1

Page No.
Date

Deepanshu 2019CS50427

Ans-1 (a) Proving that $f(n)$ is not $O(g(n))$.

Proof by contradiction

Assume that $f(n)$ is $O(g(n))$.

from definition of Big-oh

$$f(n) \geq g(n) \quad \text{for all } n \geq n_0 \quad \text{—(1)} \quad \text{for some } [n_0 \in N]$$

Choose a non-prime number $n_1 > n_0$.

$f(n_1) = n_1$
$g(n_1) = n_1^2$ [from function definition]

Clearly $f(n_1) < g(n_1)$ which contradicts — eq^n(1)

Thus, our assumption is wrong and

$f(n)$ is not $g(n)$

Hence proved

Proving that $g(n)$ is not $O(f(n))$.

Proof by contradiction

Assume that $g(n)$ is $O(f(n))$.

Again from definition of Big-oh

$$g(n) \geq f(n) \quad \text{for all } n \geq n_0 \quad \text{—(1)} \quad \text{for some } [n_0 \in N]$$

From number theory, we know that there can be arbitrarily large prime number.

choose a prime $n_p > n_0$.

$f(n_p) = n_p^3$, $g(n_p) = n_p^2$ (from function definition).

Clearly, $f(n_p) > g(n_p)$ which contradicts —(1)

Hence $g(n)$ is not $O(f(n))$.

Hence Proved

②

Deepanshu 2019CS50427

(b) $u(n) = 1 + \sin(n)$

$v(n) = 1$

checking if ~~f(n)~~ $v(n)$ is $O(u(n))$.

Assume that $v(n)$ is $O(u(n))$. Then,

$v(n) \geq u(n)$ for all $n \geq n_0$. [for some $n_0 \in N$]

$1 \geq 1 + \sin(n)$

$\sin(n) \leq 0$ for all $n \geq n_0$ which is definitely not true since $\sin(n)$ is a ~~periodic~~ periodic function and oscillates between -1 and -1.

$v(n)$ is not $O(u(n))$. — ①

checking if $u(n)$ is $O(v(n))$.

Assume that $u(n)$ is $O(v(n))$.

$u(n) \geq v(n)$ for all $n \geq n_0$ [for some $n_0 \in N$].

$1 + \sin(n) \geq 1$

$\sin(n) \geq 0$ for all $n \geq n_0$. which is again not true since $\sin(n)$ is periodic function with values between -1 and 1. Thus $\sin(n)$ will be negative for some $n \geq n_0$.

$u(n)$ is not $O(v(n))$. — ②

from ① & ②, we can say that u and v form a mutually non-dominating pair of functions.

Page No. ____
Date

Deepanshu 2019CS50427

Ans-2 » Let SMC denote Sparse Matrix Class and LL denote linked List.

```
Algorithm valueAtCell (int i, int j, SMC A, SMC B).
Ptr temp1 ← A.rowArray[i]
Ptr temp2 ← B.colArray[j]
int n ← A.rowArray.length
    float result ← 0
    while (n>0)
        result ← result + (temp1.val) x (temp2.val)
        n--
        temp1 = temp1.nextinCOL
        temp2 = temp2.nextinrow
    return result

Algorithm multiply (SMC A, SMC B).
    n ← A.rowArray.length
    SMC resultMatrix;
    resultMatrix.rowArray ← new Array[n]
        resultMatrix.colArray ← new Array[n].

    for (i←0; i<n; i++)
        generateNode
        for (j←0; j<n; j++)
    generateNode (0,0)
```

Page No.
Date

Deepanshu 2019CS50427

Algorith genarateNode (int i, intj, SMC A, SMC B, n)
LL node
float val = valueAtCell (i, j, A, B)
if (i ≥ 0 and j ≥ 0 and i < n and j < n).
node = new LinkedList (i, j, val, genarateNode(i+1, j), generateNode (i, j+1)).

~~else if (i ≥ n or i < 0)~~
~~LL node = new LinkedList (i,~~
else
node = null
return node

Algorithm multiply (SMC A, SMC B)
n ← A. rowArray. length
~~SMC resultMaal~~
SMC resultMatrix
resultMatrix. rowArray ← new Array [n]
for (i ← 0, i < n, i++)

resultMatrix. rowArray [i] = generate Node (i, 0, genarateNo ~~generate~~ null

for (i ← 0, i < n, i++)

resultMatrix ·colArray [i] = genarateNode (i, 0, null, genarate Node(0, i

Deepanshu 2019CS50427

Ans-3 ⇒ (a)

```
class BBMH{
    Node root;
BBMH() {
this.root=null;
  }

BBMH(Node root){
    this.root = root;
  }
```

```
class Node{
  int data;
 Node left;
  Node right;

  Node(){
     this.data=0
     this.left=null
      this.right=null
  }
 Node(int data){
    this.data = data
    this.left=null
     this.right=null
```

(b) Algorithm insert(int data)

```
if ( size(root.left) < size(root.right))
        while(data < root.left.data
```

(i+1,

Page No.
Date

Deepanshu 2019CS50427

(1) Algorithm Insert (int data)

```
while ( data < root.val)
    if ( size ( root.left) < size (root.right)
        insert ( root.left)
    else
        insert ( root.right)

Node n = Node (data)
~~n.left~~
~~if (ro~~
if ( size (root.left) < size (root.right))
    n.left = root
    n.right = root.right
    root.right = null.
else
    n.right = root
    n.left = root.left
    root.left = null

return
```

Page No.
Date

Deepanshu 2019CS50427

~~Algorithm totalNodes~~

(c) Algorithm delete (int data)

```
while (data < root.data and root != null)
    delete ( root. left)
    delete (root. right )
if (root = null)
    return
else (
    if (size (root. left) < size (root. right)).
        root. right. left = root. left
        free (root)
    else.
        root. left. right = root. right
        free (root).
```

From induction, insertion would preserve the 2 properties.
Also, delete preserves the condition of point 2 and also point 1 (trivial)