

FIT2004 S1_2015 Tute Week 7 Solutions

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, $0 \leq i \leq m$ and $0 \leq j \leq n$.

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.

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

Question Two

`between(tree(e,L,R), min, max)`

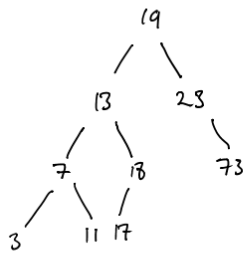
```
{
    if (tree = nil) return;
    if (min <= e && L != nil) between(L, min, max)
    if (e >= min && e <= max) print(e)
    if (e <= max && R != nil) between (R, min, max)
}
```

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

Question Three

9.3

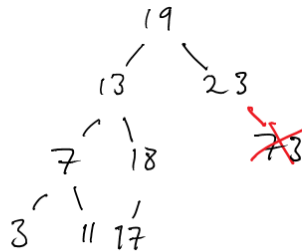
Insertions



This is an unbalanced binary tree.

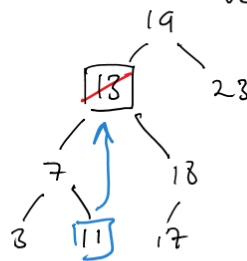
Deletions

- delete 73
case 1. leaf node, trivial (unless balanced)



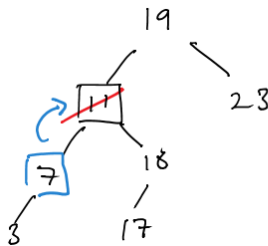
- delete 13

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



can also look for
min in child RST.

- delete 11

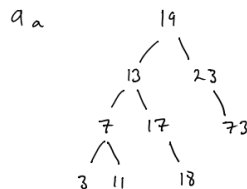
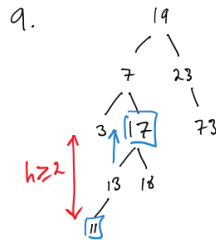
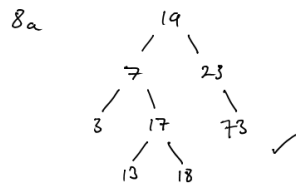
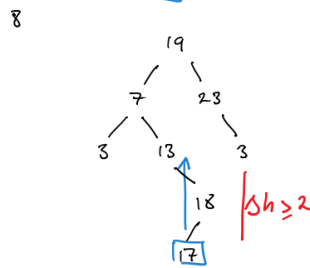
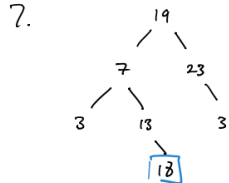
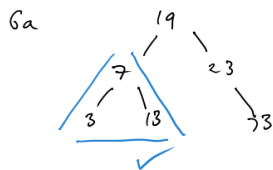
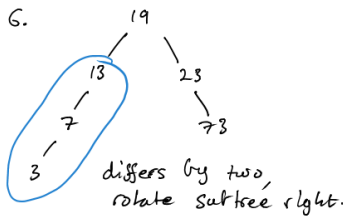
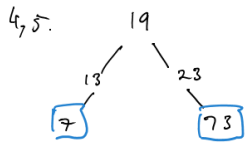
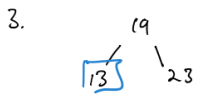
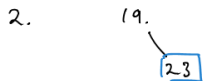


Question Four

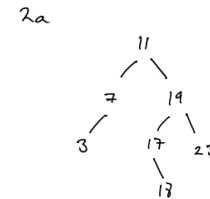
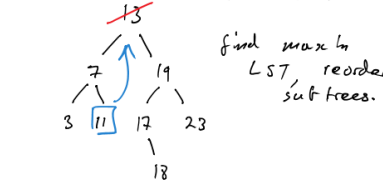
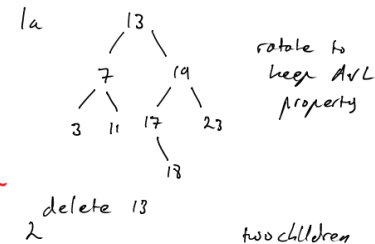
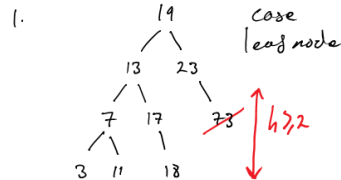
9. AVL tree, self-balancing BST.
heights of two children subtrees of any node differ by at most one.

Insert.

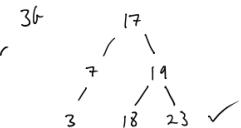
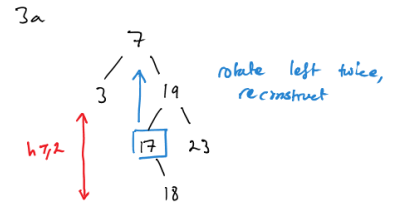
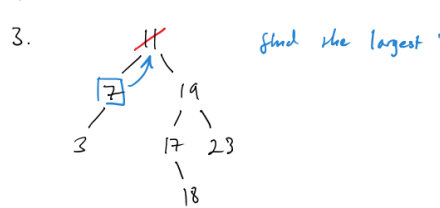
1. 19.



Delete. 73



Delete 11



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 sort these alphabetically

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

now we determine the longest common substring compared to previous items

lloomooloo	-	
loo	1	
loomooloo	3	
mooloo		0
o	0	
olloomooloo	1	
oloo	2	
omooloo	1	

oo	1
oolloomooloo	2
ooloo	3
oomooloo	2
woolloomooloo	0

\$ represents an end of string character

Note that sorting suffices and determining the longest common substring is helpful but not required to obtain the suffix tree or trie

[illegible]

with path compression we get

