INDIAN STATISTICAL INSTITUTE

MTech(CS) I year   2020-2021

Subject: Computing Laboratory

Lab Test 2 (8 February, 2021)

Total: 60 marks   Duration: 3 hours

INSTRUCTIONS

1. You may consult or use slides / programs provided to you as course material, but please do not consult or use material from other Internet sources, your classmates, or anyone else.
2. You may write your programs in C/C++ or Python.
3. Please make sure that your programs adhere strictly to the specified input and output format.
4. Please adhere to the file naming conventions discussed in class (i.e., your file names should be of the form `cs20XX-labtest2-progY` with a `.c` or `.py` extension). Note that your file names **do not** need to contain a 6-8 character long password like alphanumeric string.
5. Please upload your programs to `https://www.dropbox.com/request/N256WowL5A8PV9LbxIND`.

Q1. This question contains two parts. You should write a **single** program for this problem. Your program should take a **single letter** (`a`, or `b`) as its only command-line argument. Depending on the command-line argument, it should produce the desired output for part (a) or (b) respectively.

Consider a binary tree T, in which each node v_i contains a pair of positive integers $[x_i, y_i]$, representing the interval $I_i = \{j \mid j \in \mathbb{N},\ x_i \leq j \leq y_i\}$.

Define the *span*, $S(T)$ of such a tree T as follows.

- If T is empty, $S(T) = \{\}$.
- If T consists of a single node containing the interval I, then $S(T) = I$.
- If the root of T contains the interval I, then $S(T)$ is the smallest interval $[X, Y]$ that contains I, the span of the left subtree of T and the span of the right subtree of T. For example, if the root of T contains $[25, 30]$, and the spans of the left and right subtrees of T are $[1, 2]$ and $[14, 15]$ respectively, then $S(T) = [1, 30]$.

(a) Write a function that computes, for each node v_i of T, the span of the subtree rooted at v_i. You will get full credit only if the time complexity of your algorithm is linear in the number of nodes.

(b) Assuming that T contains at least two nodes, write a function that partitions T into 2 trees T_1 and T_2 by removing exactly one edge in such a way that $|S(T_1) \cap S(T_2)|$ is maximised.

[15+25]

Input format

For a binary tree with n nodes, the input will consist of $n+1$ lines. The first line will contain a single integer $n \geq 2$, specifying the number of nodes in the tree. Each of the next n lines corresponds to a tree node v_i, and consists of **four** integers corresponding to x_i, y_i, and the indices of its left and right child nodes, respectively. An index of -1 in either the third or fourth field indicates that the corresponding child is absent. Note that nodes are indexed starting from 0; the node with index 0 corresponds to the root of the tree.

Output format

For (a), the output format will be very similar to the input format, but for each node v, the first two fields should contain the lower and upper limits of the span of the subtree rooted at v.

For (b), your program should simply output the index of the vertex that should be detached from its parent in order to obtain the desired partition.

Sample input 0

```
7
8   8 -1  1
1  20  2  3
19 30  4  5
5  15  6 -1
5   6 -1 -1
2  10 -1 -1
8  21 -1 -1
```

Sample output 0

For part (a)

