Homework 2

Mohit Galvankar mgalvank@iu.edu

3.8 Using the definitions of big-Oh and Ω, find the upper and lower bounds for the following expressions. Be sure to state appropriate values for c and n_0.

(a) $c_1 n$
(b) $c_2 n^3 + c_3$
(c) $c_4 n \log n + c_5 n$
(d) $c_6 2^n + c_7 n^6$

3.8]

a) $C_1 n$
upper bound → n
lower bound → n.

b) $C_2 n^3 + C_3$
upper bound → n^3
lower bound → n^3

c) $C_4 n \log n + C_5 n$.
upper bound → $n \log n$
lower bound → $n \log n$

* d) $C_6 2^n + C_7 n^6$
upper bound → 2^n
lower bound → 2^n

3.10 **(a)** Is $2n = \Theta(3n)$? Explain why or why not.
(b) Is $2^n = \Theta(3^n)$? Explain why or why not.

3.10]

a) $2n = \Theta(3n)$

Yes

We know that, $\Theta(3n)$ is equal to $\Theta(n)$.
Now $\Theta(2n)$ is also equal to $\Theta(n)$.
Hence $2n = \Theta(3n)$.

b) $2^n = \Theta(3^n)$

No.

Suppose that it was true, then there exists constants c and n_0 such that $2^n \leq c\,3^n$ for all $n \geq n_0$. The last eq is equivalent to $(2/3)^n \leq c$ for all $n \geq n_0$. However $(2/3)^n \to \infty$ as $n \to \infty$ $\therefore$ $(2/3)^n \leq c$ cannot be true for all $n \geq n_0$ for any constant c.

3.12 Determine Θ for the following code fragments in the average case. Assume that all variables are of type `int`.

(a)
```
a = b + c;
d = a + e;
```

(b)
```
sum = 0;
for (i=0; i<3; i++)
   for (j=0; j<n; j++)
      sum++;
```

(c)
```
sum=0;
for (i=0; i<n*n; i++)
   sum++;
```

(d)
```
for (i=0; i < n-1; i++)
  for (j=i+1; j < n; j++) {
    tmp = AA[i][j];
    AA[i][j] = AA[j][i];
    AA[j][i] = tmp;
  }
```

(e)
```
sum = 0;
for (i=1; i<=n; i++)
  for (j=1; j<=n; j*=2)
    sum++;
```

(f)
```
sum = 0;
for (i=1; i<=n; i*=2)
  for (j=1; j<=n; j++)
    sum++;
```

(g) Assume that array **A** contains n values, `Random` takes constant time, and `sort` takes $n \log n$ steps.

```
for (i=0; i<n; i++) {
  for (j=0; j<n; j++)
    A[j] = DSutil.random(n);
  sort(A);
}
```

(h) Assume array **A** contains a random permutation of the values from 0 to $n-1$.

```
sum = 0;
for (i=0; i<n; i++)
  for (j=0; A[j]!=i; j++)
    sum++;
```

(i)
```
sum = 0;
if (EVEN(n))
  for (i=0; i<n; i++)
    sum++;
else
  sum = sum + n;
```

3.12]

a) $f(n) = 1$

b) $f(n) \Rightarrow 3n \Rightarrow n$

c) $f(n) = n^2$

* d) $f(n) = n^2$

e) $f(n) = n \log n$

f) $f(n) = n \log n$

g) $f(n) \Rightarrow (n(n + n\log n)) \Rightarrow n^2 \log n$

★ h) $f(n) = n^2$

★ i) $f(n) = n$

3.13 Show that big-Theta notation defines an equivalence relation on the set of functions.

3.13

i) Reflexive

For any function $f(n)$, $f(n) = \Theta(f(n))$

$\therefore 1 f(n) \le f(n) \le 1 f(n)$ for all n.

ii) Symmetry.

