DIVIDE AND CONQUER

CS112.O11.KHTN

Team 10

TABLE OF CONTENTS

TABLE OF CONTENTS

01

PROBLEM

CEO

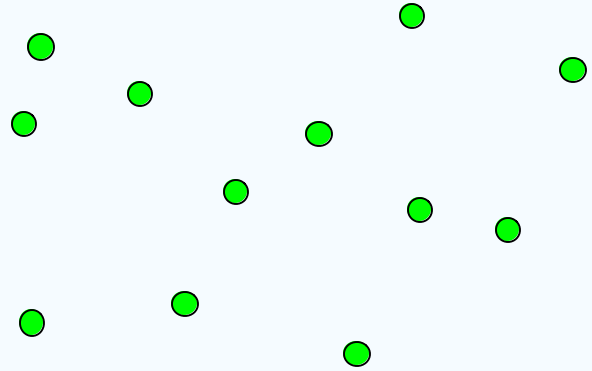
Suppose Nam is the CEO of a large company who wants to evaluate the performance of all employees in the company. The company has thousands of employees, so how will you evaluate the performance of all employees?

Finding a closest pair of points

Given a set of points {p1, . . . , pn} find the pair of points {pi , pj } that are closest together.

With the distance calculated using the formula

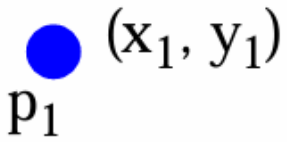
$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

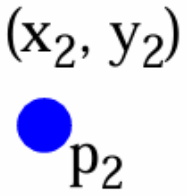


BRUTEFORCE

Check every pair of points

```
// A Brute Force method to return the
// smallest distance between two points
// in P[] of size n
float bruteForce(Point P[], int n)
{
    float min = FLT_MAX;
    for (int i = 0; i < n; ++i)
        for (int j = i+1; j < n; ++j)
            if (dist(P[i], P[j]) < min)
                min = dist(P[i], P[j]);
    return min;
}
```





$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Time Complexity: $O(N^2)$

Can we do it faster?

