

# Programming Assignment 6

## Report

### Jaideen Gerig Sec:505

#### Problem Description:

Use Dijkstra's algorithm to find the shortest path in a directed graph from a source vertex to all other vertices, then provide the results of that path.

---

#### Purpose of the Assignment:

The purpose of the assignment was to demonstrate a knowledge of the implementation of the Graph ADT and Dijkstra's algorithm. Real world applications of Dijkstra's algorithm would include GPS directions for roads, optimizing network routing to reduce latency, and flight planning for airlines.

---

#### Data Structures Description:

I used the Minimum Priority Queue based on a Binary Heap that I created in the previous programming assignment to efficiently implement Dijkstra's algorithm. The Graph ADT was implemented using a list of vertices and edges, edges were implemented with pointers to each vertex they are connected to, every vertex holds a list of all the edges connected to it.

---

#### Algorithm Description:

The main algorithm used in this assignment was Dijkstra's algorithm, which is a greedy algorithm that tests every edge and vertex to find the shortest path from a source node. Which I implemented using the Minimum Priority Queue supported by a Binary Heap, with locator structure, allowing the algorithm as a whole to run in  $O((V+E)\log(V))$  time as every edge and vertex is evaluated, and it takes  $O(\log(V))$  time to resort the binary heap after altering a key or removing the minimum, without the Locator Structure this would run in  $O((V+E)V\log(V))$  time as we would have to search for the id of every key we would want to relax, however this is not the case as my algorithm leverages the structure to replace each key in  $O(1)$  time.

---

#### Program Organization:

BinaryHeap.h and PriorityQueue.h both contain the implementation for the Minimum Priority Queue data structure. Vertex.h holds the vertex class which holds an ID, an in and out list of edges, its distance from the source node, and its parent from the source node. Edge.h holds the edge class which contains pointers to its start and end vertices as well as the weight of traveling between

them. Graph.h and Graph.cpp both hold the Graph class as well as the implementation for Dijkstra's algorithm and functions for printing the entire graph as an adjacency list and printing the shortest path between 2 nodes. GraphTest.cpp contains all the testing for the Graph ADT and Dijkstra's algorithm.

---

## How to compile + run my program:

(1) type "make all" to compile (2) type "./Main" (3) enter the source and end node you would like to find the shortest path between (4) the program will print the path and exit

---

## Input + Output Specs:

Input must be given in this format:

# of nodes

NodeID EndVertex weight -1

-1

The output will be given in this format:

(NodeID)-weight->(NodeID)...

Total Weight For Shortest Path: Weight

Total # of Comparisons: Comparisons

---

## Logical Exceptions:

Crashes when: Input is not formatted correctly from the text file

Throws exception when: Looking for a node that does not exist, Source and End vertex are the same, no path exists from Source to End Vertex

---

## C++ OO or Generic Programming features:

This program uses templates for generic programming to parameterize the type of the binary heap and priority queue implementations.

---

## Tests:

Valid Cases:

```
linux2.cse.tamu.edu - PuTTY

[jaideng]@linux2 ~/221_hw/A6> (13:11:59 04/21/14)
:: ./Main
Please enter the vertex you would like to start at
1
Please enter the vertex you would like to end at
200
Calculating path...
(1)--3-->(30)--3-->(118)--1-->(114)--1-->(65)--1-->(194)--5-->(200)
Total Weight For Shortest Path: 14
Total # of Comparisons: 18237
[jaideng]@linux2 ~/221_hw/A6> (13:12:07 04/21/14)
:: ./Main
Please enter the vertex you would like to start at
200
Please enter the vertex you would like to end at
1
Calculating path...
(200)--8-->(48)--4-->(88)--6-->(182)--1-->(1)
Total Weight For Shortest Path: 19
Total # of Comparisons: 18230
[jaideng]@linux2 ~/221_hw/A6> (13:12:21 04/21/14)
:: █
```

(tested

on random\_sparse.txt)

