॥ श्री ॥

Mani S. Palimkar
231071036

Saathi

Date ___/___/_____

TASK - 1
Inversion count.

(*) Brute force algorithm

```
NAME : InversionCount (array arr)
// I/P : An array representing the
        course code choices of
        a student.
// O/P : Number of inversions.

// We will call this function for
finding inversion count of each student
int count = 0;
    for int i = 1 to i=(len(arr)-1)
      for int j = i+1 to j = len(arr)

          if (arr[j] > arr[i])  ---> (1)
              count += 1

    return count;
```

TIME COMPLEXITY

The basic operation here is the if comparision.

~~In the j loop it is~~

To find time complexity

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \sum_{j=1}^{n-1} (n-j-1+1)$$

$$= \sum_{j=1}^{n-1} (n-j)$$

$$= (n-1) + (n-2) + (n-3) \ldots + (n-(n-1))$$

$$= (n-1) + (n-2) + (n-3) \ldots + \ldots + 1 \quad (n-n)$$

$$= n^2 - (1+2+\ldots n)$$

$$= n^2 - \frac{n(n+1)}{2} = n^2 - \frac{n^2}{2} - \frac{n}{2} = \frac{n^2-n}{2}$$

$$\neq O(n^2)$$

⊛ Divide and Conquer algorithm

NAME: InversionCount (array arr)

// I/P: An array representing the course code choice of a student.

// O/P:- No. of inversions and sorted array ~~arr~~ B

Assumption :- n is even.

⟹

Saathi

Date

i=1, j=1;
count = 0

~~for int k = 1 to len(arr) do~~
~~if~~
n = len(arr)
if n = 0 or n = 1 then
return (Arr, 0)

else
(C, leftInv) = InversionCount (first half of arr)
(D, rightInv) = InversionCount (second half of arr)

(B, splitInv) = Merge-and-CountSplitInv (C, D)

return (B, leftInv + rightInv + SplitInv).

Now, next f^n is Merge-and-countSplitInv.

NAME: Merge-and-countSplitInv ( )
// I/P: sorted arrays C and D of length n/2 each
// O/P:- sorted array B (n) and the no. of split inversions.

```
i = 1 ; j = 1 ;
splitInv = 0 ;

for K = 1 to n do
    if C[i] < D[j] then
        B[K] := C[i] , i = i + 1

else
    B[K] = D[j] , j = j + 1
    splitInv = splitInv + (n/2 - i + 1)

return (B, splitInv)
```

TIME COMPLEXITY

- First ~~each~~ the algorithm makes 2 recursive calls to sort each half of the array.
- The length of i/p is divided into half with each call.

DIVIDE :- This step just computes the middle of arr so O(1)

CONQUER : Recursively solve 2 subproblems each of size n/2.
$\Rightarrow 2T(n/2)$

COMBINE :- Merge procedure takes O(n) time.

$$T(n) = \begin{cases} \Theta(1) & , n = 1 \\ 2T(n/2) + \Theta(n) & , n > 1 \end{cases}$$

$$\downarrow$$

$$2T(n/2) + cn$$

By master theorem

$a = 2$

$b = ~~12~~ 2$

$\log_b a = \log_2 2 = 1$

$\Theta(n^{\log_b a}) = \Theta(n) = f(n)$

$\therefore$ ~~Time~~ which is 2nd case.

$\therefore$ $\boxed{\Theta(n \log n)}$

TEST CASES

For test cases, let us consider 6 courses and course codes of each of them are 1, 2, 3, 4, 5, 6.

Positive :-

1) 2, 3, 6, 4, 5, 1

Inv Count :- 7

No. of students with 0, 1, 2, 3 inv Count =

2) 2, 3, 1, 5, 6, 4
Inv Count : 4
No of students . . . =

3) 3, 4, 2, 1, 6, 5
Inv count :- 6
No of students . . . =

4) 1, 2, 3, 4, 5, 6
Inv Count :- 0
No of students . . . =

5) 6, 5, 4, 3, 2, 1
Inv count :- 15
No of students . . . =

NEGATIVE

1) -1, 2, 3, 4, 5, 6.
O/P :-

2) 1, 15, 3, 4, 5, 6
O/P

3) 1, -2, 3, 4, 5, 6
O/P

4) 1, 2, 3, -4, 5, 6
O/P

5) -11, 2, 3, 4, 5, 6
O/P :

Saathi

Date ___/___/_____

Conclusion

Brute force takes $O(n^2)$ time where as divide & conquer takes $O(n \log n)$.

Program has been written using Allman Style

Name

Saathi

TASK-2

Integer Multiplication

(a) Brute-force

NAME : IntMul ( )
// I/P : Two n-digit non-negative integers x and y
// O/P :- Product $x \cdot y$

n_1 = length of x
n_2 = length of y

```
for i from n1 - 1 to 0 :
    for j from n2 - 1 to 0 :
        product = (num1[i] * num2[j])
        sum = product + result[i+j+1]
        result[i+j+1] = sum % 10
        result[i+j] += sum // 10
```

store rem ← carry over ←

remove leading zeroes from result
if result is empty, return "0".

TIME COMPLEXITY

→ n^2 primitive operations ~~(n log n)~~
⇒ max n additions for carry over
∴ $O(n^2)$.

Page No.

Saathi

Date __/__/____

⑦ Divide & Conquer (KaratSuba).

NAME: KaratSuba ( )
// I/P: 2 n digit +ve integers x & y
// O/P: x·y
// Assumption:- n is a power of 2

* if n=1 then
  compute x·y in one step & return the result

else
a, b = first & second halves of x
c, d = first & second halves of y

compute p = a+b, q = c+d using grade school add'n

recursively compute ac = a·c, bd = b·d and pq = p·q

compute adbc := pq - ac - bd

compute $10^n \cdot ac + 10^{n/2} \cdot adbc + bd$.

return this result.

Date ___/___/_____

Saathi

TIME COMPLEXITY

This computation has 3 recursive computations of n/2 bits each. O(n) additional work.

$\therefore T(n) = 3T(n/2) + cn$

By master theorem.

$n^{\log_b a} = n^{\log 2^3} \Rightarrow$ Case 1.

$\therefore T(n) = \theta(n^{\log_2 3}) = \boxed{\theta(n^{1.585})}$

TEST CASES

Positive

1) $x = 12345678$
$y = 23567819$
$O/P = 290757211106986 79$

2) $x = 5789101$
$y = 6788912$
$O/P = 38833300921112$

3) $x = 56723410$
$y = 23723159$
$O/P = 1348677511811778$

4) $x = 11111$
$y = 22222$
$O/P = 247442258$

5) $x = 23456$
$y = 34567$
$O/P = 810849872$

Negative

1) x = −123456
y = 100000
O/P :-

2) x = −851678
y = −811110
O/P =

3) x = −567856
y = −100011
O/P =

4) x = −100001
y = 111111
O/P =

5) x = −100000
y = −555555
O/P =

Conclusion :- Grade school approach requires $O(n^2)$ time. Karatsuba requires $O(n^{1.585})$

Program written using allman coding style