Analysis of Algorithms

Divide & Conquer

Logarithms

- $\log n = \log_2 n$
- $\ln n = \log_e n$
- $\log^k n = (\log n)^k$
- $\log_b(xy) = \log_b x + \log_b y$
- $\log_b x^n = n \log_b x$
- $\log_b 1/x = -\log_b x$
- $\log_b x = \log_c x / \log_c b$
- $x = b^{\log bx}$
- $x^{\log bn} = n^{\log bx}$

Divide & Conquer

- A general paradigm for algorithm design; practiced for centuries by emperors and colonizers.
- In Divide & Conquer, we divide the problem into several independent sub problems that are similar to the original problem but smaller in size, solve the sub problems recursively, and then combine these solutions to create a solution to the original problem.

Divide & Conquer

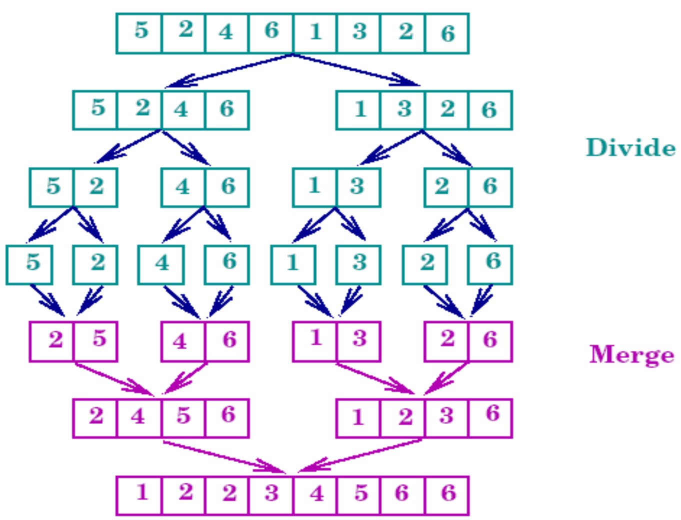
- Divide the problem into smaller problems
- Conquer by solving these problems.
- Combine these results together to solve the original problem.
- Familiar Examples: Binary Search, Merge Sort, Quicksort etc.
- Other Examples, Strasssens matrix, multiplication, Convex Hulls.

Merge Sort

```
MergeSort(A, p, r) {
  if (p < r) {
      q = ë(p + r) / 2û;
      MergeSort(A, p, q);
      MergeSort(A, q+1, r);
      Merge(A, p, q, r);
  }
}

// Merge() takes two sorted subarrays of A and
// merges them into a single sorted subarray of A
//     (how long should this take?)
```


Merge Sort: Illustration



Merge

```
Merge(A,p,q,r)
   n1 = q - p + 1
   n2 = r - q
   create arrays L[1..n1+1] and R[1..n2+1]
   for i = 1 to n1
        L[i] = A[p + i - 1]
   for j = 1 to n2
        R[j] = A[q + j]
   L[n1 + 1] = ∞
   R[n2 + 1] = ∞
   i = 1,        j = 1
   for k = p to r
        if L[i] <= R[j]
             A[k] = L[i]
             i = i + 1
        else
             A[k] = R[j]
             j = j + 1
```