Source Node       : 3
Parent Array    : 5 1 3 1 3
To Do Array     : 0 1 0 1 0
Best Cost Array : 50 60 0 80 10

**************************************

Source Node       : 4
Parent Array    : 4 4 4 4 3
To Do Array     : 1 1 0 0 1
Best Cost Array : 99999 99999 20 0 30
```

```
**************************************


Source Node       : 4
Parent Array    : 5 4 4 4 3
To Do Array     : 1 1 0 0 0
Best Cost Array : 70 99999 20 0 30


**************************************


Source Node       : 4
Parent Array    : 5 1 4 4 3
To Do Array     : 0 1 0 0 0
Best Cost Array : 70 80 20 0 30


**************************************


Source Node       : 5
Parent Array    : 5 1 5 5 5
To Do Array     : 0 1 1 1 0
Best Cost Array : 40 50 99999 99999 0


**************************************


Source Node       : 5
Parent Array    : 5 1 1 5 5
To Do Array     : 0 1 1 1 0
Best Cost Array : 40 50 110 99999 0


**************************************


Source Node       : 5
Parent Array    : 5 1 1 1 5
To Do Array     : 0 1 1 1 0
Best Cost Array : 40 50 110 70 0


**************************************


Source Node       : 5
Parent Array    : 5 1 2 1 5
To Do Array     : 0 0 1 1 0
Best Cost Array : 40 50 100 70 0
```

```
*************************************


Source Node       : 5
Parent Array    : 5 1 4 1 5
To Do Array     : 0 0 1 0 0
Best Cost Array : 40 50 90 70 0


*************************************
```

<u>Data 2, SSSFile2, and debuggingFile2</u>

Data 2:

```
8
1  2 10
2 8  2
3    2 2
6  7   2
1 5    30
8  7  2
1  4    20
3  8  10
1  3 5
8 1 6
3 7  30
4   6   3
3  4   5
4  5     5
5  6   15
7  4   4
7 6 3
6 1 5
8 6    7
1 8 20
7 1 40
5 2 10
7 5 30
5 8 3
2 7 40
2 3 35
```

```
2 1 25
6 2 5
6 3 20
4 7 25
4 8 20
```

SSSFile for data 2:

```
==================================================
Source node = 1

1   Cost = 0
1 -> 3 -> 2   Cost = 7
1 -> 3   Cost = 5
1 -> 3 -> 4   Cost = 10
1 -> 3 -> 4 -> 5   Cost = 15
1 -> 3 -> 4 -> 6   Cost = 13
1 -> 3 -> 2 -> 8 -> 7   Cost = 11
1 -> 3 -> 2 -> 8   Cost = 9
==================================================
Source node = 2

2 -> 8 -> 1   Cost = 8
2   Cost = 0
2 -> 8 -> 1 -> 3   Cost = 13
2 -> 8 -> 7 -> 4   Cost = 8
2 -> 8 -> 7 -> 4 -> 5   Cost = 13
2 -> 8 -> 7 -> 6   Cost = 7
2 -> 8 -> 7   Cost = 4
2 -> 8   Cost = 2
==================================================
Source node = 3

3 -> 2 -> 8 -> 1   Cost = 10
3 -> 2   Cost = 2
3   Cost = 0
3 -> 4   Cost = 5
3 -> 4 -> 5   Cost = 10
3 -> 4 -> 6   Cost = 8
3 -> 2 -> 8 -> 7   Cost = 6
3 -> 2 -> 8   Cost = 4
==================================================
Source node = 4

4 -> 6 -> 1   Cost = 8
4 -> 6 -> 2   Cost = 8
4 -> 6 -> 1 -> 3   Cost = 13
4   Cost = 0
4 -> 5   Cost = 5
4 -> 6   Cost = 3
4 -> 6 -> 7   Cost = 5
4 -> 5 -> 8   Cost = 8
```

```
=====================================================
Source node = 5

5 -> 8 -> 1   Cost = 9
5 -> 2   Cost = 10
5 -> 8 -> 1 -> 3   Cost = 14
5 -> 8 -> 7 -> 4   Cost = 9
5   Cost = 0
5 -> 8 -> 7 -> 6   Cost = 8
5 -> 8 -> 7   Cost = 5
5 -> 8   Cost = 3
=====================================================
Source node = 6

6 -> 1   Cost = 5
6 -> 2   Cost = 5
6 -> 1 -> 3   Cost = 10
6 -> 7 -> 4   Cost = 6
6 -> 7 -> 4 -> 5   Cost = 11
6   Cost = 0
6 -> 7   Cost = 2
6 -> 2 -> 8   Cost = 7
=====================================================
Source node = 7

