

Laboratory Program 1

Write program to do the following:

- Print all the nodes reachable from a given starting node in a digraph using the BFS method.
- Check whether a given graph is connected or not using the DFS method.

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

// Function to perform Breadth First Search (BFS)
void BFS(vector<vector<int>>& graph, int startNode) {
    int numNodes = graph.size();
    vector<bool> visited(numNodes, false); // Keep track of visited nodes
    queue<int> q; // Queue for BFS traversal

    visited[startNode] = true; // Mark the start node as visited
    q.push(startNode); // Enqueue the start node

    cout << "Nodes reachable from " << startNode << " using BFS: ";
    while (!q.empty()) {
        int currentNode = q.front();
        q.pop();
        cout << currentNode << " ";

        // Enqueue all the unvisited neighbors of the current node
        for (int neighbor = 0; neighbor < numNodes; ++neighbor) {
            if (graph[currentNode][neighbor] && !visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }
    }
    cout << endl;
}

// Depth First Search (DFS) helper function
void DFSUtil(vector<vector<int>>& graph, int node, vector<bool>& visited) {
    visited[node] = true;

    // Recursive call for all unvisited neighbors
    for (int neighbor = 0; neighbor < graph.size(); ++neighbor) {
        if (graph[node][neighbor] && !visited[neighbor]) {
```

```

        DFSUtil(graph, neighbor, visited);
    }
}

```

```

// Function to perform Depth First Search (DFS)
bool DFS(vector<vector<int>>& graph) {
    int numNodes = graph.size();
    vector<bool> visited(numNodes, false); // Keep track of visited nodes

    DFSUtil(graph, 0, visited); // Start DFS traversal from node 0

    // Check if all nodes were visited
    for (bool v : visited) {
        if (!v) {
            return false; // Graph is not connected
        }
    }
    return true; // Graph is connected
}

```

```

int main() {
    int numNodes;
    cout << "Enter the number of nodes in the graph: ";
    cin >> numNodes;

    vector<vector<int>> graph(numNodes, vector<int>(numNodes));

    cout << "Enter the adjacency matrix (0 for no edge, 1 for edge exists):" << endl;
    for (int i = 0; i < numNodes; ++i) {
        for (int j = 0; j < numNodes; ++j) {
            cin >> graph[i][j];
        }
    }

    int startNode;
    cout << "Enter the starting node for BFS: ";
    cin >> startNode;

    BFS(graph, startNode);

    bool isConnected = DFS(graph);
    if (isConnected) {
        cout << "The graph is connected." << endl;
    }
}

```

```
    } else {  
        cout << "The graph is not connected." << endl;  
    }  
  
    return 0;  
}
```

Output

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out  
Enter the number of nodes in the graph: 3  
Enter the adjacency matrix (0 for no edge, 1 for edge exists):  
0 1 0  
1 0 1  
0 1 0  
Enter the starting node for BFS: 0  
Nodes reachable from 0 using BFS: 0 1 2  
The graph is connected.  
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out  
Enter the number of nodes in the graph: 4  
Enter the adjacency matrix (0 for no edge, 1 for edge exists):  
0 1 0 0  
1 0 1 1  
0 0 0 0  
0 1 0 0  
Enter the starting node for BFS: 1  
Nodes reachable from 1 using BFS: 1 0 2 3  
The graph is connected.  
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$
```

Laboratory Program 2

Write a program to obtain the Topological ordering of vertices in a given digraph.

```
#include <bits/stdc++.h>
using namespace std;
class Graph {
    int V;
    list<int>* adj;

public:

    Graph(int V);
    void addEdge(int u, int v);
    void topologicalSort();
};

Graph::Graph(int V)
{
    this->V = V;
    adj = new list<int>[V];
}

void Graph::addEdge(int u, int v)
{
    adj[u].push_back(v);
}

void Graph::topologicalSort()
{
    vector<int> in_degree(V, 0);
    for (int u = 0; u < V; u++) {
        list<int>::iterator itr;
        for (itr = adj[u].begin();
             itr != adj[u].end(); itr++)
            in_degree[*itr]++;
    }
    queue<int> q;
    for (int i = 0; i < V; i++)
        if (in_degree[i] == 0)
            q.push(i);
    int cnt = 0;
    vector<int> top_order;
    while (!q.empty()) {
```

```

        int u = q.front();
        q.pop();
        top_order.push_back(u);
        list<int>::iterator itr;
        for (itr = adj[u].begin();
              itr != adj[u].end(); itr++)
            if (--in_degree[*itr] == 0)
                q.push(*itr);

        cnt++;
    }
    if (cnt != V) {
        cout << "There exists a cycle in the graph\n";
        return;
    }
    for (int i = 0; i < top_order.size(); i++)
        cout << top_order[i] << " ";
    cout << endl;
}

int main()
{
    int n;
    cin >> n;
    Graph g(n);
    for(int i=0;i<n;i++)
    {
        cout<<"Do you want to enter an edge? (Y/N)";
        char ch;
        cin>>ch;
        if(ch=='N')
            break;
        else if(ch=='Y')
        {
            int from,to;
            cout<<"Enter the initial vertex:";
            cin>>from;
            cout<<endl<<"Enter the target vertex:";
            cin>>to;
            g.addEdge(from,to);
        }
    }

    cout << "Topological Sort\n";
    g.topologicalSort();
}