02

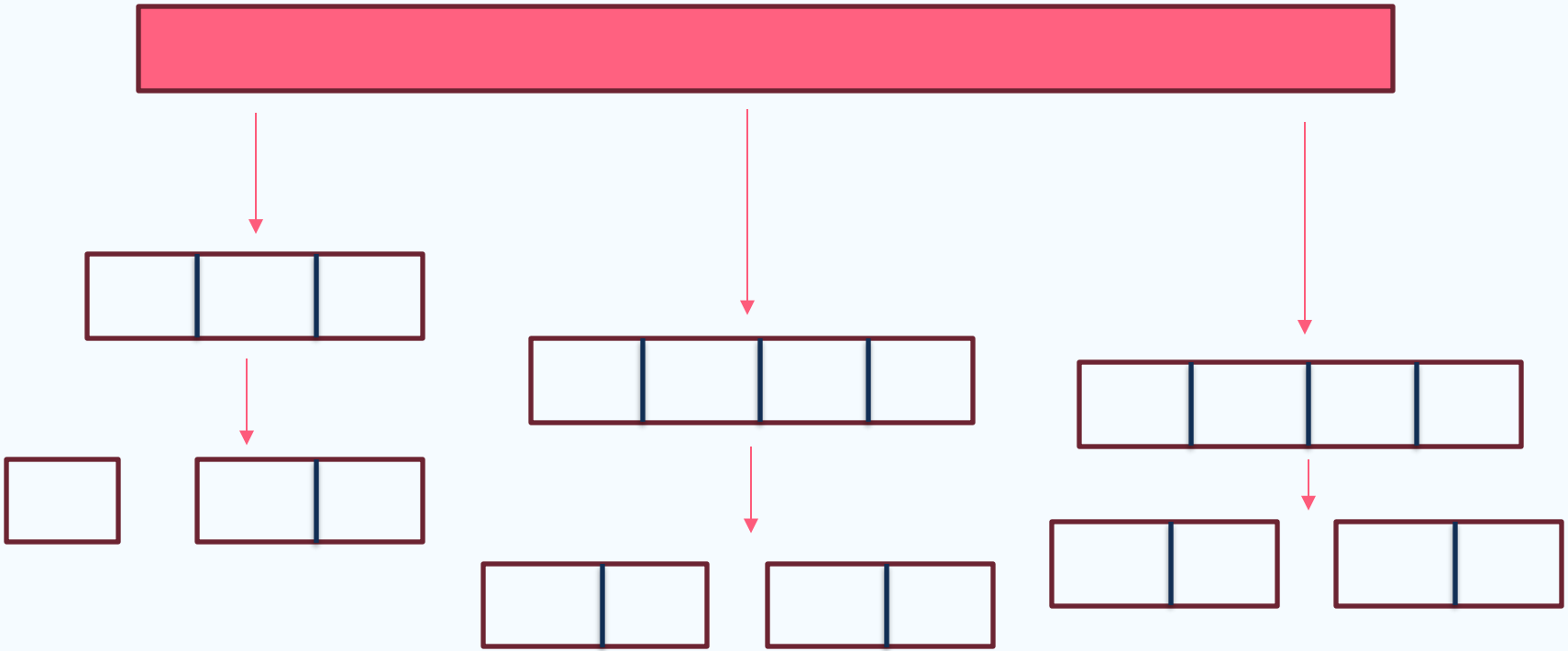
GENERAL IDEA

DIVIDE AND CONQUER

DIVIDE AND CONQUER

1. **Divide:** This involves dividing the problem into smaller sub-problems.
2. **Conquer:** Solve sub-problems by calling recursively until solved.
3. **Combine:** Combine the sub-problems to get the final solution of the whole problem.

DIVIDE AND CONQUER



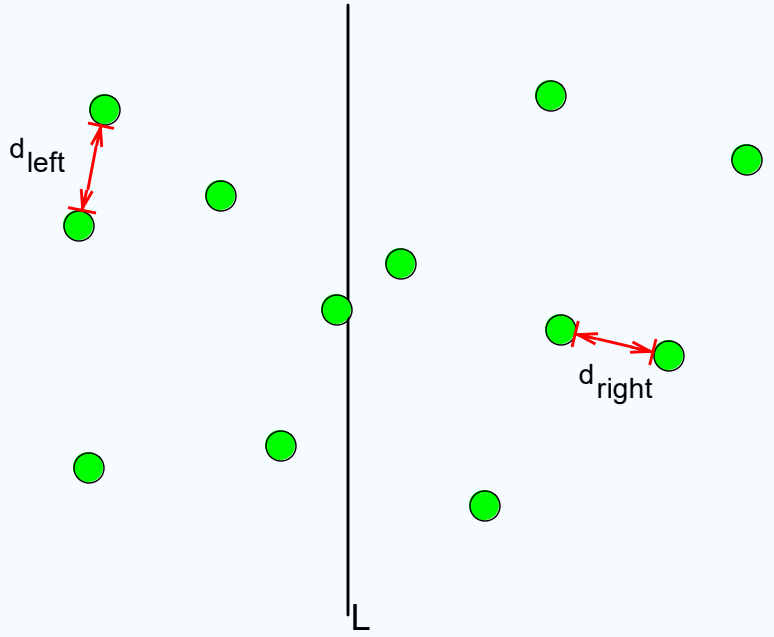
Finding a closest pair of points

Apply the idea of "**divide and conquer**":

1. Divide N points to be processed into 2 sets with a vertical line
2. Call recursively to find the closest pair of points for the left set and the right set
3. Find the shortest distance between a point in the left set and a point in the right set.

