AI Lab (Week 1)

Adit Luhadia

190911112

IT A

Lab 1

Write a Program to perform the following and find the time complexity using step count method

1. Binary Search (both iterative and recursive)

```cpp
#include <iostream>

using namespace std;

int count;

// Iterative binary search
int binarySearch(int arr[], int l, int r, int key) {
    count++;
    while (l <= r) {
        int m = l + (r - l) / 2;
        count++;

        count++;
        // If key is at m index
        if (arr[m] == key) {
            count++;
            return m;
        }

        count++;
        // If key is greater than mth element
        if (arr[m] < key) {
            count++;
            l = m + 1;
        }

        count++;
        // If key is smaller than mth element
        if (arr[m] > key) {
            count++;
            r = m - 1;
        }
```

```
    }
    count++;

    count++;
    // If not found, return -1
    return -1;
}

int main() {
    int n, key;

    count = 0;

    cout << "Enter the number of elements" << endl;
    cin >> n;

    int a[n];

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

    cout << "Enter the element to be searched" << endl;
    cin >> key;

    int result = binarySearch(a, 0, n - 1, key);

    if (result == -1) {
        cout << "Element not found" << endl;
    } else {
        cout << "Element found at index " << result << endl;
    }

    cout << "Step count: " << count << endl;

    return 0;
}
```

```
#include <iostream>

using namespace std;

int count;

// Recursive binary search
int binarySearch(int arr[], int l, int r, int key) {
```

```
    count++;
    if (r >= l) {
        count++;
        int mid = l + (r - l) / 2;

        count++;
        // If element is in the middle
        if (arr[mid] == key) {
            count++;
            return mid;
        }

        count++;
        // Search in the left subarray
        if (arr[mid] > key) {
            count++;
            return binarySearch(arr, l, mid - 1, key);
        }

        count++;
        // Search in right subarray
        return binarySearch(arr, mid + 1, r, key);
    }

    count++;
    // If not found, return -1
    return -1;
}

int main() {
    int n, key;

    count = 0;

    cout << "Enter the number of elements" << endl;
    cin >> n;

    int a[n];

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

    cout << "Enter the element to be searched" << endl;
    cin >> key;

    int result = binarySearch(a, 0, n - 1, key);
```

```
    if (result == -1) {
        cout << "Element not found" << endl;
    } else {
        cout << "Element found at index " << result << endl;
    }

    cout << "Step count: " << count << endl;

    return 0;
}
```

```
Enter the number of elements
5
Enter the elements
1 2 6 8 11
Enter the element to be searched
8
Element found at index 3
Step count: 9
```

2. Bubble Sort

```
#include <iostream>

using namespace std;

int count;

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        count++; // For outer for loop
        for (int j = 0; j < n-i-1; j++) {
            count++; // For inner for loop

            count++; // For if condition
            if (arr[j] > arr[j+1]) {
                count++;
                int temp = arr[j];
                count++;
                arr[j] = arr[j + 1];
                count++;
                arr[j + 1] = temp;
            }
        }
```

```
        count++; // For false case of inner for loop
    }
    count++; // For false case of outer for loop
}

int main() {
    int n;

    count = 0;

    cout << "Enter the number of elements" << endl;
    cin >> n;

    int a[n];

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

    bubbleSort(a, n);

    cout << "Sorted array" << endl;
    for (int i = 0; i < n; i++) {
        cout << a[i] << "\t";
    }

    cout << "Step count: " << count << endl;

    return 0;
}
```

```
Enter the number of elements
5
Enter the elements
1 2 4 3 5
Sorted array
1       2       3       4       5       Step count: 32
```

3. Selection Sort

```
#include <iostream>

using namespace std;
```

```cpp
int count;

void selectionSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        count++; // For outer for loop

        count++;
        int min_idx = i;

        for (int j = i+1; j < n; j++) {
            count++; // For inner for loop

            count++;
            if (arr[j] < arr[min_idx]) {
                count++;
                min_idx = j;
            }
        }
        count++; // For false condition of inner for loop

        // Swap the found minimum element with the first element
        count++;
        int temp = arr[min_idx];
        count++;
        arr[min_idx] = arr[i];
        count++;
        arr[i] = temp;
    }
    count++; // For false condition of outer for loop
}


int main() {
    int n;

    count = 0;

    cout << "Enter the number of elements" << endl;
    cin >> n;

    int a[n];

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

    selectionSort(a, n);
```

```
    cout << "Sorted array" << endl;
    for (int i = 0; i < n; i++) {
        cout << a[i] << "\t";
    }

    cout << "Step count: " << count << endl;

    return 0;
}
```

```
Enter the number of elements
5
Enter the elements
1 2 4 3 9
Sorted array
1       2       3       4       9       Step count: 46
```