```

```
    return 0;  
}
```

Output

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ g++ toposort.cpp  
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out  
6  
Do you want to enter an edge? (Y/N)Y  
Enter the initial vertex:1  
  
Enter the target vertex:2  
Do you want to enter an edge? (Y/N)Y  
Enter the initial vertex:1  
  
Enter the target vertex:3  
Do you want to enter an edge? (Y/N)Y  
Enter the initial vertex:1  
  
Enter the target vertex:4  
Do you want to enter an edge? (Y/N)N  
Topological Sort  
0 1 5 2 3 4  
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$
```

Laboratory Program 3

Implement Johnson Trotter algorithm to generate permutations.

```
#include <bits/stdc++.h>
#define LEFT_TO_RIGHT 1
#define RIGHT_TO_LEFT 0
using namespace std;

int searchArr(int a[], int n, int mobile)
{
    for (int i = 0; i < n; i++)
        if (a[i] == mobile)
            return i + 1;

    return -1;
}

int getMobile(int a[], bool dir[], int n)
{
    int mobile_prev = 0, mobile = 0;
    for (int i = 0; i < n; i++) {
        // direction 0 represents RIGHT TO LEFT.
        if (dir[a[i] - 1] == RIGHT_TO_LEFT && i != 0) {
            if (a[i] > a[i - 1] && a[i] > mobile_prev) {
                mobile = a[i];
                mobile_prev = mobile;
            }
        }

        if (dir[a[i] - 1] == LEFT_TO_RIGHT && i != n - 1) {
            if (a[i] > a[i + 1] && a[i] > mobile_prev) {
                mobile = a[i];
                mobile_prev = mobile;
            }
        }
    }

    if (mobile == 0 && mobile_prev == 0)
        return 0;
    else
        return mobile;
}
```

```

void printOnePerm(int a[], bool dir[], int n)
{
    int mobile = getMobile(a, dir, n);
    int pos = searchArr(a, n, mobile);

    if (dir[a[pos] - 1] == RIGHT_TO_LEFT)
        swap(a[pos - 1], a[pos - 2]);

    else if (dir[a[pos] - 1] == LEFT_TO_RIGHT)
        swap(a[pos], a[pos - 1]);

    for (int i = 0; i < n; i++) {
        if (a[i] > mobile) {
            if (dir[a[i] - 1] == LEFT_TO_RIGHT)
                dir[a[i] - 1] = RIGHT_TO_LEFT;
            else if (dir[a[i] - 1] == RIGHT_TO_LEFT)
                dir[a[i] - 1] = LEFT_TO_RIGHT;
        }
    }

    for (int i = 0; i < n; i++)
        cout << a[i]<<" ";
    cout <<endl;
}

```

```

int fact(int n)
{
    int res = 1;
    for (int i = 1; i <= n; i++)
        res = res * i;
    return res;
}

```

```

void printPermutation(int n)
{
    int a[n];
    bool dir[n];

    for (int i = 0; i < n; i++) {
        a[i] = i + 1;
        cout << a[i]<<" ";
    }
    cout << endl;
}

```



```

        for (int i = 0; i < n; i++)
            dir[i] = RIGHT_TO_LEFT;

        for (int i = 1; i < fact(n); i++)
            printOnePerm(a, dir, n);
    }
    int main()
    {
        int n;
        cout<<"Enter the value of n:";
        cin>>n;
        printPermutation(n);
        return 0;
    }

```

Output

