EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

Maximum possible marks: 200 Points

Due date:
11:59pm, Monday 05/04/2020 for Tuesday labs.
11:59pm, Wednesday 05/06/2020 for Thursday labs.

Purpose: The overall purpose of this two-part lab is to implement disjoint sets and graphs in C++. The lab also involves an application aspect.

Part A: Disjoint Sets
Points: 100

Purpose:
The purpose of Part A is to implement disjoint sets.

General Requirements:
Implement the disjoint set data structure by using the parent-pointer representation as explained in the lecture videos. The size of the disjoint set will be fixed once it is constructed.

Your data structure should use an array of pointers to point to the nodes containing the set elements. Each node should have a pointer that points to its parent. The representative element of the set should point to itself. **This data structure should not allow duplicate numbers**.

For this lab, you should build the data structure using the samples which are in the data.txt file. After that, your program should have a simple menu like this:

Please choose one of the following commands:
1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

Disjoint Set operations:
The disjoint set methods should be implemented as follows:

- MakeSet(): In general, this function constructs n disjoint sets from n different elements. In the case of this lab assignment, the data.txt file contains 8 elements, and this function should create 8 disjoint sets with one element each. Each element's value will be equal to its pointer array index. Consider Fig. 1 which shows a disjoint set of size 8. The values of each disjoint set should be read from data.txt.

- Union(X,Y): Merges the disjoint set which contains the element X and the disjoint set which contains Y into a single set by using the union by rank heuristic. For example, refer to Figs. 2 through 7. Every time this function is called on two disjoint sets, print the elements which have been merged as well as the representative element of the merged sets. For an example, refer to the second output illustration.
- Find(k): Finds an element k and prints the representative element of the set in which the element was found. If k itself is a parent, then it should print the value of itself.
- PathCompression(k): In general, this function should carry out the path compression operation on the relevant disjoint set after each find operation, implicitly, as it has been shown in the lecture videos. Here the path compression is done explicitly on the path which contains k. For example, refer to Fig. 8.
- PrintPath(k): This function should print the path of the element from the k^{th} element to the representative element.

Data file:
data.txt: 1 2 3 4 5 6 7 8

Make sure that the order of operations is the same as the one shown in the illustrations below.

MakeSet()

Fig. 1

Union(1,2)

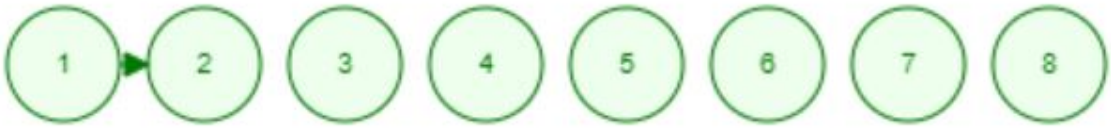


Fig. 2

Union(3,4)

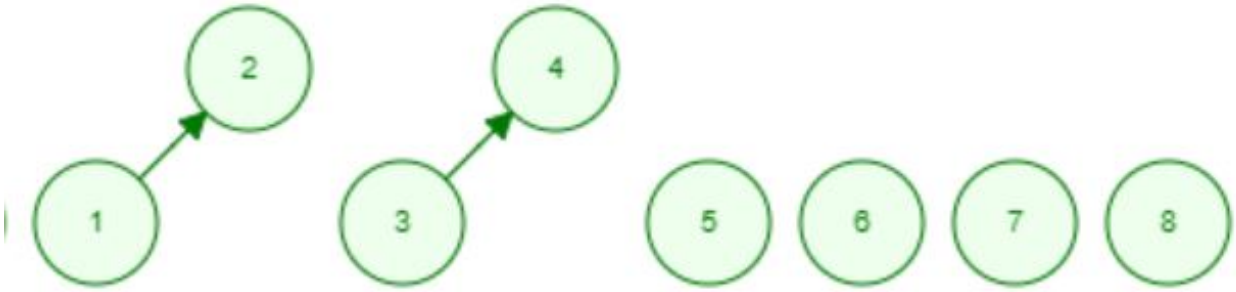


Fig. 3

Union(5,6)

