ESO 207 Midsem Exam Solutions

Q1(20)

```
makeBST(T,L){
    if not (T.root == nil)
        Sort(L)
        makeBST1(T.root,L)
}
makeBST1(x,L){
    i=1
    y=min(x)
    while (i <= L.length) do
        y.data = L[i]
        y = succ(y)
        i=i+1
} // Here min(x) and succ(y) are the functions we saw in lecture-11.
```

For the time complexity analysis, let n be the number of nodes in the tree. This is also the number of elements in L.

Sorting takes $O(n \ log \ n)$ steps, in the worst case. The while loop is executed n times. The time taken by min and total time taken by all calls to $succ$ is $O(n)$. This is because in this sequence of successive calls to $succ$ each edge in T is traversed at most twice (once in downward and once in upward direction). [See exercise 12.2-7 in the textbook.]

The total worst case time is therefore $O(n \ log \ n)$.

[There are other ways to write code which also work in $O(n\ log\ n)$ time. For example, we could keep L and an index i into L as global variables and visit the tree in inorder using usual recursive algorithm. Whenever a tree node is visited, we write into it $L[i]$ and increment i by 1.

One can also give a divide and conquer algorithm and implement it by a recursive code for this problem, but that is $O(n^2)$, if written in a straight-forward way.]

Q2(7+10+10)

(a) It is shown in figure Q2(a). Other sequences of left rotations are possible.

(b) We prove this by induction on the number of nodes in a tree T. Fig Q2(b) starts with a general case, in the first step LR is applied at the root. In the next step we inductively convert tree rooted at x to left-linear form. Next, inductive hypothesis is applied to the tree on right side of the black vertical line. The crucial observation is written in green in the figure.

(c) This is also proved by induction on the number of nodes in a tree R. Figure Q2(c), shows a series of RR (right rotations) applied to R to get a tree T s.t. root of T is the root node of U and left and right subtrees of T are left-linear. Further, left (right) subtree of T has the same set of nodes as the left (right) subtree of U. Applying inductive hypothesis on left and right subtrees of T finishes the argument.

(d)[Optional] For transformation of part (b), we can count the number of rotations inductively. Two applications of inductive hypothesis take $|T_1| + |T_2|$ and $|T_3|$ steps respectively, where $|T_i|$ is the number of node in tree T_i. Adding the very first LR rotation gives the total number of rotations as $|T_1| + |T_2| + |T_3| + 1$, which is $n - 1$. (Special cases, where some of the T_i are empty trees can also be easily verified). For part (c), the book states $O(n)$ bound but I do not have a proof of it. As an easy exercise you may show that transformation given in part (c) takes $\Omega(n^2)$ rotations in the worst case.

Q3(3+(5+10)+15)

(a) In this representation a node may be pointed to by many nodes. So, parent of a node is not defined (it may be different for different trees). To enforce, unique parent for each node will result in no sharing of nodes.

(b) In the steps of insertion/deletion whenever node u changes, all nodes on the path from the root to u are copied. Such nodes are easily identified during insertion/deletion.