```

bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ g++ john.cpp
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the value of n:4
1 2 3 4
1 2 4 3
1 4 2 3
4 1 2 3
4 1 3 2
1 4 3 2
1 3 4 2
1 3 2 4
3 1 2 4
3 1 4 2
3 4 1 2
4 3 1 2
4 3 2 1
3 4 2 1
3 2 4 1
3 2 1 4
2 3 1 4
2 3 4 1
2 4 3 1
4 2 3 1
4 2 1 3
2 4 1 3
2 1 4 3
2 1 3 4
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$

```

Laboratory Program 4

Sort a given set of N integer elements using Merge Sort technique and compute its time taken.
Run the program for different values of N and record the time taken to sort.

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

void merge(int array[], int const left, int const mid,int const right)
{
    int const subArrayOne = mid - left + 1;
    int const subArrayTwo = right - mid;

    auto *leftArray = new int[subArrayOne],
        *rightArray = new int[subArrayTwo];

    for (auto i = 0; i < subArrayOne; i++)
        leftArray[i] = array[left + i];
    for (auto j = 0; j < subArrayTwo; j++)
        rightArray[j] = array[mid + 1 + j];

    auto indexOfSubArrayOne = 0, indexOfSubArrayTwo = 0;
    int indexOfMergedArray = left;

    while (indexOfSubArrayOne < subArrayOne
        && indexOfSubArrayTwo < subArrayTwo) {
        if (leftArray[indexOfSubArrayOne]
            <= rightArray[indexOfSubArrayTwo]) {
            array[indexOfMergedArray]
                = leftArray[indexOfSubArrayOne];
            indexOfSubArrayOne++;
        }
        else {
            array[indexOfMergedArray]
                = rightArray[indexOfSubArrayTwo];
            indexOfSubArrayTwo++;
        }
        indexOfMergedArray++;
    }
}
```

```

while (indexOfSubArrayOne < subArrayOne) {
    array[indexOfMergedArray]
        = leftArray[indexOfSubArrayOne];
    indexOfSubArrayOne++;
    indexOfMergedArray++;
}

```

```

while (indexOfSubArrayTwo < subArrayTwo) {
    array[indexOfMergedArray]
        = rightArray[indexOfSubArrayTwo];
    indexOfSubArrayTwo++;
    indexOfMergedArray++;
}
delete[] leftArray;
delete[] rightArray;
}

```

```

void mergeSort(int array[], int const begin, int const end)
{
    if (begin >= end)
        return;

    int mid = begin + (end - begin) / 2;
    mergeSort(array, begin, mid);
    mergeSort(array, mid + 1, end);
    merge(array, begin, mid, end);
}

```

```

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

```

```

int main()
{
    int n;
    cout<<"Enter the number of elements:";
}

```

```

        cin>>n;
        int arr[n];
    clock_t start,end;
        //cout<<"Enter the array:";
        for(int i=0;i<n;i++){
            //cin>>arr[i];
            arr[i]=rand()%100;}
        int arr_size = sizeof(arr) / sizeof(arr[0]);
    cout << "Given array is \n";
    printArray(arr, arr_size);
        start=clock();
    mergeSort(arr, 0, arr_size - 1);
        end=clock();
    cout << "\nSorted array is \n";
    printArray(arr, arr_size);
    cout<<endl<<"Time taken is:"<<(double)(end-start)/CLOCKS_PER_SEC<<endl;
    return 0;
}

```

Output

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ g++ merge.cpp
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of elements:1000
Given array is
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23 67 35 29
57 24 95 82 45 14 67 34 64 43 50 87 8 76 78 88 84 3 51 54 99 32 60 76 68 39 12 26 86 94 39
24 28 71 32 29 3 19 70 68 8 15 40 49 96 23 18 45 46 51 21 55 79 88 64 28 41 50 93 0 34 64 2
18 28 88 69 17 17 96 24 43 70 83 90 99 72 25 44 90 5 39 54 86 69 82 42 64 97 7 55 4 48 11 2
31 81 30 33 94 60 63 99 81 99 96 59 73 13 68 90 95 26 66 84 40 90 84 76 42 36 7 45 56 79 18
72 72 50 25 85 22 99 40 42 98 13 98 90 24 90 9 81 19 36 32 55 94 4 79 69 73 76 50 55 60 42
0 46 88 97 49 90 3 33 63 97 53 92 86 25 52 96 75 88 57 29 36 60 14 21 60 4 28 27 50 48 56 2
3 44 29 90 82 40 41 69 26 32 61 42 60 17 23 61 81 9 90 25 96 67 77 34 90 26 24 57 14 68 5 58
88 38 66 70 84 56 17 6 60 49 37 5 59 17 18 45 83 73 58 73 37 89 83 7 78 57 14 71 29 0 59 18
54 30 98 25 81 16 16 2 31 39 96 4 38 80 18 21 70 62 12 79 77 85 36 4 76 83 7 59 57 44 99 11
62 75 88 19 10 32 94 17 46 35 37 91 53 43 73 28 25 91 10 18 17 36 63 55 90 58 30 4 71 61 33
0 83 72 98 30 63 47 50 30 73 14 59 22 47 24 82 35 32 4 54 43 98 86 40 78 59 62 62 83 41 48 2
4 9 39 0 78 9 53 81 14 38 89 26 67 47 23 87 31 32 22 81 75 50 79 90 54 50 31 13 57 94 81 81
85 86 85 30 90 83 14 76 16 20 92 25 28 39 25 90 36 60 18 43 37 28 82 21 10 55 88 25 15 70 3
84 95 76 5 66 28 54 28 8 93 78 97 55 72 74 45 0 25 97 83 12 27 82 21

