PAGE No
DATE

Tutorial Sheet 1

1).

3. $O(N+M)$ times, $O(1)$ space

2). $O(n)$ times, $O(1)$ space

3). $O(\log_2 n)$ times, $O(1)$ space

4). $O(\sqrt{n})$ times, $O(1)$ space

5). $n=0,\ 1$

$n=1,\ 2$ $\Rightarrow 0, 1, 3, 6, 10, \ldots n$

$n=2,\ 3$

$k^{th} = \frac{k \times (k+1)}{2}$

$\Rightarrow O(\sqrt{n})$ times, $O(1)$ space. $n = \frac{k^2+k}{2}$

$k^2 \approx n$

$k \approx \sqrt{n}$

6).

```
void recursion(int n)          -> T(n)
{ if (n==1) return;            -> 1
    recursion (n-1);           -> T(n-1)
    print(n);                  -> 1
    recursion(n-1);            -> T(n-1)
}
```

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

$T(1) = 1$

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

$T(n) = 4T(n-2) + 1 + 2$

$T(n) = 8T(n-3) + 1 + 2 + 4$

$T(n) = 2^k T(n-k) + [2^0 + 2^1 + 2^2 + \dots 2^k]$

$n - k = 1$

$k = n - 1$

$T(n) = O(2^n)$

7). it is binary search algo.

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

$T(n) = a\,T(n/b) + f(n)$ (∵ Master's Method).

so, $a = 1$, $b = 2$, $f(n) = 1$

$c = \log_b a = 0$

$n^c = f(n) = 1$

$T(n) = O(n^0 \log n)$

8)

1). $T(n) = T(n-1)+1$ $\qquad T(1)=1$

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

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

$T(n) = T(n-k)+k(1)$

$n-k=1$

$n-1=k$

$T(n) = O(n-1) = O(n)$

2). $T(n) = T(n-1)+n$ $\qquad T(1)=1$

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

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

$T(n) = T(n-k)+n+(n-1)+\dots+(n-k-1)$

$n-k=1$

$(n-1)=k$

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

$T(n) = O(n^2)$

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

$O(n^0 \log n)$

4). $T(n) = 2T(n/2)+1$

here, $a=2$ $b=2$ $f(n)=1$

$c=\log_b a = 1$

$\Rightarrow O(n)$

5). $T(n) = 2T(n-1)+1$

$T(n) = 4T(n-2)+1+2$

$T(n) = 8T(n-3)+1+2+4$

$T(n) = 2^k T(n-k)+2^0+2^1+2^2+\dots 2^{k-1}$ $\qquad k=n-1$

$T(n) = 2^{n-1}\times 1+[2^0+2^1+2^2+\dots 2^n]$

$\Rightarrow O(2^n)$

6). $T(n) = 3T(n-1), T(0) = 1$

$T(n) = 9T(n-2)$

$T(n) = 27T(n-3)$

$T(n) = 3^k T(n-k)$

$\Rightarrow O(3^n)$

7). $T(n) = T(\sqrt{n}) + 1$, $T(2) = 1$

$T(n) = T(n^{1/4}) + 1 + 1$

$T(n) = T(n^{1/8}) + 3$

$T(n) = T(n^{1/2^k}) + k$

$n^{(1/2)^k} = 1$

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

$2^k = \log n$

$k = \log_2(\log n)$

$\Rightarrow O(\log[\log n])$

8). $T(n) = T(\sqrt{n}) + n$

$T(n) = T(n^{1/4}) + n + n^{1/2}$

$T(n) = T(n^{1/8}) + n + n^{1/2} + n^{1/4}$

$T(n) = T(n^{(1/2)^k}) + \Big[\underbrace{n^{(1/2)^0} + n^{(1/2)^1} + n^{(1/2)^2} + \cdots\ n^{(1/2)^{k-1}}}_{k \text{ term}}\Big] \rightarrow G.P.$

$= n \times \left(\frac{\sqrt{n}^{k} - 1}{k - 1} \right)$

$\approx n$

$n^{1/2^k} = 2$

$\frac{1}{2^k} = \frac{1}{\log n}$

$k = \log(\log n)$

$T(n) = O(n \log(\log n))$

9). $O(n)$ times, $O(1)$ space

10). 0 → n times

1 → n-1 times

2 → n-2 "

:

:

n → 0 times

$O(n * \frac{(n+1)}{2}) \Rightarrow O(n^2)$

11). j loop run for $O(\log n)$
i loop run for $O(n/2)$

$T(n) = O(n * \log n)$

12). (a). X will always be a better choice for large inputs

13). $O(\log n)$

14). $T(n) = 7T(n/2) + 3n^2 + 2$
$a = 7,\ b = 2,\ f(n) = 3n^2 + 2$
$c = \log_2 7 = 2.81$

a, b & c all three option are correct

15). f2 > f4 > f3 > f1

16). $f(n) = 2^{(2n)}$
$= 2^n * 2^n$
So, option b is correct.

17). T(n) = 2T(n/2) + n^2

O(n^2)

18).
```
int gcd(int n, int m){
    if(n%m == 0) return m;
    if (n<m) swap(n,m);
    while (m>0){
        n=n%m;
        swap(n,m);
    } return n;
}
```

=> $\theta(\log n)$