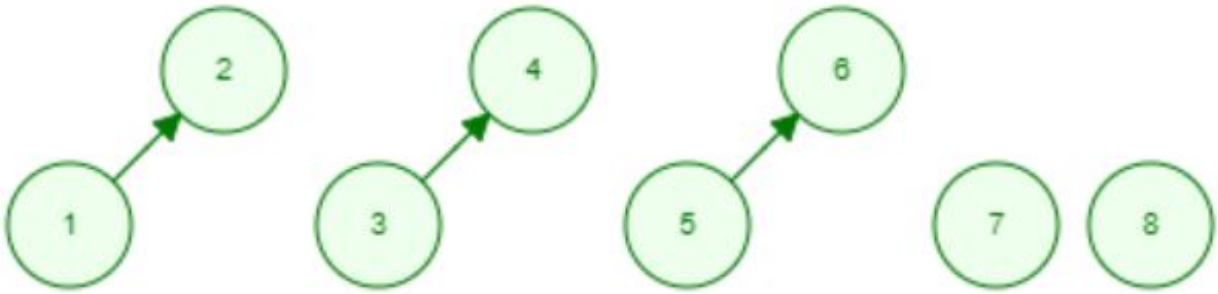


Fig. 4

Union(7,8)

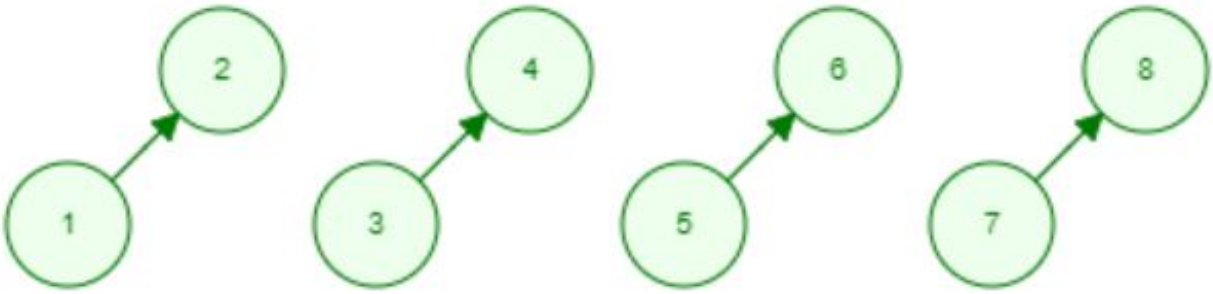


Fig. 5

Union(2,4)

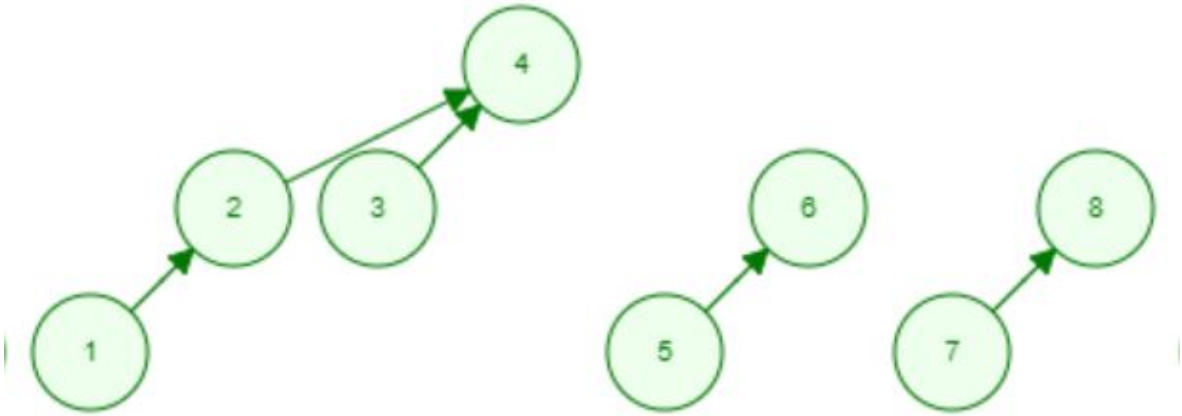


Fig. 6

Union(2,5) and then Union(6,8)

