

Dijkstras

0.1.0

Generated by Doxygen 1.8.17

1 Final Project - Dijkstra's Algorithm	1
1.1 References	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Graph Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 Graph()	7
4.1.3 Member Function Documentation	8
4.1.3.1 addEdge()	8
4.1.3.2 distance()	8
4.1.3.3 length()	9
4.1.3.4 matrixOut()	9
4.1.3.5 outEdges()	9
4.1.3.6 removeEdge()	10
5 File Documentation	11
5.1 /home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/README.md File Reference	11
5.2 /home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/src/main.cpp File Reference	11
5.2.1 Detailed Description	12
5.2.2 Function Documentation	12
5.2.2.1 Dijkstra()	12
5.2.2.2 main()	13
5.2.2.3 printShortPath()	13
Index	15

Chapter 1

Final Project - Dijkstra's Algorithm

1. This project is using Dijkstra's Algorithm to find the shortest path among any length of nodes in a graph. We decided to choose 9 different vertices starting from 0 to 8. The purpose of this program is to find the shortest distance from vertex 0 being the starting point and to other vertices 1-8.
2. For example, the shortest distance path between 0 and 8 vertices is $0 \rightarrow 1 \rightarrow 2 \rightarrow 8$, and the actual distance would be 14 since the distance from $0 \rightarrow 1 = 4$ (0,1,4), $1 \rightarrow 2 = 8$ (1,2,8), $2 \rightarrow 8 = 2$ (2,8,2) according to addEdge functions, therefore, $4 + 8 + 2 = 14$.
3. We wanted to do a project that uses something that we have learned before such as graphs data structure, and Dijkstra's algorithm was something that seemed interesting to learn and implement with our project.
4. The data structure we chose is graphs because it is the best representation of applying Dijkstra's algorithm.
5. We chose Dijkstra's algorithm because not only it is a popular algorithm in dynamic programming, but it is also the algorithm that we needed to find the shortest distance and path between any given pair of vertices.

1.1 References

1. <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>
2. http://www.gitta.info/Accessibiliti/en/html/Dijkstra_learningObject1.↵html
3. <http://www.cplusplus.com/forum/articles/7459/>

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Graph	7
---------------------------------	-------------------

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/src/ main.cpp	
This is Final Project for CPTR 227	11

Chapter 4

Class Documentation

4.1 Graph Class Reference

Public Member Functions

- [Graph](#) (int x)
- void [addEdge](#) (int i, int j, int dist)
- void [removeEdge](#) (int i, int j)
- int [distance](#) (int i, int j)
- vector< int > [outEdges](#) (int i)
- int [length](#) ()
- int ** [matrixOut](#) ()

4.1.1 Detailed Description

This class creates an integer matrix graph.

Definition at line 19 of file main.cpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Graph()

```
Graph::Graph (  
    int x ) [inline]
```

[Graph](#) constructor

Parameters

<i>x</i>	The dimensions of the graph
----------	-----------------------------

Definition at line 31 of file main.cpp.

```

31     {
32         n = x;
33         matrix = new int*[n];
34         for (int i=0; i<n; i++)
35             matrix[i] = new int[n];
36     }
37 }
```

4.1.3 Member Function Documentation

4.1.3.1 addEdge()

```

void Graph::addEdge (
    int i,
    int j,
    int dist ) [inline]
```

Adds an edge to the graph by setting [i][j] to the distance from the nodes

Parameters

<i>i</i>	Initial node
<i>j</i>	Node to connect

Definition at line 45 of file main.cpp.

```

45     {
46         matrix[i][j] = dist;
47         matrix[j][i] = dist;
48     }
```

4.1.3.2 distance()

```

int Graph::distance (
    int i,
    int j ) [inline]
```

Finds distance by checking the number stored in [i][j]

Parameters

<i>i</i>	Initial node
<i>j</i>	A connected node

Definition at line 66 of file main.cpp.

```

66     {
67         return matrix[i][j];
68     }
```

4.1.3.3 length()

```
int Graph::length ( ) [inline]
```

Returns the length by returning n

Returns

length of graph

Definition at line 91 of file main.cpp.

```
91 {  
92     return n;  
93 }
```

4.1.3.4 matrixOut()

```
int** Graph::matrixOut ( ) [inline]
```

Returns the matrix

Returns

the matrix

Definition at line 100 of file main.cpp.

```
100 {  
101     return matrix;  
102 }
```

4.1.3.5 outEdges()

```
vector<int> Graph::outEdges (  
    int i ) [inline]
```

Finds all nodes connected to i by checking all of it's rows

Parameters

<i>i</i>	Initial node
----------	--------------

Returns

A vector of all connected nodes

Definition at line 76 of file main.cpp.

```
76 {  
77     vector<int> edges;
```

```
78         for (int j=0; j<n; j++) {
79             if (matrix[i][j]) {
80                 edges.push_back(j);
81             }
82         }
83         return edges;
84     }
```

4.1.3.6 removeEdge()

```
void Graph::removeEdge (
    int i,
    int j ) [inline]
```

Removes an edge from the graph by setting [i][j] to 0

Parameters

<i>i</i>	Initial node
<i>j</i>	Connected node

Definition at line 56 of file main.cpp.

```
56                                     {
57         matrix[i][j] = 0;
58         matrix[i][j] = 0;
59     }
```

The documentation for this class was generated from the following file:

- </home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/src/main.cpp>

Chapter 5

File Documentation

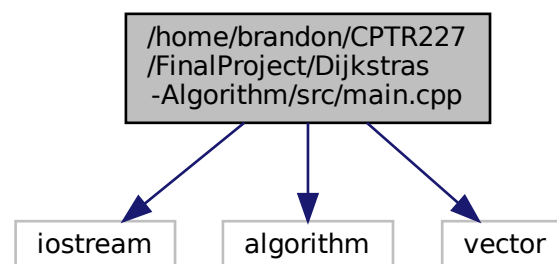
5.1 /home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/README.md File Reference

5.2 /home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/src/main.cpp File Reference

This is Final Project for CPTR 227.

```
#include <iostream>
#include <algorithm>
#include <vector>
```

Include dependency graph for main.cpp:



Classes

- class [Graph](#)

Functions

- void `Dijkstra` (`Graph` g, int source, int *dist, int *parent)
- void `printShortPath` (`Graph` g, int *dist, int *parent)
- int `main` (int, char **)

5.2.1 Detailed Description

This is Final Project for CPTR 227.

