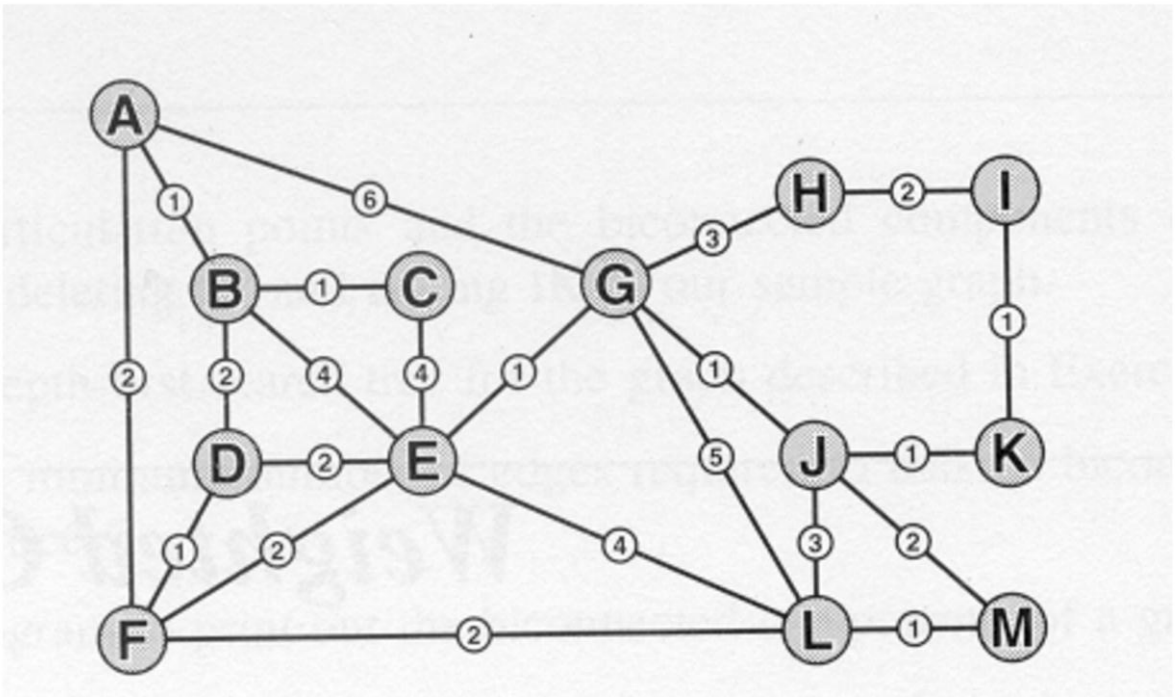
Algorithms and Data Structures Assignment

Fionn Brien C23353086

1. Introduction

This is my report on the Graph Traversal, MST & SPT Algorithm Assignment.

I was given the graph below to traverse with Depth-First Search, Breath-First Search, Prim's MST and Dijkstra's SPT.



Below is the text file version of the provided Graph. The first row tells us that there are 13 edges and 22 vertices in the graph. The rest of the rows tell us where the edges are (e.g. between A and B on row 2) and their weight (e.g. 1 on row 2).

File Edit View

```
13 22
1 2 1
1 6 2
1 7 6
2 3 1
2 4 2
2 5 4
3 5 4
4 5 2
4 6 1
5 6 2
5 7 1
5 12 4
6 12 2
7 8 3
7 10 1
7 12 5
8 9 2
9 11 1
10 11 1
10 12 3
10 13 2
12 13 1
```