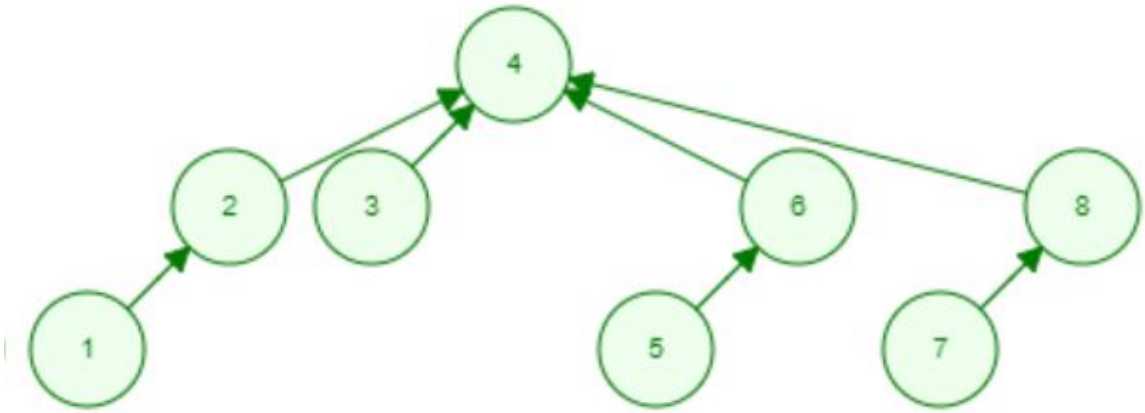


Fig. 7

PathCompression(7)

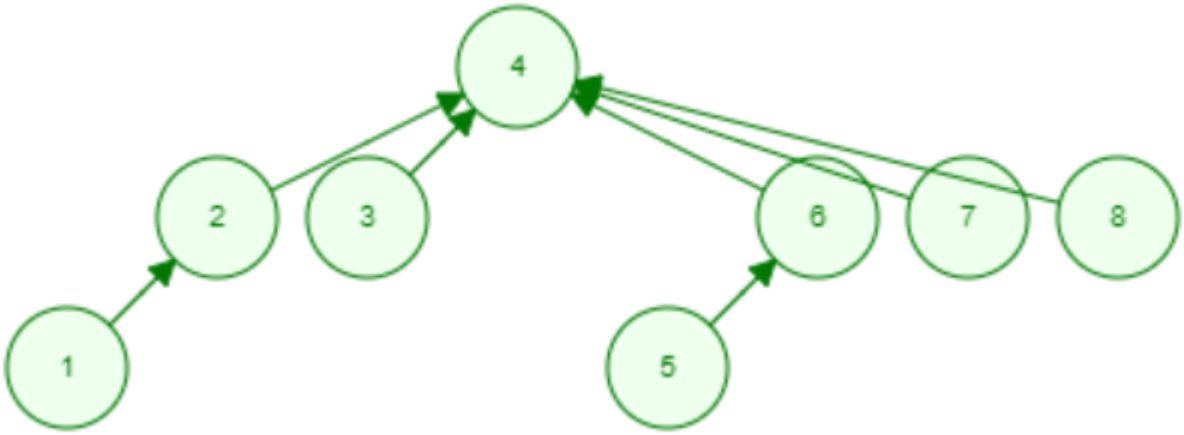


Fig. 8

Expected output for the Disjoint Set:

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>1
>Output: The disjoint sets have been constructed.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>1
>2
>Output: 1 and 2 have been merged. The representative element is 1.

EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

---

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>3
>4
>Output: Union on 3 and 4 has been done. The representative element is 3.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>5
>6
>Output: Union on 5 and 6 has been done. The representative element is 5.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>7
>8

>Output: Union on 7 and 8 has been done. The representative element is 7.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>2
>4
>Output: Union on 2 and 4 has been done. The representative element is 1.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>2
>5
>Output: Union on 2 and 5 has been done. The representative element is 1.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>2
>Output: Enter the representative elements for the two sets which you wish to union:
>6

>8
>Output: Union on 6 and 8 has been done. The representative element is 1.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>3
>Output: Enter the element you want to find:
>0
>Output: Sorry! 0 is not found!

