Design & Analysis of Algorithm LAB

PAPER CODE	:	**CIC-359**
Faculty Name	:	Dr. Moolchand Sharma
Name	:	Amit Singhal
Enrollment No.	:	11614802722
Branch	:	Computer Science & Engg.
Semester | Group	:	5C6

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

PSP Area, Plot No. 1, Sector-22, Rohini, Delhi-110086

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

Computer Science & Engineering Department

VISION

"To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values."

MISSION

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real-life problems and fulfil societal needs.

LAB Assessment Sheet

S.No.	Experiment Name	M A R K S					Total Marks	Date of Perf.	Date of Check.	Signature
		R1	R2	R3	R4	R5				

S.No.	Experiment Name	M A R K S					Total Marks	Date of Perf.	Date of Check.	Signature
		R1	R2	R3	R4	R5				

INTRODUCTION TO DESIGN AND ANALYSIS OF ALGORITHMS

Design and Analysis of Algorithms is a fundamental aspect of computer science that involves creating efficient solutions to computational problems and evaluating their performance. It focuses on designing algorithms that effectively address specific challenges and analysing their efficiently in terms of **time** and **space complexity**.

ALGORITHM ANALYSIS: -

Algorithm analysis is an important part of computational complexity theory, which provides theoretical estimation for the required resources of an algorithm to solve a specific computational problem. Analysis of algorithms is the determination of the amount of time and space resources required to execute it.

Why the Analysis of Algorithm is important?

- To predict the behavior of an algorithm without implementing it on a specific computer.
- It is much more convenient to have simple measures for the efficiency of an algorithm than to implement the algorithm and test the efficiency every time a certain parameter in the underlying computer system changes.
- It is impossible to predict the exact behavior of an algorithm. There are too many influencing factors.
- The analysis is thus only an approximation; it is not perfect.
- More importantly, by analyzing different algorithms, we can compare them to determine the best one for our purpose.

What are the Advantages of Analysis of Algorithms?

1. Efficiency Assessment: Analyzing algorithms helps determine their time and space complexity, allowing developers to evaluate how efficiently they will perform under different conditions.

2. Optimal Algorithm Selection: It provides insights that guide the selection of the most appropriate algorithm for a given problem, ensuring the best performance based on specific requirements.

3. Scalability Evaluation: Analysis reveals how an algorithm's performance changes with varying input sizes, helping to predict scalability and identify potential issues as data grows.

4. Resource Management: Understanding an algorithm's resource requirements enables better allocation of computational resources, leading to improved overall system performance.

5. Bottleneck Identification: Through analysis, developers can identify inefficient parts of an algorithm, facilitating targeted optimization efforts to enhance performance.

6. Performance Prediction: Analyzing algorithms allows for performance predictions in different scenarios, aiding in planning and system design.

7. Understanding Trade-offs: It helps clarify the trade-offs between time and space complexity, which is crucial for making informed decisions in algorithm design.

8. Facilitating Debugging: A clear understanding of how an algorithm should perform helps identify bugs or inefficiencies more easily during development.

9. Informed Learning and Improvement: Studying algorithm analysis principles fosters a deeper understanding of algorithm design techniques, promoting continuous learning and improvement.

10. Foundation for Advanced Concepts: It provides a basis for exploring more advanced topics in computer science, such as complexity theory and algorithmic optimization.

What are the Disadvantages of Analysis of Algorithms?

1. Complexity: The mathematical and theoretical nature of algorithm analysis can be complex and difficult to understand for beginners.

2. Assumptions: Analysis often relies on assumptions about input data and computational models, which may not reflect real-world scenarios.

3. Overhead: Detailed analysis can introduce additional time and resource overhead during the design and development process.

4. Limited Practicality: Theoretical performance metrics may not always translate to practical performance due to factors like hardware differences and implementation variations.

5. Neglect of Other Factors: Focusing solely on time and space complexity can lead to overlooking other important factors, such as maintainability, readability, and usability.