7 -> 6 -> 1   Cost = 8
7 -> 6 -> 2   Cost = 8
7 -> 6 -> 1 -> 3   Cost = 13
7 -> 4   Cost = 4
7 -> 4 -> 5   Cost = 9
7 -> 6   Cost = 3
7   Cost = 0
7 -> 6 -> 2 -> 8   Cost = 10
=====================================================
Source node = 8

8 -> 1   Cost = 6
8 -> 7 -> 6 -> 2   Cost = 10
8 -> 1 -> 3   Cost = 11
8 -> 7 -> 4   Cost = 6
8 -> 7 -> 4 -> 5   Cost = 11
8 -> 7 -> 6   Cost = 5
8 -> 7   Cost = 2
8   Cost = 0
```

DebuggingFile for data 2:

```
Source Node       : 1
Parent Array    : 1 3 1 1 1 1 1 1
To Do Array     : 0 1 0 1 1 1 1 1
Best Cost Array : 0 7 5 20 30 99999 99999 20
```

```
**************************************

Source Node       : 1
Parent Array    : 1 3 1 3 1 1 1 1
To Do Array     : 0 1 0 1 1 1 1 1
Best Cost Array : 0 7 5 10 30 99999 99999 20

**************************************

Source Node       : 1
Parent Array    : 1 3 1 3 1 1 3 1
To Do Array     : 0 1 0 1 1 1 1 1
Best Cost Array : 0 7 5 10 30 99999 35 20

**************************************

Source Node       : 1
Parent Array    : 1 3 1 3 1 1 3 3
To Do Array     : 0 1 0 1 1 1 1 1
Best Cost Array : 0 7 5 10 30 99999 35 15

**************************************

Source Node       : 1
Parent Array    : 1 3 1 3 1 1 3 2
To Do Array     : 0 0 0 1 1 1 1 1
Best Cost Array : 0 7 5 10 30 99999 35 9

**************************************

Source Node       : 1
Parent Array    : 1 3 1 3 1 8 3 2
To Do Array     : 0 0 0 1 1 1 1 0
Best Cost Array : 0 7 5 10 30 16 35 9

**************************************

Source Node       : 1
Parent Array    : 1 3 1 3 1 8 8 2
To Do Array     : 0 0 0 1 1 1 1 0
Best Cost Array : 0 7 5 10 30 16 11 9
```

```
*****************************************

Source Node       : 1
Parent Array    : 1 3 1 3 4 8 8 2
To Do Array     : 0 0 0 0 1 1 1 0
Best Cost Array : 0 7 5 10 15 16 11 9

*****************************************

Source Node       : 1
Parent Array    : 1 3 1 3 4 4 8 2
To Do Array     : 0 0 0 0 1 1 1 0
Best Cost Array : 0 7 5 10 15 13 11 9

*****************************************

Source Node       : 2
Parent Array    : 8 2 2 2 2 2 2 2
To Do Array     : 1 0 1 1 1 1 1 0
Best Cost Array : 8 0 35 99999 99999 99999 40 2

*****************************************

Source Node       : 2
Parent Array    : 8 2 2 2 2 8 2 2
To Do Array     : 1 0 1 1 1 1 1 0
Best Cost Array : 8 0 35 99999 99999 9 40 2

*****************************************

Source Node       : 2
Parent Array    : 8 2 2 2 2 8 8 2
To Do Array     : 1 0 1 1 1 1 1 0
Best Cost Array : 8 0 35 99999 99999 9 4 2

*****************************************

Source Node       : 2
Parent Array    : 8 2 2 7 2 8 8 2
To Do Array     : 1 0 1 1 1 1 0 0
Best Cost Array : 8 0 35 8 99999 9 4 2
```

```
***************************************


Source Node       : 2
Parent Array    : 8 2 2 7 7 8 8 2
To Do Array     : 1 0 1 1 1 1 0 0
Best Cost Array : 8 0 35 8 34 9 4 2


***************************************


Source Node       : 2
Parent Array    : 8 2 2 7 7 7 8 2
To Do Array     : 1 0 1 1 1 1 0 0
Best Cost Array : 8 0 35 8 34 7 4 2


***************************************


Source Node       : 2
Parent Array    : 8 2 6 7 7 7 8 2
To Do Array     : 1 0 1 1 1 0 0 0
Best Cost Array : 8 0 27 8 34 7 4 2


***************************************


Source Node       : 2
Parent Array    : 8 2 1 7 7 7 8 2
To Do Array     : 0 0 1 1 1 0 0 0
Best Cost Array : 8 0 13 8 34 7 4 2