4. Insertion Sort

```
#include <iostream>

using namespace std;

int count;

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        count++; // For the for loop

        count++;
        int j = i - 1;

        count++;
        int key = arr[i];

        for (; j >= 0 && arr[j] > key; j--) {
            count++;

            count++;
            arr[j + 1] = arr[j];
        }
        count++;

        count++;
        arr[j + 1] = key;
    }
    count++; // For false condition of for loop
```

```
}


int main() {
    int n;

    count = 0;

    cout << "Enter the number of elements" << endl;
    cin >> n;

    int a[n];

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

    insertionSort(a, n);

    cout << "Sorted array" << endl;
    for (int i = 0; i < n; i++) {
        cout << a[i] << "\t";
    }

    cout << "Step count: " << count << endl;

    return 0;
}
```

```
Enter the number of elements
5
Enter the elements
1 4 2 3 6
Sorted array
1       2       3       4       6       Step count: 25
```

Lab 1 additional

Write a Program to perform the following and find the time complexity using either operation counts or step counts method

1. Interpolation search

```
#include <iostream>
```

```
using namespace std;

int count = 0;

int interpolationSearch(int arr[], int lo, int hi, int x)
{
    count++;
    int pos;
    count++; // For if statement
    if (lo <= hi && x >= arr[lo] && x <= arr[hi])
    {
        count++;
        pos = lo + (((double)(hi - lo) / (arr[hi] - arr[lo])) * (x - arr[lo]))
;
        count++; // For if statement
        if (arr[pos] == x)
        {
            count++;
            return pos;
        }
        count++; // For if statement
        if (arr[pos] < x)
        {
            count++;
            return interpolationSearch(arr, pos + 1, hi, x);
        }
        count++; // For if statement
        if (arr[pos] > x)
        {
            count++;
            return interpolationSearch(arr, lo, pos - 1, x);
        }
    }
    return -1;
}
int main()
{
    int arr[20], n, index, x, i;
    cout << "enter the number of elements:";
    cin >> n;
    cout << "enter the elements:";
    for (i = 0; i < n; i++)
        cin >> arr[i];
    cout << "enter the search element:";
    cin >> x;
    index = interpolationSearch(arr, 0, n - 1, x);
    if (index != -1)
```

```
        cout << "Element found at index " << index;
    else
        cout << "Element not found.";

    cout << endl
         << "Count: " << count;
    return 0;
}
```

```
enter the number of elements:5
enter the elements:1 2 4 8 9
enter the search element:8
Element found at index 3
Count: 5
```

2. Radix Sort

```
#include <iostream>
using namespace std;

int count = 0;

void radsort(int A[], int n)
{
    int temp;
    int bucket[10][20];
    int buck_count[10];
    int i, k, j, r, np = 0, divisor = 1;
    count++; //for initialization
    int largest, pass_no;
    largest = A[0];
    for (i = 1; i < n; i++)
    {
        count++; //for the for loop
        count++; //for the if condition
        if (A[i] > largest)
        {
            largest = A[i];
            count++; //for the assignment operation
        }
    }
    count++; //for the false condition of for loop
    while (largest > 0)
    {
        count++; //for the while loop
        np++;
        count++; //for the increment operation;
        largest = largest / 10;
```

```
        count++; //for the assignment operation
    }
    count++; //for the false condition of the while loop
    for (pass_no = 0; pass_no < np; pass_no++)
    {
        count++; //for the for loop
        for (k = 0; k < 10; k++)
        {
            count++; //for the for loop
            buck_count[k] = 0;
            count++; //for the assignment operation
        }
        count++; //for the false condition of the for loop
        for (i = 0; i < n; i++)
        {
            count++; //for the for loop
            r = (A[i] / divisor) % 10;
            count++; //for the assignment operation
            bucket[r][buck_count[r]++] = A[i];
            count++; //for the assignment operation
        }
        count++; //for the false condition of the for loop
        i = 0;
        count++; //for the assignment operation
        for (k = 0; k < 10; k++)
        {
            count++; //for the for loop
            for (j = 0; j < buck_count[k]; j++)
            {
                count++; //for the for loop
                A[i++] = bucket[k][j];
                count++; //for the assignment operation
            }
            count++; //for the false condition of the for loop
        }
        count++; //for the false condition of the for loop
        divisor = divisor * 10;
        count++; //for the assignment operation
    }
    count++; //for the false condition of the for loop
}
int main()
{
    int a[20], n, i, j;
    cout << "enter the size\n";
    cin >> n;
    cout << "Enter the elements\n";
    for (i = 0; i < n; i++)
```

```
    {
        cin >> a[i];
    }
    radsort(a, n);
    cout << "The sorted  array is \n";
    for (i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << "\nNumber of steps : " << count;
    return 0;
}
```

```
enter the size
5
Enter the elements
1 2 8 5 3
The sorted  array is
1 2 3 5 8
Number of steps : 88
```