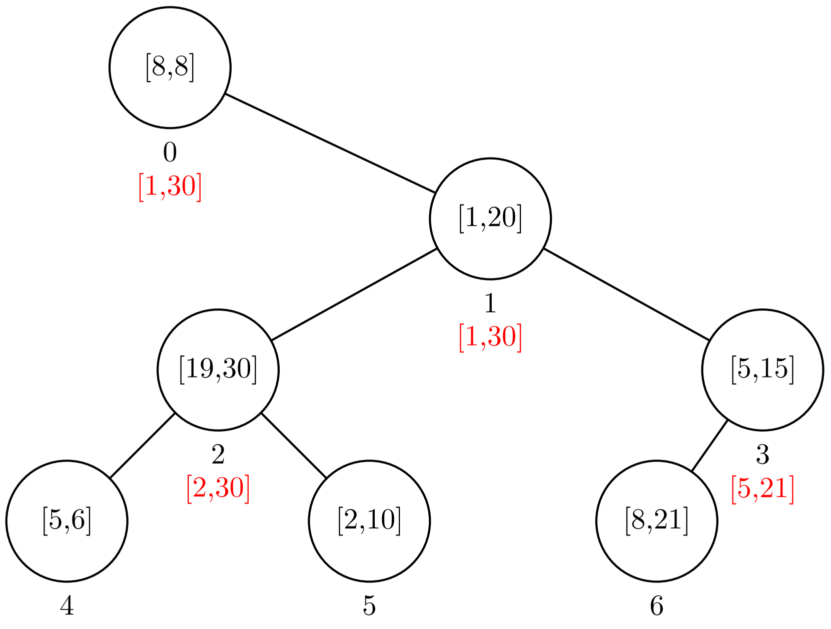
```
7
1 30  -1 1
1 30  2  3
2 30  4  5
5 21  6 -1
5  6 -1 -1
2 10 -1 -1
8 21 -1 -1
```

For part(b)

```
2
```

Explanation

The input corresponds to the binary tree shown in the following figure. The span for each non-leaf-node is shown in red (the span for a leaf-node is identical to the interval for that node).



The span at node 2 must be the smallest interval containing $[5,6]$, $[2,10]$ and $[19,30]$. Thus, $S(2) = [2,30]$. Similarly, $S(3)$ must contain both $[8,21]$ and $[5,15]$; therefore, $S(3) = [5,21]$.

For this tree, we have to choose one of the 6 edges for deletion. The set $S(T_1) \cap S(T_2)$ for the two resulting subtrees T_1 and T_2 are shown below for each case.

- Delete edge 0-1: $|S(T_1) \cap S(T_2)| = |[8,8] \cap [1,30]| = |[8,8]| = 1$
- Delete edge 1-2: $|S(T_1) \cap S(T_2)| = |[2,30] \cap [1,21]| = |[2,21]| = 20$
- Delete edge 1-3: $|S(T_1) \cap S(T_2)| = |[1,30] \cap [5,21]| = |[5,21]| = 17$
- Delete edge 2-4: $|S(T_1) \cap S(T_2)| = |[5,6] \cap [1,30]| = |[5,6]| = 2$
- Delete edge 2-5: $|S(T_1) \cap S(T_2)| = |[1,30] \cap [2,10]| = |[2,10]| = 9$
- Delete edge 3-6: $|S(T_1) \cap S(T_2)| = |[8,21] \cap [1,30]| = |[8,21]| = 14$

Deleting edge 1-2 produces the desired partition, since $|S(T_1) \cap S(T_2)|$ is the largest in this case. The desired output is therefore 2.

Q2. Let us define a pair of integers, say s_i and s_j, as `NEARLY MIRROR` if $s_i < 0, s_j > 0$, and $|s_i| - s_j = 1$. Given a set $S = \{s_1, s_2, \ldots, s_n\}$, of 8-bit binary numbers, write a program to identify whether there exists any pair of `NEARLY MIRROR` numbers. [20]

Input Format

Inputs, in the form of binary numbers, will be provided as command-line arguments. Negative numbers will be represented in 2's complement form.

Output Format

Output is to be printed on the standard output. If there exists any pair of `NEARLY MIRROR` numbers print them (in the same order of their appearance in the list). Otherwise print `NIL`. If there are multiple `NEARLY MIRROR` numbers in the input print them pairwise in separate lines.

Command-line Arguments

```
./prog1 <signed number 1> <signed number 2> ...
```

Sample Input 0

```
./prog1 00100010 11111010 11111100 00000011 10001100 00000001
```

Sample Output 0

```
11111100 00000011
```

Sample Input 1

```
./prog1 11110100 00000010 11111011 00000000 00000011 00001100 00001011 11111111
```

Sample Output 1

```
11110100 00001011
```

Sample Input 2

```
./prog1 11111011 00000101 11111100
```

Sample Output 2

```
NIL
```