Lecture 18, 22 October 2024

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Lecture 18, 22 Oct 2024

Algorithms, Complexity, Data Structures

Why complexity matters? Efficiency

Given two lists, find common elements

```
for x in l1 : n1
    for y in l2 : n2
        if x==y :
            add x to output
```

$n_1 \cdot n_2$ comparisons

Validating SIM cards

Check if every SIM has a valid Aadhaar Number

l_1 = SIM + Aadhaar — n_1

l_2 = Aadhaar data — n_2

$n_1 \cdot n_2$

What are n_1 & n_2?

```
for x in l1 : - n1
    if x ∉ l2:
        report x
    for y in l2 : - n2
        if x == y :
            mark x ok
    mark x bad
```

n_1 – SIM cards ~ $\approx$ 100 crore $\approx 1 \times 10^9$

n_2 – Aadhaar numbers ~ population $\approx$ 100 cr $\approx 1 \times 10^9$

$n_1 \cdot n_2 = 10^{18}$ comparisons

10^9 ops/sec

Typical high end desktop / laptop

10^8 to 10^9 ops/sec

At least 10^9 secs

10^9 secs ?

60 secs = 1 min

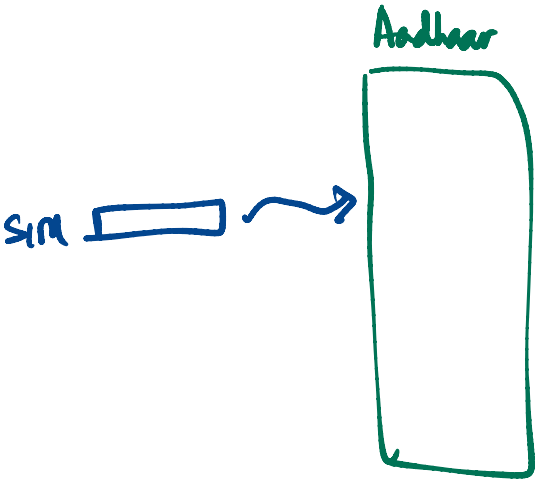
60 mn = 1 hr = 3600 sec

24 hr = 1 day = 90000

365 day $\quad 36\underline{0000}00$

$\approx$ 100 years

How To Fix ?



Sorted Aadhaar

Guess birthday

30 June

30 Sep

15 Nov

24 Oct

22 Oct

11 Oct

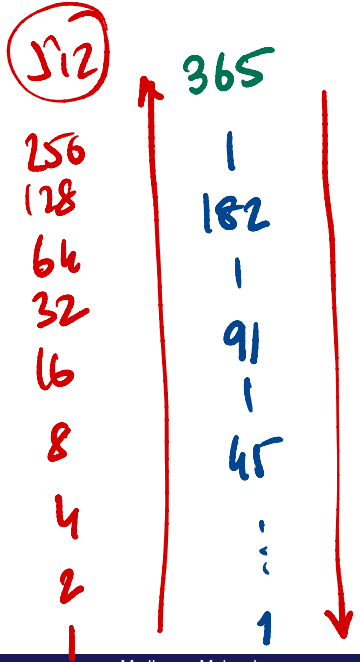
6 Oct

9 Oct

7 Oct ✓

How many questions are needed

Each step halves the search interval



$2^9 = 512$

$2^8 = 256$

$\log_2 512 = 9$

Binary search

n sorted value

$\Downarrow$

$\log_2 n$

queries

$2^{10} = 1024 \approx 1000$

$\log_2 1000 \approx 10$

$\log 1000000 \approx 20$

$\log 10^9 \approx 30$

$2^{10} \times 2^{10} = 2^{10+10}$

1000×1000

Assume Aadhaar is sorted

for x in SIM: 10^9

Check x in Aadhaar: ← binary search in 10^9

30

$10^9 \times 30 = 30$ seconds

vs 100 years

Additionally - Sort Arthaan

How long does this take?

Why not ternary?

What are we measuring?

How?

Behaviour depends on input

- How big
- How it is arranged

Measure Worst Case behaviour as a function of input size

||

Ideally measure "average" behaviour

X

Specific inputs

↓

"Worst case"

Function of size

Searching an unsorted list — Takes length of list

$f(n) = cn$ Input size = list length

Checking if a list has duplications

```
for i in range(len(L)):
    for j in range(i+1, len(L)):
        L[i] == L[j] ?
```

$(n-1) + (n-2) + \cdots + 2 + 1$

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

$c \cdot n^2$ $\quad$ $d \cdot n$

dominates as $n \to \infty$

Asymptotic complexity

$f(n)$ as $n \to \infty$

grows as $c \cdot n$

grows as $c \cdot n^2$

$f(n) = O(n)$ "Big O"

$\exists c \quad \forall n > 0 \quad f(n) \leq c \cdot n$

$f(n) = O(n^2) \quad \exists c \forall n \quad f(n) \leq cn^2$

$f(n) = 5n^2 + 6n \simeq O(n^2)$ $n = 1$

11	1
32	4
	9

c

$5n^2 + 6n$ $5n^2 + dn^2$ $c = 5$

$n \leq 6$ $3n^2$

$n > 1$ $< n^2$

Highest degree term matters

$O(n)$ better than $O(n^2)$, is better than $O(n^3)$..

$\log n$ $\sqrt{n}$

$\log n < \sqrt{n} < n$

Orders of magnitude

Input size	Values of $t(n)$						
	$\log n$	n	$n \log n$	n^2	n^3	2^n	$n!$
10	3.3	10	33	100	1000	1000	10^6
100	6.6	100	66	10^4	10^6	10^{30}	10^{157}
1000	10	1000	10^4	10^6	10^9		
10^4	13	10^4	10^5	10^8	10^{12}		
10^5	17	10^5	10^6	10^{10}			
10^6	20	10^6	10^7	10^{12}			
10^7	23	10^7	10^8				
10^8	27	10^8	10^9				
10^9	30	10^9	10^{10}				
10^{10}	33	10^{10}	10^{11}				

Input size

List - length

Multiplying 2 numbers (in base 10, say)

$k \times l$

$\overline{23}$ $\overline{47}$

```
  238
x 47
-----
 161
92x
```

No of digits in decimal = $\log_{10}(n)$

Size is log of magnitude

Naive primality testing

```
for i in range(2, n):
    check if n % i == 0
```

$\approx O(n)$

$O(10^{\log_{10}(n)})$ time