Invalid Cases:

```
linux2.cse.tamu.edu - PuTTY

[jaideng]@linux2 ~/221_hw/A6> (13:13:08 04/21/14)
:: ./Main
Please enter the vertex you would like to start at
1
Please enter the vertex you would like to end at
1
Calculating path...
terminate called after throwing an instance of 'std::runtime_error'
  what():  End and source are the same
Aborted

[jaideng]@linux2 ~/221_hw/A6> (13:13:13 04/21/14)
:: ./Main
Please enter the vertex you would like to start at
500
Please enter the vertex you would like to end at
1
Calculating path...
terminate called after throwing an instance of 'std::runtime_error'
  what():  Vertex does not exist in this graph
Aborted

[jaideng]@linux2 ~/221_hw/A6> (13:13:28 04/21/14)
:: █
```

(tested

on random\_sparse.txt)

```
Vertex.h
5
1 2 4 -1
2 3 5 -1
3 4 2 -1
4 -1
5 1 2 -1
-1

[jaideng]@linux2 ~/221_hw/A6> (13:19:02 04/21/14)
:: ./Main
Please enter the vertex you would like to start at
1
Please enter the vertex you would like to end at
5
Calculating path...
terminate called after throwing an instance of 'std::runtime_error'
  what(): No path exists
Aborted

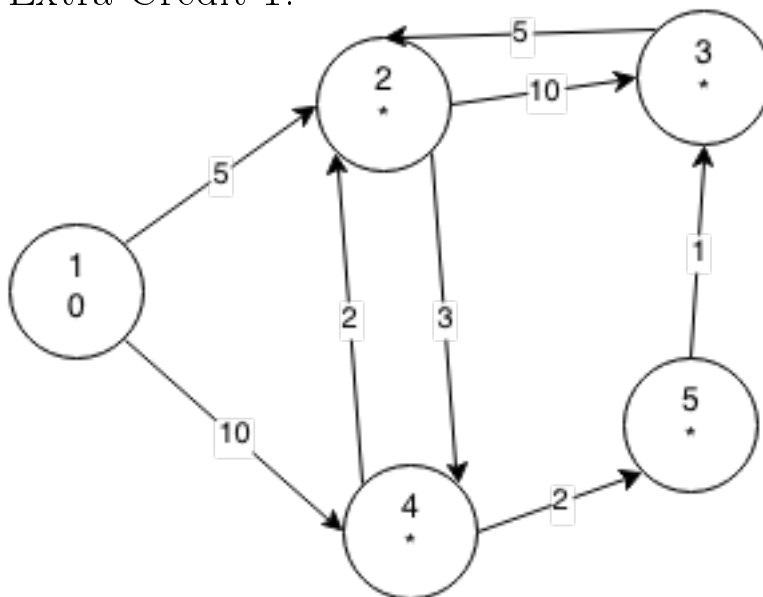
[jaideng]@linux2 ~/221_hw/A6> (13:19:07 04/21/14)
::
```

