CALIFORNIA STATE UNIVERSITY, FRESNO
DEPARTMENT OF COMPUTER SCIENCE

Class:	**Algorithms & Data Structures**		Semester:	**Fall 2023**
Points		Document author:	**Pratham Aggarwal**	
		Author's email:	**Pratham_aggarwal@mail.fresnostate.edu**	
		Laboratory number:	**Lab 11**	

1. Statement of Objectives

The primary objective of this experiment was to develop and examine two fundamental graph algorithms for producing minimal spanning trees based on an input graph represented by an adjacency matrix, Prim's and Kruskal's. The scope included converting algorithmic guidelines into functional code. The significance was in understanding and using key graph theory principles to real-world challenges. The effective implementation of both methods, resulting in output matrices that reflect the original graph and the matching minimal spanning tree, is one of the significant accomplishments. This report includes the algorithmic implementations in detail, as well as the analysis of output matrices and the overall learning outcomes derived from the experiment.

2. Experimental Procedure

a. I began by implementing the **function printMatrix**, which prints a 2D matrix with 'INF' replaced with the matching value in the matrix.
b. **Function to Find the Lowest Key**: Implement a function called minKey that finds the index of the minimum key value in a vector while excluding vertices that are already in the MST.
c. **Prim's Minimal Spanning Tree Algorithm**: Use the primsMST function. Set up arrays for the parent, key values, and the MST inclusion status. Set the beginning vertex's key value to 0. Iterate through the vertices to determine the shortest tree. Then, using the printMatrix function, create the MST matrix and print the original graph and MST.
d. **Implement kruskalMST function**: Make a matrix to hold the minimum spanning tree and a vector to hold the edges. Keep all of the edges in the vector. Sort edges in non-descending weight order. For disjoint set tracking, create a parent array. To detect cycles and generate the MST matrix, use union-find.
e. **Main Function**: Enter the number of vertices (V) and the starting vertex index (startVertex). Enter the graph's adjacency matrix line by line. Run Prim's algorithm and print the minimal spanning tree with the primsMST function. Run Kruskal's algorithm and print the minimal spanning tree with the kruskalMST function.
   Produce the adjacency matrix and the minimal spanning tree matrices.
   Check for accuracy: Handling incorrect inputs, such as entering non-integer values or 'INF' for non-infinity entries.

3. Analysis

Prim's Algorithm Implementation: The Prim's algorithm successfully generates a minimal spanning tree by iteratively selecting the vertex with the lowest key value. The resulting output matrix shows the connectedness of the minimal spanning tree, with edges chosen to minimize total weight.

Kruskal's method Implementation: Kruskal's method, which focuses on sorting edges by weight and employs the union-find strategy, effectively builds a minimal spanning tree. Edges are sorted and union-find operations are performed with precision to avoid cycles. The output matrix reflects the smallest spanning tree configuration, proving the algorithm's validity.

Screenshot in the end of the report.