3. Shell Sort

```
#include <iostream>
using namespace std;

int count = 0;

int shellSort(int arr[], int n)
{
    for (int gap = n / 2; gap > 0; gap /= 2)
    {
        count++;
        for (int i = gap; i < n; i += 1)
        {
            count++;
            int temp = arr[i];
            count++;
            int j;
            for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
            {
                arr[j] = arr[j - gap];
                count++;
            }
            count++;
            arr[j] = temp;
            count++;
        }
        count++;
    }
```

```
    count++;
    return 0;
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
}
int main()
{
    int arr[20], i, n;
    cout << "enter the number of elements:";
    cin >> n;
    cout << "enter the elements:";
    for (i = 0; i < n; i++)
        cin >> arr[i];
    cout << "Array before sorting: \n";
    printArray(arr, n);
    shellSort(arr, n);
    cout << "\nArray after sorting: \n";
    printArray(arr, n);
    cout << endl
         << "Count: " << count;
    return 0;
}
```

```
enter the number of elements:5
enter the elements:1 2 3 9 6
Array before sorting:
1 2 3 9 6
Array after sorting:
1 2 3 6 9
Count: 34
```

4. Heap Sort

```
#include <iostream>
using namespace std;

int Count = 0;

void heapify(int arr[], int n, int i)
{
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    if (l < n && arr[l] > arr[largest])
```

```
        largest = l;
    if (r < n && arr[r] > arr[largest])
        largest = r;
    if (largest != i)
    {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr, n, largest);
    }
}
void heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
    {
        Count++; //for the for loop
        Count++; //for function call
        heapify(arr, n, i);
    }
    Count++; // for the false condition of the for loop
    for (int i = n - 1; i > 0; i--)
    {
        Count++; //for the for loop
        int temp = arr[0];
        Count++; //for the assignment operation
        arr[0] = arr[i];
        Count++; //for the assignment operation
        arr[i] = temp;
        Count++; //for the assignment operation
        Count++; //for the function call
        heapify(arr, i, 0);
    }
}
int main()
{
    int a[20], n, i;
    cout << "enter the size\n";
    cin >> n;
    cout << "Enter the elements\n";
    for (i = 0; i < n; i++)
    {
        cin >> a[i];
    }
    heapSort(a, n);
    cout << "The sorted  array is \n";
    for (i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << "\nNumber of step : " << Count;
```

```
    return 0;
}
```

```
enter the size
5
Enter the elements
1 3 6 2 4
The sorted  array is
1 2 3 4 6
Number of step : 25
```

Lab 2

1. DFS

```
#include <bits/stdc++.h>

using namespace std;
void DFS(int A[10][10], int a, int b, int &c)
{
    stack<int> s;
    c++;
    int z;
    bool visited[a];
    for (int i = 0; i < a; i++)
    {
        c++;
        visited[i] = false;
    }
    c++;
    s.push(0);
    while (!s.empty())
    {
        z = s.top();
        s.pop();
        c++;
        if (!visited[z])
        {
            cout << z << " ";
            visited[z] = true;
            c++;
        }
        for (int i = 0; i < a; i++)
        {
```

```
            c++;
            if (A[z][i] && visited[i] == false)
            {
                s.push(i);
                c++;
            }
        }
    }
}
int main()
{
    int A[10][10], n, m, x, co = 0;
    ;
    cout << "Enter no. of the vertices\n"
         << endl;
    cin >> n;
    co++;
    cout << "Enter number of the edges\n"
         << endl;
    cin >> m;
    co++;
    for (int i = 0; i < n; i++)
    {
        co++;
        {
            for (int j = 0; j < n; j++)
            {
                A[i][j] = 0;
                co++;
            }
        }
    }
    co++;
    int p, q;
    for (int i = 1; i <= m; i++)
    {
        cout << "Enter Source: " << endl;
        cin >> p;
        co++;
        cout << "Enter destination: " << endl;
        cin >> q;
        co++;
        A[p][q] = 1;
    }
    DFS(A, n, m, co);
    co++;
    cout << endl
         << "\nNo of counts:" << co << endl;
```

```
}
```