Divide and Conquer

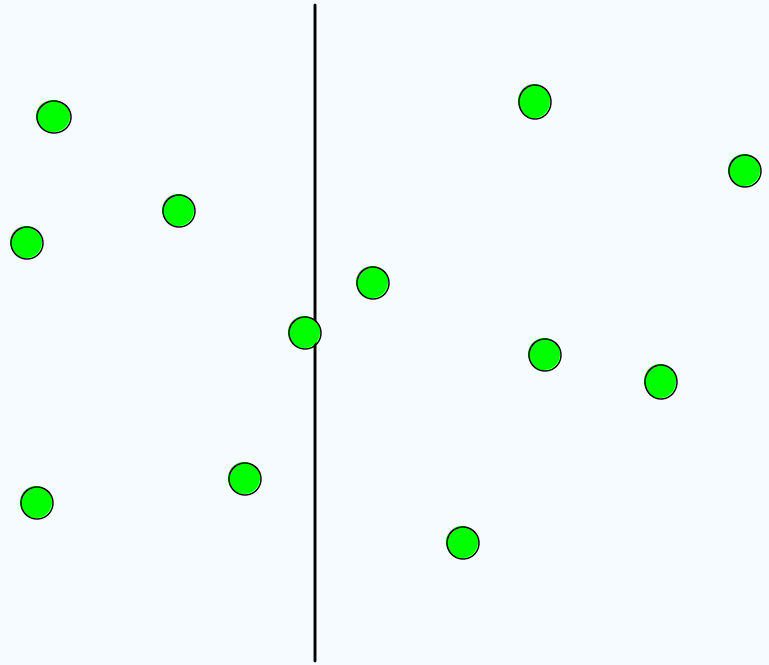
1. Split the points with line L so that half the points are on each side.
2. Recursively find the pair of points closest in each half.



Combine: the hard case

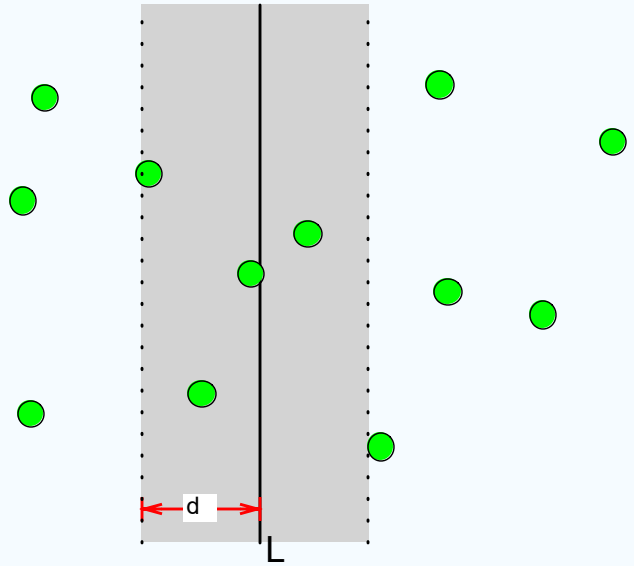
Let $d = \min\{d_{left}, d_{right}\}$.

d would be the answer, except maybe L split a close pair!



Region Near L

If there is a pair {pi , pj } with dist(pi , pj) < d that is split by the line, then both pi and pj must be within distance d of L.



Pseudo code

```
DAC(a, i, j)
{
   if(small(a, i, j))
     return(Solution(a, i, j))
   else
     mid = divide(a, i, j)          // f1(n)
     b = DAC(a, i, mid)             // T(n/2)
     c = DAC(a, mid+1, j)           // T(n/2)
     d = combine(b, c)              // f2(n)
  return(d)
}
```

