DAA

Assignment2

Ojaswi Kumar

102296002

2CS11

Write a program to solve the following problems using greedy Greedy approach:

Activity Selection

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

struct Activity {
    int start;
    int finish;
};

bool activityCompare(Activity a1, Activity a2) {
    return (a1.finish < a2.finish);
}

void printMaxActivities(Activity arr[], int n) {
    sort(arr, arr+n, activityCompare);
    int i = 0;
    cout << "Selected activities: " << i << " ";
    for (int j = 1; j < n; j++) {
        if (arr[j].start >= arr[i].finish) {
            cout << j << " ";
            i = j;
        }
    }
}

int main() {
    Activity arr[] = {{5, 9}, {1, 2}, {3, 4}, {0, 6}, {5, 7}, {8, 9}};
    int n = sizeof(arr) / sizeof(arr[0]);
    printMaxActivities(arr, n);
    return 0;
```

```
}
```

2)Job Sequencing:

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

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

bool jobCompare(Job a, Job b) {
    return (a.profit > b.profit);
}

void printJobSequence(Job arr[], int n) {
    sort(arr, arr+n, jobCompare);
    vector<int> result(n);
    vector<bool> slot(n);
    fill(slot.begin(),slot.end(),false);

    for (int i=0; i<n; i++) {
        for (int j=min(n, arr[i].deadline)-1; j>=0; j--) {
            if (!slot[j]) {
                result[j] = i;
                slot[j] = true;
                break;
            }
        }
    }

    cout << "Jobs sequence is: ";
    for (int i=0; i<n; i++)
        if (slot[i])
            cout << arr[result[i]].id << " ";
}

int main() {
    Job arr[] = {{1, 2, 50}, {2, 1, 40}, {3, 2, 35}, {4, 3, 30}};
    int n = sizeof(arr) / sizeof(arr[0]);
    printJobSequence(arr, n);
    return 0;
}
```

3) Fractional knapsack :

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

struct Item {
    int value;
    int weight;
};

bool cmp(Item a, Item b) {
    double r1 = (double)a.value / a.weight;
    double r2 = (double)b.value / b.weight;
    return r1 > r2;
}

double fractionalKnapsack(int W, Item arr[], int n) {
    sort(arr, arr + n, cmp);
    double totalValue = 0.0;
    for (int i = 0; i < n; i++) {
        if (W == 0) return totalValue;
        int a = min(arr[i].weight, W);
        totalValue += a * ((double)arr[i].value / arr[i].weight);
        W -= a;
    }
    return totalValue;
}

int main() {
    int W = 50;
    Item arr[] = {{60, 10}, {100, 20}, {120, 30}};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "Maximum value we can obtain = " << fractionalKnapsack(W, arr, n)
<< endl;
    return 0;
}
```

Additional Questions :

1)

```
1) #include <iostream>
2) using namespace std;
3)
4) int majorityElement(int arr[], int n) {
5)     int count = 0;
6)     int majority;
7)     for (int i = 0; i < n; i++) {
8)         if (count == 0) {
9)             majority = arr[i];
10)            count = 1;
11)        }
12)        else {
13)            if (arr[i] == majority) {
14)                count++;
15)            }
16)            else {
17)                count--;
18)            }
19)        }
20)    }
21)    return majority;
22)}
23)
24)int main() {
25)    int arr[] = {2, 2, 3, 5, 2, 2, 6};
26)    int n = sizeof(arr)/sizeof(arr[0]);
27)    cout << majorityElement(arr, n) << endl;
28)    return 0;
29)}
30)
```

2)

```
#include <iostream>
#include <algorithm>
#include <Vector>

using namespace std;

int minCandies(int ratings[], int n) {
    vector<int> candies(n);
    candies[0] = 1;
    for (int i = 1; i < n; i++) {
        if (ratings[i] > ratings[i-1]) {
            candies[i] = candies[i-1] + 1;
        } else {
            candies[i] = 1;
```

```
        }
    }
    for (int i = n-2; i >= 0; i--) {
        if (ratings[i] > ratings[i+1]) {
            candies[i] = max(candies[i], candies[i+1] + 1);
        }
    }
    int totalCandies = 0;
    for (int i = 0; i < n; i++) {
        totalCandies += candies[i];
    }
    return totalCandies;
}

int main() {
    int ratings[] = {1, 3, 2, 2, 4};
    int n = sizeof(ratings) / sizeof(ratings[0]);
    cout << minCandies(ratings, n) << endl;
    return 0;
}
```

3)

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

int max_courses(int start, int end, vector<vector<int>>& courses) {
    int count = 0, end_day = 0;
    for (int i = 0; i < courses.size(); i++) {
        if (courses[i][0] >= start && courses[i][1] <= end) {
            if (courses[i][1] > end_day) {
                count++;
                end_day = courses[i][1];
            }
        }
    }
    return count;
}

int main() {
    int N, Q;
    cin >> N >> Q;
    vector<vector<int>> courses(N, vector<int>(2));
    for (int i = 0; i < N; i++) {
        cin >> courses[i][0] >> courses[i][1];
```

```
    }
    for (int i = 0; i < Q; i++) {
        int start, end;
        cin >> start >> end;
        cout << max_courses(start, end, courses) << endl;
    }
    return 0;
}
```

4)

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

struct Customer {
    int arrival;
    int departure;
    int compartment;
};

bool cmp(Customer a, Customer b) {
    return a.arrival < b.arrival;
}

int main() {
    int T;
    cin >> T;
    while (T--) {
        int N, K;
        cin >> N >> K;
        vector<Customer> customers(N);
        for (int i = 0; i < N; i++) {
            cin >> customers[i].arrival >> customers[i].departure >>
customers[i].compartment;
        }
        sort(customers.begin(), customers.end(), cmp);
        vector<int> compartments(K + 1, -1);
        int count = 0;
        for (int i = 0; i < N; i++) {
            int j;
            for (j = customers[i].compartment; j <= K; j++) {
                if (compartments[j] <= customers[i].arrival) {
                    compartments[j] = customers[i].departure;
                    count++;
                    break;
                }
```

```
            }
            if (j > K) {
                for (int k = 1; k <= K; k++) {
                    if (compartments[k] <= customers[i].arrival &&
compartments[k] >= customers[i].departure) {
                        compartments[k] = customers[i].departure;
                        count++;
                        break;
                    }
                }
            }
        }
        cout << count << endl;
    }
    return 0;
}
```

Max Sub – Array :

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

int max_subarray_sum(int arr[], int n) {
    int max_sum = 0;
    int current_sum = 0;

    for (int i = 0; i < n; i++) {
        current_sum += arr[i];
        if (current_sum < 0) {
            current_sum = 0;
        }
        if (current_sum > max_sum) {
            max_sum = current_sum;
        }
    }
    return max_sum;
}

int main() {
    int arr[] = {-2, -3, 4, -1, -2, 1, 5, -3};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "Maximum sum of contiguous subarray is: " << max_subarray_sum(arr,
n);
```

```
    return 0;
}
```