4. Encountered Problems

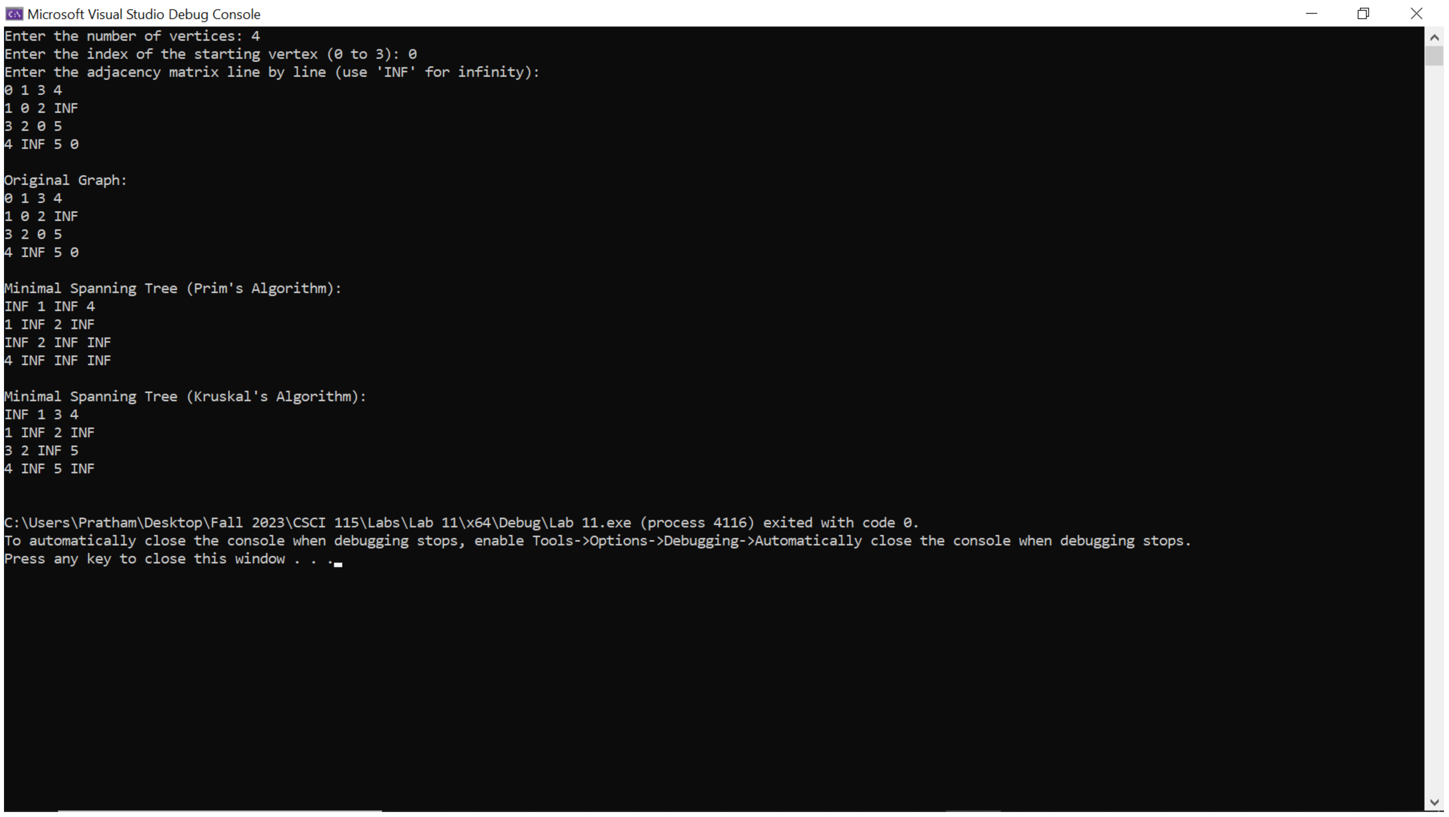
The use of predetermined functions for various operations was a considerable challenge. Some were already in the coding guidelines, but I had to find up how they were implemented. Some of the predefined functions appeared to be complicated, making it difficult to fully understand their functionality. The assignment's code guidelines encouraged a straightforward and clean implementation without the use of shortcuts. However, when looking up internet for some areas they were implemented in shorter way and it was kind of difficult to understand. I used external resources for reference to verify the accuracy and efficiency of the code, such as managing exceptions like the INF case. This required evaluating multiple online implementations of the algorithms, especially for complex parts or specified functions. In the end, I mostly faced problems with the implementation of certain areas but I was able to understand them by the help on online tools and programs.

5. Conclusions

In conclusion, this experiment achieved its objectives by providing the correct output and a better understanding of Prim's and Kruskal's algorithms. The executions of the code helped in understanding new ways of writing the same program. I learned many new predefined functions. Implementing both Prim's and Kruskal's algorithms allowed for a comparison, showing light on the various techniques employed by each. The lab demonstrated the broader importance of graph algorithms in problem-solving situations by providing a realistic application of theoretical concepts. Overall, the experience improved coding skills, algorithmic comprehension, and the application of theoretical knowledge to real-world problem solving.

6. References

N/A



Microsoft Visual Studio Debug Console

```
Enter the number of vertices: 4
Enter the index of the starting vertex (0 to 3): 0
Enter the adjacency matrix line by line (use 'INF' for infinity):
0 1 3 4
1 0 2 INF
3 2 0 5
4 INF 5 0

Original Graph:
0 1 3 4
1 0 2 INF
3 2 0 5
4 INF 5 0

Minimal Spanning Tree (Prim's Algorithm):
INF 1 INF 4
1 INF 2 INF
INF 2 INF INF
4 INF INF INF

Minimal Spanning Tree (Kruskal's Algorithm):
INF 1 3 4
1 INF 2 INF
3 2 INF 5
4 INF 5 INF


C:\Users\Pratham\Desktop\Fall 2023\CSCI 115\Labs\Lab 11\x64\Debug\Lab 11.exe (process 4116) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```