CSE 321 Introduction to Algorithm Design
Homework 1 Solution

1) a) The running time of algorithm A is at least $O(n^2)$? =>

=> Big-O notation gives us the upper bound. Upper bound meanings maximum running time. Using "at least", it is not true for big O, it gives minimum running time. It is not scientifically.

b) i) $2^{n+1} = O(2^n)$?

$2^n \cdot 2 \Rightarrow O(2^n)$ $\quad 2 \cdot 2^n \leq c \cdot 2^n \quad c=2, \forall n \geq 0$ True

ii) $2^{2n} = O(2^n)$? $\quad 2^{2n} \leq c \cdot 2^n \, / 2^n$

$2^n \leq c \Rightarrow$ it is not true for big-O False

$2^{2n} \neq O(2^n)$

c) $\max(f(n), g(n)) = \Theta(f(n)+g(n))$?

$\Theta(f(n)+g(n))$

$\max(f(n), g(n)) = O(f(n)+g(n))$ $\leq c$

$f(n) \leq f(n)+g(n)$, $g(n) \leq f(n)+g(n)$ $\Big\} \max(f(n),g(n)) \in O(f(n)+g(n))$

$\max(f(n), g(n)) = \Omega(f(n)+g(n))$ $\geq c$

$f(n)+g(n) \leq 2 \cdot \max(f(n),g(n)) \Rightarrow \max(f(n),g(n)) \in \Omega(f(n)+g(n))$

For Theta (Θ) notation, we prove O and Ω notations before.

If $f(n) \leq f(n)+g(n)$ and $g(n) \leq f(n)+g(n)$ are <u>true</u>,

$\Rightarrow \max(f(n), g(n)) \in O(f(n)+g(n))$ is true.

For Ω =

$\max(f(n), g(n)) \geq c \cdot (f(n)+g(n)) \Rightarrow f(n)+g(n) \leq 2 \cdot \max(f(n), g(n))$

$= \max(f(n), g(n)) \in \Omega(f(n)+g(n))$ is true.

O and Ω are true, then $\max(f(n), g(n)) \in \Theta(f(n)+g(n))$ is true.

2 a) $n^{1.01}$ $n\log^2 n$ $\lim \frac{n^{1.01}}{n\log^2 n} = \lim \frac{n^{0.01}}{\log^2 n} \xrightarrow{\text{L hospital}} \frac{0.01 \cdot n^{-0.99}}{2\log n \cdot \frac{1}{n \cdot \ln 10}} = \frac{0.01 \cdot n \cdot \ln 10}{n^{0.99} \cdot 2 \cdot \log n}$

$\lim \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = \Theta(g(n)) \Rightarrow n^{1.01} = \Theta(n\log^2 n)$

$= \frac{1}{\infty} = 0$

b) $n!$, 2^n $\lim_{n\to\infty} \frac{n!}{2^n} = \frac{\lim \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n}{2^n} = \lim \sqrt{2\pi n} \cdot \left(\frac{n}{2e}\right)^n = \infty$

Stirlings formula

$n! = \Omega(2^n)$

c) $\sqrt{n}$, $(\log n)^3$ $\lim_{n\to\infty} \frac{\sqrt{n}}{(\log n)^3} = \frac{\infty}{\infty}$ L'Hospital $\lim_{n\to\infty} \frac{\frac{1}{2\sqrt{n}}}{3(\log n)^2 \cdot \frac{1}{n \cdot \ln 2}}$

$\lim_{n\to\infty} \frac{\sqrt{n} \cdot \ln 2}{6 \cdot (\log n)^2} \xrightarrow{\text{L'Hospital}} \lim_{n\to\infty} \frac{\frac{\ln 2}{2\sqrt{n}}}{12(\log n)\frac{n^2}{n}} \Rightarrow \lim_{n\to\infty} \frac{(\ln 2)^2 \cdot \sqrt{n}}{24 \cdot (\log n)} \Rightarrow \lim_{n\to\infty} \frac{\frac{(\ln 2)^2}{2\sqrt{n}}}{24 \cdot \frac{1}{n \cdot \ln 2}}$

$\Rightarrow \lim_{n\to\infty} \frac{(\ln 2)^2 \sqrt{n}}{48} = \infty \longrightarrow f(\sqrt{n}) \in \Omega \cdot g((\log n)^3)$

d) $n \cdot 2^n$, 3^n $\lim_{n\to\infty} \frac{n \cdot 2^n}{3^n} = \lim_{n\to\infty} n \cdot \left(\frac{2}{3}\right)^n = \infty \cdot 0 = 0$