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>3
>Output: Enter the element you want to find:
>6
>Output: Element 6 has been found successfully. The representative element is 4.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>4
>Output: Enter the element on whose set you would want to perform path compression
>6
>Output: Path compression has been done successfully.

EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

---

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>3
>Output: Enter the element you want to search:
>1
>Output: Element 1 has been found successfully. The representative element is 4.

Please choose one of the following commands:

1- MakeSet
2- Union
3- Find
4- PathCompression
5- PrintPath

>Enter your choice:
>5
>Output: Enter the element you want to find the path for:
>2
>Output: The path for the element 2 is: 2->1.

Part B: Graphs
Points: 100

Purpose:
The purpose of Part B of this lab is to implement a graph data structure which you will use to help a civil engineer build connections or bridges between the Isla Nublar islands in Costa Rica. The civil engineer has limited resources available to him, and he should not ask the owner of Jurassic Park, John Hammond, for additional resources. Though John Hammond says, “Spare no expense”, he has spent more than he should have, and he is currently out of money. So, the engineer should use the materials effectively and efficiently to construct bridges between the islands and minimize the cost of construction. The shorter the path, the more money that is saved. Do you think you can help him connect the islands with minimum cost and distance?

EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

General Requirements:

For this part of the lab, you are required to implement a graph data structure using an adjacency matrix and then implement Kruskal's and Prim's algorithm to calculate the minimum spanning tree (MST). At the same time, you will implement depth-first search and breadth-first search traversals on the graph data structure. These implementations will help an engineer build the Jurassic Park island chain system.

Island connectivity system:

The map of the islands is given to you and specifies the distances between each pair of islands in the chain. The island connectivity system needs to build an appropriate length of bridge based on the cost required to go from one area to another. You need to construct a connectivity map with the help of Kruskal's and Prim's algorithms.

Points to remember:

1. The distance between the islands is in miles.
2. The cost per mile is 250K dollars.

Operations on a graph:

Implement the following operations for the graph data structure:

- **Buildgraph(costs, n)** - should build a graph, it accepts two arguments – a square 2D array of integer costs representing the cost of each edge representing the distance between two islands, and n, the number of islands in the graph.
- **BFS()** - the method performs a breadth-first search on the island graph. It should return an array of size 2 for which each entry contains a pointer that points to an array. The first array should contain the tree edges of the island graph. The second array should contain the cross edges.
- **DFS()** - the method performs a depth-first search on the graph. It should return an array of size 2 for which each entry contains a pointer that points to an array. The first array should contain the tree edges of the island graph. The second array should contain the back edges.
- **Kruskal MST()** - implement Kruskal's algorithm that computes a MST for a given island graph. At every step of adding a new edge, you will need to ensure that no cycle is created. You should use the disjoint set data structure and a priority queue for cycle detection and obtaining a minimum cost edge. The algorithm should return an array of edges connecting the islands and the costs of the corresponding edges. The returned array will represent the MST which is the total length of the bridge path. This function should also return the estimated cost for constructing the bridge in dollars. Example: If the estimated cost is 250,000, it should return as 250,000 $ or 250K $.
- **Prim MST()** - implement Prim's algorithm that computes a MST for a given graph. You should use a priority queue for finding a minimum cost edge. The algorithm should return an array of edges connecting the islands and the costs of the corresponding edges. The returned array will represent the MST which is the total length of the bridge

path. This function should also return the estimated cost for constructing the bridge in dollars.