Sorted array is
0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5
11 11 11 11 11 11 11 11 11 11 11 11 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13 13 14 14 14 14
18 18 18 18 18 18 18 18 18 18 18 18 19 19 19 19 19 19 19 19 20 20 20 20 20 20 21 21 21 21 21 21
4 24 24 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 26
29 29 29 29 29 29 29 29 29 29 30 30 30 30 30 30 30 30 30 30 31 31 31 31 31 31 31 31 31 31 31 31
36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 37 37 37 38 38 38 38 38 38
2 42 43 43 43 43 43 43 43 43 43 43 43 43 43 43 44 44 44 44 44 44 44 44 45 45 45 45 45 46 46 46 46
50 50 50 51 51 51 51 51 51 51 51 51 51 51 51 52 52 52 52 52 52 52 52 52 53 53 53 53 53 53 53 53 54 54 54
57 57 57 58 58 58 58 58 58 58 58 58 58 58 58 59 59 59 59 59 59 59 59 59 59 59 59 59 59 59 60 60 60 60 60
5 65 65 65 65 66 66 66 66 66 66 66 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67 67
72 72 72 72 72 72 72 72 72 72 72 72 72 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73
79 79 79 79 80 80 80 80 80 80 80 80 80 80 80 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
5 85 85 85 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86
91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91
98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98 98

Time taken is:0.000524
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$
```

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of elements:6
Given array is
83 86 77 15 93 35

Sorted array is
15 35 77 83 86 93

Time taken is:1.6e-05
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$
```

Laboratory Program 5

Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

```
#include<bits/stdc++.h>
#include<time.h>
using namespace std;
int partition(int arr[],int low,int high)
{
    int p=arr[high];
    int i=low-1;
    for(int j=low;j<=high-1;j++)
    {
        if(arr[j]<p)
        {
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1],arr[high]);
    return i+1;
}
void quicksort(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()
{
    clock_t start,end;
    int n;
    cout<<"Enter the number of elements:";
    cin>>n;
    int arr[n];
    cout<<"Enter the elements:";
    for(int i=0;i<n;i++)
    arr[i]=rand()/1000000;
    start=clock();
    quicksort(arr,0,n-1);
```

```

end=clock();
cout<<endl<<"Sorted array is:";
for(int i=0;i<n;i++)
cout<<arr[i]<<" ";
cout<<"\nTime taken is:"<<(double)(end-start)/CLOCKS_PER_SEC<<endl;
return 0;
}

```

Output

```

bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of elements:20
Enter the elements:
Sorted array is:304 424 596 719 783 846 1025 1102 1189 1303 1350 1365 1540 1649 1681 1714 1804 1957 1967 2044
Time taken is:7e-06
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ 