The provided skeleton code GraphLists.java had areas for me to complete. In the Heap class, I added the siftUp() and siftDown() methods that I had implemented in a previous lab.

```java
public void siftUp( int k)
{
    int v = a[k];

    // code yourself
    // must use hPos[] and dist[] arrays
    while (k > 1 && dist[v] < dist[a[k / 2]]) {
        a[k] = a[k / 2];
        hPos[a[k]] = k;
        k = k / 2;
    }
    a[k] = v;
    hPos[v] = k;
}
```

```
public void siftDown( int k)
{
    int v, j;

    v = a[k];

    // code yourself
    // must use hPos[] and dist[] arrays

    while (2 * k <= N) {
        j = 2 * k;
        if (j < N && dist[a[j + 1]] < dist[a[j]]) j++;
        if (dist[v] <= dist[a[j]]) break;
        a[k] = a[j];
        hPos[a[k]] = k;
        k = j;
    }
    a[k] = v;
    hPos[v] = k;
}
```

In the Graph class, I needed to write code to insert edges into the Adjacency Matrix.

```
// write code to put edge into adjacency matrix

// insert node v into u's adjacency list
 t = new Node();
 t.vert = v;
 t.wgt = wgt;
 t.next = adj[u];
 adj[u] = t;

 // insert node u into v's adjacency list (since the graph is undirected)
 t = new Node();
 t.vert = u;
 t.wgt = wgt;
 t.next = adj[v];
 adj[v] = t;
```

Then I implemented methods for Depth-First Search, Breath-First Search, Prim's MST and Dijkstra's SPT.

2. Adjacency List Diagram

```
adj[A] -> |G | 6| -> |F | 2| -> |B | 1| ->
adj[B] -> |E | 4| -> |D | 2| -> |C | 1| -> |A | 1| ->
adj[C] -> |E | 4| -> |B | 1| ->
adj[D] -> |F | 1| -> |E | 2| -> |B | 2| ->
adj[E] -> |L | 4| -> |G | 1| -> |F | 2| -> |D | 2| -> |C | 4| -> |B | 4| ->
adj[F] -> |L | 2| -> |E | 2| -> |D | 1| -> |A | 2| ->
adj[G] -> |L | 5| -> |J | 1| -> |H | 3| -> |E | 1| -> |A | 6| ->
adj[H] -> |I | 2| -> |G | 3| ->
adj[I] -> |K | 1| -> |H | 2| ->
adj[J] -> |M | 2| -> |L | 3| -> |K | 1| -> |G | 1| ->
adj[K] -> |J | 1| -> |I | 1| ->
adj[L] -> |M | 1| -> |J | 3| -> |G | 5| -> |F | 2| -> |E | 4| ->
adj[M] -> |L | 1| -> |J | 2| ->
```

This is the Adjacency List Diagram created upon reading wGraphs1.txt and putting the edges into the Adjacency Matrix.

3. MST using Prim's

Start at L

Heap contents:

L(0)

dist[]:

A= Max B= Max C= Max D= Max E= Max F= Max G= Max H= Max I= Max J= Max K= Max L=0 M= Max

parent[]:

A<- Null B<- Null C<- Null D<- Null E<- Null F<- Null G<- Null H<- Null I<- Null J<- Null K<- Null L<- Null M<- Null

L

Step 1: Go to M

Heap contents:

E(4) F(2) G(5) J(3) L(0) M(1)

dist[]:

A= Max B= Max C= Max D= Max E=4 F=2 G=5 H= Max I= Max J=3 K= Max L=0 M=1

parent[]:

A<- Null B<- Null C<- Null D<- Null E<-L F<-L G<-L H<- Null I<- Null J<-L K<- Null L<- Null M<-L

M

Step 2: Go to F

Heap contents:

E(4) F(2) G(5) J(2) L(0)

dist[]:

A= Max B= Max C= Max D= Max E=4 F=2 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<- Null B<- Null C<- Null D<- Null E<-L F<-L G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

F

Step 3: Go to D

Heap contents:

A(2) D(1) E(2) G(5) J(2) L(0)

dist[]:

A=2 B= Max C= Max D=1 E=2 F=2 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-F B<- Null C<- Null D<-F E<-F F<-L G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