You should add the following options to the menu you developed in Part A of the lab:

Please choose one of the following commands:
6- BFS
7- DFS
8- Kruskal MST
9- Prim MST
10- Exit

Data file:
data1.txt:

Tyrannosaurus Rex, 1
Velociraptors, 2
Indominous Rex, 3
Mososaurus, 4
Spinosaurus, 5

n,5
1 6 8 12 9
6 1 7 -1 13
8 7 1 10 -1
12 -1 10 1 8
9 13 -1 8 1

The line n,5 specifies the number of islands (i.e., n) to be connected. From the second line to the sixth line, the numbers in the data file represent the values of the distances between pairs of islands which are to be stored in the weight matrix. The (i, i) entries are always 1 to simply denote that it is possible to travel from the island to itself. (Note this does not mean that the distance from the island to itself is 1.) Whenever the (i, j) entry is -1 there is no direct connection possible. The other (i,j) values, such as 6, 8, 12, and 9 represent the distances between the i^{th} island and the other islands whenever there is a direct connection possible. The values for the distances between pairs of islands are to be stored in the weight matrix.

A weighted graph represented as a matrix as shown above can be used to represent the distances between the islands. Your weight matrix should be designed only with the island_number field. You should not use the island_name in designing the weighted graph matrix. You need to extract the island_number from the node in this process. The number of rows and number of columns in the weight matrix depends on the number of islands i.e., n value.

EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

Each island will be recognised with two fields: the name of the island and the number associated with it. So, in your code, for each island you need to create a node. Each node represents the island with two fields – island_name(type: String) and island_number(type: integer).

The following diagram demonstrates the graph for data1.txt:

The circles represent the island names (Tyrannosaurus Rex, Velociraptors, Indominous Rex, Mososaurus, and Spinosaurus), and the square boxes represent the distance between each pair of two dinosaur islands.

Here is the representation of the islands:

Island #Number – island_name, island_number

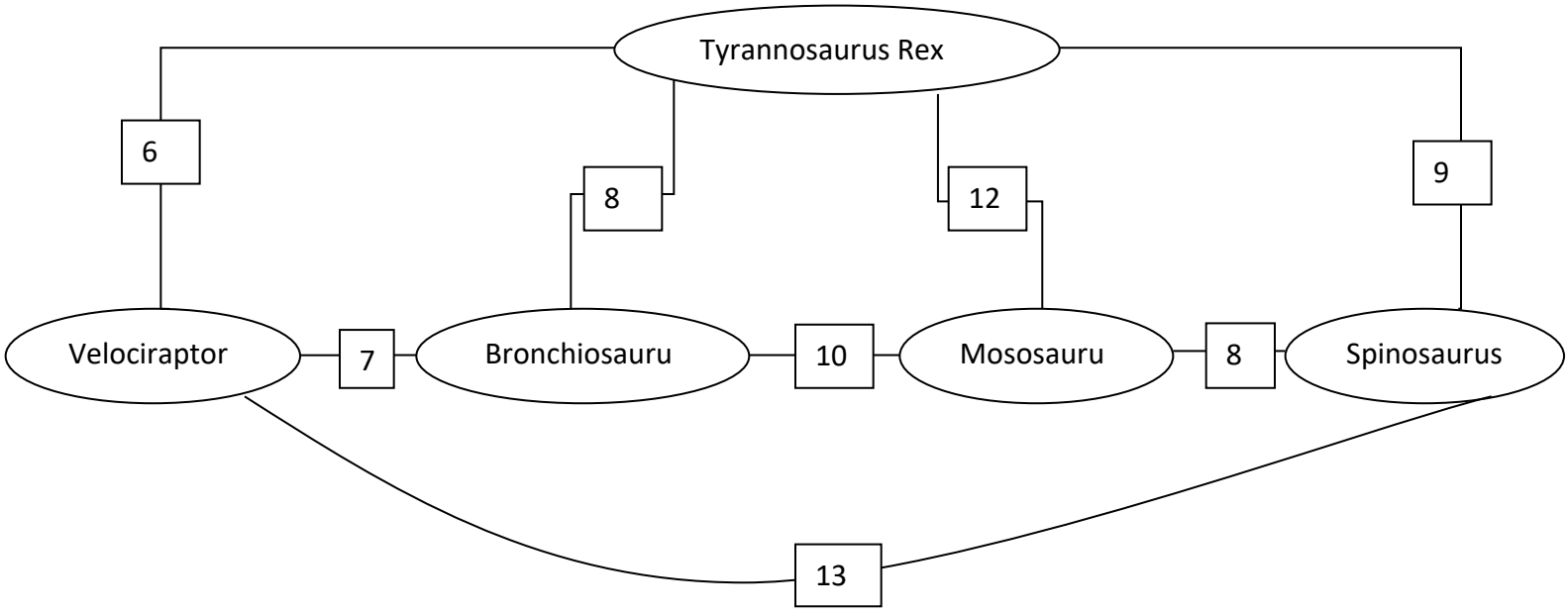
Island #1 – Tyrannosaurus Rex, 1

Island #2 – Velociraptors, 2

Island #3 – Indominous Rex, 3

Island #4 – Mososaurus, 4

Island #5 – Spinosaurus, 5



EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

Expected output

Please choose one of the following commands:

6- BFS
7- DFS
8- Kruskal MST
9- Prim MST
10- Exit

>6

Tree Edges: (Tyrannosaurus Rex, Velociraptors) (Tyrannosaurus Rex, Indominous Rex) (Tyrannosaurus Rex, Mososaurus) (Tyrannosaurus Rex, Spinosaurus)
Cross Edges: (Velociraptors, Indominous Rex) (Velociraptors, Spinosaurus) (Indominous Rex, Mososaurus) (Mososaurus, Spinosaurus)

Please choose one of the following commands:

6- BFS
7- DFS
8- Kruskal MST
9- Prim MST
10- Exit

>7

Tree Edges: (Tyrannosaurus Rex, Velociraptors) (Velociraptors, Indominous Rex) (Indominous Rex, Mososaurus) (Mososaurus, Spinosaurus)
Back Edges: (Tyrannosaurus Rex, Indominous Rex) (Tyrannosaurus Rex, Mososaurus) (Tyrannosaurus Rex, Spinosaurus) (Tyrannosaurus Rex, Spinosaurus)

Please choose one of the following commands:
6- BFS
7- DFS
8- Kruskal MST
9- Prim MST
10- Exit
>8
(Tyrannosaurus Rex, Velociraptors){6} (Velociraptors, Indominous Rex){7} (Mososaurus, Spinosaurus){8} (Tyrannosaurus Rex, Spinosaurus){9}
Total length of the route = 30 miles
Total estimate to construct the bridges in the route = 30*250K = 750K $
Note: () represents the edge between the two stops, and {} represents the distance

EECS 560 Lab 10 – Implementation of Disjoint Sets and Graphs

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

---

Please choose one of the following commands:

6- BFS
7- DFS
8- Kruskal MST
9- Prim MST
10- Exit

>9

(Tyrannosaurus Rex, Velociraptors){6} (Velociraptors, Indominous Rex){7} (Tyrannosaurus Rex, Spinosaurus){9} (Mososaurus, Spinosaurus){8}

Total length of the route = 30 miles

Total estimate to construct the bridges in the route = 30*250K = 750K $

Please choose one of the following commands:

6- BFS
7- DFS
8- Kruskal MST
9- Prim MST
10- Exit

>10

Bye!

Grading rubric:

- Full grade: The program should execute without any issues with all the options executed and with no memory leaks.
- Points will be taken off for execution errors, such as memory leaks, segmentation/program abort issues and missing handling of invalid cases.
- Programs that are compiled but do not execute will earn in the range of 0 to 50% of the possible points. Your grade will be determined based on the program design and the options implemented in the code.

Submission instructions:

- All files, i.e., the source files and Makefile, should be zipped in a folder.
- Include a ReadMe.txt if your code requires any special instructions to run.
- The naming convention of the folder should be LastName_Lab10.zip (or .tar or .rar or .gz).
- Email it to your respective lab instructor: chiranjeevi.pippalla@ku.edu (Chiru) or prashanthi.mallojula@ku.edu (Prashanthi) with subject line EECS 560 Lab10.
  Your program should compile and run on the Linux machines in **Eaton 1005D using g++.**