```

```

bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ g++ quick.cpp
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of elements:1000
Enter the elements:
Sorted array is:2 6 6 7 8 10 11 11 12 12 14 19 29 31 34 35 35 37 40 42 51 61 63 70 70 74 76
151 155 155 157 159 159 160 164 165 168 168 174 180 181 184 188 195 199 200 201 201 204 207
0 262 262 269 269 270 270 271 272 278 279 282 286 289 292 292 294 294 296 298 298 304 305 36
64 370 371 372 376 378 378 380 382 384 387 387 389 392 394 395 395 396 396 400 402 402 404 4
460 463 468 470 471 474 476 476 476 478 480 481 483 484 485 488 491 492 496 496 498 500 500
570 572 577 580 584 586 588 590 593 595 596 601 603 606 608 610 610 617 619 619 620 620 624
2 672 677 680 690 695 695 696 697 699 700 703 705 706 707 709 711 711 712 714 716 719 719 72
86 787 791 802 803 805 811 813 816 820 820 822 822 824 826 828 831 833 841 845 846 846 846 8
937 937 939 941 943 945 950 951 952 959 959 962 964 968 970 971 972 980 982 982 983 989 990
1034 1036 1037 1038 1043 1046 1046 1047 1051 1057 1057 1059 1061 1063 1065 1065 1066 1067 10
5 1129 1129 1130 1131 1131 1131 1135 1137 1139 1140 1141 1143 1143 1144 1144 1151 1152 1155
232 1237 1238 1238 1239 1240 1242 1242 1242 1243 1244 1250 1250 1251 1253 1255 1255 1260 126
1295 1297 1302 1303 1304 1307 1308 1309 1312 1312 1314 1315 1315 1319 1326 1328 1329 1329 1
69 1370 1371 1372 1373 1374 1376 1376 1380 1384 1385 1386 1387 1388 1388 1389 1390 1392 1395
1431 1432 1433 1433 1434 1434 1435 1442 1446 1447 1447 1448 1449 1450 1450 1452 1455 1460 14
3 1503 1504 1505 1510 1516 1520 1520 1527 1528 1529 1534 1540 1541 1543 1543 1544 1545 1548
605 1605 1609 1610 1610 1615 1617 1622 1623 1626 1626 1630 1630 1631 1632 1633 1633 1635 163
1675 1676 1679 1681 1682 1687 1687 1691 1694 1703 1704 1707 1707 1708 1713 1714 1717 1722 1
63 1766 1775 1776 1779 1780 1780 1781 1782 1782 1784 1785 1787 1788 1789 1789 1795 1797 1799
1858 1858 1859 1862 1862 1866 1867 1869 1874 1875 1875 1876 1877 1884 1884 1886 1887 1889 18
1 1934 1936 1937 1939 1941 1941 1943 1943 1947 1950 1953 1954 1956 1957 1959 1960 1965 1967
000 2001 2001 2002 2004 2004 2006 2006 2007 2009 2010 2013 2016 2025 2025 2027 2030 2032 203
2106 2112 2113 2114 2114 2118 2130 2135 2137 2137 2142 2143 2145 2147
Time taken is:0.000402
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ 

```

Laboratory Program 6

Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

```
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;
using namespace std::chrono;

// Function to heapify a subtree rooted at 'root'
// n is the size of the heap
void heapify(vector<int>& arr, int n, int root) {
    int largest = root; // Initialize largest as root
    int left = 2 * root + 1; // Left child
    int right = 2 * root + 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 != root) {
        swap(arr[root], arr[largest]);

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

// Function to perform heap sort
void heapSort(vector<int>& arr) {
    int n = arr.size();

    // Build a max heap (rearrange array)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

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



```

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

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

int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;

    vector<int> arr(n);

    for (int i = 0; i < n; i++)
        arr[i]=rand()/100000;

    // Measure the time taken
    auto start = high_resolution_clock::now();

    heapSort(arr);

    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start);

    cout << "Sorted array: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    cout << "Time taken by heap sort: " << duration.count() << " microseconds" << endl;