***************************************


Source Node       : 2
Parent Array    : 8 2 1 7 4 7 8 2
To Do Array     : 0 0 1 0 1 0 0 0
Best Cost Array : 8 0 13 8 13 7 4 2


***************************************


Source Node       : 3
Parent Array    : 2 3 3 3 3 3 3 3
To Do Array     : 1 0 0 1 1 1 1 1
Best Cost Array : 27 2 0 5 99999 99999 30 10
```

```
****************************************

Source Node       : 3
Parent Array    : 2 3 3 3 3 3 3 2
To Do Array     : 1 0 0 1 1 1 1 1
Best Cost Array : 27 2 0 5 99999 99999 30 4

****************************************

Source Node       : 3
Parent Array    : 8 3 3 3 3 3 3 2
To Do Array     : 1 0 0 1 1 1 1 0
Best Cost Array : 10 2 0 5 99999 99999 30 4

****************************************

Source Node       : 3
Parent Array    : 8 3 3 3 3 8 3 2
To Do Array     : 1 0 0 1 1 1 1 0
Best Cost Array : 10 2 0 5 99999 11 30 4

****************************************

Source Node       : 3
Parent Array    : 8 3 3 3 3 8 8 2
To Do Array     : 1 0 0 1 1 1 1 0
Best Cost Array : 10 2 0 5 99999 11 6 4

****************************************

Source Node       : 3
Parent Array    : 8 3 3 3 4 8 8 2
To Do Array     : 1 0 0 0 1 1 1 0
Best Cost Array : 10 2 0 5 10 11 6 4

****************************************

Source Node       : 3
Parent Array    : 8 3 3 3 4 4 8 2
To Do Array     : 1 0 0 0 1 1 1 0
Best Cost Array : 10 2 0 5 10 8 6 4
```

```
***************************************


Source Node       : 4
Parent Array    : 6 4 4 4 4 4 4 4
To Do Array     : 1 1 1 0 1 0 1 1
Best Cost Array : 8 99999 99999 0 5 3 25 20


***************************************


Source Node       : 4
Parent Array    : 6 6 4 4 4 4 4 4
To Do Array     : 1 1 1 0 1 0 1 1
Best Cost Array : 8 8 99999 0 5 3 25 20


***************************************


Source Node       : 4
Parent Array    : 6 6 6 4 4 4 4 4
To Do Array     : 1 1 1 0 1 0 1 1
Best Cost Array : 8 8 23 0 5 3 25 20


***************************************


Source Node       : 4
Parent Array    : 6 6 6 4 4 4 6 4
To Do Array     : 1 1 1 0 1 0 1 1
Best Cost Array : 8 8 23 0 5 3 5 20


***************************************


Source Node       : 4
Parent Array    : 6 6 6 4 4 4 6 5
To Do Array     : 1 1 1 0 0 0 1 1
Best Cost Array : 8 8 23 0 5 3 5 8


***************************************


Source Node       : 4
Parent Array    : 6 6 1 4 4 4 6 5
To Do Array     : 0 1 1 0 0 0 0 1
Best Cost Array : 8 8 13 0 5 3 5 8
```

```
**************************************


Source Node       : 5
Parent Array    : 8 5 5 5 5 5 5 5
To Do Array     : 1 1 1 1 0 1 1 0
Best Cost Array : 9 10 99999 99999 0 15 99999 3


**************************************


Source Node       : 5
Parent Array    : 8 5 5 5 5 8 5 5
To Do Array     : 1 1 1 1 0 1 1 0
Best Cost Array : 9 10 99999 99999 0 10 99999 3


**************************************


Source Node       : 5
Parent Array    : 8 5 5 5 5 8 8 5
To Do Array     : 1 1 1 1 0 1 1 0
Best Cost Array : 9 10 99999 99999 0 10 5 3


**************************************


Source Node       : 5
Parent Array    : 8 5 5 7 5 8 8 5
To Do Array     : 1 1 1 1 0 1 0 0
Best Cost Array : 9 10 99999 9 0 10 5 3


**************************************


Source Node       : 5
Parent Array    : 8 5 5 7 5 7 8 5
To Do Array     : 1 1 1 1 0 1 0 0
Best Cost Array : 9 10 99999 9 0 8 5 3


**************************************


Source Node       : 5
Parent Array    : 8 5 6 7 5 7 8 5
To Do Array     : 1 1 1 1 0 0 0 0
Best Cost Array : 9 10 28 9 0 8 5 3
```

