ANALYSIS OF ALGORITHMS

(SET: 1)

Solutions

Q.1 The running time of an algorithm T(n), where 'n' is the input size, is given by—

$T(n) = 8\left[\left(\frac{n}{2}\right) + qn, \text{if } n > 1\right] = p, \text{if } n = 1$

where p, q are constants. The order of this algorithm is—

(a) n^2 (b) n^n
(c) n^3 (d) n

Solution: Option (c)

Consult Master's theorem.

Q.2 An algorithm is made up of 2 modules M_1 and M_2. If order of M_1 is f(n) and M_2 is g(n) then the order of the algorithm is—

(a) max(f(n), g(n)) (b) min(f(n), g(n))
(c) f(n) + g(n) (d) f(n) * g(n)

Solution: Option (a)

Q.3 The concept of order (Big O) is important because—

(a) it can be used to decide the best algorithm that solves a given problem
(b) it determines the maximum size of a problem that can be solved in a given system, in a given amount of time
(c) it is the lower bound of the growth rate of the algorithm
(d) Both (a) and (b)

Solution: Option (d)

Q.4 The running time T(n), where 'n' is the input size, of a recursive algorithm is given as follows—

T(n)= C+T(n-1), if n>1
= d, if n$\leq$ 1

The order of the algorithm is—

(a) n^2
(b) n
(c) n^3
(d) n^n

Solution: Option (b)

T(n)= C + T(n-1)
= C + C + T(n-2)
= C + C + C + T(n-3)
= C + C + …….n time + T(1)
= nC +d
T(n)= O(n)

Q.5 There are 4 different algorithms. A1, A2, A3, A4 to solve a given problem with the order log(n), log(log(n)), nlog(n), n/log(n) respectively. Which is the best algorithm?

(a) A1
(b) A2
(c) A4
(d) A3

Solution: Option (b)

Put values of n, and check for the algorithm which gives comparatively least value for a given value of n.

Q.6 The time complexity of an algorithm T(n), where n is the input size, is given by—
T(n)= T(n-1) + 1/n, if n>1
= 1, otherwise.

The order of the algorithm is—

(a) log n
(b) n
(c) n^2
(d) n^n

Solution: Option (a)

$$\mathrm{T(n)} = \mathrm{T(n-1)} + \frac{1}{\mathrm{n}}$$
$$= \frac{1}{\mathrm{n}} + \frac{1}{\mathrm{n}-1} + \mathrm{T(n-2)}$$
$$= \frac{1}{\mathrm{n}} + \frac{1}{\mathrm{n}-1} + \frac{1}{\mathrm{n}-2} + \mathrm{T(n-3)}$$
$$= \frac{1}{\mathrm{n}} + \frac{1}{\mathrm{n}-1} + \frac{1}{\mathrm{n}-2} + \cdots + 1$$

= log(n)

Q.7 The running time of an algorithm is given by,
T(n)= T(n-1) + T(n-2) – T(n-3), if n>3
= n, otherwise.

The order of this algorithm is—

(a) n
(b) log n
(c) n^n
(d) n^2

Solution: Option (a)

T(4)= T(3) + T(2) – T(1)= 3 + 2 -1= 4

T(5)= T(4) + T(3) – T(2)= 4 + 3 – 2= 5

T(6)= T(5) + T(4) – T(3)= 5 + 4 -3= 6

By induction, we see that T(n)= n. Hence, order is n.

Q.8 What should be the relation between T(1), T(2) and T(3), so that Q.7, gives an algorithm whose order is constant?

(a) T(1)= T(2) = T(3)
(b) T(1) + T(3)= T(2)
(c) T(1) – T(3)= T(2)
(d) T(1) + T(2)= T(3)

Solution: Option (a)

Let T(1)= T(2)= T(3)= k(say)
Then, T(4)= k + k - k= k
T(5)= k + k –k= k
So, T(n)= k, a constant can be proved by induction.

Q.9 The order of a binary search algorithm is—

(a) n
(b) n^2
(c) nlog(n)
(d) log(n)

Solution: Option (d)

Q.10 The recurrence relation that arises in relation with the complexity of binary search, 'O'—

(a) $T(n)= T(n/2) + k$, where k is a constant
(b) $T(n)= 2T(n/2) + k$, where k is a constant
(c) $T(n)= T(n/2) + \log(n)$
(d) $T(n)= T(n/2) + n$

Solution: Option (a)

Q.11 Consider the following 2 functions:

$f(n)= n^3$, if $0 \leq n < 10{,}000$
$\quad = n^2$, otherwise
$g(n)= n$, if $0 \leq n < 100$
$\quad = n^2 + 5n$, otherwise

Which of the following option is correct?
(a) f(n) is $O(n^3)$
(b) g(n) is $O(n^3)$
(c) O(f(n)) is same as O(g(n))
(d) g(n) is O(1)

Solution: Option (c)

Q.12 Let T(n) be the function defined by—

$T(1)= 1$, if n=1
$$= 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \sqrt{n}, \text{ for } n \geq 2$$

Which of the following is true?
(a) $T(n)= O(\sqrt{n})$
(b) $T(n)= O(n)$
(c) $T(n)= O(\log n)$
(d) None of the above

Solution: Option (b)

Apply Master's theorem

Q.13 In the following function, let $n \geq m$.

int gcd(n, m)
{   if(n%m_2=0) return m;
n= n%m;

```
return gcd(m, n);
}
```

How many recursive calls are made by this function?

