**What is the difference between Divide and conquer?**

“Divide-and-Conquer” vs “Decrease-and-Conquer”: According to this definition, Merge Sort and Quick Sort comes under divide and conquer (because there are 2 sub-problems) and Binary Search comes under decrease and conquer (because there is one sub-problem).

The main difference between divide and conquer and dynamic programming is that the divide and conquer combines the solutions of the sub-problems to obtain the solution of the main problem while dynamic programming uses the result of the sub-problems to find the optimum solution of the main problem.

**Exercises 5.1**

1.

a) Call Algorithm MaxIndex (A[0..n − 1]) where Algorithm MaxIndex(A[l..r])

//Input: A portion of array A[0..n − 1] between indices l and r (l ≤ r)

//Output: The index of the largest element in A[l..r]

if l = r return l

else temp1 ← MaxIndex (A[l..(l + r)/2])

temp2 ← MaxIndex(A[(l + r)/2 + 1..r])

if A[temp1] ≥ A[temp2]

return temp1

else return temp2

b) This algorithm returns the index of the leftmost largest element.

c) The recurrence for the number of element comparisons is

C(n) = C(n/2) + C(n/2)+1 for n > 1, C(1) = 0.

Solving it by backward substitutions for n = 2k yields the following:

C(2k)=2C(2k−1)+1

= 2[2C(2k−2) + 1] + 1 = 22C(2k−2)+2+1

= 22[2C(2k−3) + 1] + 2 + 1 = 23C(2k−3)+22 +2+1

= ...

= 2i

C(2k−i)+2i−1 + 2i−2 + ... + 1

= ...

= 2kC(2k−k)+2k−1 + 2k−2 + ... +1=2k − 1 = n − 1.

We can verify that C(n) = n − 1 satisfies, in fact, the recurrence for every value of n > 1 by substituting it into the recurrence equation and considering separately the even (n = 2i) and odd (n = 2i + 1) cases. Let n = 2i, where i > 0. Then the left-hand side of the recurrence equation is n − 1=2i − 1. The right-hand side is C(n/2) + C(n/2)+1 = C(2i/2) + C(2i/2)+1 = 2C(i) + 1 = 2(i − 1) + 1 = 2i − 1, which is the same as the left-hand side.

Let n = 2i + 1, where i > 0. Then the left-hand side of the recurrence equation is n − 1=2i. The right-hand side is C(n/2) + C(n/2)+1 = C((2i + 1)/2) + C((2i + 1)/2)+1

= C(i + 1) + C(i)+1=(i + 1 − 1) + (i − 1) + 1 = 2i

6.

0 1 2 3 4 5 6

**E X A M P L E**

0 1 2 3 4 5 6

**E X A M P L E**

0 1 2 3 4 5 6

**E X A M P L E**

0 1 2 3 4 5

**E X A M P L**

**E X A M L P E**

**A E M X E L P**

**A E E L M P X**