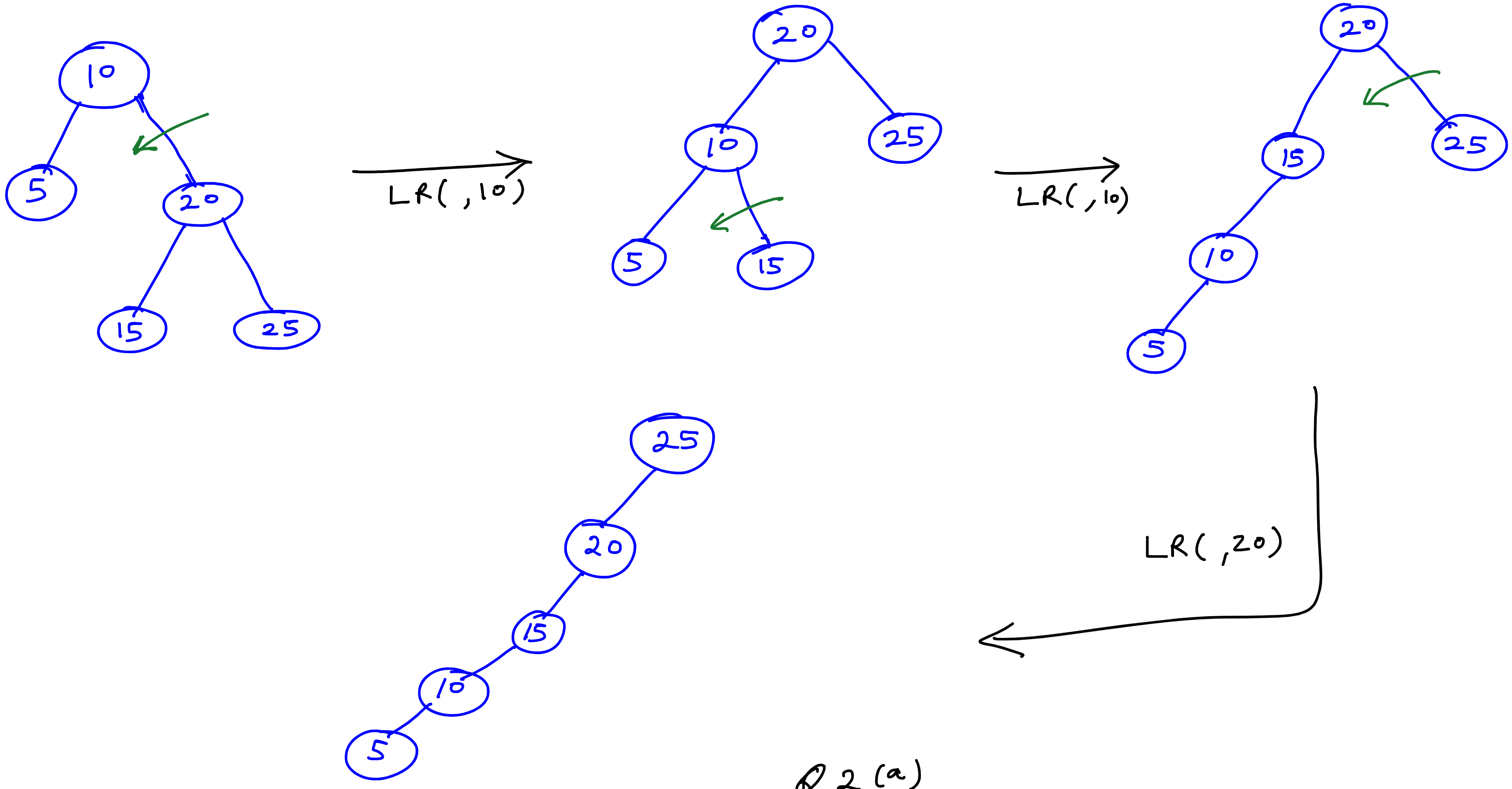
(i) Shown in figure Q3(b)(i).

(ii) Shown in figure Q3(b)(ii).

(c) This is shown in figure Q3(c). Apart from the copies made in phase-I of insertion, any node which changes during Ifixup is also copied.

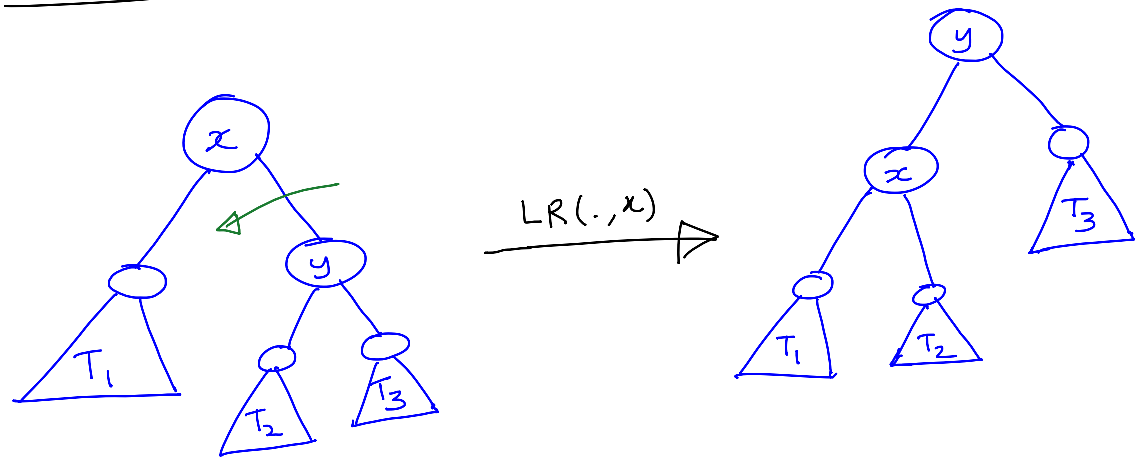
Q4(20) $T.bh$ needs to be changed only in the following places in Ifixup and Dfixup functions. Black height of T is not modified at any other place.

