CSE221

Assignment 1

24301500
Mazidul Islam Kabbo
section 9

1. We generally choose Binary search, Binary Search divides the space into 2 halves while Ternary Search divides it into 3 parts, but Ternary performs more comparison per step. Both algorithm has a time complexity of $O(\log n)$ but due to more comparison in ternary, we choose Binary. We cant always use these 2 algorithm. To use them data must satisfy certain conditions, like for binary, data must be sorted, or for ternary, data must be unimodal or, it first increases (values) then decreases.
When these conditions dont satisfy, we must use other algorithms like linear search, DFS/BFS, etc.

*. all loops independant time complexity shown first, then in the end, overall time complexity

2 a) Outer loop:

$$\frac{n}{2}$$

$$\frac{n}{2\times 6}$$

$$\frac{n}{2\times 6^2}$$

$$\frac{n}{2\times 6^3} \Rightarrow \frac{n}{2\times 6^k} = 1$$

$$\frac{n}{2} = 6^k$$

$$\log\frac{n}{2} = k\log 6$$

$$\log_6\frac{n}{2} = k$$

$\therefore O(\log n)$

middle loop:

$$2$$

$$2\times 4$$

$$2\times 4^2$$

$$2\times 4^k = n$$

$$4^k = \frac{n}{2}$$

$$k\log 4 = \log\frac{n}{2}$$

$$k = \log_4\frac{n}{2}$$

$\therefore O(\log n)$

inner loop:

$$2$$

$$2\times 3^1$$

$$2\times 3^2$$

$$2\times 3^k = j \quad [j=n]$$

$$2\times 3^k = n$$

$$k\log 3 = \log\frac{n}{2}$$

$$k = \log_3\frac{n}{2}$$

$\therefore O(\log n)$

So, overall time complexity is $O((\log n)^3)$

b) First Outerloop:

n
$n-1$
$n-2$
$n-k=1$
$n-1=k$

$O(n)$

First outerloop's inner loop:

j	j^2	p
2	4	0
3	9	n
4	16	$2n$
k	$k^2=n$	$kn = n+i$ [$i=n$ (worst case)]
	$k=\sqrt{n}$	$k\cancel{n}=2\cancel{n}$
		$k=2$

so, $O(1)$

so, First part time complexity is $O(n)$

Second part:

Outer loop:

$\frac{n}{2}$

$\frac{n}{2}\times\frac{1}{6}$

$\frac{n}{2\times 6^k}=1$

$n=2\times6^k$

$\frac{n}{2}=6^k$

$\log\frac{n}{2}=(\log 6)\times k$

$k=\log_6\frac{n}{2} \Rightarrow O(\log n)$

middle loop:

2

2×4

2×4^2

$2\times4^k=i$ [$i=n$ (worst case)]

$2\times4^k=n$

$k\log 4=\log\frac{n}{2}$

$k=\log_4\frac{n}{2} \Rightarrow O(\log n)$

inner loop:

1

1×3

1×3^2

$1\times3^k=j$ [$j=n$ (worst case)]

$3^k=n$

$k\log 3=\log n$

$k=\log_3 n$

$O(\log n)$

so, for second part overall time complexity is $O((\log n)^3)$

so, $T(n)=O(n)+O((\log n)^3)$

since $O(n)$ dominant, Overall time complexity is $O(n)$

3. $f(n) = \log n + 5$; $g(n) = \log n^2 = 2\log n$

$\log n + 5 \geq 2c_1 \log n$ — ①

$\log n + 5 \leq 2c_2 \log n$ — ⑪

①

$\log n + 5 \geq 2c_1 \log n$

for $c_1 = \frac{1}{2}$

$\log n + 5 \geq \log n$

$5 \geq 0 \log n \Rightarrow 5 \geq 0$

for all values of n, this condition satisfies

⑪

$\log n + 5 \leq 2c_2 \log n$

for $c_2 = 10$

$\log n + 5 \leq 20 \log n$

$5 \leq 19 \log n$

for all values of $n \geq 2$, this condition satisfies

so, for,

$c_1 = \frac{1}{2}$, $c_2 = 10$, $n_0 = 2$, we have proved that

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

4. $f(n) = n^2 \log n^2$

$= 2n^2 \log n$

this is

$O(n^2 \log n)$

$= \log n^3$

$= 3 \log n$

this $O(\log n)$

so $f(n)$ is not $O(\log n^3)$ or $O(\log n)$

5. i) int i=1
while (i*i <= key)
{ i=i+1; }
return i-1;

ii) int L=0; int answer=0;
int R= key;
while (L<=R)
{ if (mid* mid == key)
{ return mid; }
else if (mid* mid < key)
{ L=mid+1; answer= mid; }
else { R= mid-1; }

}
return answer;

6. step1: consider Left as index 0, right as index arr.length-1
step 2: Run a loop while L< R
Loop steps:
step 1: Find mid index using (L+R)/2
step 2: if (arr[mid] > arr[R]) then
compare { L= mid+1; }
else { R=mid; }
step 3: return arr[L];