6. Dynamic Environments: Algorithms may perform differently in dynamic environments, making pre-analysis less reliable for certain applications.

7. Diminishing Returns: For some algorithms, further analysis may yield diminishing returns in terms of insights gained versus effort spent.

Applications of Design and Analysis of Algorithms: -

1. Search Engines: Algorithms are used to index and retrieve information efficiently, optimizing search results based on user queries.

2. Data Compression: Algorithms help reduce file sizes for storage and transmission, making data transfer faster and more efficient.

3. Machine Learning: Used to develop algorithms for training models, optimizing performance, and making predictions based on data patterns.

4. Network Routing: Algorithms determine the most efficient paths for data transmission across networks, improving communication speeds and reliability.

5. Scheduling: Employed in operations research to optimize scheduling tasks in manufacturing, transportation, and project management for better resource utilization.

Lab Exercise - 1

- AIM :: WAP in C++ to implement Bubble, Merge, Quick & Insertion Sort and also evaluate time in each.

1. Bubble Sort

Source_Code ::

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

using namespace std;

void bubbleSort(vector<int>& arr)
{
    int n = arr.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

int main()
{
    cout << "5C6 - Amit Singhal (11614802722)" << endl;
```

```
    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };
    cout << "Unsorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    clock_t start = clock();
    bubbleSort(arr);
    clock_t end = clock();

    cout << "Bubble Sort:" << endl;
    cout << "Sorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    double time_taken_ms = double(end - start) * 1000.0
        / CLOCKS_PER_SEC; // Convert to milliseconds
    cout << "Time taken: " << time_taken_ms << " milliseconds" << endl;

    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ vi exp_1.1.cpp
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_1.1.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Unsorted Array: 64 34 25 12 22 11 90
Bubble Sort:
Sorted Array: 11 12 22 25 34 64 90
Time taken: 0.003 milliseconds
```

2. Merge Sort

Source_Code ::

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

using namespace std;

void merge(vector<int>& arr, int l, int m, int r)
{
    int n1 = m - l + 1;
    int n2 = r - m;

    vector<int> L(n1), R(n2);

    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int i = 0; i < n2; i++)
        R[i] = arr[m + 1 + i];

    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++;
    }

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

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(vector<int>& arr, int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int main()
{
    cout << "5C6 - Amit Singhal (11614802722)" << endl;
    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };
    cout << "Unsorted Array: ";
```

```
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    clock_t start = clock();
    mergeSort(arr, 0, arr.size() - 1);
    clock_t end = clock();

    cout << "Merge Sort:" << endl;
    cout << "Sorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    double time_taken_ms = double(end - start) * 1000.0
        / CLOCKS_PER_SEC; // Convert to milliseconds
    cout << "Time taken: " << time_taken_ms << " milliseconds" << endl;

    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_1.2.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Unsorted Array: 64 34 25 12 22 11 90
Merge Sort:
Sorted Array: 11 12 22 25 34 64 90
Time taken: 0.239 milliseconds
```

3. Quick Sort

Source_Code ::

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

using namespace std;

int partition(vector<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);
}

void quickSort(vector<int>& arr, int low, int high)
{
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

```
int main()
{
    cout << "5C6 - Amit Singhal (11614802722)" << endl;
    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };
    cout << "Unsorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;
    clock_t start = clock();
    quickSort(arr, 0, arr.size() - 1);
    clock_t end = clock();

    cout << "Quick Sort:" << endl;
    cout << "Sorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    double time_taken_ms = double(end - start) * 1000.0
        / CLOCKS_PER_SEC; // Convert to milliseconds
    cout << "Time taken: " << time_taken_ms << " milliseconds" << endl;

    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_1.3.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Unsorted Array: 64 34 25 12 22 11 90
Quick Sort:
Sorted Array: 11 12 22 25 34 64 90
Time taken: 0.026 milliseconds
```

4. Insertion Sort

Source_Code ::