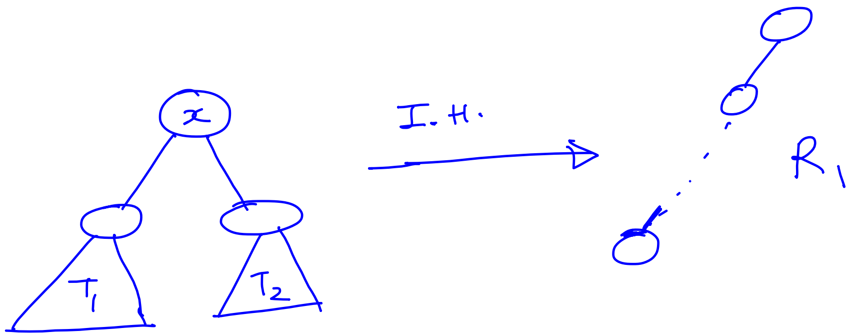
- In Ifixup(T,z), in the case $u.col == red$,
  just below statement
  $y.col = black$
  statement
  '$T.bh = T.bh + 1$'
  should be added.
- In Dfixup,
  the first **'if'** statement of the pseudo-code now needs to distinguish if x is root or x is red.
  This can be done by adding following statements at begining of Dfixup body.
  'if (x == root)
  T.bh = T.bh-1
  return'
  There are also other equivalent ways to write the modified pseudo-code.

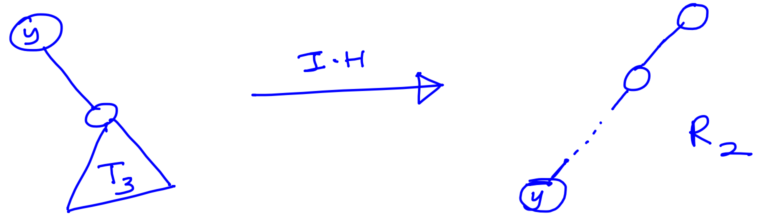


Q 2 (a)

Q2(b)





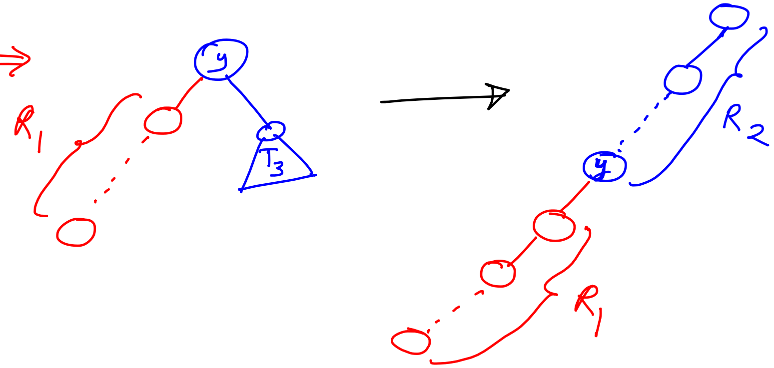


Note that in this sequence y is always either the root or the leftmost node.

⇒ All rotations are done on y or its ancestors.