    return 0;
}

```

Output

```
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ g++ ss.cpp
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of elements: 4
Sorted array: 8469 16816 17146 18042
Time taken by heap sort: 2 microseconds
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of elements: 1000
Sorted array: 24 60 69 76 89 109 116 116 122 128 149 194 297 313 347 350 357 374 406 429 512
1246 1261 1264 1345 1354 1364 1378 1465 1497 1501 1505 1515 1553 1557 1572 1592 1594 1600 1606 1617
65 2178 2179 2195 2215 2237 2267 2316 2336 2352 2356 2357 2389 2419 2425 2432 2457 2462 2557 2571
2986 3040 3051 3091 3091 3170 3185 3247 3272 3282 3322 3322 3335 3364 3374 3377 3377 3388 3400 3400
3 3873 3874 3890 3920 3946 3951 3952 3963 3964 4000 4027 4029 4041 4060 4074 4095 4101 4102 4102 4102
543 4606 4634 4687 4705 4719 4746 4761 4765 4766 4780 4802 4819 4836 4842 4855 4886 4917 4920 4920 4920
5518 5524 5529 5531 5542 5559 5593 5594 5617 5643 5673 5700 5726 5777 5805 5845 5862 5882 5882 5882 5882
95 6306 6317 6326 6357 6384 6398 6467 6480 6534 6534 6548 6554 6558 6596 6602 6617 6619 6699 6699 6699
7163 7193 7198 7223 7263 7330 7333 7334 7383 7386 7392 7407 7454 7455 7463 7492 7523 7568 7568 7568 7568
0 8206 8222 8228 8242 8260 8283 8317 8332 8411 8455 8466 8468 8469 8469 8472 8556 8563 8574 8574 8574
331 9373 9375 9398 9418 9439 9451 9503 9514 9520 9593 9599 9620 9644 9683 9709 9718 9720 9720 9720 9720
9 10234 10251 10252 10255 10264 10303 10311 10345 10349 10361 10371 10386 10433 10463 10467 10467 10467 10467 10467 10467
66 11006 11015 11025 11046 11058 11060 11083 11117 11127 11135 11171 11200 11223 11243 11258 11258 11258 11258 11258 11258
565 11583 11591 11727 11755 11769 11842 11860 11864 11866 11896 11906 11927 11949 11973 12040 12040 12040 12040 12040 12040
2508 12513 12532 12551 12553 12605 12640 12651 12652 12662 12678 12694 12724 12727 12739 12739 12739 12739 12739 12739 12739
13075 13080 13093 13126 13129 13142 13152 13156 13190 13262 13281 13291 13292 13305 13353 13353 13353 13353 13353 13353 13353
13693 13696 13709 13714 13722 13732 13743 13760 13767 13801 13840 13854 13864 13870 13883 13883 13883 13883 13883 13883 13883
2 14146 14148 14155 14170 14211 14242 14268 14314 14317 14321 14331 14339 14343 14344 14352 14352 14352 14352 14352 14352 14352
25 14727 14731 14734 14746 14761 14764 14771 14862 14870 14946 14986 15004 15012 15027 15038 15038 15038 15038 15038 15038 15038
483 15553 15624 15662 15663 15692 15722 15733 15790 15815 15821 15835 15847 15859 15869 15906 15906 15906 15906 15906 15906 15906
6306 16315 16326 16331 16339 16355 16359 16401 16425 16459 16464 16471 16497 16497 16533 16533 16533 16533 16533 16533 16533 16533
16877 16879 16914 16948 17039 17043 17070 17077 17083 17132 17146 17172 17220 17251 17269 17269 17269 17269 17269 17269 17269 17269
17612 17637 17661 17754 17768 17792 17801 17806 17819 17822 17824 17846 17855 17878 17880 17880 17880 17880 17880 17880 17880 17880
5 18415 18439 18444 18495 18509 18566 18577 18579 18585 18587 18590 18622 18628 18660 18671 18671 18671 18671 18671 18671 18671 18671
05 19052 19078 19085 19090 19098 19103 19108 19111 19117 19145 19162 19185 19274 19307 19319 19319 19319 19319 19319 19319 19319 19319
676 19682 19733 19735 19748 19759 19776 19790 19812 19822 19829 19832 19836 19836 19842 19850 19850 19850 19850 19850 19850 19850 19850
0167 20251 20255 20279 20304 20324 20328 20335 20336 20377 20386 20397 20403 20406 20448 20448 20448 20448 20448 20448 20448 20448 20448
21184 21307 21350 21371 21373 21427 21431 21451 21474 21474 21474 21474 21474 21474 21474 21474 21474 21474 21474 21474 21474 21474
Time taken by heap sort: 812 microseconds
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$
```

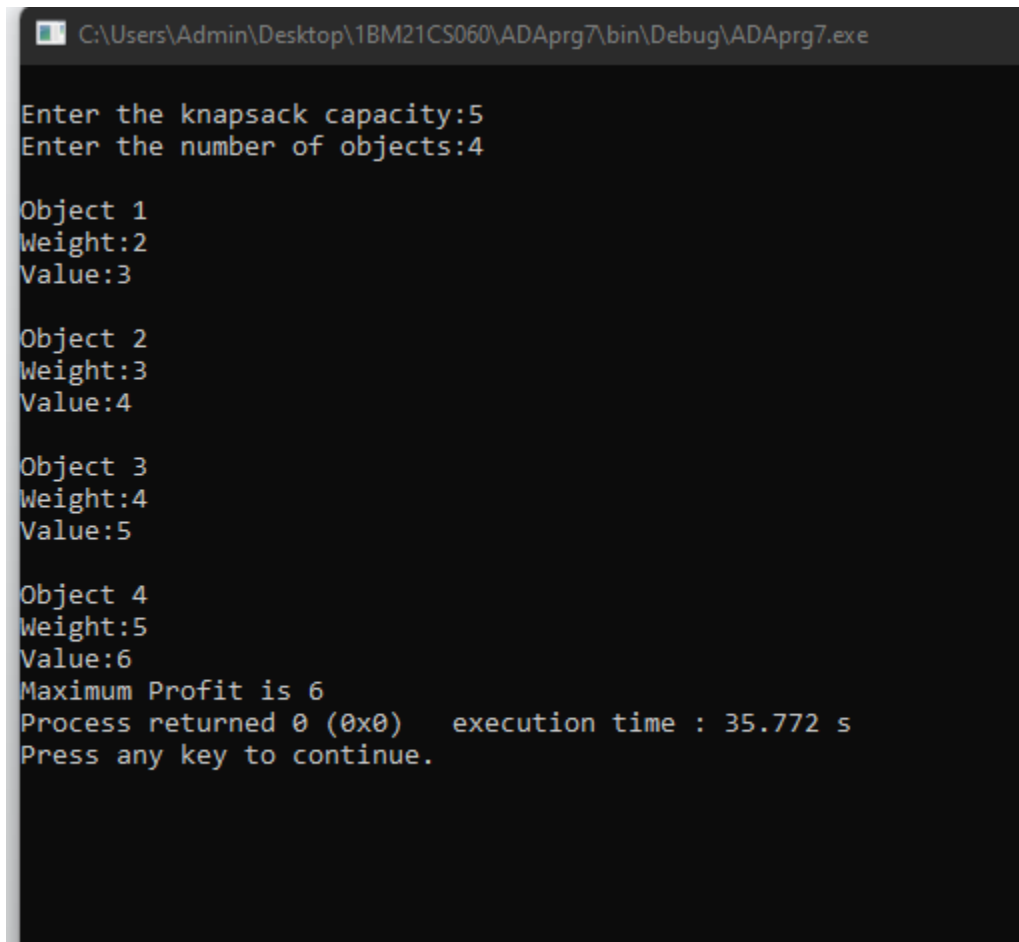
Laboratory Program 7

Implement 0/1 Knapsack problem using dynamic programming.

