Assignment 1

CS201: Data Structures and Algorithms

Pranav Kanire

Roll no.: 190010033

October 23, 2020

Question 1

Definition 1

$O(q(n)) = \{$p(n) : there exists positive constants c and n_0 such

that $0 \leq p(n) \leq cq(n), \ \forall n \geq n_0\}$

Definition 2

$\Omega(q(n)) = \{$p(n) : there exists positive constants c and n_0 such

that $0 \leq cq(n) \leq p(n), \ \forall n \geq n_0\}$

Definition 3

$\Theta(q(n)) = \{$p(n) : there exists positive constants c_1, c_2 and n_0

such that $0 \leq c1q(n) \leq p(n) \leq c2q(n), \ \forall n \geq n_0\}$

Theorem 1

$p(n) = O(q(n))$ if and only if $q(n) = \Omega(p(n))$.

Theorem 2

$p(n) = \Theta(q(n))$ if and only if $p(n) = O(q(n))$ and $p(n) = \Omega(q(n))$.

Theorem 3

$p(n) = \Theta(q(n))$ if and only if $q(n) = \Theta(p(n))$.

I have given:

$$f(n) = n^{log\ log\ n} \tag{1}$$

$$g(n) = 4^{log\ n}$$

$$\therefore g(n) = (2^2)^{log\ n}$$

$$\therefore g(n) = 2^{2\ log\ n} \tag{2}$$

(i) Is $f(n) = O(g(n))$?

Binary Answer: No

Proof:

We will prove that there exist no positive constants c and n_0 such that $f(n) \leq c.g(n) \; \forall n \geq n_0$.

Then the answer follows from definition 1.

Assume there exists such positive constants c and n_0.

$\Rightarrow 0 \leq f(n) \leq c.g(n) \; .... \; \forall n \geq n_0$

$\Rightarrow 0 \leq n^{log \; log \; n} \leq c.2^{2 \; log \; n} \; .... \; \forall n \geq n_0$

Taking log_n on both sides, we get

$$log\ log\ n \ \leq \ log_n C + 2\ log\ n.log_n 2$$

Consider $n = 2^{2^m}$

$$\therefore log\ log\ 2^{2^m} \leq log_{2^{2^m}} C + 2 log\ 2^{2^m}.log_{2^{2^m}} 2$$

$$\Rightarrow m \leq \frac{log\ C}{2^m} + \frac{2^{m+1}}{2^m}$$

$$\Rightarrow m.2^m \leq log\ C + 2^{m+1}$$

$$\Rightarrow (m-2)2^m \leq log\ c$$

But, $\forall C$ we can always find m for which $(m-2)2^m > log\ C$

and hence this leads to contradiction.

$\therefore f(n) \neq O(g(n))$

(ii) Is $f(n) = \Omega(g(n))$?

Binary Answer: Yes

Proof:

From the above answer for part(i):

$0 \leq f(n) \leq c.g(n)$

$\Rightarrow (m-2)2^m \leq log\ C \quad \ldots$ where $n = 2^{2^m}$

Similarly,

$f(n) \geq c.g(n) \geq 0$

$\Rightarrow (m-2)2^m \geq log\ C \quad \ldots$ where $n = 2^{2^m}$

Now, let $C = 1$, then $log\ C = 0$

$\therefore \forall m \geq 3 \quad (m-2)2^m \geq log\ C$

$\Rightarrow n_0 = 2^{2^m} = 2^{2^3} = 256$

Therefore, as per definition 2, $f(n) = \Omega(g(n))$

(iii) Is $f(n) = \Theta(g(n))$?

Binary Answer: No

Proof:

Follows from the answer to (i) and Theorem 2.

(iv) Is $g(n) = O(f(n))$?

Binary Answer: Yes

Proof:

Follows from the answer to (ii) and Theorem 1.

(v) Is $g(n) = \Omega(f(n))$?

Binary Answer:

Proof: No

Follows from the answer to (i) and Theorem 1.

(vi) Is $g(n) = \Theta(f(n))$?

Binary Answer: No

Proof:

Follows from the answer to (iii) and Theorem 3

Question 2

I have given:

$$Preorder\ \ Traversal:\ \ [5,\ 4,\ 2,\ 1,\ 3,\ 7,\ 6,\ 9,\ 8]$$

We can obtain the BST from the given pre-order traversal Using

the following algorithm or code snippet:

(The complete code is in diffrent file.)

```
struct node{
```

```
int data;
struct node *left,*right;
}
struct node *newnode(int data){
struct node *temp = (struct node*)malloc(sizeof(struct node));
temp->data = data;
temp->left = temp->right = NULL;
}
struct node *Constructbst(int order[], int start, int last){
```

```
if(start > last)
   return NULL;
else if(start == last)
   return newnode(order[start]);
\\ Recursion
struct node *root = newnode(order[start]);
int i = 0;
while (i≤end & & arr[i] < arr[start])
   i++;
```

```
    root->left = Constructbst(order, start+1, i-1);

    root->right = Constructbst(order, i, last);

    return root;

}

int main(){

    int preorder[ ] = {5, 4, 2, 1, 3, 7, 6, 9, 8}

    int n = sizeof(preorder)/sizeof(preorder[0]);

    struct node *mainroot = Constructbst(preorder, 0, n-1);
```

\\ We get bst from given preorder with root as mainroot

\\ We can get anything from this bst now

}

If u want to check the code for different preorders, then just change the initialized preorder array in the 'main function'.

(i)BST corresponding to given preorder:

(it is explained on the next page clearly)

5

4 7

2 6 9

1 3 8

$5 \rightarrow left = 4\&5 \rightarrow right = 7$

$4 \rightarrow left = 2\&7 \rightarrow left = 6\&7 \rightarrow right = 9$

$2 \rightarrow left = 1\&2 \rightarrow right = 3\&9 \rightarrow left = 8$

The inorder traversal for the same would be:

$$[1, \ 2, \ 3, \ 4, \ 5, \ 6, \ 7, \ 8, \ 9]$$

(ii) Post-order Traversal:

the post-order traversal for the given pre-order traversal is:

$$[1, 3, 2, 4, 6, 8, 9, 7, 5]]$$