Tutorial 7 Solution

Question One

Given two strings s1[1..m] and s2[1..n]..

Given a matrix L(i, j), where i corresponds to a character at position i in s1, and j corresponds to the character at position j in s2. When either i or j are 0, they correspond to the empty character. Simply, 0 <= i <= m and 0 <= j <=n.

Each cell (i, j) contains both the longest common subsequence and a pointer to the cell from which it was derived.

Given this, the longest common subsequence is in L(m, n), where the entire matrix is iteratively defined using the following dynamic programming recurrence relationship.

$$L(i,j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ L(i-1, j-1) + s1[i] & \text{if } s1[i] = s2[j] \\ longest\,[L(i, j-1), L(i-1, j)] & \text{if } s1[i] \neq s2[j] \end{cases}$$

Question Two

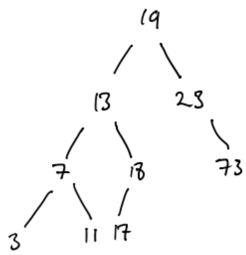
```
between(tree(e,L,R), min, max)
{
        if (tree = nil) return;
        if (min <= e && L != nil) between(L, min, max)
        if (e >= min && e <= max) print(e)
        if (e<=max && R != nil) between (R, min, max)
}
```

Note that this will print out the items as per an in order traversal

Question Three

q 3

Insertions

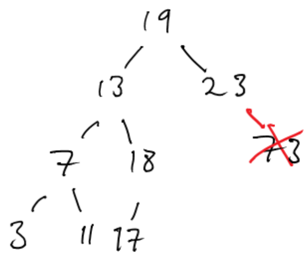


This is an unbalanced binary tree.

Deletions

- delete 73

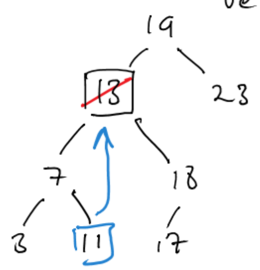
  case 1. leaf node, trivial (unless balanced)



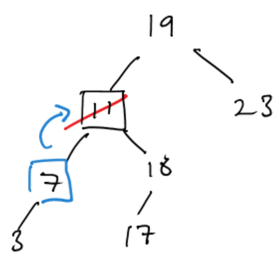
- delete 13

  leaf has two children
  find max in LST. of node to be deleted

  can also look for min in child RST.