$n \cdot 2^n = O(3^n)$

e) $\sum_{i=1}^{n} i^k$, $n^{k+1} \rightarrow \underbrace{(n \cdot n \cdots n)}_{k+1}$

$\underbrace{1^k + 2^k + 3^k + \cdots n^k}_{n \text{ terms}} \leq \underbrace{n \cdot n^k}_{n^{k+1}} \rightarrow \sum_{i=1}^{n} i^k \in O(n^{k+1})$ ✓

$f(n) = \sum 1^k + 2^k + 3^k \cdots n^k$

$\sum_{i=1}^{n} i^k \Rightarrow \sum_{i=1}^{n/2} i^k + \sum_{i=\frac{n}{2}+1}^{n} i^k \geq \sum_{i=\frac{n}{2}+1}^{n} i^k \Longrightarrow \underbrace{\left(\frac{n}{2}+1\right)^k + \left(\frac{n}{2}+2\right)^k \cdots + n^k}_{} \geq \frac{n}{2}\left(\frac{n}{2}\right)^k = \frac{n^{k+1}}{2^{k+1}}$

$n/2$ terms, each terms at least $\left(\frac{n}{2}+1\right)^k \geq \left(\frac{n}{2}\right)^k$

big O and Ω =

$\Rightarrow \sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$

$\Rightarrow \sum_{i=1}^{n} i^k \in \Omega(n^{k+1})$ ✓

(2)

f) 2^n , 2^{n+1}

$2^n \le c \cdot 2^{n+1}$ $\quad f(n) = O(g(n))$ ✓
$c \geq 1$
if $c > 0$ $\quad f(n) = \Omega(g(n))$ ✓

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

g) $n^{1/2}$, $5^{\log_2 n}$

$\lim\limits_{n\to\infty} \frac{n^{\frac{1}{2}}}{5^{\log_2 n}} = \lim\limits_{n\to\infty} \frac{n^{\frac{1}{2}}}{n^{\log_2 5}} = \lim\limits_{n\to\infty} n^{\frac{1}{2} - \log_2 5} = \lim\limits_{n\to\infty} n^{\frac{1}{2} - 2{,}32} = \lim\limits_{n\to\infty} n^{-1{,}821}$

$= \lim\limits_{n\to\infty} \frac{1}{n^{1{,}821}} = \frac{1}{\infty} = 0 \quad n^{\frac{1}{2}} = O(5^{\log_2 n})$

h) $\log 2n$, $\log 3n$

$\lim\limits_{n\to\infty} \frac{\log 2n}{\log 3n} = \lim\limits_{n\to\infty} \frac{\frac{\cancel{2}}{\ln 2 \cdot 2n}}{\frac{3}{\ln 2 \cdot 3n}} = 1$ constant.

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

(3) $\log n$, $\sqrt{n+10}$, $n+10$, 10^n, 100^n, $n^2 \log n$, $32^{\log n}$, n^6
(a) (b) (c) (d) (e) (f) (g) (h)

$32^{\log n} \to 2^{5\log n}$

a) $\frac{f(n+1)}{f(n)} = \frac{\log(n+1)}{\log n}$

e) $\frac{f(n+1)}{f(n)} = \frac{100^{n+1}}{100^n} = 100$

b) $\frac{f(n+1)}{f(n)} = \frac{\sqrt{n+11}}{\sqrt{n+10}} = \sqrt{\frac{n+11}{n+10}} = \sqrt{1 + \frac{1}{n+10}}$

(f) $\frac{f(n+1)}{f(n)} = \frac{(n+1)^2 \cdot \log(n+1)}{n^2 \cdot \log n}$

c) $\frac{f(n+1)}{f(n)} = \frac{n+11}{n+10} = 1 + \frac{1}{n+10}$

g) $\frac{f(n+1)}{f(n)} = \frac{32^{\log(n+1)}}{32^{\log n}} = \frac{(n+1)^{\log 32}}{n^{\log 32}} = \left(\frac{n+1}{n}\right)^{\log 32} = \left(\frac{n+1}{n}\right)^{1{,}50}$

d) $\frac{f(n+1)}{f(n)} = \frac{10^{n+1}}{10^n} = 10$

h) $\frac{f(n+1)}{f(n)} = \frac{(n+1)^6}{n^6} = \left(\frac{n+1}{n}\right)^6$

$\Longrightarrow$

Now, we compare their results:

- a-b compare: we found $a = \frac{\log(n+1)}{\log n}$ and $b = \sqrt{1 + \frac{1}{n+10}}$ $b > a$, where $n > 1$

$\Rightarrow$ then $\boxed{\sqrt{n+10} > \log n}$

- b-c compare: $b = \sqrt{1 + \frac{1}{n+10}}$ & $c = 1 + \frac{1}{n+10}$ $c > b$, where n is positive number.