D

Step 4: Go to A

Heap contents:

A(2) B(2) E(2) F(1) G(5) J(2) L(0)

dist[]:

A=2 B=2 C= Max D=1 E=2 F=1 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-F B<-D C<- Null D<-F E<-F F<-D G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

F

Step 5: Go to B

Heap contents:

A(2) B(2) E(2) G(5) J(2) L(0)

dist[]:

A=2 B=2 C= Max D=1 E=2 F=1 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-F B<-D C<- Null D<-F E<-F F<-D G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

A

Step 6: Go to C

Heap contents:

B(1) E(2) G(5) J(2) L(0)

dist[]:

A=2 B=1 C= Max D=1 E=2 F=1 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-F B<-A C<- Null D<-F E<-F F<-D G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

B

Step 7: Go to J

Heap contents:

A(1) C(1) E(2) G(5) J(2) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=2 F=1 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-F F<-D G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

C

Step 8: Go to K

Heap contents:

A(1) E(2) G(5) J(2) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=2 F=1 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-F F<-D G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

A

Step 9: Go to G

Heap contents:

E(2) G(5) J(2) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=2 F=1 G=5 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-F F<-D G<-L H<- Null I<- Null J<-M K<- Null L<- Null M<-L

E

Step 10: Go to I

Heap contents:

G(1) J(2) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=2 F=1 G=1 H= Max I= Max J=2 K= Max L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-F F<-D G<-E H<- Null I<- Null J<-M K<- Null L<- Null M<-L

G

Step 11: Go to E

Heap contents:

E(1) H(3) J(1) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=1 F=1 G=1 H=3 I= Max J=1 K= Max L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-G F<-D G<-E H<-G I<- Null J<-G K<- Null L<- Null M<-L

J

Heap contents:

E(1) H(3) K(1) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=1 F=1 G=1 H=3 I= Max J=1 K=1 L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-G F<-D G<-E H<-G I<- Null J<-G K<-J L<- Null M<-L

E

Heap contents:

H(3) K(1) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=1 F=1 G=1 H=3 I= Max J=1 K=1 L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-G F<-D G<-E H<-G I<- Null J<-G K<-J L<- Null M<-L

K

Heap contents:

H(3) I(1) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=1 F=1 G=1 H=3 I=1 J=1 K=1 L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-G F<-D G<-E H<-G I<-K J<-G K<-J L<- Null M<-L

I

Step 12: Go to H

Heap contents:

H(2) L(0)

dist[]:

A=1 B=1 C=1 D=1 E=1 F=1 G=1 H=2 I=1 J=1 K=1 L=0 M=1

parent[]:

A<-B B<-A C<-B D<-F E<-G F<-D G<-E H<-I I<-K J<-G K<-J L<- Null M<-L

H Heap contents:

H(2) L(0)

4. SPT using Dijkstra's

Start at L:

Heap contents:

L(0)

dist[]:

A= Max B= Max C= Max D= Max E= Max F= Max G= Max H= Max I= Max J= Max K= Max L=0 M= Max

parent[]:

A<- Null B<- Null C<- Null D<- Null E<- Null F<- Null G<- Null H<- Null I<- Null J<- Null K<- Null L<- Null M<- Null

Step 1:

Heap contents:

E(4) F(2) G(5) J(3) M(1)

dist[]:

A= Max B= Max C= Max D= Max E=4 F=2 G=5 H= Max I= Max J=3 K= Max L=0 M=1

parent[]:

A<- Null B<- Null C<- Null D<- Null E<-L F<-L G<-L H<- Null I<- Null J<-L K<- Null L<- Null M<-L

Step 2:

Heap contents:

E(4) F(2) G(5) J(3)

dist[]:

A= Max B= Max C= Max D= Max E=4 F=2 G=5 H= Max I= Max J=3 K= Max L=0 M=1

parent[]:

A<- Null B<- Null C<- Null D<- Null E<-L F<-L G<-L H<- Null I<- Null J<-L K<- Null L<- Null M<-L

Step 3:

Heap contents:

E(4) G(5) J(3)

dist[]:

A= Max B= Max C= Max D= Max E=4 F=2 G=5 H= Max I= Max J=3 K= Max L=0 M=1

parent[]:

A<- Null B<- Null C<- Null D<- Null E<-L F<-L G<-L H<- Null I<- Null J<-L K<- Null L<- Null M<-L

Step 4:

Heap contents:

A(4) D(3) E(4) G(5)

dist[]:

A=4 B= Max C= Max D=3 E=4 F=2 G=5 H= Max I= Max J=3 K= Max L=0 M=1

parent[]:

A<-F B<- Null C<- Null D<-F E<-L F<-L G<-L H<- Null I<- Null J<-L K<- Null L<- Null M<-L

Step 5:

Heap contents:

A(4) E(4) G(4) K(4)

dist[]:

A=4 B= Max C= Max D=3 E=4 F=2 G=4 H= Max I= Max J=3 K=4 L=0 M=1

parent[]:

A<-F B<- Null C<- Null D<-F E<-L F<-L G<-J H<- Null I<- Null J<-L K<-J L<- Null M<-L

Step 6:

Heap contents:

A(4) B(5) E(4) G(4)

dist[]:

A=4 B=5 C= Max D=3 E=4 F=2 G=4 H= Max I= Max J=3 K=4 L=0 M=1

parent[]:

A<-F B<-D C<- Null D<-F E<-L F<-L G<-J H<- Null I<- Null J<-L K<-J L<- Null M<-L

Step 7:

Heap contents:

B(5) E(4) G(4) I(5)

dist[]:

A=4 B=5 C= Max D=3 E=4 F=2 G=4 H= Max I=5 J=3 K=4 L=0 M=1

parent[]:

A<-F B<-D C<- Null D<-F E<-L F<-L G<-J H<- Null I<-K J<-L K<-J L<- Null M<-L

Step 8:

Heap contents:

B(5) G(4) I(5)

dist[]:

A=4 B=5 C= Max D=3 E=4 F=2 G=4 H= Max I=5 J=3 K=4 L=0 M=1

parent[]:

A<-F B<-D C<- Null D<-F E<-L F<-L G<-J H<- Null I<-K J<-L K<-J L<- Null M<-L

Step 9:

Heap contents:

B(5) C(8) I(5)

dist[]:

A=4 B=5 C=8 D=3 E=4 F=2 G=4 H= Max I=5 J=3 K=4 L=0 M=1

parent[]:

A<-F B<-D C<-E D<-F E<-L F<-L G<-J H<- Null I<-K J<-L K<-J L<- Null M<-L

Step 10:

Heap contents:

B(5) C(8) H(7)

dist[]:

A=4 B=5 C=8 D=3 E=4 F=2 G=4 H=7 I=5 J=3 K=4 L=0 M=1

parent[]:

A<-F B<-D C<-E D<-F E<-L F<-L G<-J H<-G I<-K J<-L K<-J L<- Null M<-L

Step 11:

Heap contents:

C(8) H(7)

dist[]:

A=4 B=5 C=8 D=3 E=4 F=2 G=4 H=7 I=5 J=3 K=4 L=0 M=1

parent[]:

A<-F B<-D C<-E D<-F E<-L F<-L G<-J H<-G I<-K J<-L K<-J L<- Null M<-L

Step 12:

Heap contents:

H(7)

