|  |  |  |  |
| --- | --- | --- | --- |
| **UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, DEHRADUN** | | | |
| Assignment #1, September 2019Programme Name: B.Tech(CSE- Big Data) Semester : IIICourse Name : Design & Analysis of AlgorithmsCourse Code : CSEG2003 Max. Marks :Nos. of page(s) : 03 **Instructions :** **Answer the following questions** | | | |
|  | | | |
| S. No. |  | **Marks** | **CO** |
| **Q1** | Solve the following recurrence relations:   1. T(n) = 2T() + log n 2. T(n)=3T(n/2) + n2 3. T(n)=16T(n/4) + n 4. T(n)=8T(n/4) - n2 5. T(n)=2T(n/2) + n log n 6. T(n)=T() + 1 7. T(n)=2T(n/2) + 8. T(n)= 0.5 T(n/2) + 1/n 9. T(n) = 2T() + 1 10. T(n) = 2T(n/3)+n 11. T(n) = 12. T(n) = 13. T (n)=3T(n/4) +n | **12** | **CO2** |
| **Q2** | Obtain the asymptotic upper bound using recursion tree for:   1. T(n)= 2T(n/2) + cn 2. T(n)= 3T(n/4) + cn2 3. T(n)= T(n/3) + T(2n/3) + cn 4. T(n)= 2T(n/2) + n2 5. T(n)=2T(n−1)+1 6. *T*(*n*)=4*T*(⌊*n*/2⌋)+*cn* | **06** | **CO2** |
| **Q3** | Find the time complexity of the below recurrence | **04** | **CO2** |
| **Q4** | Find big oh (O) notations, big omega and theta notations for the following function: | **06** | **CO2** |
| **Q5** | Compute Big-Oh for following code:  **1.** for(i=0;i<n;i++)  for(j=i;j<n;j++) for(k=j;k<n;k++)  A++;   1. void function(int n)   {  int i, j, k, count = 0; for(i=n/2; i<=n ; i++)  for(j=1; j+n/2 <= n; j=j++)  for(k=1; k<=n; k=k\*2)  count++;  }   1. void function(int n)   {  int i, j, k, count = 0; for(i=n/2; i<=n ; i++)  for(j=1; j <= n; j=j\*2)  for(k=1; k<=n; k=k\*2)  count++;  }   1. void function(int n)   {  If(n ==1) return n; int i, j, k, count = 0; for(i=1; i<=n ; i++)  { for(j=1; j <= n; j=j++)  {  Printf(“\*”); Break;  }  }  }   1. void function(int n)   { int i = 1, s =1; while(s<=n)  {  i++;  s = s + 1; printf(“\*”);  }  }   1. void function(int n)   { int i = 1, count =0; for(i=1; i\*i <= n ; i++)  count++;  } | **12** | **CO2** |
| **Q6.** | **Setup a recurrence relation for the algorithm basic operation count and solve:**   1. You have 81 balls. 80 balls have the same weight. 1 ball is the lightest one. Design a divide and conquer algorithm to find the lightest ball. 2. Given a sorted array in which all elements appear twice (one after one) and one element appears only once. Find that element in O(log n) complexity. 3. Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write efficient functions to find floor of x. 4. A sorted array is rotated at some unknown point, find the minimum element in it. 5. Given an integer x, find square root of it. If x is not a perfect square, then return floor(√x). | **15** | **CO3**  **, CO4** |
| **Q10.** | Explain Fibonacci Number generation problem with an example. | **10** | **CO4** |