$\Rightarrow$ then $\boxed{n+10 > \sqrt{n+10}}$

"d" and "e" are constant and they are highest.
Now we compare "c" with a close value. This is "f"

- c-f compare: $c = 1 + \frac{1}{n+10}$ and $f = \frac{(n+1)^2 \cdot \log(n+1)}{n^2 \cdot \log n}$, $f > c$, where $n > 1$

$\Rightarrow$ then $\boxed{n^2 \log n > n+10}$

- f-g compare: $f = \frac{(n+1)^2 \cdot \log(n+1)}{n^2 \log n}$ and $g = \left(\frac{n+1}{n}\right)^{\log 32} = \left(\frac{n+1}{n}\right)^{1,5}$

$g > f$, where $n > 1$
$\Rightarrow$ then $\boxed{32^{\log n} > n^2 \log n}$

- g-h compare $g = \left(\frac{n+1}{n}\right)^{1,5}$ and $h = \left(\frac{n+1}{n}\right)^6$ $h > g$ where $n > 0$

$\Rightarrow$ then $\boxed{n^6 > 32^{\log n}}$

- d-h compare:

$d = 10$ and $h = \left(\frac{n+1}{n}\right)^6$ $d > h$, where $n \geq 0$

$\Rightarrow$ then $\boxed{10^n > n^6}$

d-e compare. $d = 10$ $e = 100$ $e > d$ where $n \geq 0$

$\Rightarrow$ then $100^n > 10^n$

The Result:

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

④

a) procedure findMin (root):

this = root

while (this.left is not Null) do

this = this.left

end while

return this

end

The tree's maximum heigh is n.
Then the complexity is $T(n)$

$T(n) \in O(n)$

b) procedure searchNode (root, target):

if (root is equal target OR root is Null) do } 3

return root } 1

end if

if (root.value is greater then target) do

return searchNode (root.right, target) } recursive call 1

else

return searchNode (root.left, target) } recursive call 2

end if

end

Tree's heigh is n. The Time Complexity is $T(n)$.

$T(n) = 2T(\frac{n}{2}) + 4$

We know "Master Theorem", $T(n) = a.T(\frac{n}{b}) + f(n)$

$a=2 \quad b=2 \quad d=0$

$a > b^d \Rightarrow 2 > 2^0 \rightarrow 2 > 1$

$\Rightarrow T(n) = \Theta(n^{\log_2 2})$

④ d)

```
procedure mergeBST(root1, root2):
    merge create new list root1.size + root2.size                        } 1

    while (i1 < root1.size AND i2 < root2.size) do                      \
        if (root1 in index i1 <= root2 in index i2) do } 1               |
            merge.add (root1 in index i1) } 2                            | n
            i1 ++                                                        |
        else                                                             |
            merge.add(root2 in index i2) } 2                             |
            i2 ++                                                        /
        endif
    end while

    while (i1 < root1.size) do                                           \
        merge.add (root1 in index i1) } 2                                | n
        i1++                                                             /
    end while

    while (i2 < root2.size) do                                           \
        merge.add (root2 in index i2) } 2                                | n
        i2++                                                             /
    end while

    return merge                                                         } 1

end
```

$T(n) = 1 + n + n + n + 1 + c$

$3n + 2 = 3n = \underline{n \in O(n)}$

(5)

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

In for loop, "i" starts 2. first run, first "if" condition executes and "count" variable increases then "count" will be 1.
Second run, "i" is 3, then "else" condition executes, and $i = (i-1) * i$ will run, this time "i" will be $2 * 3 = 5$.
In for loop line, "i" increase one more time then "i" will be 6.
Third run for loop, $i = 6$, "else" condition executes again, then "i" will be $5 * 6 = 30$. Then "i" increases, "i" will be 31.
In other steps "i" will be odd number very time and else condition executes every step. For this result for loop's complexity is log(log n). Then this function's complexity is $O(\log(\log n))$.

$i = (i-1) * i$

$E = O$ E
+1
Odd

$E = E$ O
+1
Odd

Ahmet Tuşa Telli
151044092

CamScanner ile tarandı