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

using namespace std;

void insertionSort(vector<int>& arr)
{
    int n = arr.size();
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

int main()
{
    cout << "5C6 - Amit Singhal (11614802722)" << endl;
    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };
```

```
    cout << "Unsorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    clock_t start = clock();
    insertionSort(arr);
    clock_t end = clock();

    cout << "Insertion Sort:" << endl;
    cout << "Sorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    double time_taken_ms = double(end - start) * 1000.0
        / CLOCKS_PER_SEC; // Convert to milliseconds
    cout << "Time taken: " << time_taken_ms << " milliseconds" << endl;

    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_1.4.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Unsorted Array: 64 34 25 12 22 11 90
Insertion Sort:
Sorted Array: 11 12 22 25 34 64 90
Time taken: 0.003 milliseconds
```

Lab Exercise - 2

- AIM :: WAP in C++ to implement Linear & Binary Search and also evaluate time in each.

1. Linear Search

Source_Code ::

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

using namespace std;

int linearSearch(const vector<int>& arr, int x)
{
    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] == x) {
            return i;
        }
    }
    return -1; // Element not found
}

int main()
{
    cout << "5C6 - Amit Singhal (11614802722)" << endl;

    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };
```

```
    cout << "Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    int x;
    cout << "Enter the element to search: ";
    cin >> x;
    clock_t start = clock();
    int index = linearSearch(arr, x);
    clock_t end = clock();
    if (index != -1) {
        cout << "Element found at index: " << index << endl;
    } else {
        cout << "Element not found" << endl;
    }
    double time_taken_ms = double(end - start) * 1000.0
        / CLOCKS_PER_SEC; // Convert to milliseconds
    cout << "Time taken: " << time_taken_ms << " milliseconds" << endl;
    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_2.1.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Array: 64 34 25 12 22 11 90
Enter the element to search: 25
Element found at index: 2
Time taken: 0.003 milliseconds
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Array: 64 34 25 12 22 11 90
Enter the element to search: 90
Element found at index: 6
Time taken: 0.026 milliseconds
```

2. Binary Search

Source_Code ::

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

using namespace std;

int binarySearch(const vector<int>& arr, int x)
{
    int left = 0, right = arr.size() - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            return mid;
        }
        if (arr[mid] < x) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1; // Element not found
}

int main()
{
    cout << "5C6 - Amit Singhal (11614802722)" << endl;