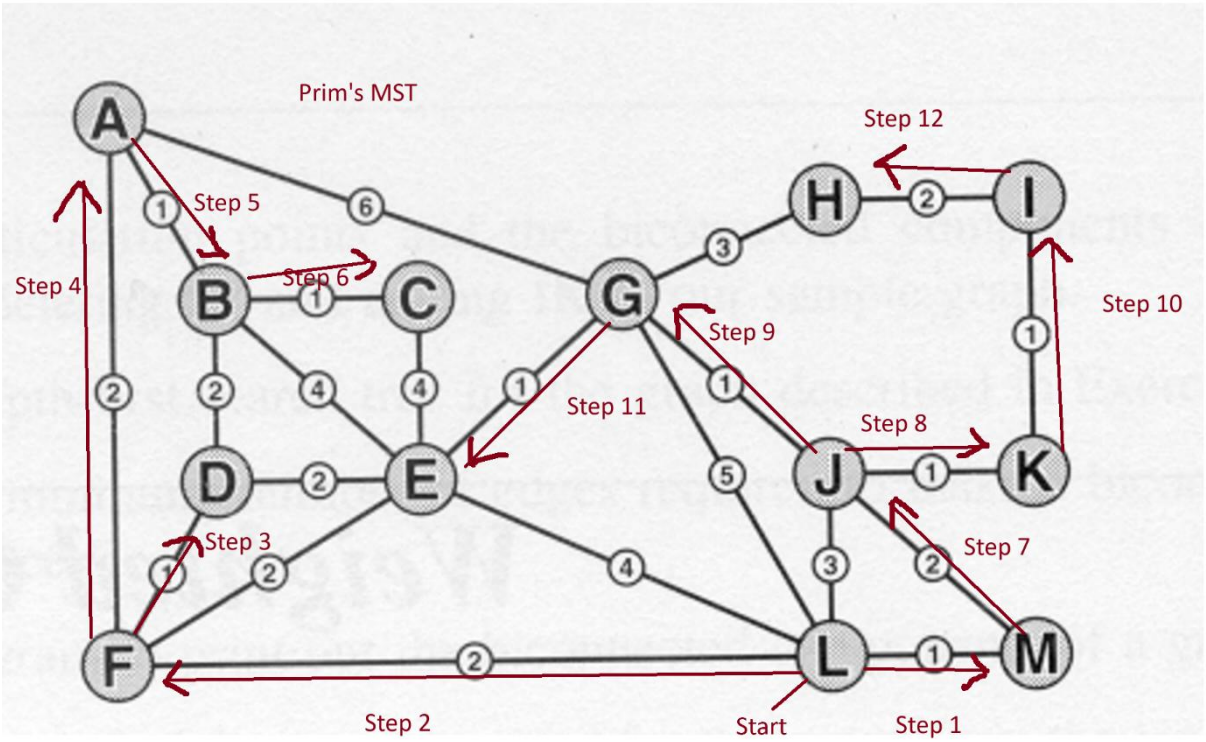
dist[]:

