CSE 321 - Homework 1

1) (a) Is the following statement scientifically sound?: The running-time of algorithm A is at least $O(n^2)$"?

→ A scientific sentence must contain conclusive and correct judgments. Complexity of the algorithm that $O(n^2)$ is upper bound. So, The running time of Algorithm A is the most quatratic at $O(n^2)$. (Bigger polynomial). Big-O notation is a upper bound that can not be least. Therefore, Not scientifically sound.

(b) Are the following true? (c is constant)

i) $2^{n+1} = O(2^n)$? $\Rightarrow 2^n . 2 \leq c . 2^n$ $c>0$, $n \geq n_0$, $n \geq 0$ $\Rightarrow \cancel{2^n} . \cancel{2} \leq \cancel{2} . \cancel{2^n}$ $\boxed{1 \leq 1}$ ✓ $\boxed{c=2}$ $n \geq n_0$ for all n is true.

ii) $2^{2n} = O(2^n)$? $\Rightarrow 2^{2n} \leq c . 2^n$ $c>0$, $n>0$, $n \geq n_0$ $\Rightarrow \frac{2^{\cancel{2n}}}{\cancel{2^n}} \leq \frac{c . \cancel{2^n}}{\cancel{2^n}} \Rightarrow \boxed{2^n \leq c} \to 2^n$ is not bounded it complexity by constant number false

(c) Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Is the equation $\max(f(n), g(n)) = \Theta(f(n)+g(n))$ true? (c_1 and c_2 constant)

Theta has omega and big-O $\Rightarrow$ (1) $\to \max(f(n), g(n)) \geq \Omega(f(n)+g(n))$
$\to \max(f(n), g(n)) \leq O(f(n)+g(n))$

$f, g \in N \Rightarrow$ (2) i) $\max(f(n), g(n)) \geq c_1(f(n)+g(n))$ ✓
ii) $\max(f(n), g(n)) \leq c_2(f(n)+g(n))$ ✓

i) Proof

g(n) or f(n) must equel or smaller than $\max(f(n), g(n))$

$\max(f(n), g(n)) \geq g(n)$ and $\max(f(n), g(n)) \geq f(n)$

+

$2\max(f(n), g(n)) \geq f(n)+g(n)$

$\max(f(n), g(n)) \geq \frac{1}{2}(f(n)+g(n))$ $\boxed{c_1 = \frac{1}{2}}$

ii) $\max(f(n), g(n)) \leq c_2.(f(n)+g(n)) \to c_2$ can be 1 or bigger ✓

$\boxed{c_2 \geq 1}$

2 ideas are proved
So, this equation is true.

- Hamza YOĞURTCUOĞLU -
- 171044086 -

2) In each of the following situations, indicate whether $f \in O(g)$ or $f \in \Omega(g)$ or both (in which case $f \in \Theta(g)$).

(a) $f(n) = n^{1.01}$, $g(n) = n \cdot \log^2 n$ $\Rightarrow \lim\limits_{n\to\infty} \dfrac{n^{1.01}}{\cancel{n} \cdot \log^2 n} = \dfrac{n^{0.01}}{\log^2 n} = \dfrac{\infty}{\infty}$

L'Hospital are used $\rightarrow \lim\limits_{n\to\infty} \dfrac{0.01 \cdot n^{-0.99}}{2 \cdot \log n \cdot \frac{1}{n \cdot \ln 2}} = \lim\limits_{n\to\infty} \dfrac{0.01 \cdot \ln 2 \cdot n^{0.01}}{2 \cdot \log n} = \dfrac{\infty}{\infty}$

Again L'Hospital $\rightarrow \lim\limits_{n\to\infty} \dfrac{(0.01)^2 \cdot \ln 2 \cdot n^{-0.99}}{2 \cdot \frac{1}{n \cdot \ln 2}} = \boxed{\dfrac{(0.01)^2 (\ln 2)^2 \cdot n^{0.01}}{2}} =$

