Academic Year	Module	Assessment Number	Assessment Type
S20	Introductory Data Structures and Algorithms (DipIT02)	A1	Assignment Submission

[Assignment Submission]

Student Id : [NP03A190299]
Student Name : [Yogesh Shrestha]
Section : [DC8]
Module Leader : [Mr. Prakash Gautam]

Submitted on : 06-03-2020

Page No.
Date : / /

Tutorial 1

1 → $n = 1$ $0, 2^1 = 2$
$n = 2$ 1 2^2

$n = n$ 2^n

2 →. Strength (AVL Tree)
a) Balanced BST so searching of any element will give $O(\log n)$ in worst case.
b) Finding minimum and maximum can be done in $O(\log n)$ time.
c) Insertion and deletion of element can be done in $O(\log n)$ time.

Limitation.
a) After every insertion, deletion, balanced property need to be checked and in case it ~~violat~~ violate, rotations are required which increases complexity.

3 → If police are searching for the criminal there should be best solution but if we are measuring the length between two points "approximately" is good enough.

4 → Let $T(n)$ be the running time of the algorithm A and let a function $f(n) = O(n^2)$. That is $T(n)$ is an upper bond of $f(n)$. Since, $f(n)$ could be the function smaller than n^2. we can correct the statement at constant.

Page No.
Date: / /

5)

a. Is $2^{n+1} = O(2^n)$ Is $2^{2n} = O(2^n)$?

$2^n . 2^1 = O(2^n)$ Or, $2^{2n} = O(2^n)$

Yes.

$n=0$	2^n	2^n		$n=1$	2^2	2^1
$n=1$	2	2		$n=2$	2^4	2^2
$n=2$	4	4	its true.		So. its false	

6=> Input Insertion Sort Quick sort

n $8n^2$ $64n \log_2 n$

Dividing both by $8n$

	n	$8 \log_2 n$
$n=2$	2	~~$8 \log_2 n$~~ $8 \times 1 = 8$
$n=32$	32	$8 \times 4 = 34$
$n=64$	64	$8 \times 6 = 48$
$n=43$	43	43.04

So, IS is better in $n \leq 43$ than QS.

7->

n	First algorithm ($100n^2$)	Second algorith (2^n)
1	100	2
2	400	4
10	10000	1024
16	25600	65536
15	22500	32768

After $n=15$ second algorithm is better.

Page No.
Date : / /

8→. Soln.

$\Theta(g(n)) \Rightarrow C_1 g(n) \leq f(n) \leq C_2 g(n)$

$C_1 g(n) \leq f(n) \Rightarrow \Omega(g(n))$

$f(n) \leq C_2 g(n) \Rightarrow O(g(n))$

9→. Soln.

$f(n) = f(n-1) + n$

$f(1) = 1$

when, $f(1) = 1$ $n = 1$

$f(1) = f(0) + 1$

$= 1$

$n = 2$

$f(2) = f(2-1) + 2$

$= f(1) + 2$

$= 1 + 2$

$n = 3$ $f(3) = f(3-1) + 3$

$= 1 + 2 + 3$

$n = n$ $f(n) = 1 + 2 + 3 + \ldots + n$ — (i)

$f(n) = 1 + (n-1) + (n-2) + (n-3) + \ldots + 1$ (ii)

Adding (i) & (ii)

$2f(n) = (1+n) + (n+1) + (n+1) + \ldots + (n+1)$

$= (n+1)\{1 + 1 + 1 + \ldots + 1\}$

$= (n+1)(n)$

$= \frac{n(n+1)}{2}$

$f(n) = \frac{n^2}{2} + \frac{n}{2} = O(n^2)$ $\therefore f(n) = O(n^2)$

Page No.
Date : / /

12 → Sol^n

$f(n) = 1 + 2 + 3 + \ldots + n$ — (i)

$f(n) = n + (n-1) + (n-2) + (n-3) + \ldots + 1$ — (ii)

Add (i) & (ii)

$2f(n) = (n+1) + (n+1) + (n+1) + \ldots + (n+1)$

$= (n+1)\{1+1+1 \ldots + 1\}$

$= (n+1)(n)$

$= \frac{n(n+1)}{2}$

$= \frac{n^2}{2} + \frac{1}{2}$

Thus we can say $O(n^2)$.