```
#include<stdio.h>
int main()
{
    int obj,i,j,knapsack,w[50],v[50];
    printf("\nEnter the knapsack capacity:");
    scanf("%d",&knapsack);
    printf("Enter the number of objects:");
    scanf("%d",&obj);
    for(i=0;i<obj;i++)
    {
        printf("\nObject %d\n",i+1);
        printf("Weight:");
        scanf("%d",&w[i]);
        printf("Value:");
        scanf("%d",&v[i]);
        /*w[i]=rand() % 100 + 1;
        v[i]=rand() % 100 + 1;
        */
    }
    int arr[obj+1][knapsack+1];
    for(i=0;i<=obj;i++)
    {
        for(j=0;j<=knapsack;j++)
        {
            if(i==0||j==0)
                arr[i][j]=0;
            else if(w[i]>j)
            {
                arr[i][j]=arr[i-1][j];
            }
            else
            {
                if(arr[i-1][j]>(arr[i-1][j-w[i]]+v[i]))
                    arr[i][j]=arr[i-1][j];
                else
                    arr[i][j]=arr[i-1][j-w[i]]+v[i];
            }
        }
    }
    printf("Maximum Profit is %d",arr[obj][knapsack]);
}
```

```
    return 0;  
}
```

Output



```
C:\Users\Admin\Desktop\1BM21CS060\ADAPrg7\bin\Debug\ADAPrg7.exe  
  
Enter the knapsack capacity:5  
Enter the number of objects:4  
  
Object 1  
Weight:2  
Value:3  
  
Object 2  
Weight:3  
Value:4  
  
Object 3  
Weight:4  
Value:5  
  
Object 4  
Weight:5  
Value:6  
Maximum Profit is 6  
Process returned 0 (0x0)   execution time : 35.772 s  
Press any key to continue.
```

Laboratory Program 8

Implement All Pair Shortest paths problem using Floyd's algorithm

```
#include<bits/stdc++.h>
#include<algorithm>
#define INF 99999
using namespace std;
int main()
{
    int v,i,j,k;
    cout<<"Enter the number of vertices:";
    cin>>v;
    int arr[5][5],dist[5][5];
    cout<<"Enter the graph:"<<endl;
    for(i=0;i<v;i++)
    {
        for(j=0;j<v;j++){
            cin>>arr[i][j];}

    }
    cout<<endl;
    for(i=0;i<v;i++)
    {
        for(j=0;j<v;j++)
            dist[i][j]=arr[i][j];
    }
    for(k=0;k<v;k++)
    {
        for(i=0;i<v;i++)
        {
            for(j=0;j<v;j++)
            {
                dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j]);
            }
        }
    }
    cout<<"Distance matrix is:"<<endl;
    for(i=0;i<v;i++)
    {
        for(j=0;j<v;j++)
        {
            if(dist[i][j]==INF)
```

```

        cout<<"INF"<<" ";
        else
        cout<<dist[i][j]<<" ";
    }
    cout<<endl;
}
return 0;
}

```

Output

```

bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ ./a.out
Enter the number of vertices:5
Enter the graph:
0 99999 3 2 99999
99999 0 99999 99999 99999
99999 99999 0 99999 99999
99999 99999 99999 0 99999
1 99999 99999 5 0

Distance matrix is:
0 INF 3 2 INF
INF 0 INF INF INF
INF INF 0 INF INF
INF INF INF 0 INF
1 INF INF 5 0
bmscecse@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~/ADALab_CS259$ 

```

Laboratory Program 9

Find Minimum Cost Spanning Tree of a given undirected graph using Prim's and Kruskal's algorithm.