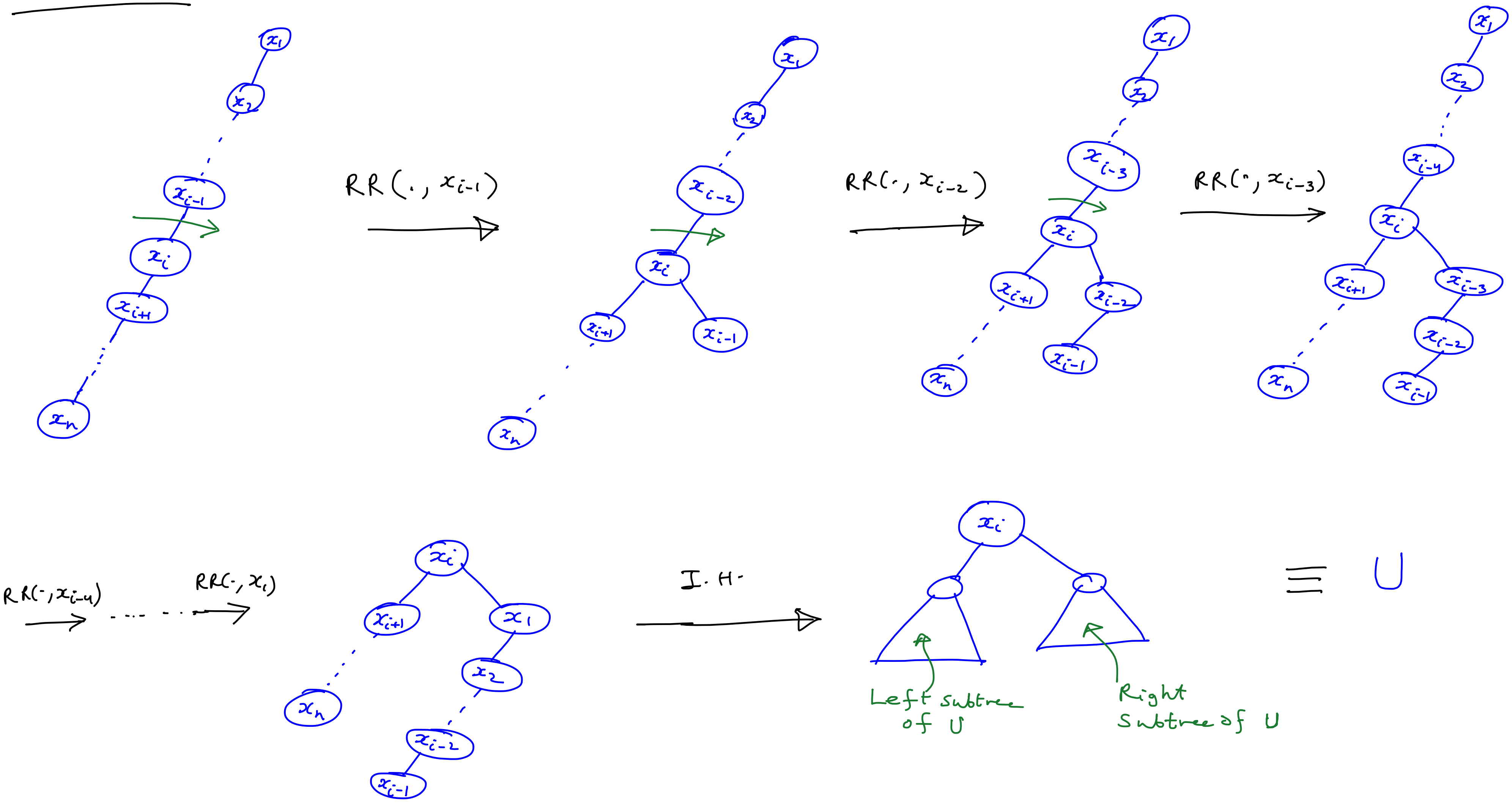
⇒ If a subtree is attached to y as its left child then all these rotations preserve that subtree and its relation to y.