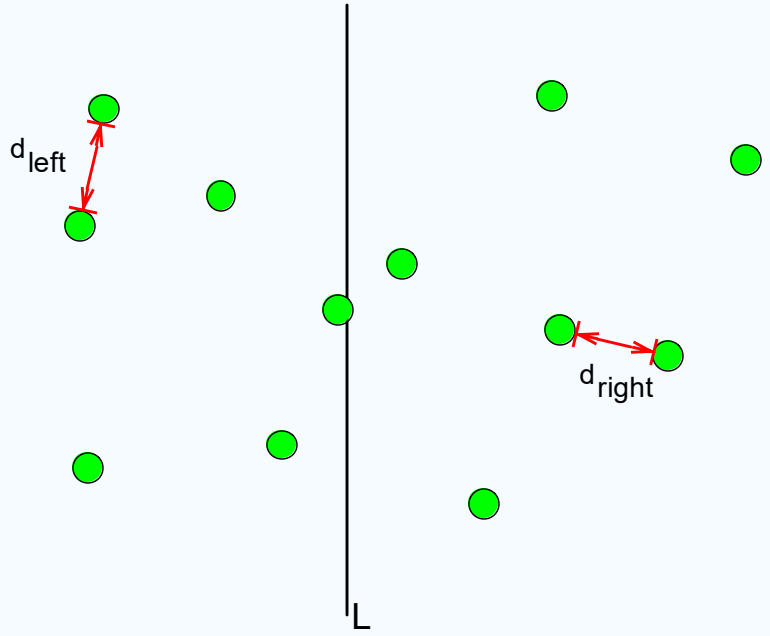
IDENTITY

1. **Divide**: Can the problem be divided into smaller sub-problems of the same type?
2. **Conquer**: Can the sub-problems be directly solved when they are small enough?
3. **Combine**: Can the solutions of the sub-problems be combined to form a solution for the original problem?

Examples: binary search, quick-sort…

Finding a closest pair of points

1. Divide
2. Conquer
3. Combine



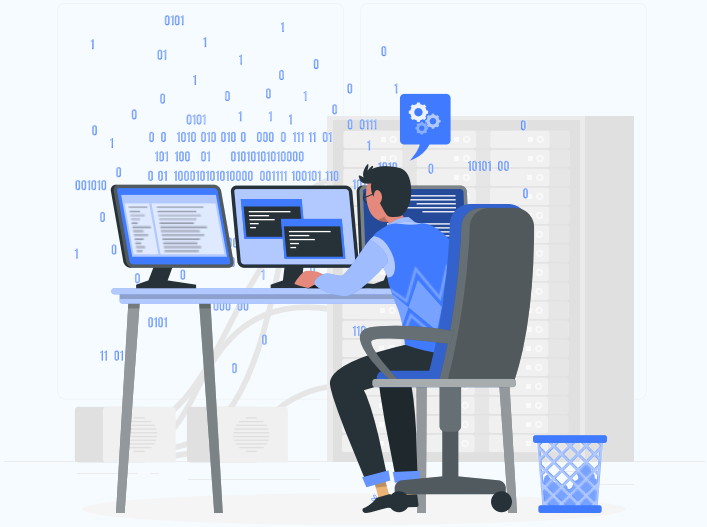
Variation of DAC

1. Dynamic Programming

Divide large problems into overlapping sub-problems and optimal sub-structures.
These problems often have many correct solutions and each solution has 1 evaluation value

Variation of DAC

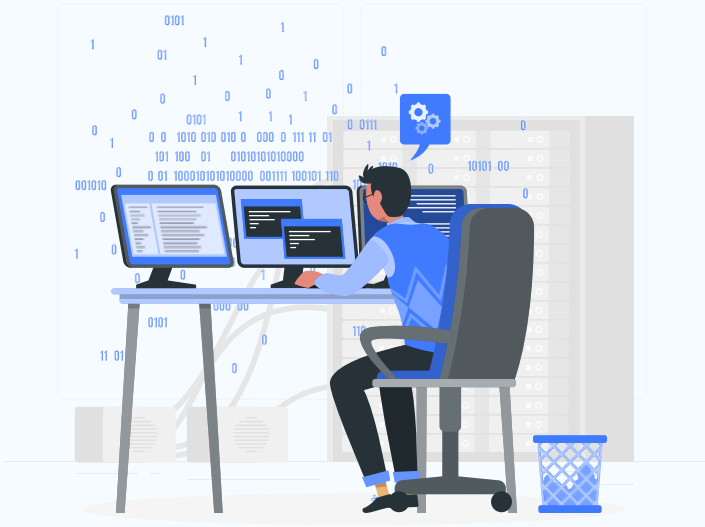
2. Decrease and conquer



A technique used to solve problems by reducing the size of the input data at each step of the solution process
Ex: binary search,..