A=4 B=5 C=6 D=3 E=4 F=2 G=4 H=7 I=5 J=3 K=4 L=0 M=1

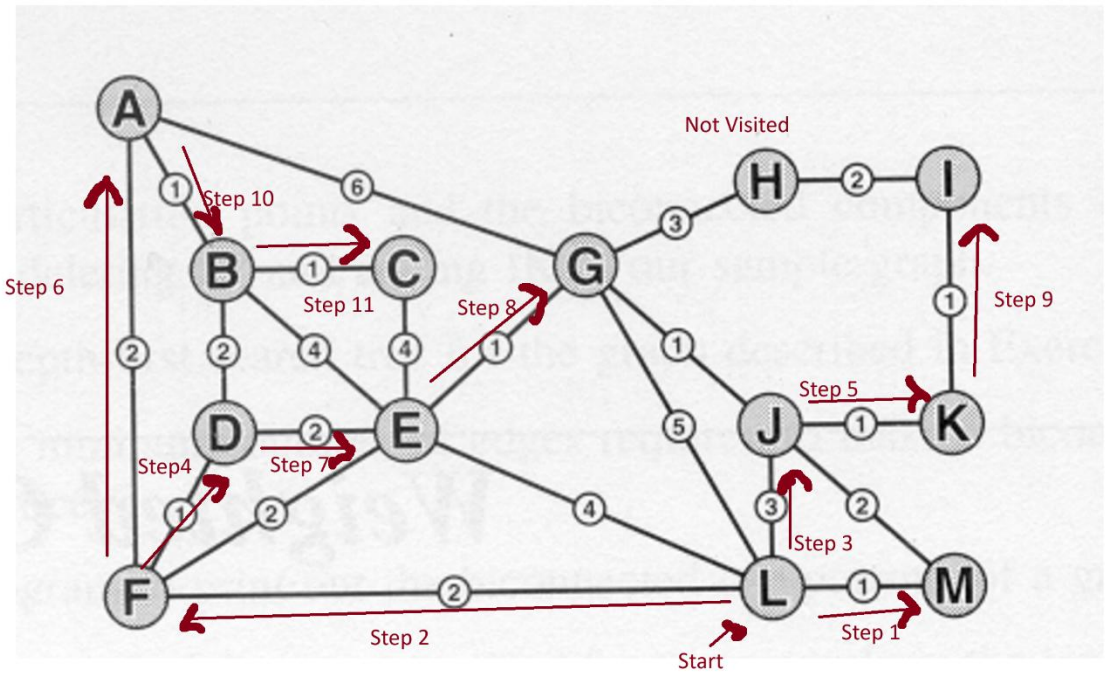
parent[]:

A<-F B<-D C<-B D<-F E<-L F<-L G<-J H<-G I<-K J<-L K<-J L<- Null M<-L

5. MST Diagram



6. SPT Diagram



7. Screen captures of output

User inputs file name and the starting vertex. The program reads the first line of the txt, which lets it know the total number of vertices and edges. Then program reads the edges from the file and displays the edges with their weights.

```
Enter the filename: wGraph1.txt
Enter the starting vertex: 12
Parts[] = 13 22
Reading edges from text file
```

```
Reading edges from text file
Edge A--(1)--B
Edge A--(2)--F
Edge A--(6)--G
Edge B--(1)--C
Edge B--(2)--D
Edge B--(4)--E
Edge C--(4)--E
Edge D--(2)--E
Edge D--(1)--F
Edge E--(2)--F
Edge E--(1)--G
Edge E--(4)--L
Edge F--(2)--L
Edge G--(3)--H
Edge G--(1)--J
Edge G--(5)--L
Edge H--(2)--I
Edge I--(1)--K
Edge J--(1)--K
Edge J--(3)--L
Edge J--(2)--M
Edge L--(1)--M
```

The program displays the Depth First and Breath First traversal of the graph.

```
Depth-first
B E L M J K I H G A F D C
Breadth-first
B E D C A L G F M J H K I
```

Prim's MST

```
Prim's MST
L M F D F A B C A E G J E K I H
MST total weight: 19


Minimum Spanning tree parent array is:
A -> B
B -> A
C -> B
D -> F
E -> G
F -> D
G -> E
H -> I
I -> K
J -> G
K -> J
L -> @
M -> L
```