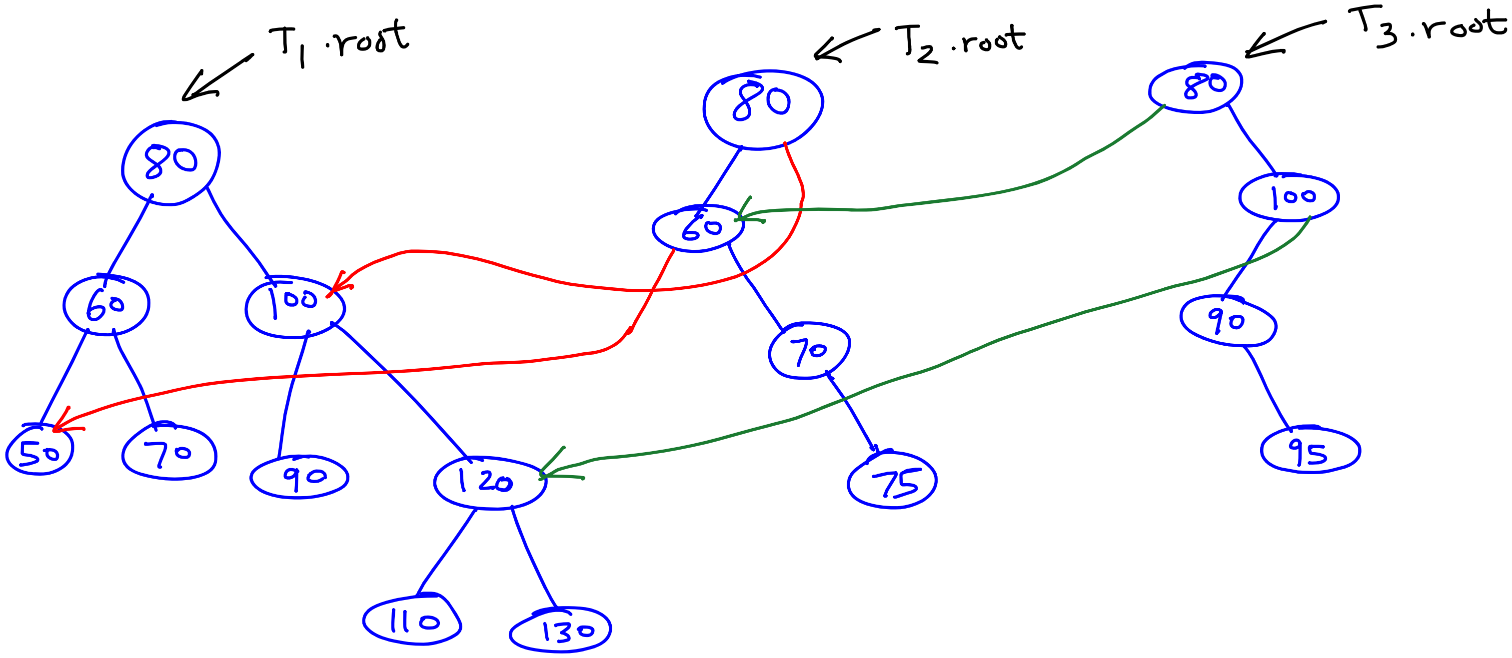
⇒



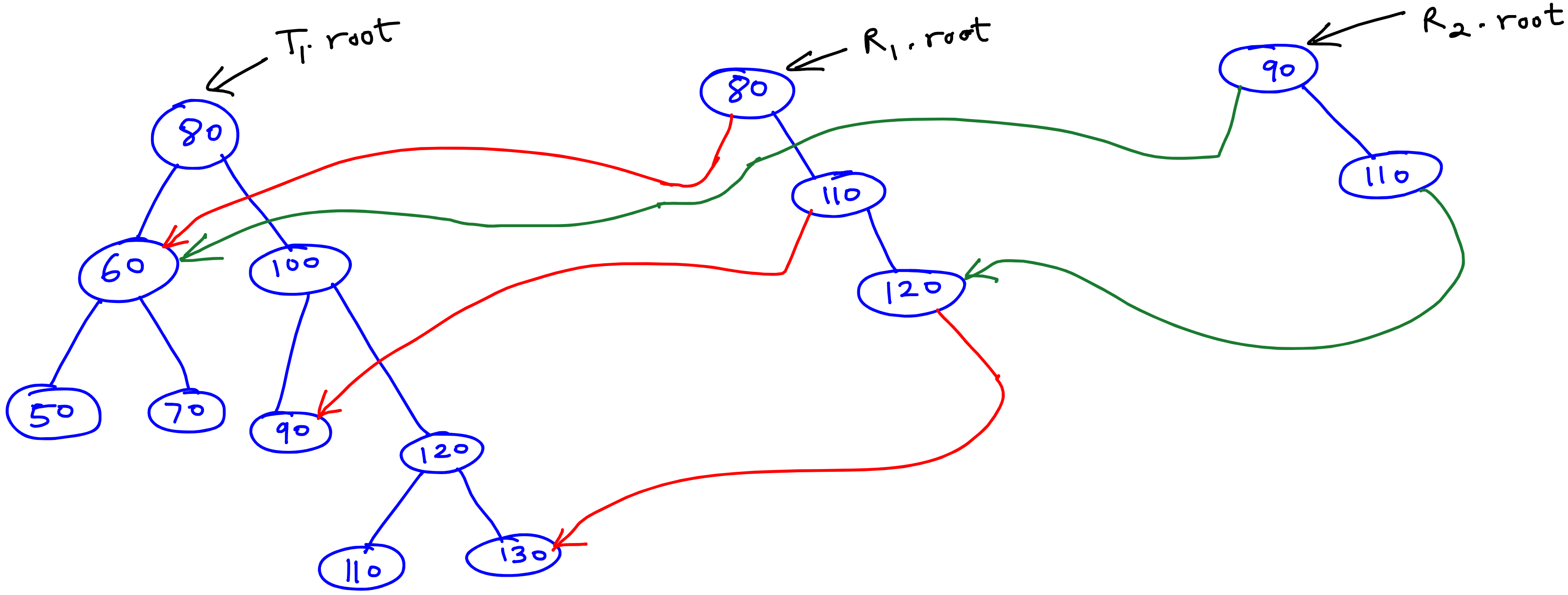
Q 2(c)

Let x_i be the root of BST U.