Variation of DAC

3. Transform and conquer



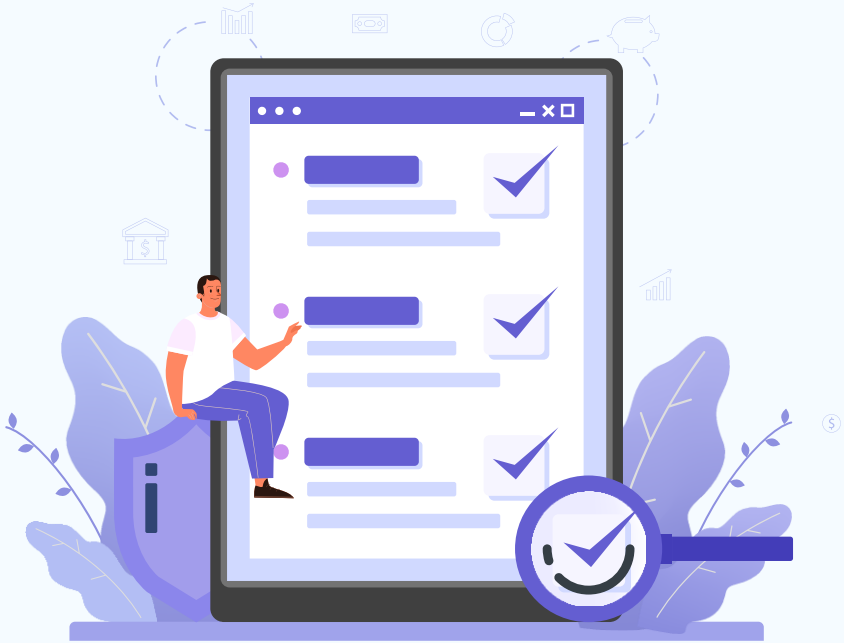
The problem-solving strategy of "Transform and Conquer" involves dividing the problem into smaller parts and transforming the data or structure of the original problem into a simpler form to solve it more effectively.

03

WHY?

Advantages

- Reusable Solutions
- Good Time Performance
- Optimization Support
- Versatility



Disadvantages

- Slow recursion
- Complicates sometimes
- Difficult to implement
- Only suitable for certain problems

Compare

“Divide and Conquer” and “Brute Force”

Feature	Divide and Conquer	Brute Force
Efficiency	Typically more efficient for large or complex problems	Typically less efficient for large or complex problems
Approach	Breaks a problem down into smaller subproblems	Tries every possible solution
Complexity	Often has a logarithmic time complexity (O(log n) or On(log n))	Often has a linear time complexity (O(n) or O(n^2))
Examples	Binary search, merge sort, quick sort	Bubble sort, insertion sort, selection sort

04

COMPLEXITY

The recurrence

$$T(n) = aT(n/b) + f(n)$$

where n = size of the problem
a = number of subproblems in the recursion and a >= 1
n/b = size of each subproblem
f(n) = cost of work done outside the recursive calls like dividing into subproblems and cost of combining them to get the solution.

Master theorem

$$T(n) = \begin{cases} O(n^d \log(n)) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b(a)}) & \text{if } a > b^d \end{cases}$$

Finding a closest pair of points

Classic divide and conquer recurrence:

$$T(n) = 2T(n/2) + O(n)$$

Apply Master Theorem, we have complexity is:

$$O(nlogn)$$

05

EXAMPLE

Application

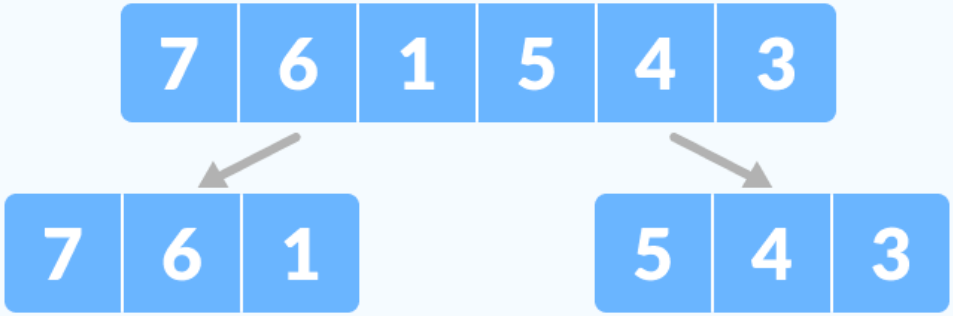
Sorting (Merge Sort, Quick Sort)
Large Integer Multiplication - Karatsuba Algorithm
Matrix Multiplication - Strassen’s algorithm
Finding a closest pair of points

Merge Sort

1. Let the given array be:

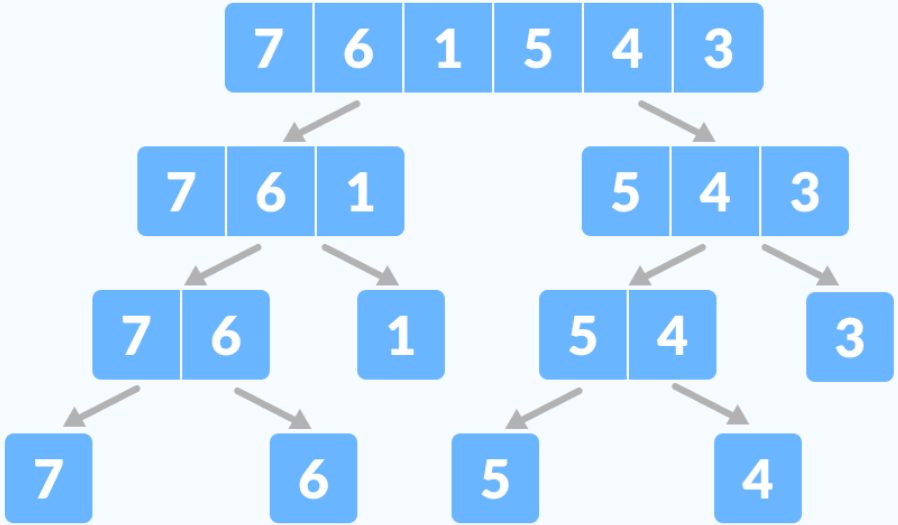
Merge Sort

2. **Divide** the array into two halves.



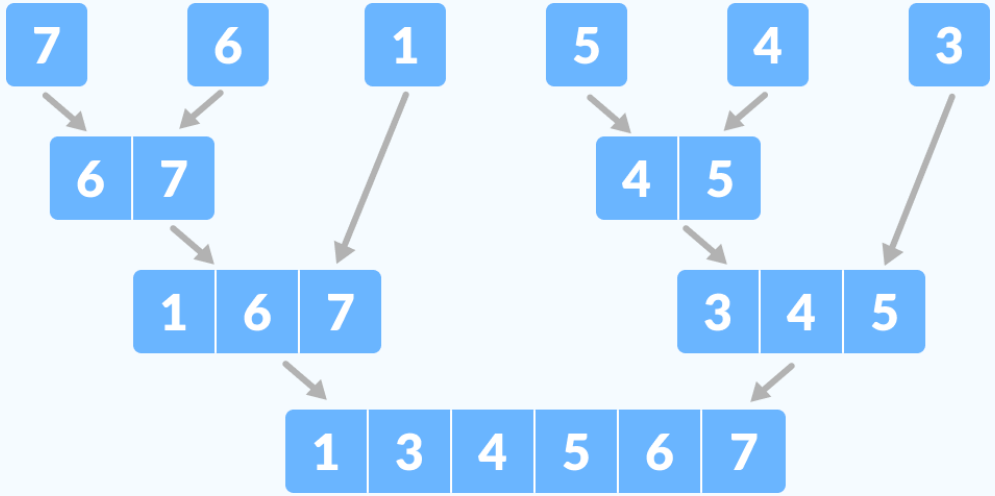
Merge Sort

Again, divide each subpart recursively into two halves until you get individual elements.