$= \infty \Rightarrow f \in \Omega(g)$

Notation Rules:

$\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = \text{Result}$

1. Result $= 0 \Rightarrow f(n) \in O(g(n))$
2. " $= \infty \Rightarrow f(n) \in \Omega(g(n))$
3. " $= \text{constant} \Rightarrow f(n) \in \Theta(g(n))$

(b) $f(n) = n!$, $g(n) = 2^n$ $\Rightarrow \lim\limits_{n\to\infty} \dfrac{n!}{2^n} = \dfrac{\infty}{\infty} \Rightarrow$ (Stirling formula) $\lim\limits_{n\to\infty} \dfrac{\sqrt{2\pi n} \cdot \frac{n^n}{e^n}}{2^n} \Rightarrow \lim\limits_{n\to\infty} \dfrac{(2\pi)^{1/2} \cdot n^{n+1/2}}{2^n \cdot e^n} = \infty$

Stirling Formula: $n! \cong \sqrt{2\pi n} \cdot \left(\dfrac{n}{e}\right)^n$

Because grown rate of exponential "n^n" is faster than grown rate of exponential "2^n"

Ex: n^n: $2^2 = 4$, $3^3 = 27$, $4^4 = 256$ ...
2^n: $2^2 = 4$, $2^3 = 8$, $2^4 = 16$ ...

$f \in \Omega(g)$

(c) $f(n) = \sqrt{n}$, $g(n) = (\log n)^3$ $\Rightarrow \lim\limits_{n\to\infty} \dfrac{\sqrt{n}}{(\log n)^3} = \dfrac{\infty}{\infty} \Rightarrow$ (1) L'Hospital $\lim\limits_{n\to\infty} \dfrac{\frac{1}{2} \cdot n^{-0.5}}{3(\log n)^2 \cdot \frac{1}{n \cdot \ln 2}} = \lim\limits_{n\to\infty} \dfrac{n^{0.5} \cdot \ln 2}{6 \cdot (\log n)^2} = \dfrac{\infty}{\infty}$

(2) L'Hospital Again
$\lim\limits_{n\to\infty} \dfrac{\frac{1}{2} \cdot n^{-0.5} \cdot \ln 2}{12 \cdot \log \cdot \frac{1}{n \cdot \ln 2}} = \lim\limits_{n\to\infty} \dfrac{n^{0.5} (\ln 2)^2}{24 \cdot \log n} \rightarrow$ (3) L'Hospital Again $\lim\limits_{n\to\infty} \dfrac{\frac{1}{2} \cdot n^{-0.5} \cdot (\ln 2)^2}{24 \cdot \frac{1}{n \cdot \ln 2}} \Rightarrow \lim\limits_{n\to\infty} \dfrac{n^{0.5} (\ln 2)^3}{48} = \infty \Rightarrow f \in \Omega(g)$

(d) $f(n) = n \cdot 2^n$, $g(n) = 3^n$ $\Rightarrow \lim\limits_{n\to\infty} \dfrac{n \cdot 2^n}{3^n} = \dfrac{\infty}{\infty} \Rightarrow$ (1) $\lim\limits_{n\to\infty} e^{\ln \frac{n \cdot 2^n}{3^n}} =$ (2) $\lim\limits_{n\to\infty} e^{(\ln(n 2^n) - \ln 3^n)}$ → we will deal with it

ex: $2^x = (e^{\ln 2})^x$

$= \lim\limits_{n\to\infty} \log n + \log 2^n - \log 3^n = \lim\limits_{n\to\infty} \log n + n \cdot \log 2 - n \cdot \log 3 \Rightarrow \lim\limits_{n\to\infty} n\left(\log 2 - \log 3 + \dfrac{\log n}{n}\right)$, $\dfrac{\log n}{n} \to 0$

$\boxed{\log 2 - \log 3 < 0} \Rightarrow \lim\limits_{n\to\infty} \dfrac{n \cdot 2^n}{3^n} = 0 \quad f \in O(g)$

