600.363/463 Key to Mid Sem 1

I 1. Yes 2. NO 3. Yes 4. NO 5. NO 6. Yes
7. NO 8. Yes 9. Yes 10. Yes

II 1. $T(n) \le 3T(\frac{n}{2}) + n^2$
By master theorem $T(n) = O(n^2)$.

2. $T(n) \le 5T(\frac{n}{2}) + n^2$
By MT: $T(n) = O(n^{\log_2 5})$

3. $T(n) \le 2T(\frac{n}{4}) + T(\frac{n}{3})$
Let $T(n) \le cn^\alpha$, α to be chosen to satisfy the inductive step & c to be chosen to satisfy the base cases.

To satisfy the inductive step
needs: $2c(\frac{n}{4})^\alpha + c(\frac{n}{3})^\alpha \le cn^\alpha$
i.e. needs $2(\frac{1}{4})^\alpha + (\frac{1}{3})^\alpha \le 1$
$\alpha \simeq 0.86$
Hence $T(n) = O(n^{0.86})$

Note: I meant to ask for
$T(n) \le 2T(\frac{n}{4}) + T(\frac{n}{3}) + n$.
then since $2(\frac{1}{4}) + \frac{1}{3} < 1$, $T(n) = O(n)$.
This can be proved by induction.
Some students proved this bound for the given recurrence by induction. $T(n) \le cn$ is not a tight bound for the given recurrence, but we have given credit for the solution.

II 4. $T(n) \le 2n\,T(n-1)$

$\le 2n\,2(n-1)\,T(n-2) = 2^2 n(n-1)\,T(n-2)$

$\le 2^3 n(n-1)(n-2)\,T(n-3)$

$\vdots$

$\le 2^{n-1} n(n-1)\cdots 3.2\,T(1)$

Hence $T(n) = O(2^n n!)$.

5. $T(n) \le 2T(n-1) + T(n-2)$

Assuming it to be an equality & $T(n) = c^n$

$c^n = 2c^{n-1} + c^{n-2}$

i.e. $c^2 - 2c - 1 = 0$

ie. $c = 1 \pm \sqrt{2}$

Hence $T(n) = a(1+\sqrt{2})^n + b(1-\sqrt{2})^n$

since $(1-\sqrt{2})^n = O(1)$. $T(n) = O((1+\sqrt{2})^n)$.

III $c[i,j]$ $x_1 \cdots x_i$ / $y_1 \cdots y_j$

If x_i couples with y_j then $c[i-1, j-1] + \delta(x_i, y_j)$

If x_i couples with a symbol in $y_1 \cdots y_{j-1}$ $c[i, j-1]$

Hence $c[i,j] = \max\{c[i-1,j-1] + \delta(x_i,y_j),\ c[i,j-1]\}$ if $i > 1$ & $i \le j$

$c[1,j] = \max\{\delta(x_1,y_j),\ c[1,j-1]\}$

Base: $c[1,1] = \delta(x_1, y_1)$

Compute $c[1,1], c[2,2], \ldots, c[n,n], c[1,2], \ldots$; i.e.

in increasing value of $j-i$

Each $c[i,j]$ requires $O(1)$ steps

Hence Speed $= O(n(m-n)) = O(nm)$.

IV 1. a) Start at the root with REM = k.

Repeat
At any node, access the size of the left subtree
If size ≥ k move to the left child;
If size = k-1, return the value at the root.
If size < k-1
rem = rem - size - 1,
move to the right child.

1. b) First compute the rank of x,
Then compute the rank of y,
Then return rank(y) - rank(x) + 1.

computation of the rank of a given number z:
start at the root, set count = 0;
Follow the path for z as in any binary search tree;
If right child is taken, count = count + Size(left child) + 1
If it stops at that node, count = count + 1 & return count

IV 2) b)

a) is false since a balanced binary tree is never more space efficient.

c) is false ~~but~~ since a BST can never require less space. It usually takes more space since you need to store some flags. For example, in a Red-Black tree you need to store a bit for the color. Also, a complete BT can be stored in a RAM as an array without using pointers.