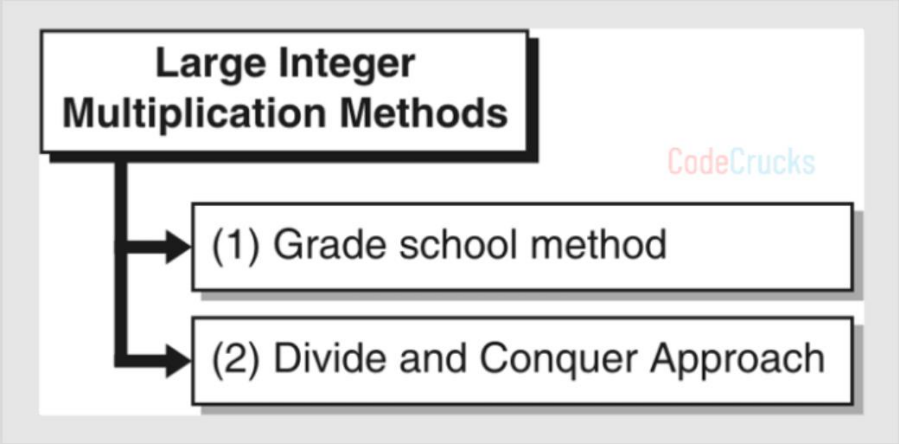
Merge Sort

3. Now, combine the individual elements in a sorted manner. Here, **conquer** and **combine** steps go side by side.

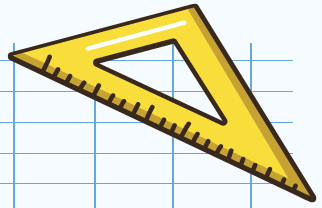


Large Integer Multiplication

- Common procedure in computer-assisted problem solving.
- Multiplying big numbers is not only difficult, but also time-consuming and error-prone.



Traditional Multiplication Method.



American	British
456 × 1234	456 × 1234
1824	456
+ 1368	+ 0912
+ 0912	+ 1368
0456	+ 1824
562704	562704

Traditional Multiplication Method.

- time-consuming
- inefficient.

Complexity: O(mn)

Divide and Conquer Approach

Suppose we have to multiply 2 integers of size 2n decimal places
By dividing a and b we can write a and b in the form

$$a = a_1.10^n + a_2,$$
$$b = b_1.10^n + b_2$$

in there
a_1, a_2, b_1, b_2 are n-digit numbers
Apply to the Karatsuba formula:
$ab = a_1b_1 * 10^{2n} + [(a_1 + a_2)(b_1 + b_2) - a_1b_1 - a_2b_2] * 10^n + a_2b_2$

Divide and Conquer Approach

For example: a= 5678 and b= 6789.

a.b?

- Time-consuming
- Inefficient

Divide and Conquer Approach

For example: a= 5678 and b= 6789.

1. Our algorithm separates the terms into $a_1 = 56$, $a_2 = 78$, $b_1 = 67$ and $b_2 = 89$.

2. Three halves multiplications that need to be performed are:

p = $a_1b_1 = 56 \times 67 = 3752$

q = $a_2b_2 = 78 \times 89 = 6942$

and

r = $(a_1 + a_2) \times (b_1 + b_2) = 134 \times 156 = 20904$

Divide and Conquer Approach

For example: a= 5678 and b= 6789.

