



**DELHI TECHNICAL CAMPUS
GREATER NOIDA**
Affiliated to GGSIPU and approved by AICTE & COA



PRACTICAL FILE

SESSION: 2023-24

Design and Analysis of Algorithm Lab

(CIC 359)

III Year, V Sem

Submitted to:

Name: Ms. Anshul

Designation: Assistant Professor

Submitted by:

Name: Adityan Verma

Enrollment No. 07818002721

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

INDEX

S.NO.	PROGRAM NAME	REMARKS	DATE OF EXPERIMENT	DATE OF SUBMISSION	SIGN.
1	To implement following algorithm using array as a data structure and analyse its time complexity. a. Merge sort b. Quick sort c. Bubble sort d. Selection sort e. Heap sort				
2	To implement Linear search and Binary search and analyse its time complexity.				
3	To implement Huffman Coding and analyse its time complexity.				
4	To implement Minimum Spanning Tree and analyse its time complexity.				
5	To implement Dijkstra's algorithm and analyse its time complexity.				
6	To implement Bellman Ford algorithm and analyse its time complexity.				
7	Implement N Queen's problem using Back Tracking.				
8	To implement Matrix Multiplication and analyse its time complexity.				
9	To implement Longest Common Subsequence problem and analyse its time complexity.				
10	To implement naive String-Matching algorithm, Rabin Karp algorithm and Knuth Morris Pratt algorithm and analyse its time complexity.				
11	To implement Sorting Network.				

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 1

Aim: To implement following algorithm using array as data structure & analyze its time complexity.

a) Merge Sort

Code: -

```
#include <iostream>
#include <ctime>

using namespace std;

void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    // Create temporary arrays
    int L[n1], R[n2];

    // Copy data to temporary arrays L[] and R[]
    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    // Merge the temporary arrays back into arr[l..r]
    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Copy the remaining elements of L[], if there are any
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}

// Copy the remaining elements of R[], if there are any
while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for large l and r
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        // Merge the sorted halves
        merge(arr, l, m, r);
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main()
{
    int n, i;

    cout << "Enter the number of elements you want to enter: ";
    cin >> n;
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
int a[n];
clock_t t1, t2;
double t;

cout << "Enter the elements:-" << endl;
for(i=0; i<n; i++) {
    cout << "a[" << i << "]: ";
    cin >> a[i];
}

cout << "\nArray before sort: ";
for(i=0; i<n; i++) {
    cout << a[i] << " ";
}

t1 = clock();

mergeSort(a, 0, n-1); // Pass Array in Sorting Function

cout << "\nArray after sort: ";
for(i=0; i<n; i++) {
    cout << a[i] << " ";
}

t2 = clock();
t = (double) (t2 - t1)/CLOCKS_PER_SEC;
cout << "\nTotal time taken by code: " << t << " seconds.";

return 0;
}
```

Name: Adityan Verma

Enrollment No. 07818002721

Output: -

```
Enter the number of elements you want to enter: 6
Enter the elements:-
a[0]: 95
a[1]: 48
a[2]: 56
a[3]: 13
a[4]: 48
a[5]: 79
Array before sort: 95 48 56 13 48 79
Array after sort: 13 48 48 56 79 95
Total time taken by code: 0.000006 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

b) Quick Sort

Code: -

```
#include <iostream>
#include <ctime>

using namespace std;

// Function to partition the array into two halves
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }

    swap(arr[i + 1], arr[high]);
    return (i + 1);
}

// Function to perform quicksort on the array
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // Find the partitioning index
        int pi = partition(arr, low, high);

        // Recursively sort the elements before and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main()
{
    int n, i;
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
cout << "Enter the number of elements you want to enter: ";
cin >> n;

int a[n];
clock_t t1, t2;
double t;

cout << "Enter the elements:-" << endl;
for(i=0; i<n; i++) {
    cout << "a[" << i << "]: ";
    cin >> a[i];
}

cout << "\nArray before sort: ";
for(i=0; i<n; i++) {
    cout << a[i] << " ";
}

t1 = clock();

quickSort(a, 0, n-1); // Pass Array in Sorting Function

cout << "\nArray after sort: ";
for(i=0; i<n; i++) {
    cout << a[i] << " ";
}

t2 = clock();
t = (double) (t2 - t1)/CLOCKS_PER_SEC;
cout << "\nTotal time taken by code: " << t << " seconds.";

return 0;
}
```


Name: Adityan Verma

Enrollment No. 07818002721

Output: -

```
Enter the number of elements you want to enter: 6
Enter the elements:-
a[0]: 95
a[1]: 48
a[2]: 56
a[3]: 13
a[4]: 67
a[5]: 79
Array before sort: 95 48 56 13 67 79
Array after sort: 13 48 56 67 79 95
Total time taken by code: 0.000007 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

c) Bubble Sort

Code: -

```
#include <iostream>
```

```
#include <ctime>
```

```
using namespace std;
```

```
void bubbleSort(int arr[], int n) {  
    for (int i = 0; i < n - 1; i++) {  
        for (int j = 0; j < n - i - 1; j++) {  
            // Swap if the element found is greater than the next element  
            if (arr[j] > arr[j + 1]) {  
                swap(arr[j], arr[j + 1]);  
            }  
        }  
    }  
}
```

```
int main()  
{  
    int n, i;  
  
    cout << "Enter the number of elements you want to enter: ";  
    cin >> n;  
  
    int a[n];  
    clock_t t1, t2;  
    double t;  
  
    cout << "Enter the elements:-" << endl;  
    for(i=0; i<n; i++) {  
        cout << "a[" << i << "]: ";  
        cin >> a[i];  
    }  
  
    cout << "\nArray before sort: ";  
    for(i=0; i<n; i++) {  
        cout << a[i] << " ";  
    }  
}
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
}  
  
t1 = clock();  
  
bubbleSort(a, n); // Pass Array in Sorting Function  
  
cout << "\nArray after sort: ";  
for(i=0; i<n; i++) {  
    cout << a[i] << " ";  
}  
  
t2 = clock();  
t = (double) (t2 - t1)/CLOCKS_PER_SEC;  
cout << "\nTotal time taken by code: " << t << " seconds."  
  
return 0;  
}
```