```
#include<stdio.h>
#include<conio.h>
void prims();
int cost[10][10],vt[10],et[10][10],vis[10],j,n;
int sum=0;
int x=1;
int e=0;
int find(int v,int parent[10])
{
    while(parent[v]!=v)
    {
        v=parent[v];
    }
    return v;
}

void union1(int i,int j,int parent[10])
{
    if(i<j)
        parent[j]=i;
    else
        parent[i]=j;
}

void kruskal(int n,int a[10][10])
{
    printf("Kruskal's Algorithm\n");
    int count,k,min,sum,i,j,t[10][10],u,v,parent[10];
    count=0;
    k=0;
    sum=0;
    for(i=0;i<n;i++)
        parent[i]=i;
    while(count!=(n-1))
    {
        min=999;
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
```

```

        {
            if(a[i][j]<min && a[i][j]!=0)
            {
                min=a[i][j];
                u=i;
                v=j;
            }
        }
    }
    i=find(u,parent);
    j=find(v,parent);
    if(i!=j)
    {
        union1(i,j,parent);
        t[k][0]=u;
        t[k][1]=v;
        k++;
        count++;
        sum=sum+a[u][v];
    }
    a[u][v]=a[v][u]=999;
}
if(count==n-1)
{
    printf("Minimal spanning tree is:\n");
    for(i=0;i<n-1;i++)
    {
        printf("%d %d\n",t[i][0],t[i][1]);
    }
    printf("Cost of the minimal spanning tree is %d\n",sum);
}
else
    printf("Spanning tree does not exist.\n");
}

```

```

int main()
{
    int i,n1,ch,j;
    int cost1[10][10];
    printf("1.Prim's \n2.Kruskal's \nEnter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {

```



```

    case 1:
        printf("Enter the number of vertices:");
        scanf("%d",&n);
        printf("Enter the cost adjacency matrix\n");
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                scanf("%d",&cost[i][j]);
            }
            vis[i]=0;
        }
        prims();
        printf("Prim's Algorithm\n");
        printf("Edges of the minimal spanning tree are:");
        for(i=1;i<=e;i++)
        {
            printf("%d,%d\t",et[i][0],et[i][1]);
        }
        printf("Weight of the minimal spanning tree is %d\n",sum);
        break;
    case 2:

```

```

        printf("Enter the number of vertices:");
        scanf("%d",&n1);
        printf("Enter the cost adjacency matrix\n");
        for(i=0;i<n1;i++)
        for(j=0;j<n1;j++)
            scanf("%d",&cost1[i][j]);
        kruskal(n1,cost1);

```

```

default:return 0;
}

```

```

}
/* Prim's algorithm */
void prims()
{
    int s,min,m,k,u,v;
    vt[x]=1;
    vis[x]=1;
    for(s=1;s<n;s++)

```

```

{
    j=x;
    min=999;
    while(j>0)
    {
        k=vt[j];
        for(m=2;m<=n;m++)
        {
            if(vis[m]==0)
            {
                if(cost[k][m]<min)
                {
                    min=cost[k][m];
                    u=k;
                    v=m;
                }
            }
        }
        j--;
    }
    vt[++x]=v;
    et[s][0]=u;
    et[s][1]=v;
    e++;
    vis[v]=1;
    sum=sum+min;
}
}

```

Output

```
C:\Users\Admin\Desktop\1BM21CS060\ADAprg9\bin\Debug\ADAprg9.exe
1.Prim's
2.Kruskal's
Enter your choice:2
Enter the number of vertices:4
Enter the cost adjacency matrix
0 4 3 7
4 0 10 999
3 10 0 11
7 999 17 0
Kruskal's Algorithm
Minimal spanning tree is:
0 2
0 1
0 3
Cost of the minimal spanning tree is 14

Process returned 0 (0x0)   execution time : 24.396 s
Press any key to continue.
```

```
C:\Users\Admin\Desktop\1BM21CS060\ADAprg9\bin\Debug\ADAprg9.exe
1.Prim's
2.Kruskal's
Enter your choice:1
Enter the number of vertices:4
Enter the cost adjacency matrix
0 4 3 7
4 0 10 999
3 10 0 11
7 999 17 0
Prim's Algorithm
Edges of the minimal spanning tree are:1,3    1,2    1,4    Weight of the minimal spanning tree is 14

Process returned 0 (0x0)   execution time : 23.309 s
Press any key to continue.
```

Laboratory Program 10

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

Laboratory Program 11

Implement "N-Queens Problem" using Backtracking.