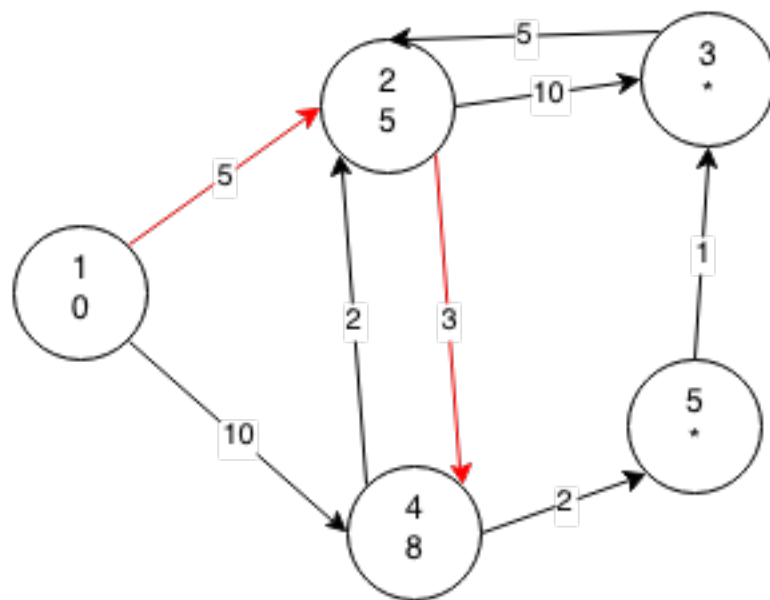
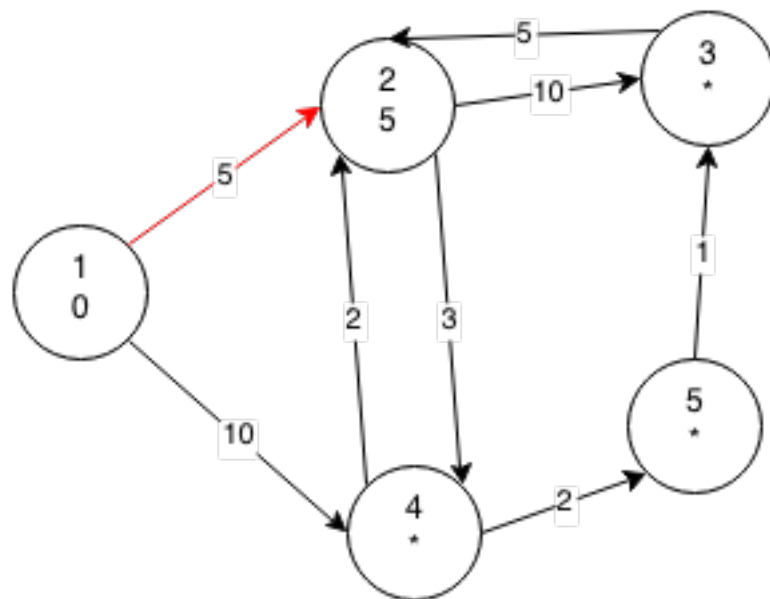
on modified simple.txt)

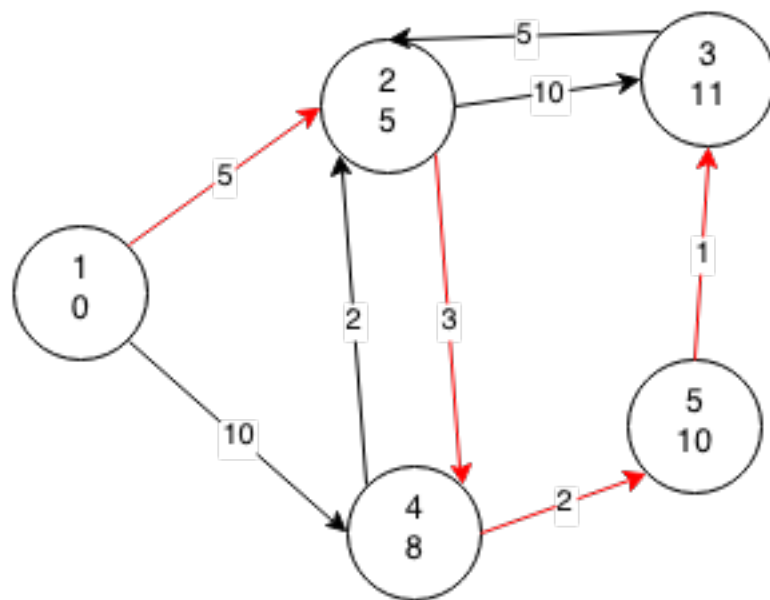
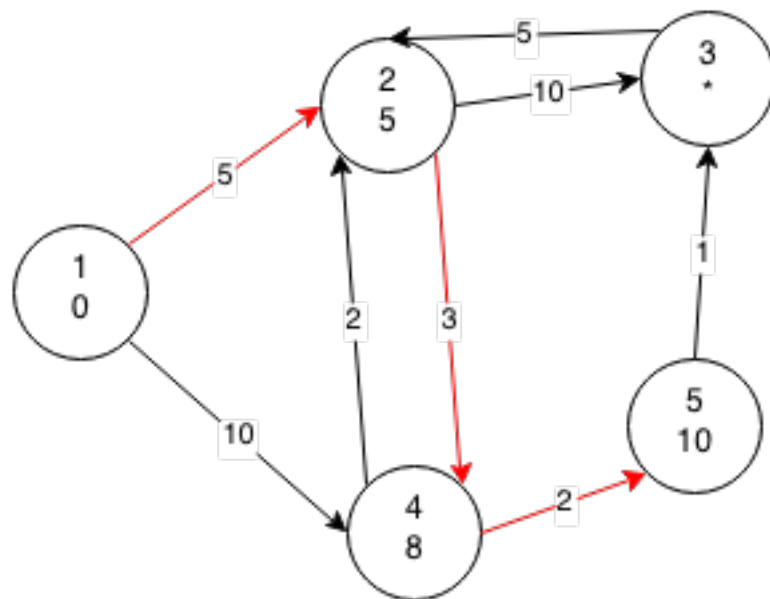
Other test files included in folder