13 → Sol^n

$n+1 \quad n \rightarrow n+n+1$

$\sqrt{n} \quad n \rightarrow \sqrt{n}+n$

$\sqrt{n} \quad n+1+n+1+n \rightarrow \sqrt{n}+n+1+n+1+n$

$n+n+1+\sqrt{n}+n+\sqrt{n}+1$

$= O(n)$

Page No.
Date : / /

Soln.

10 a) $T(n) = T(n-1) + n$

$T(0) = 1$ (when $n=0$, function does nothing & execution constant T)

$T(1) = T(0) + 1$
$= 1+1$

$T(2) = T(1) + 2 = 1+1+2$

$T(3) = T(2) + 3 = 1+1+2+3$

$T(n) = 1+1+2+3+\ldots+n$.(i)

$T(n) = n+(n-1)+(n-2)+(n-3)+\ldots+1$.(ii)

Adding (i) & (ii)

$2T(n) = (n+1)+(n+1)+(n+1)+\ldots+(n+1)$
$= (n+1)\{1+1+1+\ldots+1\}$
$= (n+1)(n)$
$= \frac{(n)(n+1)}{2}$
$= \frac{n^2}{2} + \frac{1}{2}$
$O(n^2)$

b) $T(n) = 3^n + n\log_2 n^4 + 7$

$3^n + n\log_2 n^4 \leq 300 * 3^n$ ~~O( )~~ So, $O(3^n)$.

c) $T(n) = n\frac{\pi}{2}\log_2 n + (n-1)$

$= n\frac{\pi}{2}\log_2 n + (n-1) < 100n\frac{\pi}{2}\log n$

$O(n\log_2 n)$.

Page No.
Date : / /

d). $T(n) = 3n^2 + \frac{n}{10^3} + 47.$

$T(n) \leq C.g(n)$ for $n \geq n_0$

here we take, $C = 51$

$g(n) = n^2$

$3n^2 + \frac{n}{10^3} + 47 \leq 51n^2$

Thus, we can say $O(n^2)$

e) $T(n) = 200n^2 + n - 9$

$T(n) \leq C.g(n)$ for $n \geq n_0$

here we take, $C = 200$

$g(n) = n^2$

$100n^2 + n - 9 \leq 200n^2$

Thus, we say $O(n^2)$

f) $T(n) = (n-2)^2 + \frac{n}{3^4}$

$= n^2 - 4n + 4 + \frac{n}{3^4}$

$= n^2 - 4n + \frac{n}{3^4} + 4$

$T(n) \leq C.g(n)$ for $n \geq n_0$

$C = 200$

$g(n) = n^2$

$n^2 - 4n + \frac{n}{3^4} + 4 \leq 200n^2$ Thus, $O(n^2)$.

Page No.
Date: / /

1-> let's take $T(n) = 3 + T(n/2)$ Binary Search.
So we

we know $T(1) = 1$.

$T(n) = 3 + T(n/2)$ - (1)

$T(n/2) = 3 + T(n/4)$ - (2)

$T(n/4) = 3 + T(n/8)$ - (3)

From eqn (i), (ii) & (iii)

$T(n) = 3 + T(n/2)$

$= 3 + 3 + T(n/4)$

$= 3 + 3 + 3 + T(n/8)$

So, in general,

$T(n) = 3k + T(n/2^k)$ - (4)

At the same point we get $\frac{n}{2^k} = 1$

Then, $\frac{n}{2^k} = 1$

$n = 2^k$

taking log on both side

$\log n = \log_2 (2^k)$

$\log n = k$ -- (v)

in eqn (4) & (5)

$T(n) = 3\log n + T(n/2^{\log n})$

$= 3\log n + T(n/n)$

$= 3\log n + T(1)$

$= 3\log n + 1$

For eqn (6) we can choose $c\,g(n)$ such that $T(n) \le c.g(n)$ for $n \ge n_0$

Page No.
Date: / /

Where: $C = 4$
$g(n) = \log n$
$3\log n + 1 \leq 4\log n$
Then, we can say $O(\log n)$

la linear Search
$T(n) = 4n + 3$
$T(n) \leq C g(n)$ for $n \geq n_0$
We take,
$C = 8$
$g(n) = n$.
So, $4n + 3 \leq 8n$
Thus, we can say $O(n)$