2. BFS

```
#include <bits/stdc++.h>

using namespace std;

void BFS(int A[10][10], int a, int b, int &x)
{
    queue<int> q;
    x++;
    int z;
    bool visited[a];
    for (int i = 0; i < b; i++)
    {
        x++;
        visited[i] = false;
    }
    x++;
    q.push(0);
    while (!q.empty())
    {
        z = q.front();
        q.pop();
        x++;
        if (!visited[z])
        {
            cout << z << " ";
            visited[z] = true;
            x++;
        }
        for (int i = 0; i < a; i++)
        {
            x++;
            if (A[z][i] && visited[i] == false)
            {
                q.push(i);
                x++;
            }
        }
    }
}
int main()
{
```

```
    int A[10][10], n, m, x, co = 0;
    ;
    cout << "enter no. of vertice\n";
    cin >> n;
    co++;
    cout << "enter number of edges\n";
    cin >> m;
    co++;
    for (int i = 0; i < n; i++)
    {
        co++;
        {
            for (int j = 0; j < n; j++)
            {
                A[i][j] = 0;
                co++;
            }
        }
    }
    co++;
    int p, q;
    for (int i = 1; i <= m; i++)
    {
        cout << "Enter Source: ";

        cin >> p;
        co++;
        cout << "Enter destination: ";
        cin >> q;
        co++;
        A[p][q] = 1;
    }
    BFS(A, n, m, co);
    co++;
    cout << endl
         << "No of counts: " << co << endl;
}
```

3. To find mother vertex in a graph

```
#include <bits/stdc++.h>

using namespace std;

void dfs(int **adj, bool *visited, int n, int sv)
{
    visited[sv] = true;
```

```
    for (int i = 0; i < n; i++)
        if (adj[sv][i] == 1 && !visited[i])
            dfs(adj, visited, n, i);
}

int main()
{
    int v, e;
    cin >> v >> e;
    cout << endl;

    int *adj = new int[v];
    for (int i = 0; i < v; i++)
        adj[i] = new int[v];

    for (int i = 0; i < v; i++)
        for (int j = 0; j < v; j++)
            adj[i][j] = 0;

    for (int i = 0; i < e; i++)
    {
        int l, r;
        cin >> l >> r;
        adj[l][r] = 1;
    }

    bool *visited = new bool[v];
    for (int i = 0; i < v; i++)
        visited[i] = false;

    int flag;

    for (int i = 0; i < v; i++)
    {
        flag = 0;
        visited[i] = true;
        dfs(adj, visited, v, i);

        for (int j = 0; j < v; j++)
        {
            if (visited[j] == false)
            {
                flag = 1;
                break;
            }
        }

        if (flag == 0)
```

```
            cout << i << "  ";

        // reset
        for (int j = 0; j < v; j++)
            visited[j] = false;
    }

    delete[] visited;
    for (int i = 0; i < v; i++)
        delete[] adj[i];

    delete[] adj;

    return 0;
}
```

4. To find transpose matrix of given graph

```
#include <bits/stdc++.h>
using namespace std;

int c;

void dfs(int **adj, bool *visited, int n, int sv)
{
    c++;
    visited[sv] = true;
    cout << sv << "\t";
    c++;

    for (int i = 0; i < n; i++, c++)
        if (adj[sv][i] == 1 && !visited[i])
            dfs(adj, visited, n, i);
}

int main()
{
    c = 0;
    int v, e;
    c++;
    cin >> v >> e;
    c++;
    cout << endl;

    int *adj = new int[v];
```

```
    for (int i = 0; i < v; i++, c++)
        adj[i] = new int[v];

    for (int i = 0; i < v; i++, c++)
        for (int j = 0; j < v; j++, c++)
            adj[i][j] = 0;

    for (int i = 0; i < e; i++, c++)
    {
        c++;
        int l, r;
        cin >> l >> r;
        c++;
        adj[r][l] = 1;
    }

    c++;

    bool *visited = new bool[v];
    for (int i = 0; i < v; i++, c++)
        visited[i] = false;

    int sv;
    c++;
    cout << "Enter start vertex: ";
    cin >> sv;
    c++;

    cout << "dfs of transpose: ";
    dfs(adj, visited, v, sv);

    cout << "\nStep count: " << c;

    return 0;
}
```