We proved with logarithm

- Hamza YOĞURTCUOĞLU -

ⓔ $\dfrac{f(n)}{\sum_{i=1}^{n} i^k}$ $\dfrac{g(n)}{n^{k+1}}$ $\boxed{\sum_{i=1}^{n} i^k = 1^k + 2^k \quad + n^k}$ $\Rightarrow \lim_{n\to\infty} \dfrac{1^k + 2^k + \cdots + (n-2)^k + (n-1)^k + n^k}{n^k \cdot n} = \underset{\text{constant}}{\int}$

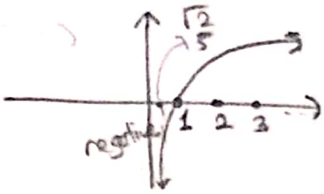
it means sum up from 1^k to n^k

• Both $f(n)$ and $g(n)$ are n term and both degree are k. There is proportion between $g(n)$ and $f(n)$. $f \in \Theta(g)$

ⓕ $\dfrac{f(n)}{2^n}$ $\dfrac{g(n)}{2^{n+1}}$ $\Rightarrow \lim_{n\to\infty} \dfrac{2^n}{2^{n+1}} = \lim_{n\to\infty} \dfrac{\cancel{2^n}}{\cancel{2^n}.2} = \lim_{n\to\infty} \dfrac{1}{2} = \boxed{\dfrac{1}{2}} \Rightarrow f \in \Theta(g)$

ⓖ $\dfrac{f(n)}{n^{1/2}}$ $\dfrac{g(n)}{5^{\log_2 n}}$ $\Rightarrow \lim_{n\to\infty} \dfrac{n^{1/2}}{5^{\log_2 n}} \Rightarrow \lim_{n\to\infty} \dfrac{n^{\log_2 \sqrt{2}}}{n^{\log_2 5}} = \lim_{n\to\infty} n^{\log_2 \sqrt{2}} \cdot n^{-\log_2 5}$

$= \lim_{n\to\infty} n^{\log_2 \sqrt{2} - \log_2 5} = \lim_{n\to\infty} n^{\log_2 \frac{\sqrt{2}}{5}} \xrightarrow{-1.8} \Rightarrow \lim_{n\to\infty} \dfrac{1}{n^{+1.8}} = 0$:)



$\Downarrow$

$f \in O(g)$

ⓗ $\dfrac{f(n)}{\log 2n}$, $\dfrac{g(n)}{\log 3n}$ $\Rightarrow \lim_{n\to\infty} \dfrac{\log 2n}{\log 3n} = \dfrac{\infty}{\infty} \Rightarrow$ L'Hospital $\lim_{n\to\infty} \dfrac{\dfrac{1}{2\cancel{n}.\cancel{\ln 2}}}{\dfrac{1}{3\cancel{n}.\cancel{\ln 2}}} = \dfrac{3}{2} \Rightarrow f \in \Theta(g)$

3) List the following functions according to their order of growth and prove your assertions

① $\log n$, ② $\sqrt{n+10}$, ③ $n+10$, ④ 10^n, ⑤ 100^n, ⑥ $n^2 \log n$, ⑦ $32^{\log n}$, ⑧ n^6

Firstly, Growth rates must be found by $f(n+1)$ over $f(n)$ for each functions.

① $\dfrac{f(n+1)}{f(n)} = \dfrac{\log(n+1)}{\log n} \Rightarrow$ $n=2 \Rightarrow \dfrac{\log 3}{\log 2} \cong 1.58$, $n=3 = \dfrac{\log 4}{\log 3} = 1.26$

② $\dfrac{f(n+1)}{f(n)} = \dfrac{\sqrt{n+11}}{\sqrt{n+10}} \Rightarrow$ $n=2 \Rightarrow \dfrac{\sqrt{13}}{\sqrt{12}} \cong 1.04$, $n=3 \Rightarrow \dfrac{\sqrt{14}}{\sqrt{13}} \cong 1.03$