(a) $\Theta(\log_2 n)$ (b) $\Omega(n)$
(c) $\Theta(\log_2 \log_2 n)$ (d) $\Theta(\sqrt{n})$

Solution: Option (a)

The best method is taking values of n and m and counting the no. of comparisons and matching.

Q.14 What is the time complexity of the following recursive function?

```
int Dosomething (int n) {
if(n≤2)
return 1;
else
return (Dosomething (floor(sqrt(n))) + n);
}
```

(a) $\Theta(n^2)$ (b) $\Theta(n\log_2 n)$
(c) $\Theta(\log_2 n)$ (d) $\Theta(\log_2 \log_2 n)$

Solution: Option (d)

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

We continue until this cannot be continued any more. This happens when the power of n evaluates to 2(or less). Let it happen after k steps. after k steps, the power of n(in the first term) on the right hand side will be, ½ x.

We stop when—
$n^{1/2x}= 2$

Taking log on both sides—
$½ x\log_2 n, \log_2 2$

Taking log on both sides—
$\log_2 \log_2 n= k$

Q.15 An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers needs to be determined. Which of the following is TRUE about the no. of comparisons needed?

(a) Atleast 2n-C comparisons are needed
(b) Atmost 1.5n-2 comparisons are needed
(c) Atleast $n\log_2 n$ comparisons are needed
(d) None of the above

Solution: Option (b)

This can be achieved by comparing pairwise numbers.

Q.16 Consider the following C code segment:

```
int IsPrime (n)
{
   int i, n;
   for (i=2; I ≤ √n; i++)
   {
       if(n% i= =0)
       {
           printf("Not prime\n");
           return 0;
        }
       return 1;
    }
}
```

Let T(n) denote the no. of times the 'for' loop is executed by the program on input n. Which of the following is TRUE?

(a) $T(n)= O(\sqrt{n})$ and $T(n)= \Omega(\sqrt{n})$
(b) $T(n)= O(\sqrt{n})$ and $T(n)= \Omega(1)$
(c) $T(n)= O(n)$ and $T(n)= \Omega(\sqrt{n})$
(d) None of the above

Solution: Option (b)

The worst case is when 'n' is prime and the best case is when 'n' is even.

Q.17 The minimum no. of comparisons required to determine if an integer appears more than n/2 times in a sorted array of n integers.

(a) $\Theta(n)$
(b) $\Theta(\log n)$
(c) $\Theta(\log \log n)$
(d) $\Theta(1)$

Solution: Option (d)

If we just check index at n/2 +1 then if the no. is present there, it appears more than n/2 times.

The next two questions(Q.18 & Q.19) are based on the following:

```
int f1 (int n)
{
  if (n= =0 || n= =1)
      return n;
   else
       return (2*f1 (n-1) + 3*f1 (n-2));
}
int f2 (int n)
{
   int i;
   int X[N], Y[N], Z[N];
   X[0]=Y[0]=Z[0]=0;
  X[1]=1; Y[1]=2; Z[1]=3;
  for (i=2, i≤ n, i++) {
      X[i]= Y[i-1] + Z[i-2];
      Y[i]= 2 * X[i];
      Z[i] = 3 * X[i];
  }
 return X[n];
```

Q.18 The running time of f1(n) and f2(n) are—

(a) $\Theta(n)$ and $\Theta(n)$
(b) $\Theta(2^n)$ and $\Theta(n)$
(c) $\Theta(n)$ and $\Theta(2^n)$
(d) $\Theta(2^n)$ and $\Theta(2^n)$

Solution: Option (b)

The recursive and iterative approach of the same algorithm.

Q.19 f1(8) and f2(8) returns the value of—

(a) 1661 and 1640
(b) 59 and 59
(c) 1640 and 1640
(d) 1640 and 1641

Solution: Option (c)

Q.20 The running time of the algorithm is represented by following recurrence relation:

$$T(n)=\begin{cases} n, & \text{if } n\leq 3 \\ T(n/3) + Cn, & \text{otherwise} \end{cases}$$

Which is the time complexity?

(a) $\Theta(n)$
(b) $\Theta(n\log n)$
(c) $\Theta(n^2)$
(d) $\Theta(n^2\log n)$

Solution: Option (a)

Apply Master's theorem.