1.(a) Draw the Binary Tree for the following array using the strategy defined below:

(i) root is at A[1]

(ii) left child of node at A[i] is at A[2i]

(iii) left child of node at A[i] is at A[2i+1]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 10 | 21 | 5 | 18 | 8 | 11 | 9 | 2 | 33 |

1.(b) Write down the pseudo-code for Max-Heapify(A,i) operation and then run Max-Heapify(A,2) for the initial tree you obtained in question 1(a).

2. Analyse the time complexity of Quick-Sort for 2:8 proportional split. Note that you need to write down the recurrence relation of it and solve that relation using recurrence tree method. While solving the problem you need to answer the following queries.

(i) Draw the Recurrence Tree.

(ii) State the length of shortest and largest height, minimum and maximum no. of levels of the tree in terms of input size ***n***. Mention these terms with respect to the Array A [given in 1(a)]

3.(a) Describe running time of following Code Segments w.r.t to Best Case, Worst Case.

Search\_and\_Sum (key, A, n) ►A[1 . . n]{

intfun(intkey,int[] A)

{

int sum = 0;

**for**(**int**i=0;i<=m+m+1000;i++)

{

**for**(**int** j=A.length; j>=1;j/=2)

{

**if**(key == A[j])

break;

**else**

sum=+j;

}

}

return sum;

}

(b) Order the following terms in ascending order.

O(1),O(100),O(nlogn),O(logn),O(n!),O(n3)