Merge Sort:

Divide and Conquer Method:

1.) _____ the problem in smaller problems

2.) _____ these problems

3.) _____ the answers

How does it relate to Merge Sort?

Merge Sort:

1.) _____ the problems into 2 halves

2.) _____ each half

3.) _____ the sorted halves

Ex:

Array: _____

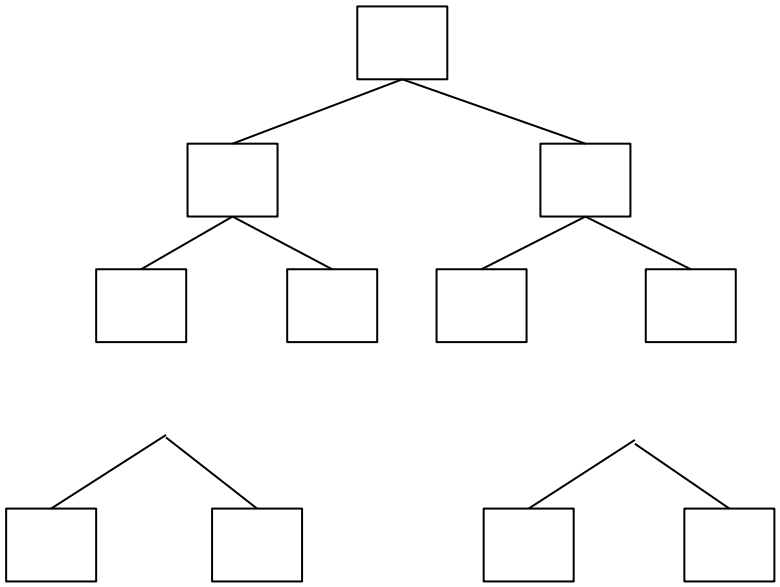
Iterations:									
1									
2									
3									
4									
5									
6									
7									
8									

Code:

TC:

-
-
-

Recurrence Formula and Recursion Tree:



Level	Arg/Problem Size	Nodes per Level	1 Node TC	Level TC

SC:

Stable? _____ Adaptive? _____

Usable Data Types:

Variations:

```
int binary_search(int A[], int left, int right, int v){
   int m = left+(right-left)/2;
   if (left > right) return -1;
   if (v == A[m]) return m;
   if (v < A[m])
      return binary_search(A, left, m-1, v);
   else
      return binary_search(A, m+1, right, v);
}


Merge_sort(A, le, r)  //N = ri-le+1
   if (le>=ri) return
   else
      m = floor(le+(ri-le)/2)
      Merge sort(A, le,  m);
      Merge_sort(A, m+1, ri);
      Merge(A, le, m, ri);

Merge(A, le, m, ri)
 1  n1=m-le+1+1   // +1 for inf
 2  n2=ri-m+1     // +1 for inf
 3  let L[n1], R[n2] be arrays
 4  for j=0 to n1-2
 5     L[j]=A[le+j]
 6  for j=0 to n2-2
 7     R[j]=A[m+1+j]
 8  L[n1] = inf
 9  R[n2] = inf
10  j=0,
11  i=0
12  for k=le to ri
13     if L[i] ≤ R[i]
14        A[k]=L[i]
15        i++
16     else
17        A[k] = R[j]
18        j++

// pseudocode
// - indentation => instruction group in {}
// - loops:
//    for k=le to ri
//    for(k=le; k<=ri; k++)
```