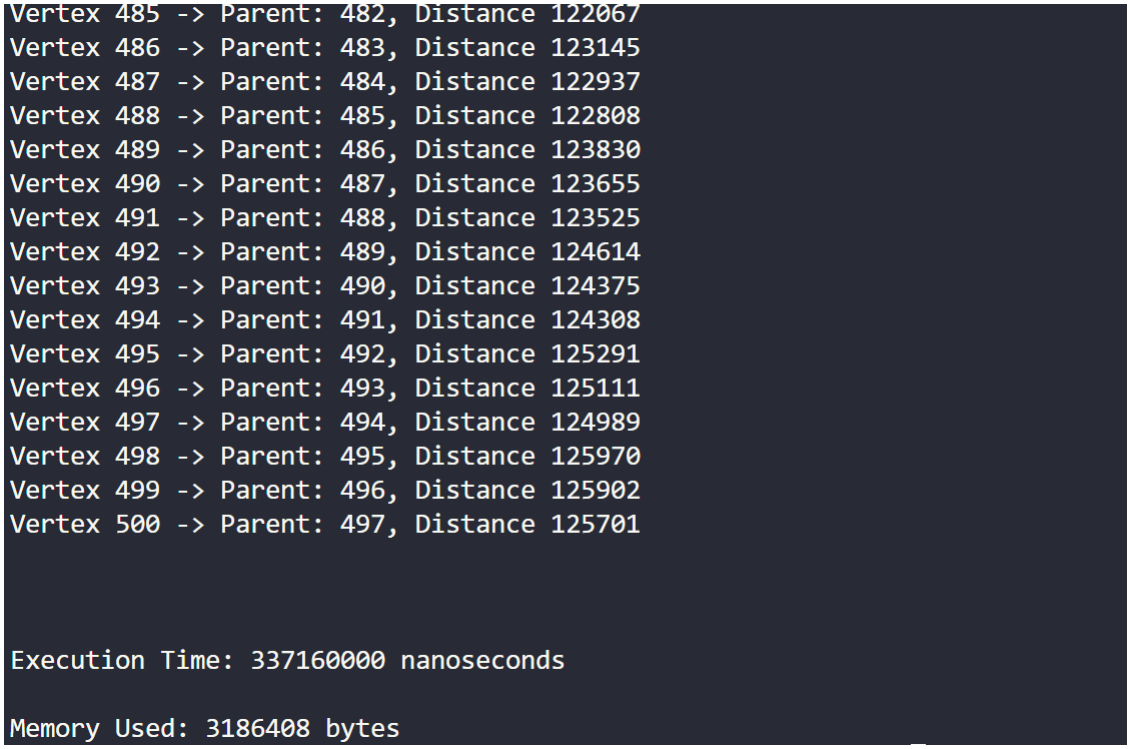
Dijkstra's SPT

```
Dijkstra's SPT

L L M F J D K A E G I B C

Vertex A -> Parent: F, Distance 4
Vertex B -> Parent: D, Distance 5
Vertex C -> Parent: B, Distance 6
Vertex D -> Parent: F, Distance 3
Vertex E -> Parent: L, Distance 4
Vertex F -> Parent: L, Distance 2
Vertex G -> Parent: J, Distance 4
Vertex H -> Parent: G, Distance 7
Vertex I -> Parent: K, Distance 5
Vertex J -> Parent: L, Distance 3
Vertex K -> Parent: J, Distance 4
Vertex L -> Parent: @, Distance 0
Vertex M -> Parent: L, Distance 1
```

8. Challenging World Graph

For a large real-world graph, I found a graph of roads in Rome online. I created a separate java file and copied the Heap class and the Graph class with only Dijkstra's SPT to traverse the graph. As the graph is so massive, I removed any used of the toChar function as it would just run out of

characters after the first handful. I added the nanoTime() System method before and after Dijkstra's to note the time it took for the graph to be traversed. It returns 337160000 nanoseconds or 0.33716 seconds. I also added totalMemory() and freeMemory() runtime methods to get the memory used. It returned 3186408 btyes or 3.186408 mb.



9. Reflection

-I reinforced what I had already learned about graph traversal in the previous labs.

-I learned about the real world applications for graph traversal algorithms i.e. traversal of road maps.

-I discovered websites such as http://snap.stanford.edu/data/index.html, where I can get large real world graphs if I ever need one.

-I got a better understanding of Prim's MST and Dijkstra's SPT when I had to break them down step by step and see what the contents of the heap, parent array and dist array were at each step of the process.

-I learned how to find out memory usage and execution time with built in methods in Java.

-I got lots of practice at debugging code.