Output: -

```
Enter the number of elements you want to enter: 6  
Enter the elements:-  
a[0]: 95  
a[1]: 48  
a[2]: 56  
a[3]: 13  
a[4]: 67  
a[5]: 79  
Array before sort: 95 48 56 13 67 79  
Array after sort: 13 48 56 67 79 95  
Total time taken by code: 0.000019 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

d) Selection Sort

Code: -

```
#include <iostream>
#include <ctime>

using namespace std;

void selection(int array[], int n)
{
    int i, j, pos;
    for(i=0; i<n-1; i++) {
        pos = i;
        for(j=i+1; j<n; j++) {
            if(array[pos] > array[j]) {
                pos = j;
            }
        }
        if(pos!=i) {
            int c = array[i];
            array[i] = array[pos];
            array[pos] = c;
        }
    }
}

int main()
{
    int n, i;

    cout << "Enter the number of elements you want to enter: ";
    cin >> n;

    int a[n];
    clock_t t1, t2;
    double t;

    cout << "Enter the elements:-" << endl;
    for(i=0; i<n; i++) {
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
    cout << "a[" << i << "]: ";  
    cin >> a[i];  
}  
  
cout << "\nArray before sort: ";  
for(i=0; i<n; i++) {  
    cout << a[i] << " ";  
}  
  
t1 = clock();  
  
selection(a, n); // Pass Array in Sorting Function  
  
cout << "\nArray after sort: ";  
for(i=0; i<n; i++) {  
    cout << a[i] << " ";  
}  
t2 = clock();  
t = (double) (t2 - t1)/CLOCKS_PER_SEC;  
cout << "\nTotal time taken by code: " << t << " seconds."  
return 0;  
}
```

Output: -

```
Enter the number of elements you want to enter: 6  
Enter the elements:-  
a[0]: 95  
a[1]: 48  
a[2]: 56  
a[3]: 13  
a[4]: 48  
a[5]: 79  
Array before sort: 95 48 56 13 48 79  
Array after sort: 13 48 48 56 79 95  
Total time taken by code: 0.000018 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

e) Heap Sort

Code: -

```
#include <iostream>
#include <ctime>

using namespace std;

// Heapify a subtree rooted with node i which is an index in arr[]
void heapify(int arr[], int n, int i) {
    int largest = i; // Initialize largest as root
    int left = 2 * i + 1; // Left child
    int right = 2 * i + 2; // Right child

    // If left child is larger than root
    if (left < n && arr[left] > arr[largest])
        largest = left;

    // If right child is larger than largest so far
    if (right < n && arr[right] > arr[largest])
        largest = right;

    // If largest is not root
    if (largest != i) {
        swap(arr[i], arr[largest]);

        // Recursively heapify the affected sub-tree
        heapify(arr, n, largest);
    }
}

// Main function to perform heap sort
void heapSort(int arr[], int n) {
    // Build heap (rearrange array)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // One by one extract an element from the heap
    for (int i = n - 1; i > 0; i--) {
        // Move the current root to the end
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
swap(arr[0], arr[i]);

// Call max heapify on the reduced heap
heapify(arr, i, 0);
}
}

int main()
{
    int n, i;

    cout << "Enter the number of elements you want to enter: ";
    cin >> n;

    int a[n];
    clock_t t1, t2;
    double t;

    cout << "Enter the elements:-" << endl;
    for(i=0; i<n; i++) {
        cout << "a[" << i << "]: ";
        cin >> a[i];
    }

    cout << "\nArray before sort: ";
    for(i=0; i<n; i++) {
        cout << a[i] << " ";
    }

    t1 = clock();

    heapSort(a, n); // Pass Array in Sorting Function

    cout << "\nArray after sort: ";
    for(i=0; i<n; i++) {
        cout << a[i] << " ";
    }

    t2 = clock();
```

Name: Adityan Verma

Enrollment No. 07818002721

```
t = (double) (t2 - t1)/CLOCKS_PER_SEC;  
cout << "\nTotal time taken by code: " << t << " seconds.";  
  
return 0;  
}
```

Output: -

```
Enter the number of elements you want to enter: 6  
Enter the elements:-  
a[0]: 95  
a[1]: 48  
a[2]: 56  
a[3]: 13  
a[4]: 67  
a[5]: 79  
Array before sort: 95 48 56 13 67 79  
Array after sort: 13 48 56 67 79 95  
Total time taken by code: 0.000007 seconds.
```


Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 2

Aim: To implement Linear search & Binary search & analyze its TC.

