Greedy Algorithms

Prepared by Jakir Hasan

14/11/21

Content

1. Conference Scheduling
2. Job Scheduling with Deadline
3. Fractional Knapsack

Conference Scheduling

Given a set of activities (conferences) - each conference has a start time and a finish time.

Conf. id	1	2	3	4	5	6	7	8	9	10	11
Start	1	3	0	5	3	5	6	8	8	2	12
Finish	4	5	6	7	8	9	10	11	12	13	14

What is the maximum number of activities that can be completed?

Implementation

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


struct conference
{
    int id;
    int start;
    int finish;
};


bool comparison(conference a, conference b)
{
    return (a.finish <= b.finish);
```

```cpp
}


int main()
{
    int n, i, j, id, start, finish, previous_finish;
    conference conferences[10000];
    vector<int> ids;
    cin>> n;

    for (i = 0; i < n; i++)
    {
        cin>> start >> finish;

        conferences[i].id = i;
        conferences[i].start = start;
        conferences[i].finish = finish;
    }

    sort(conferences, conferences + n, comparison);
    id = conferences[0].id;
    previous_finish = conferences[0].finish;

    ids.push_back(id);

    for (i = 1; i < n; i++)
    {
        id = conferences[i].id;
        start = conferences[i].start;
        finish = conferences[i].finish;

        if (start >= previous_finish)
        {
            ids.push_back(id);
            previous_finish = finish;
        }
    }

    for (i = 0; i < ids.size(); i++)
        cout<< ids[i] << " ";
    cout<< "\n";

    return 0;
```

```
}


/*
11
1 4
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
*/
```

@Credit: Slides of Masum Sir

Job Scheduling with Deadlines

Statement
If there are a set of jobs which are associated with deadlines di >= 0 and profit pi > 0. For any job, profit is only earned if and only if the job is completed by its deadline.

Objective
Find a sequence of jobs, which are completed within their deadlines and give maximum profit.

Constraint
Any job takes a single unit of time to execute and any job cannot be completed beyond it’s deadline.

Task	Deadline	Profit
1	9	15

2	2	2
3	5	18
4	7	1
5	4	25
6	2	20
7	5	8
8	7	10
9	4	12
10	3	5

Implementation

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

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


bool comparison(job a, job b)
{
    return (a.profit > b.profit);
}

int main()
{
    int n, i, id, deadline, profit, slots[10000], total_profit = 0;
    vector<int> ids;
    job jobs[10000];

    cin>> n;
    for (i = 0; i < n; i++)
    {
        cin>> deadline >> profit;

        jobs[i].id = i;
        jobs[i].deadline = deadline;
        jobs[i].profit = profit;
    }

    sort(jobs, jobs + n, comparison);
    memset(slots, 0, sizeof(slots));

    for (i = 0; i < n; i++)
    {
        id = jobs[i].id;
        deadline = jobs[i].deadline;
        profit = jobs[i].profit;

        while (deadline >= 1)
        {
            if (slots[deadline] == 0)
            {
                total_profit += profit;
                slots[deadline] = 1;

                ids.push_back(id);
                break;
            }
            deadline -= 1;
        }
    }

    cout<< total_profit << "\n";

    for (i = 0; i < ids.size(); i++)
        cout<< ids[i] << " ";
    cout<< "\n";

    return 0;
}

/*
10
```

```
9 15
2 2
5 18
7 1
4 25
2 20
5 8
7 10
4 12
3 5
*/
```

@Credit: Slides of Masum Sir

Fractional Knapsack

Knapsack Problem

A thief robbing a store and can carry a maximal weight of w into his knapsack. There are n items and the ith item weighs wi and its worth is vi dollars. What items should the thief take?

Constraint

The knapsack weight capacity is not exceeded and the total benefit is maximal.

Fractional Knapsack

Items are divisible.

Implementation

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

struct item
{
    int id;
    double weight;
    double profit;
    double density;
};


bool comparison(item a, item b)
{
    return (a.density > b.density);
}


int main()
{
    int n, i, j, id, index;
    item items[10000];
    double weight, profit, density, capacity, total_profit = 0,
total_weight;

    cin>> n >> capacity;

    for (i = 0; i < n; i++)
    {
        cin>> weight >> profit;
        density = profit / weight;

        items[i].id = i;
        items[i].weight = weight;

        items[i].profit = profit;
        items[i].density = density;
    }

    sort(items, items + n, comparison);

    for (i = 0; i < n; i++)
    {
        id = items[i].id;
        weight = items[i].weight;

        profit = items[i].profit;
        density = items[i].density;
        if (weight <= capacity)
        {
            total_profit += profit;
            capacity -= weight;
        }
        else
        {
            total_profit += (capacity * density);
            break;
        }
    }

    printf("Maximum profit is %lf\n", total_profit);

    return 0;
}

/*
3 50
20 100
10 60
30 120
*/
```

@Credit: Slides of Masum Sir