```
**************************************

Source Node       : 5
Parent Array    : 8 5 1 7 5 7 8 5
To Do Array     : 0 1 1 1 0 0 0 0
Best Cost Array : 9 10 14 9 0 8 5 3

**************************************

Source Node       : 6
Parent Array    : 6 6 6 7 6 6 6 6
To Do Array     : 1 1 1 1 1 0 0 1
Best Cost Array : 5 5 20 6 99999 0 2 99999

**************************************

Source Node       : 6
Parent Array    : 6 6 6 7 7 6 6 6
To Do Array     : 1 1 1 1 1 0 0 1
Best Cost Array : 5 5 20 6 32 0 2 99999

**************************************

Source Node       : 6
Parent Array    : 6 6 1 7 7 6 6 6
To Do Array     : 0 1 1 1 1 0 0 1
Best Cost Array : 5 5 10 6 32 0 2 99999

**************************************

Source Node       : 6
Parent Array    : 6 6 1 7 7 6 6 1
To Do Array     : 0 1 1 1 1 0 0 1
Best Cost Array : 5 5 10 6 32 0 2 25

**************************************

Source Node       : 6
Parent Array    : 6 6 1 7 7 6 6 2
To Do Array     : 0 0 1 1 1 0 0 1
Best Cost Array : 5 5 10 6 32 0 2 7
```

```
****************************************


Source Node       : 6
Parent Array    : 6 6 1 7 4 6 6 2
To Do Array     : 0 0 1 0 1 0 0 1
Best Cost Array : 5 5 10 6 11 0 2 7


****************************************


Source Node       : 7
Parent Array    : 6 7 7 7 7 7 7 7
To Do Array     : 1 1 1 1 1 0 0 1
Best Cost Array : 8 99999 99999 4 30 3 0 99999


****************************************


Source Node       : 7
Parent Array    : 6 6 7 7 7 7 7 7
To Do Array     : 1 1 1 1 1 0 0 1
Best Cost Array : 8 8 99999 4 30 3 0 99999


****************************************


Source Node       : 7
Parent Array    : 6 6 6 7 7 7 7 7
To Do Array     : 1 1 1 1 1 0 0 1
Best Cost Array : 8 8 23 4 30 3 0 99999


****************************************


Source Node       : 7
Parent Array    : 6 6 6 7 4 7 7 7
To Do Array     : 1 1 1 0 1 0 0 1
Best Cost Array : 8 8 23 4 9 3 0 99999


****************************************


Source Node       : 7
Parent Array    : 6 6 6 7 4 7 7 4
To Do Array     : 1 1 1 0 1 0 0 1
Best Cost Array : 8 8 23 4 9 3 0 24
```

```
**************************************


Source Node       : 7
Parent Array    : 6 6 1 7 4 7 7 4
To Do Array     : 0 1 1 0 1 0 0 1
Best Cost Array : 8 8 13 4 9 3 0 24


**************************************


Source Node       : 7
Parent Array    : 6 6 1 7 4 7 7 2
To Do Array     : 0 0 1 0 1 0 0 1
Best Cost Array : 8 8 13 4 9 3 0 10


**************************************


Source Node       : 8
Parent Array    : 8 8 8 7 8 8 8 8
To Do Array     : 1 1 1 1 1 1 0 0
Best Cost Array : 6 99999 99999 6 99999 7 2 0


**************************************


Source Node       : 8
Parent Array    : 8 8 8 7 7 8 8 8
To Do Array     : 1 1 1 1 1 1 0 0
Best Cost Array : 6 99999 99999 6 32 7 2 0


**************************************


Source Node       : 8
Parent Array    : 8 8 8 7 7 7 8 8
To Do Array     : 1 1 1 1 1 1 0 0
Best Cost Array : 6 99999 99999 6 32 5 2 0


**************************************


Source Node       : 8
Parent Array    : 8 6 8 7 7 7 8 8
To Do Array     : 1 1 1 1 1 0 0 0
Best Cost Array : 6 10 99999 6 32 5 2 0
```

```
*************************************

Source Node      : 8
Parent Array    : 8 6 6 7 7 7 8 8
To Do Array     : 1 1 1 1 1 0 0 0
Best Cost Array : 6 10 25 6 32 5 2 0

*************************************

Source Node      : 8
Parent Array    : 8 6 1 7 7 7 8 8
To Do Array     : 0 1 1 1 1 0 0 0
Best Cost Array : 6 10 11 6 32 5 2 0

*************************************

Source Node      : 8
Parent Array    : 8 6 1 7 4 7 8 8
To Do Array     : 0 1 1 0 1 0 0 0
Best Cost Array : 6 10 11 6 11 5 2 0

*************************************
```