Contents

1 Solution summary

Q No	O(n)
1	log n
2	log n
3	log n
4	n^2*log n
5	n*log n
6	$n^{(1/2)}$
7	n^2

2 Detailed Solution

2.1 Find T(n) and Big-Oh

```
A()
{
  while(n>1)
    {
      n=n/2;
    }
}
```

Statement	O(n)
$n > 1$	$(\log_2 n) + 1$
$n = n/2$	$\log_2 n$

$T(n) = \sum O(n)$
$T(n) = \log_2 n + 1 + \log_2 n$
$T(n) = 2*\log_2 n + 1$
let $f(n) = T(n)$
$f(n) = O(g(n))$ if $f(n) <= c(g(n))$
$2*\log_2 n + 1 <= c*\log_2 n$
dividing by $\log_2 n$ both sides
$c = 4$ and and $N >= 2$
$f(n) <= 4*\log_2 n$ for $N >= 2$
Time complexity is $= O(\log n)$

2.2 Find T(n) and Big-Oh

```
A()
{
  for (i=1; i<=n; i=i*2)
    print("welcome");
}
```

Statement	O(n)
$i = 1$	1
$i <= n$	$\log_2 n + 1$
$i = i*2$	$\log_2 n$
print	$\log_2 n$

$T(n) = \sum O(n)$
$T(n) = 1 + \log_2 n + 1 + \log_2 n + \log_2 n$
$T(n) = 3*\log_2 n + 2$
Time complexity $= O(\log n)$

2.3 Find T(n) and Big-Oh

```
A()
{
  for (i=1; i<=n; i=i*3)
    print("welcome");
}
```

Statement	O(n)
i = 1	1
i <= n	$\log_3$ n +1
i = i*3	$\log_3$ n
print	$\log_3$ n

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

T(n) = 1 + $\log_3$ n + 1 + $\log_3$ n + $\log_3$ n

T(n) = 3*$\log_3$ n + 2

Time complexity = O(log n)

2.4 Find T(n) and Big-Oh

```
A()
{
  int i, j, k;
  for (i=n/2; i<=n; i++)
    for (j=1; j<=n/2; j++)
      for (k=1; k<=n; k=k*2)
{
  print("welcome");
}
}
```

Statement	O(n)
i = n/2	1
i <= n	n/2+1
i++	n/2
j = 1	n/2(1)
j <= n/2	n/2(n/2+1)
j++	n/2(n/2)
k = 1	n^2/4(1)
k <= n	n^2/4($\log_2$ n+1)
k = k*2	n^2/4($\log_2$ n)
print	n^2/4(1)

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

T(n) = n^2/4($\log_2$ n)+ ...

Time complexity O(n^2 * log n)

2.5 Find T(n) and Big-Oh

```
A()
{
  int i,j,k;
  for (i=n/2; i<=n; i++)
    {
      for (j=1; j<=n; j=2*j)
{
  for (k=1; k<=n; k=k*2)
    print("welcome");
}
    }
}
```

Statement	O(n)
i,j,k	3
i=n/2	1
i <= n	n/2+1
i++	n/2
j = 1	n/2(1)
j <= n	n/2($\log_2$ n + 1
j = j*2	n/2($\log_2$ n)
k = 1	n/2($\log_2$ n)(1)
k <= n	n/2($\log_2$ n)($\log_2$ n)+1
k = k*2	n/2($\log_2$ n)($\log_2$ n)

$T(n) = \sum O(n)$
$T(n) = 3 + 1 + n/2+1 + n/2 + n/2 + n/2(\log_2 n + 1) \ldots$
$T(n) = n^*(\log_2 n)^2 + \log_2 n + n + c$
$T(n) = 2^*n(\log_2 n) + \log_2 n + n + c$
$f(n) = T(n)$
$2^*n(\log_2 n) + \log_2 n + n <= c^*n(\log_2 n)$
$c = 3.5$ when $n > 2$
Time complexity = O(n*log n)

2.6 Find T(n) and Big-Oh

```
AC()
{
  i=1,s=1;
```

```
  while (s<=n)
    {
      i++;
      s=s+i;
      print("welcome");
    }
}
```

Statement	O(n)
i =1, s=1	2
s <= 2	k(k+1)/2 +1 > n
i++	k(k+1)/2 > n
s = s+i	k(k+1)/2 > n

$T(n) = \sum O(n)$
$T(n) = 2 + 3*k(k+1)/2 > n$
$k*(k+1) > c*n$
$k^2 = c*n$
$k = n^{(1/2)}$
Time complexity $= O(n^{(1/2)})$

2.7 Find T(n) and Big-Oh

```
A()
{
  int i, j, k, n;
  for (i=1; i<=n; i++)
    {
      for (j=1; j<=i; j++)
{
  for (k=1; k<=100; k++)
    {
      print("welcome");
    }
}
    }
}
```

Statement	O(n)
i,j,k,n	4
i=1	1
i <= n	n+1
i++	n
j=1	n(1)
j <= 1	n(n+1)/2+1
j++	n(n+1)/2
k = 1	n(n+1)/2
k <= 100	n(n+1)/2*(100)+1
k++	n(n+1)/2*(100)
print	n(n+1)/2*(100)

$T(n) = \sum O(n)$
$T(n) = c*n(n+1)$
Time complexity = $O(n^2)$