a) Linear Search

Code: -

```
#include <iostream>
#include <ctime>

using namespace std;

int main() {
    int a[50], n, i, k;
    clock_t t1, t2;
    float t;

    t1 = clock();

    cout << "Enter size of array" << endl;
    cin >> n;

    cout << "Enter the values" << endl;
    for (i = 0; i < n; i++) {
        cin >> a[i];
    }

    cout << "Enter the element to be searched" << endl;
    cin >> k;
    for (i = 0; i < n; i++) {
        if (a[i] == k)
        {
            cout << "Element found" << endl;
            break;
        }
    }
    if (i == (n - 1)) {
        cout << "Element not found" << endl;
    }
}
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
t2 = clock();  
t = (float)(t2 - t1) / CLOCKS_PER_SEC;  
  
cout << "Processor time taken in searching position " << t << " seconds";  
  
return 0;  
}
```

Output: -

```
Enter number of elements you want to enter: 6  
Enter Values:-  
a[0]: 12  
a[1]: 23  
a[2]: 45  
a[3]: 56  
a[4]: 78  
a[5]: 89  
Enter an element you want to search: 78  
Element Found!!... Location --> 4  
Processor time taken in searching is 0.000028 seconds.
```

```
Enter number of elements you want to enter: 8  
Enter Values:-  
a[0]: 89  
a[1]: 65  
a[2]: 48  
a[3]: 23  
a[4]: 76  
a[5]: 45  
a[6]: 26  
a[7]: 37  
Enter an element you want to search: 20  
Element not found...:(  
Processor time taken in searching is 0.000036 seconds.
```



**DELHI TECHNICAL CAMPUS
GREATER NOIDA**
Affiliated to GGSIPU and approved by AICTE & COA



Name: Adityan Verma

Enrollment No. 07818002721

b) Binary Search

Code: -

```
#include <iostream>
```

```
#include <ctime>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, i, k, mid, beg = 0, end;
```

```
    clock_t t1, t2;
```

```
    float t;
```

```
    cout << "\nEnter number of elements you want to enter: ";
```

```
    cin >> n;
```

```
    int a[n];
```

```
    cout << "\nEnter Values:-" << endl;
```

```
    for (i = 0; i < n; i++) {
```

```
        cout << "a[" << i << "]: ";
```

```
        cin >> a[i];
```

```
    }
```

```
    cout << "\nEnter an element you want to search: ";
```

```
    cin >> k;
```

```
    t1 = clock();
```

```
    end = n - 1;
```

```
    mid = (beg + end) / 2;
```

```
    if (a[beg] == k) {
```

```
        cout << "\nElement Found!!... Location --> " << beg << endl;
```

```
    }
```

```
    else if (a[end] == k) {
```

```
        cout << "\nElement Found!!!... Location --> " << end << endl;
```

```
    }
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
else {
    while (beg <= end) {
        if (a[mid] == k) {
            cout << "Element Found!!... Location --> " << mid << endl;
            break;
        }
        else if (k < a[mid]) {
            end = mid + 1;
            mid = (beg + end) / 2;
        }
        else if (a[mid] < k) {
            beg = mid - 1;
            mid = (beg + end) / 2;
        }
    }
}

if (end <= beg) {
    cout << "Element not found...:(" << endl;
}

t2 = clock();
t = (float)(t2 - t1) / CLOCKS_PER_SEC;

cout << "Processor time taken in searching position" << t << "seconds";

return 0;
}
```

Name: Adityan Verma

Enrollment No. 07818002721

Output: -

```
Enter the number of elements in the array: 5
Enter the sorted array elements:
1
2
3
4
5
Enter the target element to search: 4
Element found at index 3
Time taken for binary search: 0.001 seconds
```

Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 3

Aim: To implement Huffman coding & analyze its TC.

Code: -

```
#include <iostream>
#include <iomanip>
#include <queue>
#include <unordered_map>
#include <vector>
#include <ctime>

using namespace std;

// A Huffman tree node
struct HuffmanNode {
    char data;
    unsigned frequency;
    HuffmanNode* left;
    HuffmanNode* right;
    HuffmanNode(char data, unsigned frequency) : data(data), frequency(frequency), left(nullptr),
right(nullptr) {}
};

// Comparison function for priority queue
struct Compare {
    bool operator()(HuffmanNode* l, HuffmanNode* r) {
        return l->frequency > r->frequency;
    }
};

// Function to build the Huffman Tree and return the root
HuffmanNode* buildHuffmanTree(const string& text) {
    // Count the frequency of each character in the text
    unordered_map<char, unsigned> freq;
    for (char c : text) {
        freq[c]++;
    }

    // Create a priority queue to store the nodes with their frequencies
    priority_queue<HuffmanNode*, vector<HuffmanNode*>, Compare> pq;

    // Create a leaf node for each character and push it to the priority queue
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
for (auto& entry : freq) {
    pq.push(new HuffmanNode(entry.first, entry.second));
}

// Build the Huffman Tree
while (pq.size() > 1) {
    HuffmanNode* left = pq.top();
    pq.pop();

    HuffmanNode* right = pq.top();
    pq.pop();

    HuffmanNode* internalNode = new HuffmanNode('$', left->frequency + right->frequency);
    internalNode->left = left;
    internalNode->right = right;

    pq.push(internalNode);
}

// The root of the Huffman Tree is the only node left in the priority queue
return pq.top();
}

// Traverse the Huffman Tree and store the Programs in a map
void generateHuffmanPrograms(HuffmanNode* root, string Program, unordered_map<char,
string>& huffmanPrograms) {
    if (root) {
        if (!root->left && !root->right) {
            huffmanPrograms[root->data] = Program;
        }

        generateHuffmanPrograms(root->left, Program + "0", huffmanPrograms);
        generateHuffmanPrograms(root->right, Program + "1", huffmanPrograms);
    }
}

// EnProgram the input text using Huffman Programs
string enProgramText(const string& text, const unordered_map<char, string>&
huffmanPrograms) {
    string enProgramdText = "";
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
for (char c : text) {
    enProgramdText += huffmanPrograms.at(c);
}
return enProgramdText;
}

// DeProgram the enProgramd text using the Huffman Tree
string deProgramText(HuffmanNode* root, const string& enProgramdText) {
    string deProgramdText = "";
    HuffmanNode* current = root;
    for (char bit : enProgramdText) {
        if (bit == '0') {
            current = current->left;
        } else {
            current = current->right;
        }

        if (!current->left && !current->right) {
            deProgramdText += current->data;
            current = root;
        }
    }
    return deProgramdText;
}

int main() {
    string text;

    cout << "\nEnter the text to be enProgramd: ";
    getline(cin, text);

    clock_t t1,t2;
    float t;
    t1 = clock();

    // Build Huffman Tree
    HuffmanNode* root = buildHuffmanTree(text);

    // Generate Huffman Programs
    unordered_map<char, string> huffmanPrograms;
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
generateHuffmanPrograms(root, "", huffmanPrograms);

// EnProgram the text
string enProgramdText = enProgramText(text, huffmanPrograms);
cout << "\nEnProgramd Text: " << enProgramdText << endl;

// DeProgram the text
string deProgramdText = deProgramText(root, enProgramdText);
cout << "DeProgramd Text: " << deProgramdText << endl;

t2 = clock();
t = (float)(t2-t1) / CLOCKS_PER_SEC;
cout << "\nProcessing Time is " << setprecision(6) << t << " seconds.";

return 0;
}
```