③ $\dfrac{f(n+1)}{} = \dfrac{n+11}{}$ $n=2 \Rightarrow \dfrac{13}{} = 1.08$ ④

⑤ $\frac{f(n+1)}{f(n)} = \frac{100^{n+1}}{100^n} = 100 \rightarrow$ constant grown rate

⑥ $\frac{f(n+1)}{f(n)} = \frac{(n+1)^2 \cdot \log n+1}{n^2 \cdot \log n} \Rightarrow$ $n=2 \Rightarrow \frac{9 \cdot \log 3}{4 \cdot \log 2} = 3.56$

$n=3 \Rightarrow \frac{16 \cdot \log 4}{9 \cdot \log 3} = 2.24$

⑦ $32^{\log_2 n} \rightarrow$ we can swap n and 32 $\rightarrow n^{\log_2 32} \rightarrow n^5 \Rightarrow \frac{f(n+1)}{f(n)} = \frac{(n+1)^5}{n^5} \Rightarrow n=2 \Rightarrow \frac{3^5}{2^5} = 7.59$

$n=3 \Rightarrow \frac{4^5}{3^5} = 4.21$

⑧ $\frac{f(n+1)}{f(n)} = \frac{(n+1)^6}{n^6} = \; n=2 \Rightarrow \frac{3^6}{2^6} = 11.39$

$n=3 \Rightarrow \frac{4^6}{3^6} = 5.61$

Let's compare the closest growth then find the list.

- $\boxed{\underbrace{n+10}_{a} > \underbrace{\sqrt{n+10}}_{\sqrt{a}}} \Rightarrow a \geq \sqrt{a} \quad n>0$ all n
- $\boxed{100^n > 10^n} \Rightarrow 100 > 10$
- We simply $n^{\log_2 32} \rightarrow n^5$ so, $\boxed{n^{\log_2 32} > n^2 \cdot \log n}$ for all $n \geq 2$
- Then, $n^{\log_2 35} \rightarrow n^5$ so, $\boxed{n^6 > n^{\log_2 35}}$ $6>5$

$\rightarrow \log n$ is smallest polynomial function "

Not! There is no constant function

$$\log n < \sqrt{n+10} < n+10 < n^2 \cdot \log n < 32^{\log n} < n^6 < 10^n < 100^n$$

4) Analyze the complexity in time (big-Oh notation) of the following operations at a given binary search three (BST) that has height n:

a) Find Min.

b) Searching a Node.

c) Delete a leaf node.

d) Merging with another BST that has height n.

-Hamza YOĞURTCUOĞLU -
-171044086-

ⓐ Find Min

// The algorithm that is finding minimum node of Binary Search Tree

```
procedure FindMin(root)
    while root.left != null do
        root=root.left
    end while
    return root
end
```

//• The complexity of algorithm is T(n). Because, Algorithm traverse root to leftmost
//node and that could be traversed n (node numbers) time that happens worst case
// for left trees.

So, Complexity of the ~~a~~ algoritm. $T(n) \in O(n)$

ⓑ Searching a Node

// The algorithm that is searching desired node of Binary Search Tree.

```
procedure SearchNode(root, desiredNode)
①      if root.data== desiredNode.data OR root== NULL then
            return root
②      else if root.data> desiredNode.data then
            return (call SearchNode(root.Left, desired Node))
③      else
            return (call SearchNode(root.right, desiredNode))
```

c) Delete a leaf node.

```
// The algorithm that is removing desired leaf node of binary search tree.
(1)  procedure DeleteALeafNode (root, parentNode, deleteleaf)

(2)        if root == NULL     // if there is no what you want leaf node.
               return NULL
           end if

(3)        if root.data==deleteleaf.data AND root.left==NULL AND root.right then
(4)            if parent.leaf.data == deleteleaf.data then
                     delete parent.left
               else
                     delete parent.right
               endif
           else if root.data < deleteleaf.data
(5)              call DeleteALeafNode (root.right , root , deleteleaf)
           else
                 call DeleteALeafNode(root.left , root , deleteleaf)
           endif
           endif
     end
```

(1)- function gets parameters which are root (for checking leaf node), parentNode (the node is deleted node of parent), and deleteleaf (a leaf node).

