East West University
Department of Computer Science and Engineering

Course: CSE 246(Algorithms)
Section - 02

Submitted To :	**Submitted By :**
Dr. Md. Tauhid Bin Iqbal Assistant Professor Department of CSE East West University	**Name** : Hafsa Ferdousi **Student ID** : 2023-3-60-321 **Department** : CSE **Date** : 22 / 11 / 2025

Lab Report:06

Activity Selection Problem:

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

struct Activity {
    int start, finish;
};

bool compareActivities(Activity a, Activity b) {
    return a.finish < b.finish;
}

void GreedyActivitySelection(vector<Activity> &activities) {



    sort(activities.begin(), activities.end(), compareActivities);

    cout << "Selected Activities:\n";


    cout << "(Start: " << activities[0].start << ", Finish: "
         << activities[0].finish << ")\n";

    int lastSelected = 0;

    for (int i = 1; i < activities.size(); i++) {
        if (activities[i].start >= activities[lastSelected].finish) {
            cout << "(Start: " << activities[i].start << ", Finish: " << activities[i].finish << ")\n";
            lastSelected = i;
        }
    }
}

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

    vector<Activity> activities(n);

    cout << "Enter start and finish times:\n";
    for (int i = 0; i < n; i++) {
```

```
        cin >> activities[i].start >> activities[i].finish;
    }

    GreedyActivitySelection(activities);

    return 0;
}
```

```
Enter number of activities: 5
Enter start and finish times:
3 5
2 4
1 5
1 3
2 3
Selected Activities:
(Start: 1, Finish: 3)
(Start: 3, Finish: 5)

Process returned 0 (0x0)   execution time : 60.788 s
Press any key to continue.
```

Job Sequencing Problem:

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

struct Job {
    char id;
    int deadline;
    int profit;
};

bool compare(Job a, Job b) {

    return a.profit > b.profit;
}

void JobScheduling(vector<Job> &jobs) {
```

```
    sort(jobs.begin(), jobs.end(), compare);


    int maxDeadline = 0;

    for (auto &job : jobs) {
        maxDeadline = max(maxDeadline, job.deadline);
    }


    vector<char> slot(maxDeadline + 1, '-');

    int totalProfit = 0;
    vector<char> result;


    for (auto &job : jobs) {


        for (int t = job.deadline; t >= 1; t--) {

            if (slot[t] == '-') {

                slot[t] = job.id;

                totalProfit += job.profit;
                result.push_back(job.id);
                break;
            }
        }
    }

    cout << "Selected Jobs: ";
    for (char x : result) cout << x << " ";
    cout << endl;

    cout << "Total Profit: " << totalProfit << endl;
}

int main() {
    vector<Job> jobs = {
        {'A', 2, 100},
```

```
        {'B', 1, 19},
        {'C', 2, 27},
        {'D', 1, 25},
        {'E', 3, 15}
    };

    JobScheduling(jobs);

    return 0;
}
```

```
Selected Jobs: A C E
Total Profit: 142

Process returned 0 (0x0)   execution time : 0.143 s
Press any key to continue.
```

Huffman Coding Problem:

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

#define MAX 256
struct Node {
    char ch;
    int freq;
    int left;
    int right;
};

void findTwoMin(Node nodes[], int n, bool used[], int &min1, int &min2) {
    min1 = -1;
    min2 = -1;

    for (int i = 0; i < n; i++) {
        if (used[i]) continue;
        if (min1 == -1 || nodes[i].freq < nodes[min1].freq) {
            min2 = min1;
```

```
            min1 = i;
        } else if (min2 == -1 || nodes[i].freq < nodes[min2].freq) {
            min2 = i;
        }
    }
}
void printCodes(Node nodes[], int root, string code) {
    if (root == -1) return;


    if (nodes[root].left == -1 && nodes[root].right == -1) {
        cout << nodes[root].ch << " : " << code << endl;
        return;
    }

    printCodes(nodes, nodes[root].left, code + "0");
    printCodes(nodes, nodes[root].right, code + "1");
}

int main() {

    char text[] = "Algorithms";

    int freq[MAX] = {0};
    for (int i = 0; text[i] != '\0'; i++)
        freq[(unsigned char)text[i]]++;


    Node nodes[MAX];
    bool used[MAX] = {0};
    int n = 0;


    for (int i = 0; i < MAX; i++) {
        if (freq[i] > 0) {
            nodes[n].ch = (char)i;
            nodes[n].freq = freq[i];
            nodes[n].left = -1;
            nodes[n].right = -1;
            n++;
        }
    }

  int totalNodes = n;
```

```
    while (true) {
        int min1, min2;
        findTwoMin(nodes, totalNodes, used, min1, min2);
        if (min2 == -1) break;

        nodes[totalNodes].ch = '\0';
        nodes[totalNodes].freq = nodes[min1].freq + nodes[min2].freq;
        nodes[totalNodes].left = min1;
        nodes[totalNodes].right = min2;


        used[min1] = true;
        used[min2] = true;
        used[totalNodes] = false;

        totalNodes++;
    }


    int root = -1;
    for (int i = 0; i < totalNodes; i++)
        if (!used[i]) { root = i; break; }


    cout << "Huffman Codes:\n";
    printCodes(nodes, root, "");

    return 0;
}
```

```
Huffman Codes:
l : 000
m : 001
o : 010
r : 011
s : 100
t : 101
A : 1100
g : 1101
h : 1110
i : 1111

Process returned 0 (0x0)   execution time : 0.188 s
Press any key to continue.
```