Algorithm Design
Divide and Conquer

Solaiman Hossain
Sheikh Hasina Institute of ICT
Department of Computer Science and Engineering
ID:18ICTCSE020

Introduction

Divide and Conquer is an algorithmic pattern. In algorithmic methods, the design is to take a dispute on a huge input, break the input into minor pieces, decide the problem on each of the small pieces, and then merge the piecewise solutions into a global solution. This mechanism of solving the problem is called the Divide and Conquer Strategy.

Divide and Conquer algorithm consists of a dispute using the following three steps.

1. Divide: the original problem into a set of subproblems.

2. Conquer: Solve every subproblem individually, recursively.

3. Combine: Put together the solutions of the sub problems to get the solution to the whole problem.

Examples: The specific computer algorithms are based on the Divide Conquer approach:

1. Maximum and Minimum Problem
2. Binary Search
3. Sorting (merge sort, quick sort)
4. Tower of Hanoi.

01.Fundamental of Divide and Conquer Stray

There are two fundamental of Divide and Conquer Strategy:

1. Relational Formula
2. Stopping Condition

1. **Relational Formula:** It is the formula that we generate from the given technique. After generation of Formula we apply D and C Strategy, i.e. we break the problem recursively and solve the broken subproblems.

2. **Stopping Condition:** When we break the problem using Divide Conquer Strategy,then we need to know that for how much time, we need to apply divide and Conquer.So the condition where the need to stop our recursion steps of D and C is called as Stopping Condition.

Max - Min Problem

Problem: Analyze the algorithm to find the maximum and minimum element from an array.

Algorithm: Max ?Min Element (a [])
Max: a [i]
Min: a [i]
For i= 2 to n do
If a[i]¿ max then
max = a[i]
if a[i] ¡ min then
min: a[i] return (max, min)

Analysis

Method 1: if we apply the general approach to the array of size n, the number of comparisons required are 2n-2.

Method 2: In another approach, we will divide the problem into sub-problems and find the max and min of each group, now max. Of each group will compare with the only max of another group and min with min.

next prev Max - Min Problem Problem: Analyze the algorithm to find the maximum and minimum element from an array.

Algorithm: Max ?Min Element (a []) Max: a [i] Min: a [i] For i= 2 to n do If a[i]¿ max then max = a[i] if a[i] ¡ min then min: a[i] return (max, min) Analysis: Method 1: if we apply the general approach to the array of size n, the number of comparisons required are 2n-2.

Method-2: In another approach, we will divide the problem into sub-problems and find the max and min of each group, now max. Of each group will compare with the only max of another group and min with min.

Let n = is the size of items in an array

Let T (n) = time required to apply the algorithm on an array of size n. Here we

divide the terms as T(n/2).

2 here tends to the comparison of the minimum with minimum and maximum with maximum as in above example.

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + 2$
$T(n) = 2T(\frac{n}{2})$

$T(2) = 1$, time required to compare two elements/items. (Time is measured in units of the number of comparisons)

$T(n/2) = 2T(\frac{n}{2^2}) + 2$

Put eq (ii) in eq (i)

$T(n) = 2[2T(\frac{n}{2^2} + 2] + 2$

$=2^2T(\frac{n}{2^2} + 2^2 + 2$

Similarly, apply the same procedure recur-

sively on each subproblem or anatomy

{Use recursion means, we will use some stopping condition to stop the algorithm}

$T(n) = 2^i_T(\frac{n}{2^i}) + 2^i - 1 + 2..... + 2$

Recursion will stop, when $\frac{n}{2^i} = 2$
$\Rightarrow n = 2^(i + 1) \rightarrow (Eq.4)$

Put the equ.4 into equation 3.

$T(n) = 2^iT(2)+2^i+2^(i-1)+.....+2$
$= 2^i.1 + 2^i + 2^(i - 1) + ... + 2$
$= 2^i + \frac{2(2^i+1}{2-1}$
$= 2^(i + 1) + 2^i - 2$
$= n + \frac{n}{2} - 2$
$= \frac{3n}{2} - 2$

Number of comparisons requires applying the divide and conquering algorithm on n elements/items $= \frac{3n}{2} - 2$

Number of comparisons requires applying general approach on n elements
$= (n-1) + (n-2) = 2n - 2$

From this example, we can analyze, that how to reduce the number of comparisons by using this technique.

Analysis: suppose we have the array of size 8 elements.
Method 1: requires $(2n-2), (2*8)-2 = 14$
Method 2: requires $\frac{3*8}{2} - 2 = 10$

It is evident; we can reduce the number of comparisons (complexity) by using a proper technique.

Binary Search

1. In Binary Search technique, we search an element in a sorted array by recursively dividing the interval in half.

2. Firstly, we take the whole array as an interval.

3. If the Pivot Element (the item to be searched) is less than the item in the middle of the interval, We discard the second half of the list and recursively repeat the process for the first half of the list by calculating the new middle and last element.

4. If the Pivot Element (the item to be searched) is greater than the item in the middle of the interval, we discard the first half of the list and work recursively on the second half by calculating the new beginning and middle element.

5. Repeatedly, check until the value is found or interval is empty

Analysis

1.Input : an array A of size n, already sorted in the ascending or descending order.

2.Output : analyze to search an element item in the sorted array of size n.
3.Logic : Let T (n) = number of comparisons of an item with n elements in a sorted array.

Set BEG = 1 and END = n
Find mid=int ($\frac{beg+end}{2}$)