(2) Base condition that there is no desired leaf node and return Nothing.

(3) if current node (root) is equal leaf node then, check current node is leaf node or not.

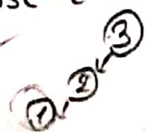
(4) if current node (root) is desired leaf node. In order to delete desired node, we have to check whether left or right of parent. Thus, we will delete a desired leaf.

(5) We are searching to delete a leaf through comparing with current node (root) and leaf node data.

each compare operation will take O(1) but worst case T(n) because that can be following example

(6) So, complexity → $T(n) \in O(n)$

ex:



• Binary search tree could be like right example.

- Hamza YOĞURTCUOĞLU -
- 171044086 -

d) Merging with another BST that has height n.

// If we merge the BST with another BST that will be done inorder traversal.

(1) procedure MergeAnotherBST (root , anotherBSTRoot)

(2) nodes = call convertSortedListInorderTraverser(anotherBSTRoot)

(3) while node : nodes do

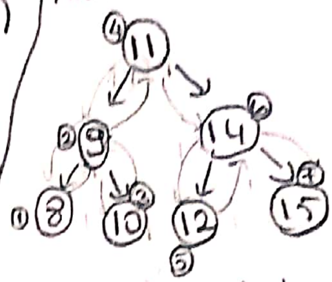
call insert (root, node)

end while

end

ex:

Another BST



This tree is perfect tree. that worst case for converting sort list. Every node traversed in $\boxed{2^{n+1}-1}$ with inorder traverse.

(1) → MergeAnotherBST functions is get arguments which first BST root and another BST root

(2) → In convert.... function traverse in AnotherBST and using the inorder traversal. with $2^{n+1}-1$ nodes. The function returns sorted list of nodes. (n height) → $T(2^n)$

(3) Each nodes of BST are inserted one by one. When we insert a node, running is taking long time → Each inserting

$\boxed{k}$ → is node of first BST ⇒ $T(k) + T(k+1) + \dots T(k+n) \Rightarrow O(k+n)$

5) Find the time complexity (big-Oh notation) of the following program.

```
void function (int n)
{
    int count = 0;
    for (int i=2; i<=n; i++)
        if (i%2==0)
        {
            count++;
        }
        else
        {
            i=(i-1)*i;
        }
}
```

// 2, 3, 7, 43, 1807 ...

// 2

// i = 6, 42, 1806 ..., i, i^2+i ...

42^2+42

In the loop, "i%2==0" is done just one time. → T(1) Then, all the time, index is increasing

i^2+i

- Hamza YOĞURTCUOĞLU -
- 171044086 -

Induction Proof $\rightarrow 2, 3, 7, 42, 1806 \ldots \quad i, i^2+i$

// we know "i" is a index, Let's say "t" iteration of "i"

$$i[t]^2 \leq i[t+1] = i[t]^2 + i[t]$$

$\rightarrow$ where c is a constant to be determined.

$\rightarrow$ Then taking the base-2 logarithm

$$2\log_2 i[t] \leq 2\log_2 i[t] + \log_2 c$$

$\rightarrow$ Taking algorithm again,

$$\log_2 2 \cdot \log i[t] \leq \log_2 2\log i[t] + \log_2 \log_2 c$$

$\rightarrow$ As, t represents the number of iterations of i for $n = i[t]$

$$\log_2 \log n \leq \log_2 \log n + \log_2 \log_2 c$$

$\rightarrow$ we can take $c \geq 2$ lets take 4

$$\log_2 \log n \leq \log_2 \log_2 n + \underbrace{\log_2 \log_2 4}_{1} \Rightarrow f(n) \in O(\log_2 \log_2 n)$$

- Hamza Yogurtcuoğlu -

- 171044086 -

Scanned by CamScanner