Output: -

```
Enter the text to be enProgramd: abaacd
EnProgramd Text: 01110010110
DeProgramd Text: abaacd

Processing Time is 0.000097 seconds.
```



**DELHI TECHNICAL CAMPUS
GREATER NOIDA**
Affiliated to GGSIPU and approved by AICTE & COA



Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 4

Aim: To implement Minimum Spanning Tree & analyze its TC.

Code: -

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <ctime>

using namespace std;

// Structure to represent an edge in the graph
struct Edge {
    int src, dest, weight;
};

// Structure to represent a subset for union-find
struct Subset {
    int parent, rank;
};

// Helper function to find the subset of an element
int find(Subset subsets[], int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}

// Helper function to perform union of two subsets
void unionSets(Subset subsets[], int x, int y) {
    int rootX = find(subsets, x);
    int rootY = find(subsets, y);

    if (subsets[rootX].rank < subsets[rootY].rank)
        subsets[rootX].parent = rootY;
    else if (subsets[rootX].rank > subsets[rootY].rank)
        subsets[rootY].parent = rootX;
    else {
        subsets[rootX].parent = rootY;
        subsets[rootY].rank++;
    }
}
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
}  
}  
  
// Comparator function to sort edges based on their weight  
bool compareEdges(const Edge& a, const Edge& b) {  
    return a.weight < b.weight;  
}  
  
// Kruskal's algorithm to find the Minimum Spanning Tree  
void kruskalMST(vector<Edge>& edges, int V) {  
    // Sort the edges in non-decreasing order of their weight  
    sort(edges.begin(), edges.end(), compareEdges);  
  
    // Allocate memory for subsets  
    Subset* subsets = new Subset[V];  
    for (int i = 0; i < V; i++) {  
        subsets[i].parent = i;  
        subsets[i].rank = 0;  
    }  
  
    vector<Edge> mst; // Store the edges of the MST  
  
    // Process each edge in sorted order  
    for (const Edge& edge : edges) {  
        int rootSrc = find(subsets, edge.src);  
        int rootDest = find(subsets, edge.dest);  
  
        // If including this edge doesn't cause a cycle, add it to the MST  
        if (rootSrc != rootDest) {  
            mst.push_back(edge);  
            unionSets(subsets, rootSrc, rootDest);  
        }  
    }  
  
    // Print the MST  
    cout << "Minimum Spanning Tree:\n";  
    for (const Edge& edge : mst) {  
        cout << edge.src << " -- " << edge.dest << " : " << edge.weight << "\n";  
    }  
}
```

Name: Adityan Verma

Enrollment No. 07818002721

```
    delete[] subsets;
}

int main() {
    clock_t t1,t2;
    float t;
    t1=clock();

    // Original graph represented as an edge list
    vector<Edge> edges = {
        {0, 1, 2},
        {0, 2, 4},
        {1, 2, 1},
        {1, 3, 5},
        {2, 3, 3}
    };

    // Number of vertices in the original graph
    int V = 4;

    // Print the original graph
    cout << "\nOriginal Graph:\n";
    for (const Edge& edge : edges) {
        cout << edge.src << " -- " << edge.dest << " : " << edge.weight << "\n";
    }
    cout << "\n";

    // Apply Kruskal's algorithm to find the MST
    kruskalMST(edges, V);

    t2=clock();
    t=float(t2-t1)/CLOCKS_PER_SEC;
    cout << "\nProcessing Time is " << t << " seconds.\n\n";

    return 0;
}
```

Name: Adityan Verma

Enrollment No. 07818002721

Output: -

Original Graph:

```
0 -- 1 : 2
0 -- 2 : 4
1 -- 2 : 1
1 -- 3 : 5
2 -- 3 : 3
```

Minimum Spanning Tree:

```
1 -- 2 : 1
0 -- 1 : 2
2 -- 3 : 3
```

Processing Time is 0.000058 seconds.



Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 5

Aim: To implement Dijkstra's algo & analyze TC.

Code: -

```
#include <iostream>
#include <vector>
#include <queue>
#include <limits>
#include <ctime>

using namespace std;

// Structure to represent a graph edge
struct Edge {
    int destination;
    int weight;
};

// Structure to represent a graph vertex
struct Vertex {
    int index;
    int distance;
};

// Comparison function for the priority queue
struct CompareVertex {
    bool operator()(const Vertex& a, const Vertex& b) {
        return a.distance > b.distance;
    }
};

// Function to perform Dijkstra's algorithm
void dijkstra(const vector<vector<Edge>>& graph, int source) {
    int V = graph.size();
    vector<int> distance(V, numeric_limits<int>::max());
    vector<bool> visited(V, false);
    priority_queue<Vertex, vector<Vertex>, CompareVertex> pq;
    pq.push({source, 0});
    distance[source] = 0;
    while (!pq.empty()) {
        int u = pq.top().index;
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
    pq.pop();
    if (visited[u]) {
        continue;
    }
    visited[u] = true;
    for (const Edge& edge : graph[u]) {
        int v = edge.destination;
        int weight = edge.weight;

        if (!visited[v] && distance[u] != numeric_limits<int>::max() && distance[u] + weight <
distance[v]) {
            distance[v] = distance[u] + weight;
            pq.push({v, distance[v]});
        }
    }
}

// Print the distances from the source to all vertices
cout << "Vertex\tDistance from Source\n";
for (int i = 0; i < V; ++i) {
    cout << i << "\t" << distance[i] << "\n";
}
}

int main() {
    clock_t t1,t2;
    float t;
    t1=clock();
    // Example graph represented as an adjacency list
    vector<vector<Edge>> graph = {
        {{1, 2}, {2, 4}},
        {{0, 2}, {2, 1}, {3, 5}},
        {{0, 4}, {1, 1}, {3, 3}},
        {{1, 5}, {2, 3}}
    };

    // Print the original graph
    cout << "\nOriginal Graph:-\n\n";
    for (int i = 0; i < graph.size(); ++i) {
        cout << "Vertex " << i << ": ";
```

Name: Adityan Verma

Enrollment No. 07818002721

```
for (const Edge& edge : graph[i]) {  
    cout << "(" << edge.destination << ", " << edge.weight << ") ";  
}  
    cout << "\n";  
}  
cout << "\n";  
  
// Specify the source vertex  
int source = 0;  
  
// Apply Dijkstra's algorithm  
dijkstra(graph, source);  
t2=clock();  
t=float(t2-t1)/CLOCKS_PER_SEC;  
cout << "\nProcessing Time is " << t << " seconds.\n\n";  
return 0;  
}
```

Output: -

```
Original Graph:-  
  
Vertex 0: (1, 2) (2, 4)  
Vertex 1: (0, 2) (2, 1) (3, 5)  
Vertex 2: (0, 4) (1, 1) (3, 3)  
Vertex 3: (1, 5) (2, 3)  
  
Vertex  Distance from Source  
0      0  
1      2  
2      3  
3      6  
  
Processing Time is 0.000139 seconds.
```


Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 6

Aim: To implement Dijkstra's algo & analyze TC.

Code: -

```
#include <iostream>
#include <vector>
#include <limits>
#include <ctime >

using namespace std;

// Structure to represent an edge in the graph
struct Edge {
    int source, destination, weight;
};

// Function to perform Bellman-Ford algorithm
void bellmanFord(const vector<Edge>& edges, int V, int source) {
    vector<int> distance(V, numeric_limits<int>::max());
    distance[source] = 0;
    // Relax all edges V-1 times
    for (int i = 0; i < V - 1; ++i) {
        for (const Edge& edge : edges) {
            if (distance[edge.source] != numeric_limits<int>::max() &&
                distance[edge.source] + edge.weight < distance[edge.destination]) {
                distance[edge.destination] = distance[edge.source] + edge.weight;
            }
        }
    }
    // Check for negative-weight cycles
    for (const Edge& edge : edges) {
        if (distance[edge.source] != numeric_limits<int>::max() &&
            distance[edge.source] + edge.weight < distance[edge.destination]) {
            cout << "Graph contains a negative-weight cycle.\n";
            return;
        }
    }
    // Print the original graph
    cout << "Original Graph:\n";
    for (const Edge& edge : edges) {
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
    cout << edge.source << " -- " << edge.destination << " : " << edge.weight << "\n";
}
cout << "\n";
// Print the distances from the source to all vertices
cout << "Vertex\tDistance from Source\n";
for (int i = 0; i < V; ++i) {
    cout << i << "\t" << distance[i] << "\n";
}
}

int main() {
    clock_t t1,t2;
    float t;
    t1=clock();
    // Example graph represented as an edge list
    vector<Edge> edges = {
        {0, 1, 2},
        {0, 2, 4},
        {1, 2, 1},
        {1, 3, 5},
        {2, 3, -3}
    };
    // Number of vertices in the graph
    int V = 4;
    // Specify the source vertex
    int source = 0;
    // Apply Bellman-Ford algorithm
    bellmanFord(edges, V, source);

    t2=clock();
    t=float(t2-t1)/CLOCKS_PER_SEC;

    cout << "\nProcessing Time is " << t << " seconds.\n\n";

    return 0;
}
```

Output: -

Name: Adityan Verma

Enrollment No. 07818002721

Original Graph:

```
0 -- 1 : 2
0 -- 2 : 4
1 -- 2 : 1
1 -- 3 : 5
2 -- 3 : -3
```

Vertex Distance from Source

```
0    0
1    2
2    3
3    0
```

Processing Time is 0.000043 seconds.



Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 7

Aim: Implement N Queen problem using backtracking.

Code: -

```
#include <iostream>
#include <vector>
#include <ctime>

using namespace std;

// Function to check if placing a queen at (row, col) is safe
bool isSafe(const vector<vector<int>>& board, int row, int col, int N) {
    // Check for queens in the same column
    for (int i = 0; i < row; ++i) {
        if (board[i][col] == 1) {
            return false;
        }
    }

    // Check for queens in the left diagonal
    for (int i = row, j = col; i >= 0 && j >= 0; --i, --j) {
        if (board[i][j] == 1) {
            return false;
        }
    }

    // Check for queens in the right diagonal
    for (int i = row, j = col; i >= 0 && j < N; --i, ++j) {
        if (board[i][j] == 1) {
            return false;
        }
    }

    return true;
}

// Function to print the order of queens in a solution
void printQueenOrder(const vector<vector<int>>& board, int N) {
    vector<int> queenOrder(N, 0);
    for (int i = 0; i < N; ++i) {
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
        for (int j = 0; j < N; ++j) {
            if (board[i][j] == 1) {
                queenOrder[i] = j + 1;
                break;
            }
        }
    }

    cout << "(";
    for (int i = 0; i < N; ++i) {
        cout << queenOrder[i];
        if (i < N - 1) {
            cout << ",";
        }
    }
    cout << ")";
}

// Function to print the chessboard
void printChessboard(const vector<vector<int>>& board, int N) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            cout << (board[i][j] == 1 ? "Q" : ".") << " ";
        }
        cout << "\n";
    }
}

// Function to solve the N-Queens problem using backtracking
void solveNQueens(vector<vector<int>>& board, int row, int N, int& solutionsCount) {
    if (row == N) {
        // All queens are placed successfully
        cout << "Solution " << solutionsCount++ << ": ";
        printQueenOrder(board, N);
        cout << "\n";
        printChessboard(board, N);
        cout << "\n\n";
        return;
    }
}
```

Name: Adityan Verma

Enrollment No. 07818002721

```
for (int col = 0; col < N; ++col) {
    if (isSafe(board, row, col, N)) {
        // Place the queen
        board[row][col] = 1;

        // Recur to place the rest of the queens
        solveNQueens(board, row + 1, N, solutionsCount);

        // Backtrack
        board[row][col] = 0;
    }
}

int main() {
    clock_t t1,t2;
    float t;
    int N;
    cout << "Enter the size of the chessboard (N): ";
    cin >> N;
    t1=clock();

    vector<vector<int>> board(N, vector<int>(N, 0));
    int solutionsCount = 1;

    solveNQueens(board, 0, N, solutionsCount);

    if (solutionsCount == 1) {
        cout << "No solutions found.\n";
    }
    t2=clock();
    t=float(t2-t1)/CLOCKS_PER_SEC;
    cout << "\nProcessing Time is " << t << " seconds.\n\n";
    return 0;
}
```

Name: Adityan Verma

Enrollment No. 07818002721

Output: -

```
Enter the size of the chessboard (N): 3
No solutions found.

Processing Time is 0.000041 seconds.
```

```
Enter the size of the chessboard (N): 4
Solution 1: (2,4,1,3)
. Q . .
. . . Q
Q . . .
. . Q .

Solution 2: (3,1,4,2)
. . Q .
Q . . .
. . . Q
. Q . .

Processing Time is 0.000102 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 8

Aim: To implement Matrix Multiplication & analyze TC.

Code: -

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
// Function to perform matrix multiplication
vector<vector<int>>> multiplyMatrix(const vector<vector<int>>>& mat1, const
vector<vector<int>>>& mat2) {
    int row1 = mat1.size();
    int col1 = mat1[0].size();
    int row2 = mat2.size();
    int col2 = mat2[0].size();
    // Check if matrix multiplication is possible
    if (col1 != row2) {
        cout << "Error: Matrix dimensions are not compatible for multiplication.\n";
        return { };
    }
    // Initialize the result matrix with zeros
    vector<vector<int>>> result(row1, vector<int>(col2, 0));
    // Perform matrix multiplication
    for (int i = 0; i < row1; ++i) {
        for (int j = 0; j < col2; ++j) {
            for (int k = 0; k < col1; ++k) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
    return result;
}
// Function to print a matrix
void printMatrix(const vector<vector<int>>>& matrix) {
    for (const auto& row : matrix) {
        for (int val : row) {
            cout << val << " ";
        }
        cout << "\n";
    }
}
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
}  
int main() {  
    clock_t t1,t2;  
    float t;  
    t1=clock();  
    // Example matrices  
    vector<vector<int>> matrix1 = {  
        { 1, 2, 3},  
        { 4, 5, 6},  
        { 7, 8, 9}  
    };  
    vector<vector<int>> matrix2 = {  
        { 9, 8, 7},  
        { 6, 5, 4},  
        { 3, 2, 1}  
    };  
    // Perform matrix multiplication  
    vector<vector<int>> result = multiplyMatrix(matrix1, matrix2);  
    // Print the matrices and the result  
    cout << "Matrix 1:\n";  
    printMatrix(matrix1);  
    cout << "\nMatrix 2:\n";  
    printMatrix(matrix2);  
    cout << "\nResult of Matrix Multiplication:\n";  
    printMatrix(result);  
    t2=clock();  
    t=float(t2-t1)/CLOCKS_PER_SEC;  
    cout << "\nProcessing Time is " << t << " seconds.\n\n";  
    return 0;  
}
```

Name: Adityan Verma

Enrollment No. 07818002721

Output: -

```
Matrix 1:
1 2 3
4 5 6
7 8 9

Matrix 2:
9 8 7
6 5 4
3 2 1

Result of Matrix Multiplication:
30 24 18
84 69 54
138 114 90

Processing Time is 0.000079 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 9

Aim: To implement Longest Common Subsequence problem & analyze its TC.