Assume $f(n) = \Theta(g(n))$

Then there exists $A, B > 0$ such that

$A\, g(n) \le f(n) \le B\, g(n)$ for sufficiently large n

$\therefore f(n) \le B\, g(n) \implies 1/B\, f(n) \le g(n)$ &

$A\, g(n) \le f(n) \implies g(n) \le 1/A\, f(n)$.

we have $1/B\, g(n) \le g(n) \le 1/A\, g(n)$ for sufficiently large n.

$\because 1/A, 1/B > 0$, $\therefore$ we conclude $g(n) = \Theta(f(n))$

iii) Transitive.

Suppose that $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$

Then, there exist $A, B, C, D > 0$ such that for sufficient n,

$A\, g(n) \le f(n) \le B\, g(n)$ & $C h(n) \le g(n) \le D h(n)$

$\therefore f(n) \ge A\, g(n) \ge A(C h(n)) = (AC)\, h(n)$.

& $f(n) \le B\, g(n) \le B(D h(n)) = (BD)\, h(n)$

Hence for sufficiently large n, $AC\, h(n) \le f(n) \le BD\, h(n)$

$\because AC, BD > 0$, we conclude $f(n) = \Theta(h(n))$

Hence proved, Θ defines an equivalence relation on set.

5.2 Define the degree of a node as the number of its non-empty children. Prove by induction that the number of degree 2 nodes in any binary tree is one less than the number of leaves.

5.2].

Let h be the height of the binary tree.

When h=1:
Only one tree with one leaf node and no full node. ∴ By induction we can say the statement holds true.

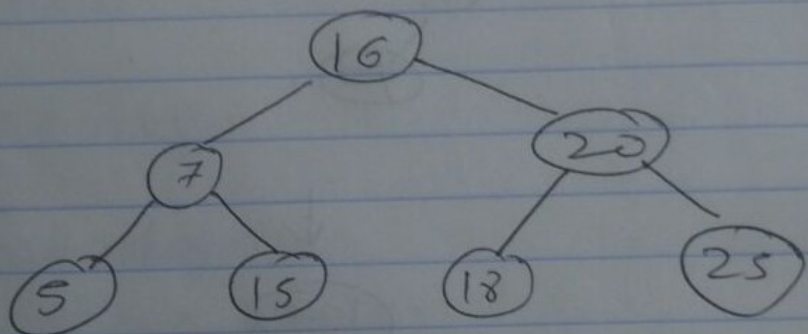
When h=k+1
case 1: root is not a full node.
Assume the tree doesn't have a right child. ∴ the no. of leaf nodes is equal to the ~~base~~ number of full-nodes. By induction we can prove that the diff is 1 since the height of the left subtree is k.

case 2: root is a full node.
total no. of leaf nodes = leaf node in the tree rooted at its left & right.
total no. of full nodes = 1 (root) + the no. of full nodes to its left & right
∴ the diff is 1.

Hence proved by induction.

5.15 (a) Show the BST that results from inserting the values 15, 20, 25, 18, 16, 5, and 7 (in that order).
(b) Show the enumerations for the tree of (a) that result from doing a preorder traversal, an inorder traversal, and a postorder traversal.