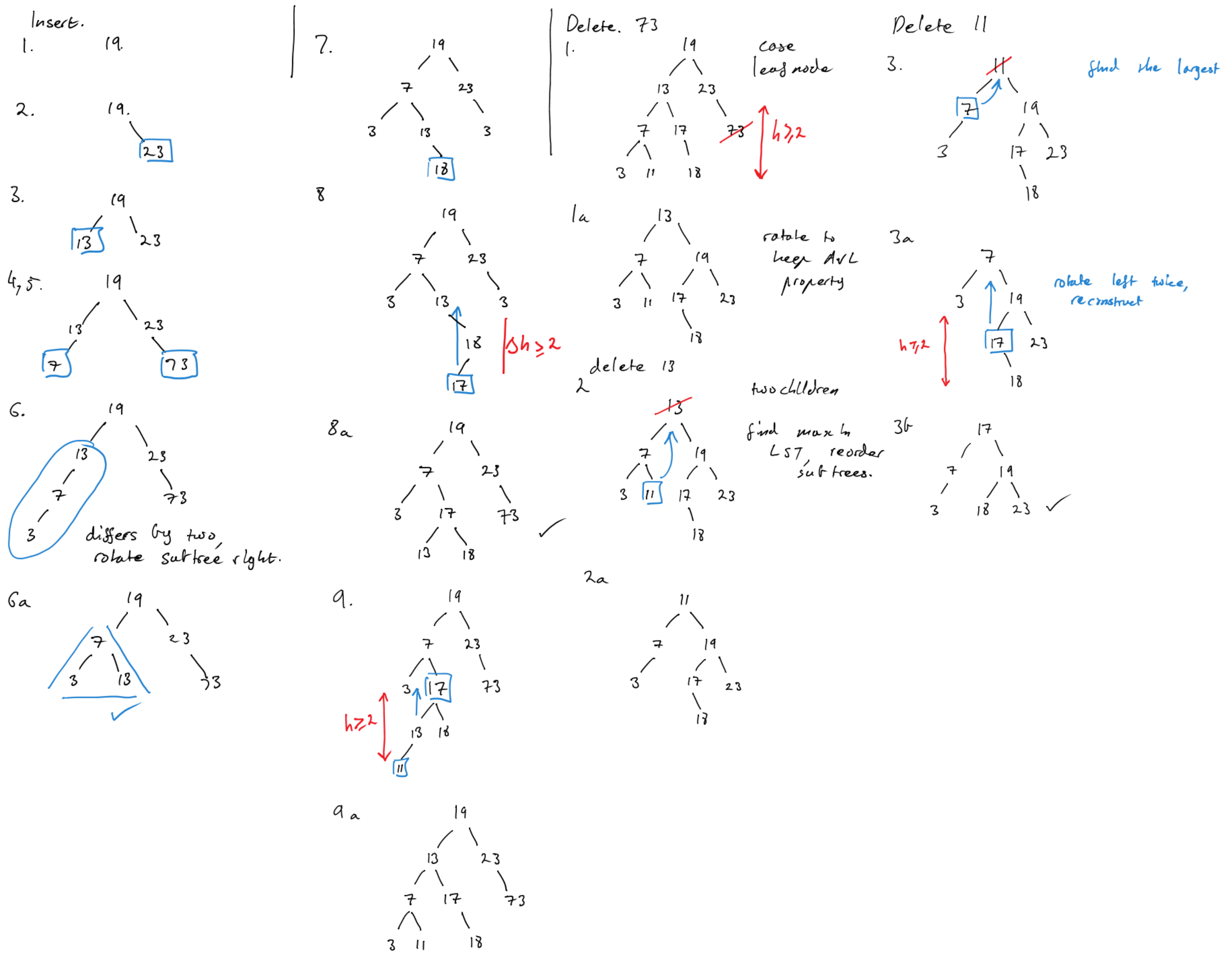
- delete 11



Question Four

q 4.

AVL tree, self-balancing BST.
heights of two children subtrees of
any node differ by at most one

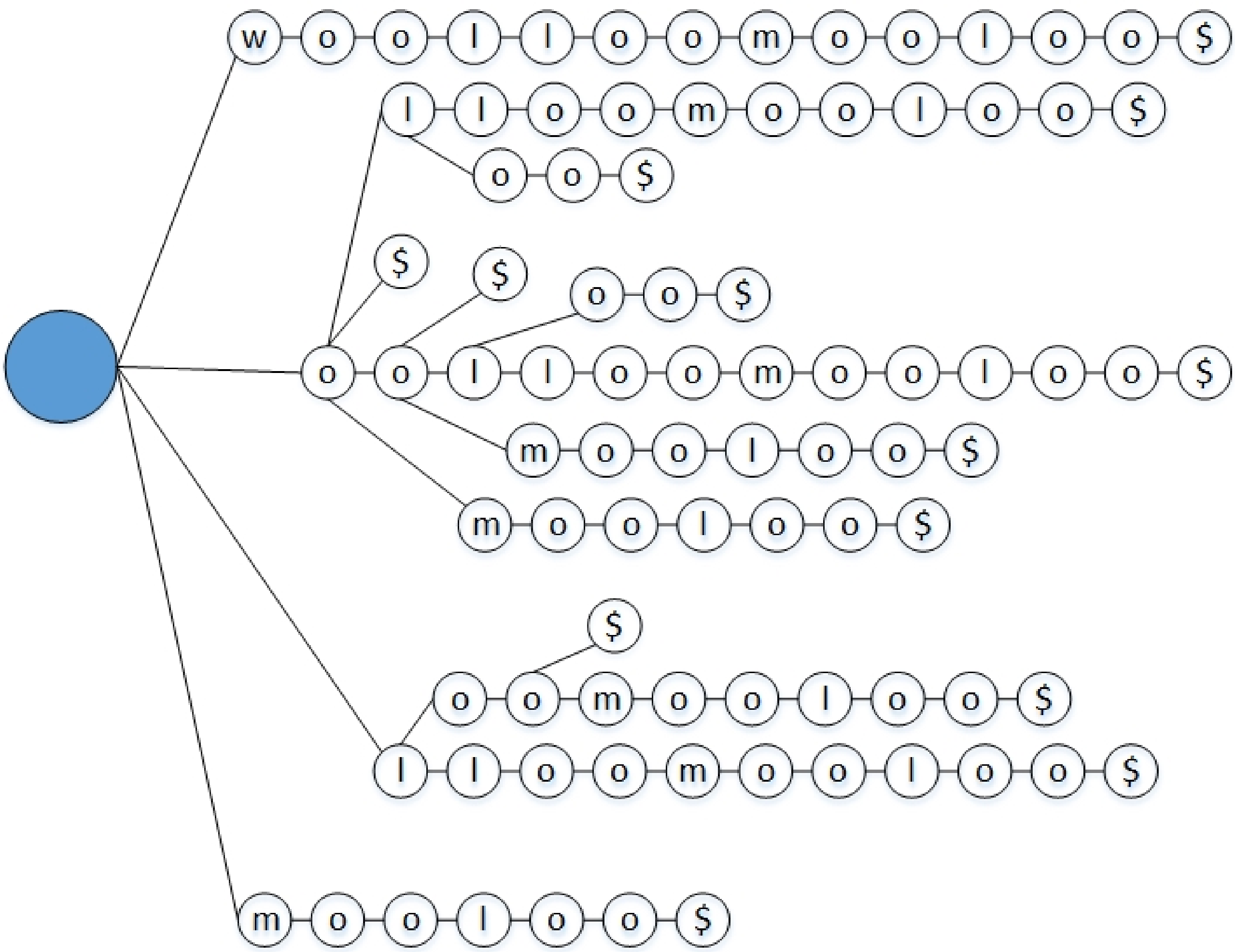


Question Five
Construct a suffix trie by hand of the string `woolloomooloo'. Path compress this trie into a suffix tree.

We will first construct every possible suffix:

Woolloomooloo$
oolloomooloo $
olloomooloo $
lloomooloo $
loomooloo $
oomooloo $
omooloo $
mooloo $
ooloo $
oloo $
loo $
oo $
o $

Now we will insert each suffix into the trie.



Note: $ represents an end of string character.

Now we will compress the branches by merging the nodes that have only one child. For this, replace every substring with numbers (x,y) where x is the starting index of the substring and y is its length.