Assignment 7: Greedy Algorithms

Due: 12th week

Note: You are strongly encouraged to learn how to solve problems for all sections for the quiz and CSE 203 final.

Problem 1 (for all sections)

Huffman coding: Given a set of n characters and their frequencies $f_1, ..., f_n$ in some text, construct Huffman codes for the characters (Reference: Cormen *et al.* – Section 16.3). You need to use the heap you implemented earlier in the course and your algorithm should run in time O($n \log n$).

Input:

First line of the input file will contain the number of characters, n and followed by n lines each containing a character and its frequency separated by a space. For example:

```
6
a 45
b 13
c 12
d 16
e 9
f 5
```

Output:

Huffman codes for the characters. Output for the above input should be

```
a 0
b 101
c 100
d 111
e 1101
f 1100
```

Note: The actual codes may be different but they should have the same lengths as the codes given above.

Problem 2 for Section B2

Interval scheduling: You have a set of *n* requests where the i-th request is specified by an interval starting at s_i and finishing at f_i. A subset of the requests is compatible if no two of them overlap in time. Give an algorithm to find a compatible subset of maximum size that runs in time O(*n log n*). (Reference: Kleinberg & Tardos – Section 4.1).

Input:

First line of the input file will contain the number of requests, *n* followed by starting and finishing times of a request in each line separated by a space. For example:

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

Output:

The number of intervals selected and their starting and finishing times. Output for the above input should be

```
4
1 4
5 7
7 11
12 14
```

Problem 2 for Section B1

Coin changing problem: You are given *n* types of notes given by their denominations $d_1, ..., d_n$ (for example 1 tk, 2 tk, 5 tk, 10 tk, 20 tk, 50 tk, 100 tk, 500 tk, 1000 tk) and a value, *V*. Give an algorithm to make change for *V* using fewest number of notes that runs in O(*n log n*) time. You have unlimited number of notes for each type and you may assume that the coin system is canonical *i.e.* greedy algorithm will give an optimal solution.

Input:

First line of the input file will contain the number of types of notes, *n* and second line will contain *n* integers separated by spaces giving the denominations (they may not be sorted). The third line will have the value, *V*. For example:

```
9
5 2 10 1000 1 500 20 100 50
2667
```

Output:

The total number of notes needed and the number of times each note is picked. Output for the above input should be

```
8
1000 2
500 1
100 1
50 1
10 1
5 1
2 1
```

Problem 2 for Section A2

Minimizing waiting times: A server has *n* customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer *i*. So if, for example, the customers are served in order of increasing *i*, then the *i*-th customer has to wait $\sum_{j=1}^{i} t_j$ minutes.

We wish to minimize the total waiting time $T = \sum_{i=1}^{n}(waiting\ time\ for\ customer\ i)$

Give an algorithm to find the order in which customers should be serviced to minimize total waiting time that runs in O(*n log n*) time.

Input:

First line of the input file will contain the number of customers, *n* followed by service times of customers in the second line separated by spaces. For example:

11
10 24 5 11 67 21 8 97 32 9 41

Output: The order in which customers should be serviced (customer number starting at 1).

3 7 10 1 4 6 2 9 11 5 8

Problem 2 Section A1

Fractional Knapsack: You are given *n* items, their weights $w_1, ..., w_n$ and their values $v_1, ..., v_n$ as well as a capacity of a knapsack, W. You want to pack items in the knapsack in such way that total value of items taken is maximized and total weight of items taken does not exceed knapsack capacity. You are allowed to take fractions of items. Give an algorithm for the problem that runs in O(*n log n*) time. (Reference: Cormen *et al.* – Section 16.2)

Input:

First line of the input file will contain the number of items, *n* followed by two numbers per line giving weight and values of items separated by spaces. The last line will contain capacity of the knapsack. For example:

5
10 50
4 60
20 40
3 21
8 48
20

Output: To total value obtained. The items taken, how much (weights) of each item taken and value obtained from each item.

154
2 4 60
4 3 21
5 8 48
1 5 25

- Contact Atif Hasan Rahman if you have queries