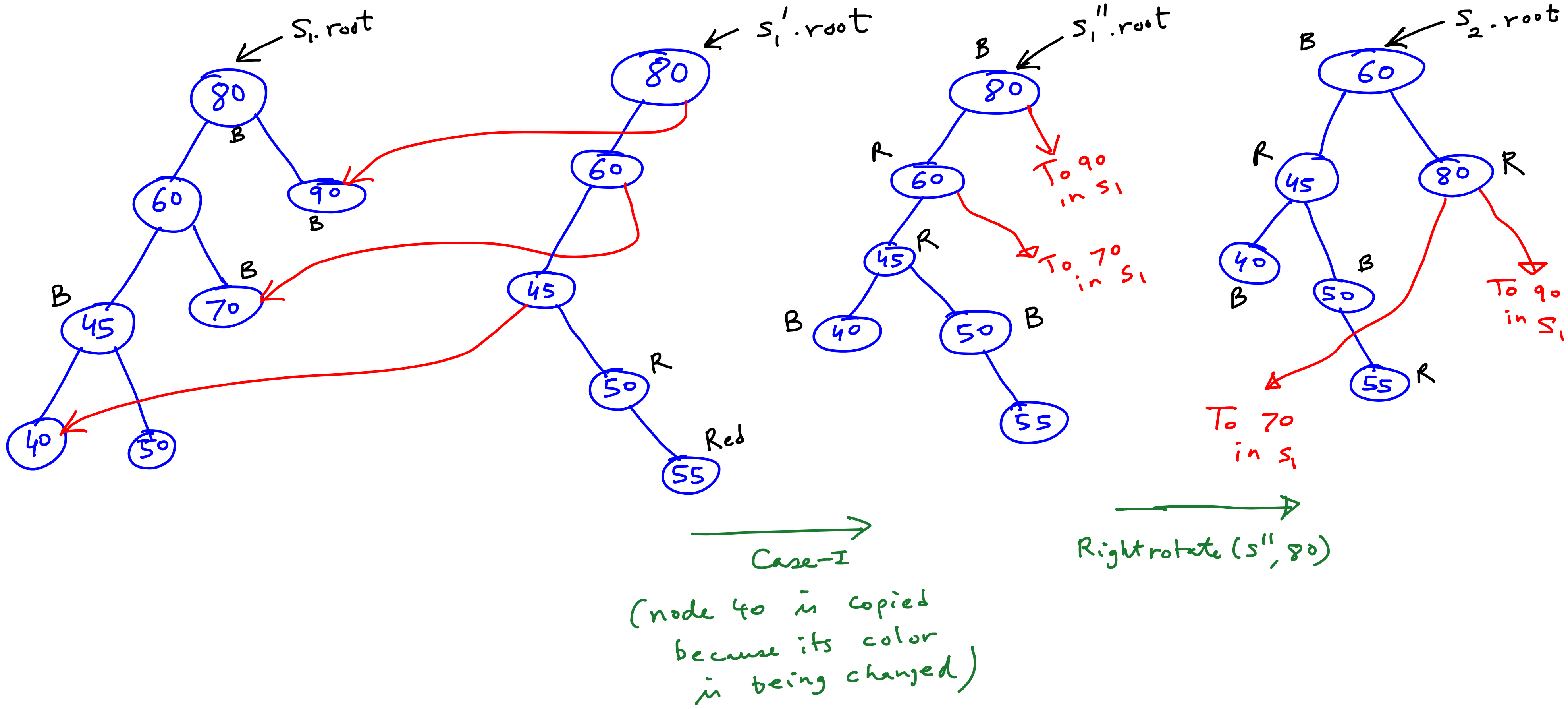



T1.root
T2.root
T3.root
80
60
100
50
70
90
120
110
130
80
60
70
75
80
100
90
95

Q3 (b)(i)

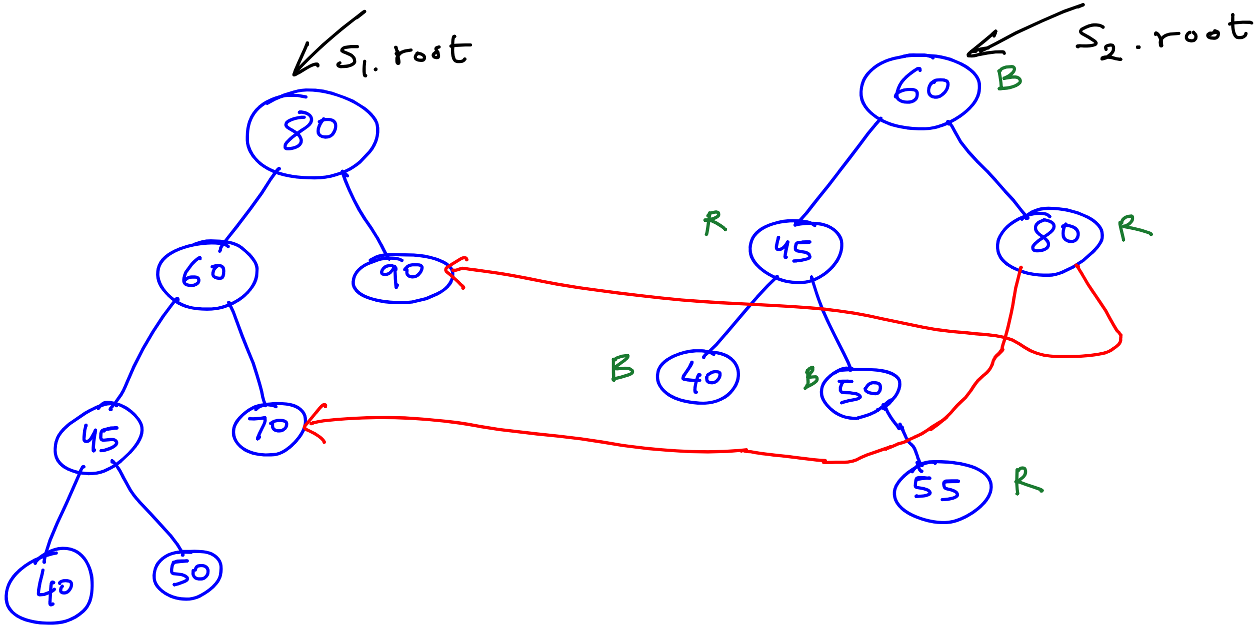

T1. root
R1. root
R2. root
80
60
100
50
70
90
120
110
130
80
110
120
90
110

Q3 (b) (ii)



[Common nodes in S_1', S_1'' and S_2 are the same not copies]

Q 3 (c)



Final picture

(after insert fixup on S_2
is finished)

Q3(e)