Therefore

ab = 3752*10000 + (20904 - 3752 - 6942)* 100 + 6942

= 3752 0000 + 10210 00+ 6942

= 38547942

We have:

$a \times b = (a_1 10^n + a_2)(b_1 10^n + b_2)$

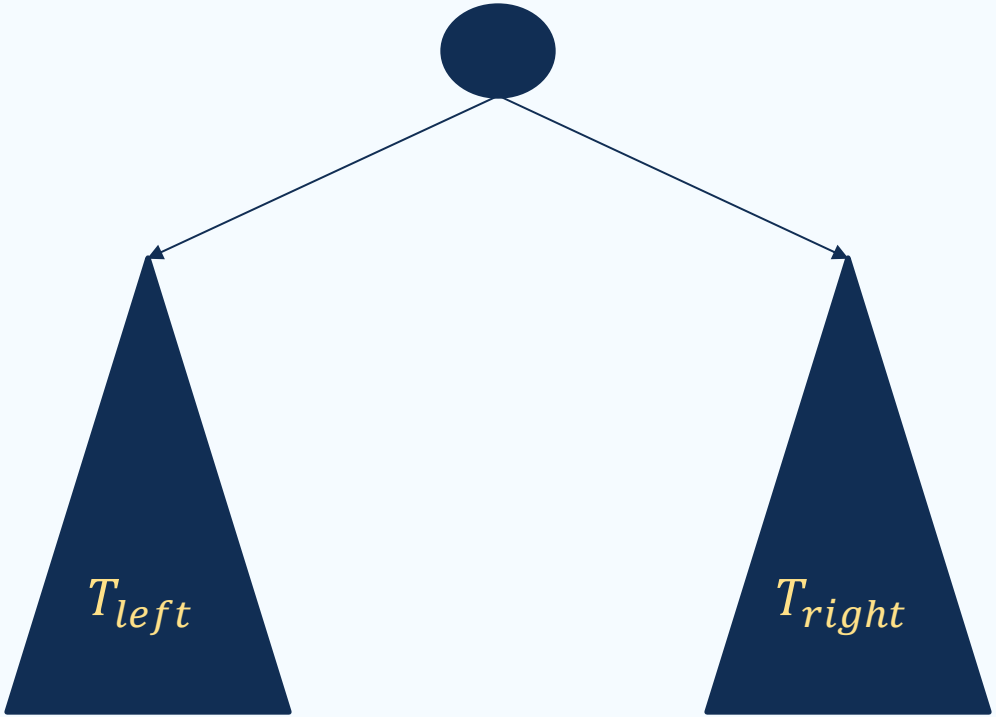
$= a_1b_1 10^{2n} + (a_1b_2 + a_2b_1) 10^n + a_2b_2$

Divide and Conquer Approach

For example: a= 5678 and b= 6789.

So we have seen a multiplication with **2n digits** about **4 multiplication** with **n digits** and **2 additions**.
Complexity: O(n^1.59)

Binary Search Tree



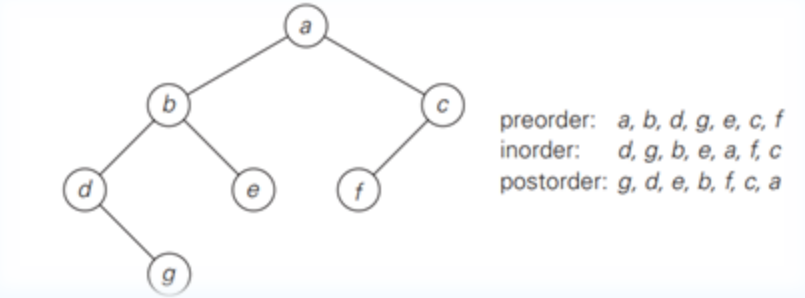
Binary Search Tree

The divide and conquer algorithm is the most important binary tree algorithm for the three basic traversals:

Preorder: often called NLR browsing

Inorder: often called LNR browsing

Postorder: often called LRN browsing



06

Q&A

QUIZ

In the divide and conquer process, breaking the problem into smaller sub-problems is the responsibility of

A. Sorting/Divide

B. Conquer/Solve

C. Merge/Combine

D. Divide/Break

QUIZ

The running time of merge sort can be reccursively represented by

A. $T(n)=2T(n/2)+O(n)$

B. $T(n)=2T(n/4)+O(n)$

C. $T(n)=3T(n/2)+O(n)$

D. $T(n)=3T(n/2)+O(n)$

Q&A

any question?

THANK YOU FOR YOUR WATCHING