Lab 3

1. Finding a path in the graph

```
#include <iostream>
#include <queue>

using namespace std;

class Node
```

```
{
public:
    int val;
    Node *next = NULL;
};

void add(Node *&head, int val)
{
    if (!head)
    {
        head = new Node;
        head->val = val;
        return;
    }
    Node *temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    Node *newn = new Node;
    newn->val = val;
    temp->next = newn;
}

bool path(Node *graph[], int visited[], int src, int dest)
{
    Node *temp = graph[src];
    while (temp != NULL)
    {
        if (visited[temp->val] == 0)
        {
            if (temp->val == dest || path(graph, visited, temp->val, dest))
            {
                cout << temp->val << " ";
                visited[temp->val] = 1;
                return true;
            }
        }
        temp = temp->next;
    }
    return false;
}

int main()
{
    int n;
    cout << "Enter size:";
    cin >> n;
    Node *graph[n + 1] = {NULL};
```

```
    cout << "Enter edges:" << endl;
    int a = 0, b = 0;
    while (true)
    {
        cin >> a >> b;
        if (a != -1 && b != -1)
            add(graph[a], b);
        else
            break;
    }
    cout << "Path between:";
    cin >> a >> b;
    int visited[n + 1] = {0};
    cout << path(graph, visited, a, b);
}
```

2. Finding a cycle in the graph

```
#include <iostream>
#include <queue>

using namespace std;

class Node
{
public:
    int val;
    Node *next = NULL;
};

void add(Node *&head, int val)
{
    if (!head)
    {
        head = new Node;
        head->val = val;
        return;
    }
    Node *temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    Node *newn = new Node;
    newn->val = val;
    temp->next = newn;
}
```

```
bool path(Node *graph[], int visited[], int src, int dest)
{
    Node *temp = graph[src];
    while (temp != NULL)
    {
        if (visited[temp->val] == 0)
        {
            if (temp->val == dest || path(graph, visited, temp->val, dest))
            {
                visited[temp->val] = 1;
                return true;
            }
        }
        temp = temp->next;
    }
    return false;
}

bool cycles(Node *graph[], int n)
{
    bool found = false;
    for (int i = 1; i < n; i++)
    {
        int visited[n] = {0};
        if (path(graph, visited, i, i))
            found = true;
    }
    return found;
}

int main()
{
    int n;
    cout << "Enter size:";
    cin >> n;
    Node *graph[n] = {NULL};
    cout << "Enter edges:" << endl;
    int a = 0, b = 0;
    while (true)
    {
        cin >> a >> b;
        if (a != -1 && b != -1)
            add(graph[a], b);
        else
            break;
    }
    cycles(graph, n) ? cout << "Cycle exists" : cout << "Cycle absent";
}
```

3. Check whether the given graph is connected or not.

```
#include <iostream>
using namespace std;

int Count = 0;

void traverse(int graph[][30], int n, int u, bool visited[])
{
    visited[u] = true;
    for (int v = 0; v < n; v++)
    {
        if (graph[u][v])
        {
            if (!visited[v])
                traverse(graph, n, v, visited);
        }
    }
}
bool isConnected(int a[][30], int n)
{
    bool *vis = new bool[n];
    for (int u; u < n; u++)
    {
        Count++; // For outer for loop
        for (int i = 0; i < n; i++)
        {
            Count++; // For inner for loop
            vis[i] = false;
            Count++;
        }
        Count++; // For false conditionn of inner for loop
        Count++;
        traverse(a, n, u, vis);
        for (int i = 0; i < n; i++)
        {
            Count++; // For inner for loop
            if (!vis[i])
            {
                Count++;
                Count++;
                return false;
            }
        }
        Count++; // For false conditionn of inner for loop
    } // For false condition of outer for loop
    Count++;
```

```
    return true;
}
int main()
{
    int n, g[30][30];
    cout << "Enter the number of vertex:\n";
    cin >> n;
    cout << "Fill the adjacency table:\n";
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> g[i][j];
    if (isConnected(g, n))
        cout << "The Graph is connected.";
    else
        cout << "The Graph is not connected.";
    cout << "\nNumber of steps : " << Count;
}
```

```
Enter the number of vertex:
3
Fill the adjacency table:
0 1 1
1 0 1
1 1 0
The Graph is connected.
Number of steps : 1
```