Problem Sheet 4

18XD56 : Design and Analysis of Algorithms Lab

Greedy Algorithms

1. *Activity Selection Problem*

You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.

Example:

Example 1 : Consider the following 3 activities sorted by

by finish time.

```
start[]  =  {10, 12, 20};

finish[] =  {20, 25, 30};
```

A person can perform at most **two** activities. The maximum set of activities that can be executed is {0, 2}.

Example 2 : Consider the following 6 activities

sorted by by finish time.

```
start[]  =  {1, 3, 0, 5, 8, 5};

finish[] =  {2, 4, 6, 7, 9, 9};
```

A person can perform at most **four** activities. The maximum set of activities that can be executed is {0, 1, 3, 4}.

2. Activity selection problem with K persons

Given two arrays **S[]** and **E[]** of size **N** denoting starting and closing time of the shops and an integer value **K** denoting the number of people, the task is to find out the maximum number of shops they can visit in total if they visit each shop optimally based on the following conditions:

- A shop can be visited by only one person
- A person cannot visit another shop if its timing collide with it

Examples:

Input: *S[] = {1, 8, 3, 2, 6}, E[] = {5, 10, 6, 5, 9}, K = 2*

Output: *4*
Explanation: *One possible solution can be that first person visits the 1st and 5th shops meanwhile second person will visit 4th and 2nd shops.*

Input: *S[] = {1, 2, 3}, E[] = {3, 4, 5}, K = 2*

Output: *3*

Explanation: *One possible solution can be that first person visits the 1st and 3rd shops meanwhile second person will visit 2nd shop.*

3.Huffman Coding

Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.
Let us understand prefix codes with a counter example. Let there be four characters a, b, c and d, and their corresponding variable length codes be 00, 01, 0 and 1. This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to

a and b. If the compressed bit stream is 0001, the de-compressed output may be “cccd” or “ccb” or “acd” or “ab”. Write an algorithm to

1) Build a Huffman Tree from input characters.
2) Traverse the Huffman Tree and assign codes to characters.

4. Find Intersection of all Intervals

Given **N** intervals of the form of **[l, r]**, the task is to find the intersection of all the intervals. An intersection is an interval that lies within all of the given intervals. If no such intersection exists then print **-1**.
Examples:
Input: *arr[] = {{1, 6}, {2, 8}, {3, 10}, {5, 8}}*
Output: *[5, 6]*
[5, 6] is the common interval that lies in all the given intervals.
Input: *arr[] = {{1, 6}, {8, 18}}*
Output: *-1*
No intersection exists between the two given ranges.

5. Assign Mice to Holes

There are N Mice and N holes are placed in a straight line. Each hole can accommodate only 1 mouse. A mouse can stay at his position, move one step right from x to x + 1, or move one step left from x to x -1. Any of these moves consumes 1 minute. Assign mice to holes so that the time when the last mouse gets inside a hole is minimized.

Examples:

```
Input : positions of mice are:
          4 -4 2
        positions of holes are:
          4 0 5
Output :  4
Assign mouse at position x = 4 to hole at
position x = 4 : Time taken is 0 minutes
Assign mouse at position x=-4 to hole at
position x = 0 : Time taken is 4 minutes
```

```
Assign mouse at position x=2 to hole at
position x = 5 : Time taken is 3 minutes
After 4 minutes all of the mice are in the holes.
Since, there is no combination possible where
the last mouse's time is less than 4,
answer = 4.

Input :  positions of mice are:
        -10, -79, -79, 67, 93, -85, -28, -94
          positions of holes are:
         -2, 9, 69, 25, -31, 23, 50, 78
Output : 102
```

6. Fitting Shelves Problem

Given length of wall w and shelves of two lengths m and n, find the number of each type of shelf to be used and the remaining empty space in the optimal solution so that the empty space is minimum. The larger of the two shelves is cheaper so it is preferred. However cost is secondary and first priority is to minimize empty space on wall.

Examples:

```
Input : w = 24 m = 3 n = 5
Output : 3 3 0
We use three units of both shelves
and 0 space is left.
3 * 3 + 3 * 5 = 24
So empty space  = 24 - 24 = 0
Another solution could have been 8 0 0
but since the larger shelf of length 5
is cheaper the former will be the answer.

Input : w = 29 m = 3 n = 9
```

```
Output : 0 3 2

0 * 3 + 3 * 9 = 27

29 - 27 = 2


Input : w = 24 m = 4 n = 7

Output : 6 0 0

6 * 4 + 0 * 7 = 24

24 - 24 = 0
```

7. Minimum Swaps for Bracket Balancing

You are given a string of 2N characters consisting of N '[' brackets and N ']' brackets. A string is considered balanced if it can be represented in the for S2[S1] where S1 and S2 are balanced strings. We can make an unbalanced string balanced by swapping adjacent characters. Calculate the minimum number of swaps necessary to make a string balanced.

Examples:

```
Input  : []][][

Output : 2

First swap: Position 3 and 4    [][]][

Second swap: Position 5 and 6   [][][]


Input  : [[][]]

Output : 0

String is already balanced.
```