5.15] 15, 20, 25, 18, 16, 5, 7.
5 7 15 16 18 20 25

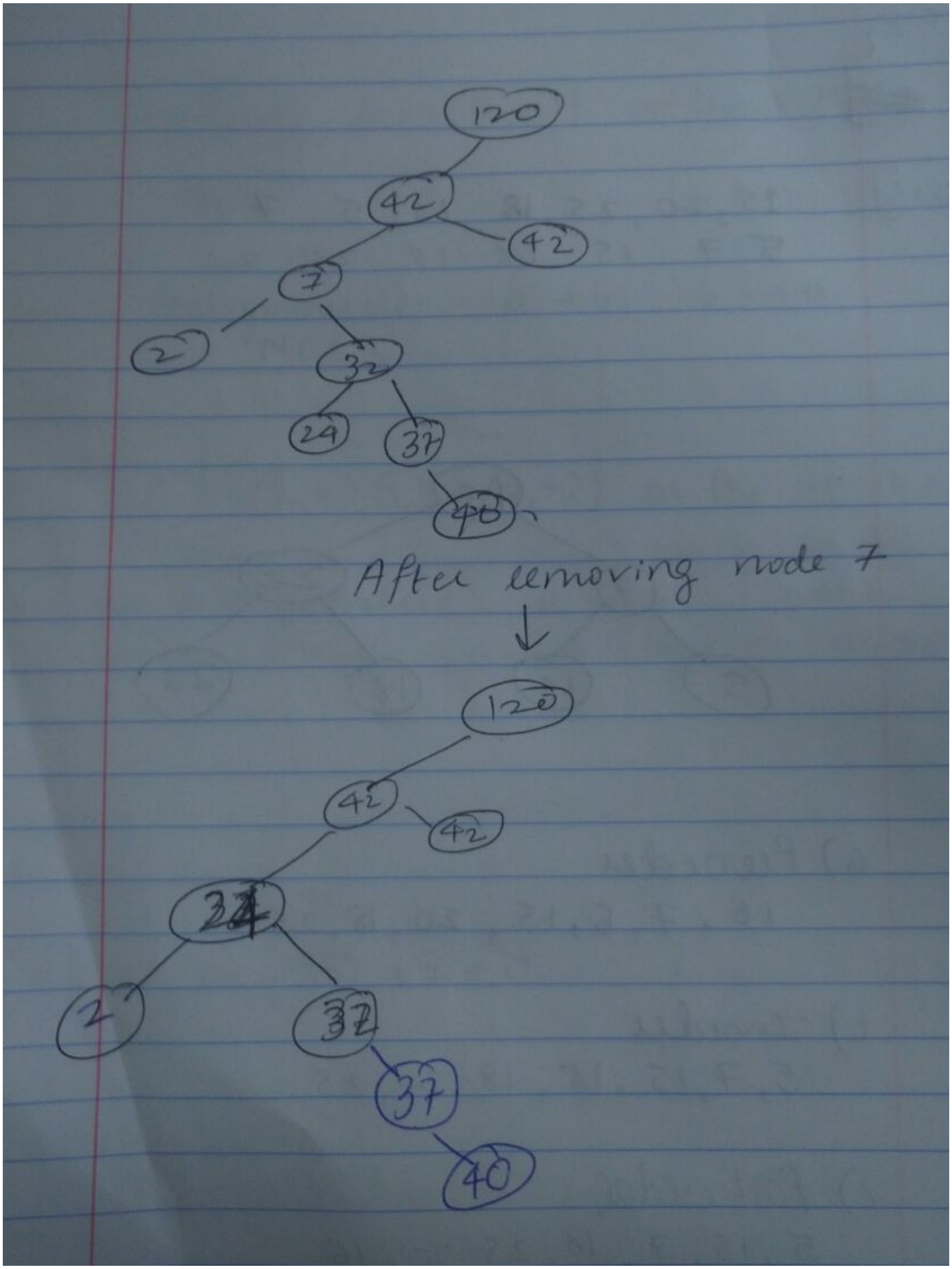


a) Preorder
16, 7, 5, 15, 20, 18, 25.

b) Inorder
5, 7, 15, 16, 18, 20, 25.

c) Postorder
5, 15, 7, 18, 25, 20, 16.

5.17 Draw the BST that results from deleting the value 7 from the BST of Figure 5.13(b).



5.19 Write a recursive function named **smallcount** that, given the pointer to
the root of a BST and a key K, returns the number of nodes having key values less than or equal to K.
Function **smallcount** should visit as few nodes in the BST as possible.
DONE
Used in order traversal to traverse through the tree. If the node is smaller than or equal to the key, increment count. If node is greater than key, terminate program and return the count.

Uploaded code on github. Java file : BinaryTree.java