This is implementation of Dijkstra's Algorithm to find the shortest path.

Author

Nathan Quick & Brandon Yi

Date

5/5/2021

5.2.2 Function Documentation

5.2.2.1 Dijkstra()

```
void Dijkstra (
    Graph g,
    int source,
    int * dist,
    int * parent )
```

Finds the shortest path in a graph using Dijkstras algorithm

Parameters

<i>g</i>	The graph
<i>source</i>	Initial node
<i>dist</i>	An empty array with the same size as g to store distance values
<i>parent</i>	An empty array with the same size as g to store path values

Definition at line 114 of file main.cpp.

```
114                                     {
115     int index = 0;
116     bool used[g.length()]; // An array that stors if the node's shortest path has already be found
117
118     for (int i=0; i<g.length(); i++){
119         dist[i] = -1;
120         used[i] = false;
121     }
122
123     dist[source] = 0; // The distance from source to source is 0
```



```

124     used[source] = true; // Mark source as used
125     parent[source] = 0; // Source has no path to source
126
127     for (int i=0; i<g.length(); i++){
128
129         vector<int> edge = g.outEdges(index); // Stores all of index's connected nodes
130
131         // Updates the distances from source to the edge
132         for (int ii=0; ii<edge.size(); ii++){
133             int u = dist[index] + g.distance(index, edge[ii]);
134             if (u < dist[edge[ii]] || dist[edge[ii]] == -1){
135                 parent[edge[ii]] = index;
136                 dist[edge[ii]] = dist[index] + g.distance(index, edge[ii]);
137             }
138         }
139         // Reset min
140         int min = -1; // Stores the smallest distance value
141
142         // Finds the shortest path of the discovered distances
143         for (int ii=0; ii<g.length(); ii++){
144             if (!used[ii] && dist[ii] != -1){
145                 if (dist[ii] < min || min == -1){
146                     min = dist[ii];
147                     index = ii;
148                 }
149             }
150         }
151         // Marks shortest node path as used
152         used[index] = true;
153     }
154 }

```

5.2.2.2 main()

```

int main (
    int ,
    char ** )

```

Definition at line 190 of file main.cpp.

```

190     {
191         Graph g(9);
192
193         int dist[g.length()];
194         int parent[g.length()];
195
196         g.addEdge(0,1,4);
197         g.addEdge(1,2,8);
198         g.addEdge(2,3,7);
199         g.addEdge(3,4,9);
200         g.addEdge(4,5,10);
201         g.addEdge(5,6,2);
202         g.addEdge(6,7,1);
203         g.addEdge(7,8,7);
204         g.addEdge(7,0,8);
205         g.addEdge(1,7,11);
206         g.addEdge(2,5,4);
207         g.addEdge(3,5,14);
208         g.addEdge(2,8,2);
209         g.addEdge(6,8,6);
210         g.addEdge(0,7,8);
211
212         Dijkstra(g, 0, dist, parent);
213         printShortPath(g, dist, parent);
214     }

```

5.2.2.3 printShortPath()

```

void printShortPath (
    Graph g,
    int * dist,
    int * parent )

```

Prints the distances and shortest path of each node from source

Parameters

<i>g</i>	The graph
<i>dist</i>	The list of smallest distances
<i>parent</i>	A list that hold path info

Definition at line 163 of file main.cpp.

```
163                                     {
164     cout << "Distance Shortest" << endl;
165     cout << "    from 0      path" << endl;
166     cout << "-----" << endl;
167     int index = 0;
168     vector<int> items;
169     for (int i=1; i<g.length(); i++){
170         cout << i << ": " << dist[i] << string(5, ' ');
171         items.clear();
172         index = i;
173         do
174         {
175             items.push_back(parent[index]);
176             index = parent[index];
177         } while (index > 0);
178         reverse(items.begin(),items.end());
179         for (int ii=0; ii<=items.size()-1; ii++){
180             if (ii==0 && dist[i] < 10){
181                 cout << ' ';
182             }
183             cout << items[ii] << " + ";
184         }
185         cout << i << endl;
186     }
187 }
```

Index

/home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/README.md,
[11](#)

/home/brandon/CPTR227/FinalProject/Dijkstras-Algorithm/src/main.cpp,
[11](#)

addEdge
 Graph, [8](#)

Dijkstra
 main.cpp, [12](#)

distance
 Graph, [8](#)

Graph, [7](#)
 addEdge, [8](#)
 distance, [8](#)
 Graph, [7](#)
 length, [8](#)
 matrixOut, [9](#)
 outEdges, [9](#)
 removeEdge, [10](#)

length
 Graph, [8](#)

main
 main.cpp, [13](#)

main.cpp
 Dijkstra, [12](#)
 main, [13](#)
 printShortPath, [13](#)

matrixOut
 Graph, [9](#)

outEdges
 Graph, [9](#)

printShortPath
 main.cpp, [13](#)

removeEdge
 Graph, [10](#)