MergeSort

- Merging involves the combination of two or more ordered files into a single ordered file.
- Example: Merge the two files:
  - 503, 703 , 765
  - 087, 512, 677
- This is solved by comparing the two smallest items, output the smallest, and then repeat the process.
- Be careful when one of the two files become exhausted!

Exercise:

- Write a function which merges two sorted arrays of length n (n >0) and m (m >0) respectively.
- The function fill have the following form:
- MERGE (A,p,q,r) /*merges the sorted subarrays A[p...q] and A[q+1…r] to form a single sorted array A[p … r] */
- This Algorithm should be off complexity

Divide and Conquer

- Divide:To solve a problem, it is subdivided into sub problems each of which is similar to the given problem.
- Conquer: Each of the sub problems is solved independently
- Combine: The solutions to the sub problems are combined in order to obtain the solution to the original problem
- Divide and conquer algorithms are often implemented using recursion. However not all recursive functions are divide and conquer algorithm. Generally, the sub problems solved by divide and conquer algorithm are **non overlapping**

Towers of Hanoi 1

- Given 3 pegs A,B,C
- Peg A has n disks, starting with the largest one on the bottom and successively smaller ones on top.
- The object of the puzzle is to move the disks one at a time from peg to peg, never placing a larger one on top of a smaller one, eventually ending with all the disks on peg B
- Solution1: Imagine the pegs arranged in a triangle. On odd numbered moves, move the smallest disk one peg clockwise. On even numbered moves make the only legal move not involving the

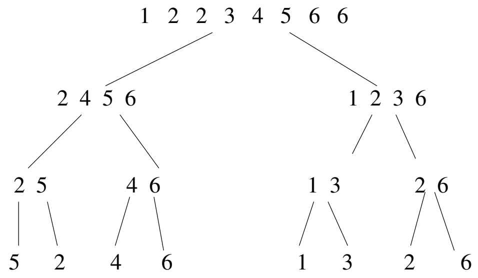
Towers of Hanoi 2

- Solution2: The divide and conquer technique.
- The problem of moving the n smallest disks from A to B can be thought of as consisting of 2 sub problems of size n-1
- First move the n-1 smallest disks from A to C exposing the nth smallest disk on peg A.
- Move that disk from A to B
- Then move the n-1 smallest disks from C to B
- The movement of the n-1 smallest disks is accomplished by a recursive application of the

Merge Sort

- **Divide:** Divide the n element sequence to be sorted into two sub sequences of n/2 elements each
- **Conquer:** Sort the two sub sequences recursively using merge sort
- **Combine:** Merge the two sorted sub sequences to produce a sorted answer
- Note: The recursion bottoms out when the length of the sequence to be sorted is 1.

Merge sort 5,2,4,6,1,3,2,6



Mergesort

- MERGESORT(A,p,r) sorts the elements in the subarray A[p..r]
- If p>=r, the subarray has at most one element and is therefore already sorted. Otherwise the divide step calculates an index q that partitions A[p…r] into two subarrays containing $\lceil n/2 \rceil$ elements and A[q+1 …r] containing $\lfloor n/2 \rfloor$ elements.
- MERGESORT(A,0,n-1) sorts the array A of

```
MERGESORT(A,p,r)
{
if p < r
  then q = ⌊ (p+r) / 2⌋
    MERGESORT(A,p,q)
    MERGESORT(A,q+1,r)
    MERGE(A,p,q,r)
}
```

When an algorithm contains a recursive call to itself its running time can be described by a recurrence equation or recurrence