# DEPARTMENT OF MATHEMATICS AND COMPUTING

# LAB MANUAL FOR ADVANCED DATA STRUCTURES AND ALGORITHMS

# COURSE CODE: MCC536

**Department of Mathematics and Computing**
**Lab manual for Advanced Structures and Algorithms**
**Course Code: MCC536**
**M. Tech (Data Analytics)**

**NOTE:**The course on Advanced Data Structures and Algorithms is to provide the concepts on advanced data structures and algorithms used in practical life. The associated lab course is to implement all these data structures and algorithm using any programming language, preferably one among C/C++, Python, Java.

**Do's:**

1. Connect to the 172.16.64.12 server.
2. Login with your own username created.
3. Update lab file with every lab class.
4. Use the computer properly to keep it in good working order.

**Don't:**

1. Don't open the internet browser.
2. Don't change the computer settings.
3. Don't get out of your seat unless you are told to.

**Apparatus Required (Common to all):**

Computer with LINUX/Windows Operating system.

## List of Experiments:

| Sl. No. | Name of Experiment | No. of Lab Classes | Page No. |
|---|---|---|---|
| **1** | Implementation of randomized quicksort algorithm | **1** | **3** |
| **2** | Implementation of hash functions and associated algorithms | **1** | **3** |
| **3** | Implementation of operations on splay trees | **2** | **4** |
| **4** | Implementation of operations on Fibonacci heaps | **2** | **4** |
| **5** | Implementation of operations on binary heaps | **1** | **4** |
| **6** | Implementation on operations on B-Trees | **2** | **5** |
| **7** | Implementation of operations on partition ADT and union-find data structures | **1** | **5** |
| **8** | Implementation of Bellman-Ford algorithm | **1** | **5** |
| **9** | Implementation of Ford-Fulkerson algorithm | **1** | **6** |
| **10** | Implementation of Edmonds-Karp algorithm | **1** | **6** |
| | | | |

**Reference Books:**


1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to algorithms, PHI, 3rd Edition, 2010.
2. M. A. Weiss, Data Structures and Algorithm Analysis, Addison-Wesley, 2nd Edition, 2013V.
3. Aho, J. E. Hopcroft, J. D. Ullman, Data Structures and Algorithms, Addison-Wesley, 1st Edition, 1982
4. S. S. Skiena: The Algorithm Design Manual, Springer, 2nd Edition, 2008
5. J. Kleinberg, E. Tardos, Algorithm Design, Addison-Wesley, 1st Edition, 2005

| Experiment No-01 | |
|---|---|
| **Aim** | Implementation of randomized quicksort algorithm |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Enter the codes for randomized quicksort algorithm. The codes should calculate the time taken by randomized quicksort algorithm and it should also compare the same input string with respect to normal quicksort algorithm.<br>➢ Write the output. |

| Experiment No-02 | |
|---|---|
| **Aim** | Implementation of hash functions and its associated algorithms |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | Given a File of **N** employee records with a set **K** of Keys(4-digit) which uniquely determine the records in file **F**. Assume that file **F** is maintained in memory by a Hash Table(HT) of **m** memory locations with **L** as the set of memory addresses (2-digit) of locations in HT. Let the keys in **K** and addresses in **L** are Integers. Design and develop a Program in C that uses Hash function **H: K * L** as H(**K**)=**K** mod **m (remainder** method), and implement hashing technique to map a given key **K** to the address space **L.** Resolve the collision (if any) using **linear probing**. |

| Experiment No-03 | |
|---|---|
| **Aim** | Implementation of operations on splay trees |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Input the binary trees (will be used in other programs also)<br>➢ Convert the tree as a splay trees using the SPLAY function<br>➢ Use the **insert(), delete()** operation<br>➢ Compare with binary search tree for random inputs |

| Experiment No-04 | |
|---|---|
| **Aim** | Implementation of operations on B/B+-trees. |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Input the binary trees (will be used in other programs also)<br>➢ Convert the tree as a B+- trees<br>➢ Use the **insert(), delete()** operation<br>➢ Compare with binary search tree for random inputs |

| Experiment No-05 | |
|---|---|
| **Aim** | Implementation of operations on binomial heaps |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Input the binary trees/binomial trees satisfying the heap-property.<br>➢ Convert the tree as a binomial heap<br>➢ Use the **insert(), delete()** operations |

| | |
|---|---|
| | ➢ Compare with binary search tree for random inputs |

| **Experiment No-06** | |
|---|---|
| **Aim** | Implementation of operations on Fibonacci heaps |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Input the binary trees satisfying the heap-property.<br>➢ Convert the tree as a Fibonacci heap<br>➢ Use the **insert(), delete()** operations<br>➢ Compare with binary search tree for random inputs |

| **Experiment No-07** | |
|---|---|
| **Aim** | Implementation of disjoint set data structures |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Input the graph as a linked list<br>➢ Write codes for **Find(), Join()** operations<br>➢ Write codes to use **Find(), Join()** in minimum spanning tree algorithms |

| **Experiment No-08** | |
|---|---|
| **Aim** | Implementation of Bellman-Ford algorithm |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Write codes to input the graph as a linked list. Here node of the graph should also contain a field to store cost.<br>➢ Implement the Bellman-Ford Algorithm for solving the single-source shortest path algorithm<br>➢ Check the output for negative cost edges/negative cycles |

| Experiment No-09 | |
|---|---|
| **Aim** | Implementation of Ford-Fulkerson algorithm |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Write codes to input the network as a directed graph as a linked list.<br>➢ Write the codes to implement the Ford-Fulkerson algorithmfor solving the maximum flow problem in a network.<br>➢ Write codes to compute the minimum capacity of the cut in the network<br>➢ Compare the maximum flow with the capacity of the minimum cut. |

| Experiment No-10 | |
|---|---|
| **Aim** | Implementation of Edmonds-Karp algorithm |
| **Tools/Apparatus** | A computer with C/C++/Java/Python compiler |
| **Procedure** | ➢ Write codes to input the graph as a linked list.<br>➢ Write the codes to implement the Edmonds-Karp algorithm for solving the maximum matching problem in a graph. |