Code: -

```
#include <iostream>
#include <vector>
#include <ctime>

using namespace std;

// Function to find the length of the Longest Common Subsequence
int longestCommonSubsequence(const string& str1, const string& str2) {
    int m = str1.length();
    int n = str2.length();

    // Create a 2D vector to store the lengths of LCS
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    // Build the dp table
    for (int i = 1; i <= m; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (str1[i - 1] == str2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }

    return dp[m][n];
}

// Function to find the Longest Common Subsequence
string findLCS(const string& str1, const string& str2) {
    int m = str1.length();
    int n = str2.length();

    // Create a 2D vector to store the lengths of LCS
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
// Build the dp table
for (int i = 1; i <= m; ++i) {
    for (int j = 1; j <= n; ++j) {
        if (str1[i - 1] == str2[j - 1]) {
            dp[i][j] = dp[i - 1][j - 1] + 1;
        } else {
            dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }
}

// Reconstruct the LCS
int i = m, j = n;
string lcs;

while (i > 0 && j > 0) {
    if (str1[i - 1] == str2[j - 1]) {
        lcs = str1[i - 1] + lcs;
        --i;
        --j;
    } else if (dp[i - 1][j] > dp[i][j - 1]) {
        --i;
    } else {
        --j;
    }
}

return lcs;
}

int main() {
    clock_t t1, t2;
    float t;
    t1 = clock();
    string str1, str2;

    cout << "Enter the first string: ";
    cin >> str1;

    cout << "Enter the second string: ";
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
cin >> str2;
int length = longestCommonSubsequence(str1, str2);
cout << "Length of Longest Common Subsequence: " << length << "\n";
string lcs = findLCS(str1, str2);
cout << "Longest Common Subsequence: " << lcs << "\n";
t2=clock();
t=float(t2-t1)/CLOCKS_PER_SEC;
cout << "\nProcessing Time is " << t << " seconds.\n\n";
return 0;
}
```

Output: -

```
Enter the first string: ABCDBAB
Enter the second string: ABCDA
Length of Longest Common Subsequence: 5
Longest Common Subsequence: ABCDA

Processing Time is 0.000312 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 10

Aim: To implement naïve String Matching algo, Rabin Karp algo & Knuth Morris Pratt algo & analyze it TC.

a) Naïve String Matching Algorithm

Code: -

```
#include <iostream>
#include <string>
#include <ctime>

using namespace std;

void naiveStringMatching(const string& text, const string& pattern) {
    int n = text.length();
    int m = pattern.length();

    for (int i = 0; i <= n - m; ++i) {
        int j;
        for (j = 0; j < m; ++j) {
            if (text[i + j] != pattern[j])
                break;
        }

        if (j == m)
            cout << "Pattern found at index " << i << "\n";
    }
}

int main() {
    clock_t t1,t2;
    float t;
    t1=clock();
    string text = "ABABCABABABCABABC";
    string pattern = "ABABC";

    cout<<"String text: "<<text<<"\n";
    cout<<"String pattern: "<<pattern<<"\n";
    cout << "Naïve String Matching Algorithm: \n";
    naiveStringMatching(text, pattern);
}
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
t2=clock();  
t=float(t2-t1)/CLOCKS_PER_SEC;  
cout << "\nProcessing Time is " << t << " seconds.\n\n";  
return 0;  
}
```

Output: -

```
String text: ABABCABABABCABABC  
String pattern: ABABC  
Naïve String Matching Algorithm:  
Pattern found at index 0  
Pattern found at index 7  
Pattern found at index 12  
  
Processing Time is 0.000053 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

a) Robin Karp Algorithm

Code: -

```
#include <iostream>
#include <string>
#include <ctime>

using namespace std;

const int prime = 101; // Prime number for hashing

void rabinKarp(const string& text, const string& pattern) {
    int n = text.length();
    int m = pattern.length();
    const int d = 256; // Number of characters in the input alphabet

    int h = 1;
    for (int i = 0; i < m - 1; ++i)
        h = (h * d) % prime;

    int patternHash = 0;
    int textHash = 0;

    // Calculate the initial hash values
    for (int i = 0; i < m; ++i) {
        patternHash = (d * patternHash + pattern[i]) % prime;
        textHash = (d * textHash + text[i]) % prime;
    }

    for (int i = 0; i <= n - m; ++i) {
        if (patternHash == textHash) {
            int j;
            for (j = 0; j < m; ++j) {
                if (text[i + j] != pattern[j])
                    break;
            }

            if (j == m)
                cout << "Pattern found at index " << i << "\n";
        }
    }
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
if (i < n - m) {  
    textHash = (d * (textHash - text[i] * h) + text[i + m]) % prime;  
    if (textHash < 0)  
        textHash += prime;  
}  
}  
}  
  
int main() {  
    clock_t t1,t2;  
    float t;  
    t1=clock();  
    string text = "ABABCABABABCABABC";  
    string pattern = "ABABC";  
    cout<<"String text "<<text<<"\n";  
    cout<<"String pattern "<<pattern<<"\n";  
    cout<<"Rabin-Karp Algorithm:-\n";  
    rabinKarp(text, pattern);  
    t2=clock();  
    t=float(t2-t1)/CLOCKS_PER_SEC;  
  
    cout << "\nProcessing Time is " << t << " seconds.\n\n";  
    return 0;  
}
```

Output: -

```
String text ABABCABABABCABABC  
String pattern ABABC  
Rabin-Karp Algorithm:-  
Pattern found at index 0  
Pattern found at index 7  
Pattern found at index 12  
  
Processing Time is 0.000124 seconds.
```

Name: Adityan Verma

Enrollment No. 07818002721

a) Knuth Morris Pratt Algorithm

Code: -

```
#include <iostream>
#include <vector>
#include <string>
#include <ctime>

using namespace std;

vector<int> computeLPS(const string& pattern) {
    int m = pattern.length();
    vector<int> lps(m, 0);

    int len = 0;
    int i = 1;

    while (i < m) {
        if (pattern[i] == pattern[len]) {
            len++;
            lps[i] = len;
            i++;
        } else {
            if (len != 0)
                len = lps[len - 1];
            else {
                lps[i] = 0;
                i++;
            }
        }
    }

    return lps;
}

void kmp(const string& text, const string& pattern) {
    int n = text.length();
    int m = pattern.length();

    vector<int> lps = computeLPS(pattern);
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

```
int i = 0; // Index for text
int j = 0; // Index for pattern

while (i < n) {
    if (pattern[j] == text[i]) {
        j++;
        i++;
    }

    if (j == m) {
        cout << "Pattern found at index " << i - j << "\n";
        j = lps[j - 1];
    } else if (i < n && pattern[j] != text[i]) {
        if (j != 0)
            j = lps[j - 1];
        else
            i++;
    }
}

int main() {
    clock_t t1,t2;
    float t;
    t1=clock();
    string text = "ABABCABABABCABABC";
    string pattern = "ABABC";
    cout<<"string text "<<text<<"\n";
    cout<<"string pattern "<<pattern<<"\n";
    cout << "Knuth-Morris-Pratt (KMP) Algorithm:\n";
    kmp(text, pattern);
    t2=clock();
    t=float(t2-t1)/CLOCKS_PER_SEC;
    cout << "\nProcessing Time is " << t << " seconds.\n\n";
    return 0;
}
```



**DELHI TECHNICAL CAMPUS
GREATER NOIDA**
Affiliated to GGSIPU and approved by AICTE & COA



Name: Adityan Verma

Enrollment No. 07818002721

Output: -

```
String text ABABCABABABCABABC
String pattern ABABC
Knuth-Morris-Pratt (KMP) Algorithm:
Pattern found at index 0
Pattern found at index 7
Pattern found at index 12

Processing Time is 0.000084 seconds.
```

**Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida**

Name: Adityan Verma

Enrollment No. 07818002721

EXPERIMENT 11

Aim: To implement Sorting Network.

Code: -

```
#include <iostream>
#include <vector>
```

```
using namespace std;
```

```
// Function to perform a comparison and swap if necessary
```

```
void compareAndSwap(int& a, int& b) {
    if (a > b) {
        swap(a, b);
    }
}
```

```
// Function to perform a batcher's merge operation
```

```
void batchersMerge(vector<int>& arr, int l, int r, int d) {
    if (d > 0) {
        int k = 1 << (d - 1);
        for (int i = l; i + k <= r; ++i) {
            compareAndSwap(arr[i], arr[i + k]);
        }
        batchersMerge(arr, l, r, d - 1); // Recursively merge smaller subproblems
        batchersMerge(arr, l + k, r + k, d - 1);
    }
}
```

```
// Function to perform Batcher's odd-even mergesort
```

```
void batchersOddEvenMergesort(vector<int>& arr, int l, int r, int d) {
    if (d == 0) {
        compareAndSwap(arr[l], arr[r]);
    } else {
        int m = (l + r) / 2;
        batchersOddEvenMergesort(arr, l, m, d - 1); // Odd phase
        batchersOddEvenMergesort(arr, m + 1, r, d - 1); // Even phase
        batchersMerge(arr, l, r, d - 1);
    }
}
```

Department of Computer Science and Engineering
Delhi Technical Campus, Greater Noida

Name: Adityan Verma

Enrollment No. 07818002721

```
// Function to perform sorting network
void sortingNetwork(vector<int>& arr, int n) {
    int d = 1; // Number of bits in the binary representation of the array size

    while ((1 << d) < n) {
        batchersOddEvenMergesort(arr, 0, n - 1, d);
        d++;
    }
}

// Function to print the array
void printArray(const vector<int>& arr) {
    for (int num : arr) {
        cout << num << " ";
    }
    cout << "\n";
}

int main() {
    vector<int> arr = {5, 2, 9, 1, 5, 6};

    cout << "Original Array: ";
    printArray(arr);

    sortingNetwork(arr, arr.size());

    cout << "Sorted Array: ";
    printArray(arr);
    return 0;
}
```

Output: -

```
Original Array: 5 2 9 1 5 6
Sorted Array: 2 5 1 5 6 9
```