    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };
```

```
    sort(arr.begin(), arr.end()); // Binary search requires a sorted array
    cout << "Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;
    int x;
    cout << "Enter the element to search: ";
    cin >> x;
    clock_t start = clock();
    int index = binarySearch(arr, x);
    clock_t end = clock();
    if (index != -1) {
        cout << "Element found at index: " << index << endl;
    } else {
        cout << "Element not found" << endl;
    }
    double time_taken_ms = double(end - start) * 1000.0
        / CLOCKS_PER_SEC; // Convert to milliseconds
    cout << "Time taken: " << time_taken_ms << " milliseconds" << endl;
    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_2.2.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Array: 11 12 22 25 34 64 90
Enter the element to search: 25
Element found at index: 3
Time taken: 0.025 milliseconds
amit@Toshiba-Satellite-C850:~/Downloads$ ./a
5C6 - Amit Singhal (11614802722)
Array: 11 12 22 25 34 64 90
Enter the element to search: 90
Element found at index: 6
Time taken: 0.003 milliseconds
```

Lab Exercise - 3

- AIM :: WAP in C++ to implement Huffman Coding & also evaluate its time complexity.

Source_Code ::

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

using namespace std;

// Node of Huffman Tree
struct Node {
    char ch;
    int freq;
    Node *left, *right;

    Node(char ch, int freq, Node* left = nullptr, Node* right = nullptr)
    {
        this->ch = ch;
        this->freq = freq;
        this->left = left;
        this->right = right;
```

```
    }
};

// Comparison function for priority queue
struct compare {
    bool operator()(Node* left, Node* right)
    {
        return left->freq > right->freq;
    }
};

// Function to build the Huffman Tree
Node* buildHuffmanTree(const unordered_map<char, int>& freq)
{
    priority_queue<Node*, vector<Node*>, compare> pq;

    for (auto pair : freq) {
        pq.push(new Node(pair.first, pair.second));
    }

    while (pq.size() != 1) {
        Node* left = pq.top();
        pq.pop();
        Node* right = pq.top();
        pq.pop();

        int sum = left->freq + right->freq;
        pq.push(new Node('\0', sum, left, right));
    }
```

```
    return pq.top();
}

// Function to encode the input string
void encode(
    Node* root, const string& str, unordered_map<char, string>& huffmanCode)
{
    if (root == nullptr)
        return;

    if (!root->left && !root->right) {
        huffmanCode[root->ch] = str;
    }

    encode(root->left, str + "0", huffmanCode);
    encode(root->right, str + "1", huffmanCode);
}

// Function to decode the encoded string
string decode(Node* root, const string& str)
{
    string result = "";
    Node* curr = root;
    for (char bit : str) {
        if (bit == '0') {
            curr = curr->left;
        } else {
            curr = curr->right;
        }
```

```
        if (!curr->left && !curr->right) {
            result += curr->ch;
            curr = root;
        }
    }
    return result;
}

int main()
{
    cout << "\n5C6 - Amit Singhal (11614802722)" << endl;

    string text;
    cout << "\nEnter the text to encode: ";
    getline(cin, text);

    unordered_map<char, int> freq;
    for (char ch : text) {
        freq[ch]++;
    }

    clock_t start = clock();
    Node* root = buildHuffmanTree(freq);
    clock_t end = clock();
    double time_taken_build_tree
        = double(end - start) * 1000.0 / CLOCKS_PER_SEC;

    unordered_map<char, string> huffmanCode;
    start = clock();
    encode(root, "", huffmanCode);
```

```
end = clock();
double time_taken_encoding = double(end - start) * 1000.0 / CLOCKS_PER_SEC;

cout << "\nCharacter Encoding Table:" << endl;
cout << "------------------------------" << endl;
cout << setw(10) << "Character" << setw(20) << "Huffman Code" << endl;
cout << "------------------------------" << endl;
for (auto pair : huffmanCode) {
    cout << setw(10) << pair.first << setw(20) << pair.second << endl;
}
cout << "------------------------------" << endl;

cout << "Time taken to build Huffman Tree: " << time_taken_build_tree
     << " milliseconds" << endl;

string encodedString = "";
for (char ch : text) {
    encodedString += huffmanCode[ch];
}

cout << "\nEncoded String: " << encodedString << endl;
cout << "Time taken for encoding: " << time_taken_encoding
     << " milliseconds" << endl;

start = clock();
string decodedString = decode(root, encodedString);
end = clock();
double time_taken_decoding = double(end - start) * 1000.0 / CLOCKS_PER_SEC;

cout << "\nDecoded String: " << decodedString << endl;
```

```
    cout << "Time taken for decoding: " << time_taken_decoding
         << " milliseconds" << endl;

    return 0;
}
```

Output ::

```
amit@Toshiba-Satellite-C850:~/Downloads$ g++ exp_3.cpp -o a
amit@Toshiba-Satellite-C850:~/Downloads$ ./a

5C6 - Amit Singhal (11614802722)

Enter the text to encode: Amit Singhal

Character Encoding Table:
------------------------------
 Character        Huffman Code
------------------------------
         l                1110
         i                 110
         n                1111
         a                1010
                          1001
         g                1011
         t                 010
         S                1000
         m                 001
         A                 011
         h                 000
------------------------------
Time taken to build Huffman Tree: 0.034 milliseconds

Encoded String: 01100111001010011000110111111011000010101110
Time taken for encoding: 0.022 milliseconds

Decoded String: Amit Singhal
Time taken for decoding: 0.004 milliseconds
```