DAA Lab - Session 5 - Mergesort and Binary Search

**Divide-and-Conquer:** Implementation of Merge Sort algorithm.

**Divide-and-Conquer:** Implementation of Binary Search algorithm.

**Problem Definition:** Sort an array of records (a record is a structure with an “id” and a “value” field) using Merge Sort algorithm in nondecreasing order on the “value” field of the records. In the sorted array, search for a record with “value” as the key and return a record which matches the key.

**Input:** Input begins with n (1 ≤ n ≤ 220) of number of records indicating the size of the input array. The following n lines has a record per line with an 8-digit id field and a 6-digit value field separated by a space. The following line has an integer k which is the key to be searched in the sorted array.

**Output:** Print the sorted array. Each record is displayed in a single line with a space between the id and the value field. On successful search, print the index, id and value of the record. Print “NOT FOUND”, otherwise. Finally, the last line is for the execution time of sort and search put together upto 6 decimal places.

**Sample Input:**

5

1 20

2 40

3 20

4 10

5 30

30

**Sample Output:**

4 10

1 20

3 20

5 30

2 40

3 5 30

0.000000 sec.

**Sample Input:**

3

92323245 400100

12345678 201030

53413233 800909

123456

**Sample Output:**

12345678 201030

92323245 400100

53413233 800909

NOT FOUND

0.000000 sec.

**Algorithm MergeSort(A[0..n-1])**

//Sorts array A[0..n-1] by recursive Merge Sort

//Procedure Merge(A[0..n-1], m) merges two

// sorted subarrays A[0..m-1] and A[m..n-1]

// into a sorted array A[0..n-1].

**if(n ≤ 1)return**

**m = ⌊n/2⌋**

**MergeSort(A[0..m-1])**

**MergeSort(A[m..n-1])**

**Merge(A[0..n-1], m)**

**Algorithm Merge(A[0..n-1], m)**

//Merges two sorted arrays A[0..m-1] and A[m..n-1] into

//the sorted array A[0..n-1]

**i ← 0, j ← m, k ← 0**

**while(i < m and j < n) do**

**if(A[i] ≤ A[j]) B[k] ← A[i]; i ← i+1**

**else B[k] ← A[j]; j ← j+1**

**k ← k+1**

**if(j = n) Copy A[i..m-1] to B[k..n-1]**

**else Copy A[j..n-1] to B[k..n-1]**

**Copy B[0..n-1] to A[0..n-1]**

**Algorithm BinarySearch(A[l..r], k)**

**if(r-l+1 < 1)**

**return -1**

**m = ⌊(l+r) / 2⌋**

**if(k = A[m])**

**return m**

**else if(k < A[m])**

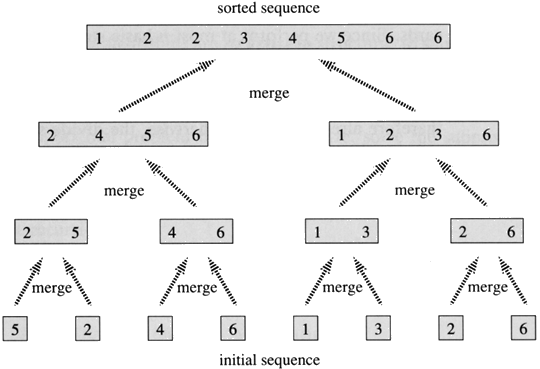
**return BinarySearch(A[l..m-1], k)**

**else**

**return BinarySearch(A[m+1..r], k)**

**Practice Problems:**

1. Implement the bottom-up way of merge sort. Here is an example of merge sort in bottom up approach.



1. Count the number of element-to-element comparisons made by the Bubble sort, Selection sort and Merge sort algorithms.