---

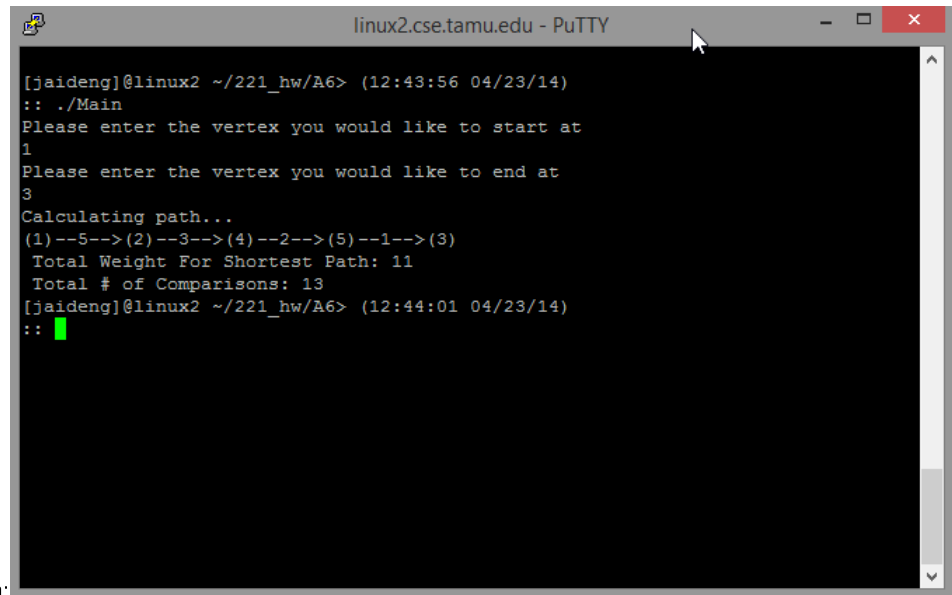
### Extra Credit 1:







Final



A screenshot of a PuTTY terminal window titled "linux2.cse.tamu.edu - PuTTY". The terminal shows a user named "jaideng" at a prompt. The user enters a command to run a program. The program prompts for a start vertex (1) and an end vertex (3). It then calculates the shortest path, displaying the path (1) --5--> (2) --3--> (4) --2--> (5) --1--> (3), the total weight (11), and the total number of comparisons (13). The terminal ends with a green cursor.

```
[jaideng]@linux2 ~/221_hw/A6> (12:43:56 04/23/14)
:: ./Main
Please enter the vertex you would like to start at
1
Please enter the vertex you would like to end at
3
Calculating path...
(1) --5--> (2) --3--> (4) --2--> (5) --1--> (3)
Total Weight For Shortest Path: 11
Total # of Comparisons: 13
[jaideng]@linux2 ~/221_hw/A6> (12:44:01 04/23/14)
:: █
```

Confirmation: