Q => You are given an integer list of size n. ($n \leq 10^7$). After taking input of the list, you will get another integer 'Q'. Q represents no. of queries that means you will Q queries & inside each query you will get 2 numbers denoting index of list (L) & (R). You have print sum of elements from L to R for each query.

$n \leq 10^7$
$Q \leq 10^5$
$0 \leq L_i, R_i \leq n-1$

$$\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \underbrace{[1, & 2, & 7, & 1, & 3, & 2, & 1]}_{} \end{array}$$

$\underline{\underline{q \Rightarrow 3}}$

$2 \quad 4$

$3 \quad 6$

$5 \quad 5$

$\hookrightarrow$ 11, 7, 2] $\underline{\underline{\text{ans}}}$

$\underline{\underline{q \le 10^5}}$

$\underline{\underline{n \le 10^7}}$

>>> $\underline{\underline{10^8}}$

Sum

Sum $\rightarrow$

$\underline{\underline{O(n)}}$

$\downarrow$

$\underline{\underline{O(qn)}}$

Sum(l, R)

$O(R-l)$

$\underline{\underline{O(n)}}$

query

$[1, 2, \overset{l}{7}, 1, 3, \overset{r}{2}, 1]$

$f(l, r) = \underbrace{f(0, r)}_{} - f(0, l) + arr[l]$

returns sum of elements from l to R

we have this already with us

prefix sums / cumulative sums

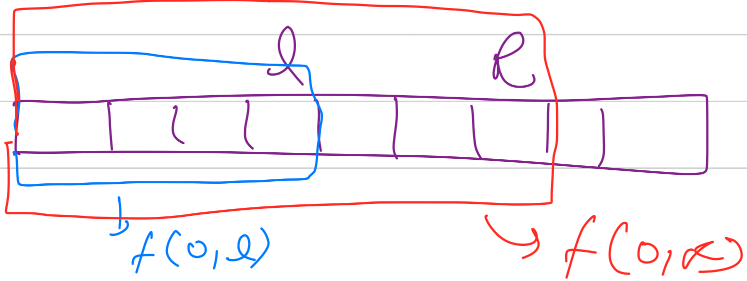
$$\sum_{i=0}^{r} a[i]$$

→ O(1) time

~~O(1) time~~

φ

O(Ø)



Problem boils down to calculating the prefix sum:

$[a_1, a_2, \boxed{a_3}, \boxed{a_4}, \ldots\ldots a_n]$

↪ iterate →

$sum = 0 + a_1 + a_2 + a_3 + \ldots$

$sum = sum + arr[i]$

$O(n)$

$O(n+q)$ final time complexity

[1, 2, 3, 4, 5, 6]

prefix list → [1, 3, 6, 10 - - - - -]

sum = ~~0~~ ~~1~~ ~~3~~ ~~6~~ 10

$O(\underbrace{n}_{} + q)$

q → O(1)

Q:- Given a list of integers of size n, Print all the Subsets of the given list.

Ex

[1, 2, 3]

$\rightarrow$ [ ]

[1]

[2]

[3]

[1, 2]

[2, 3]

[1, 3]

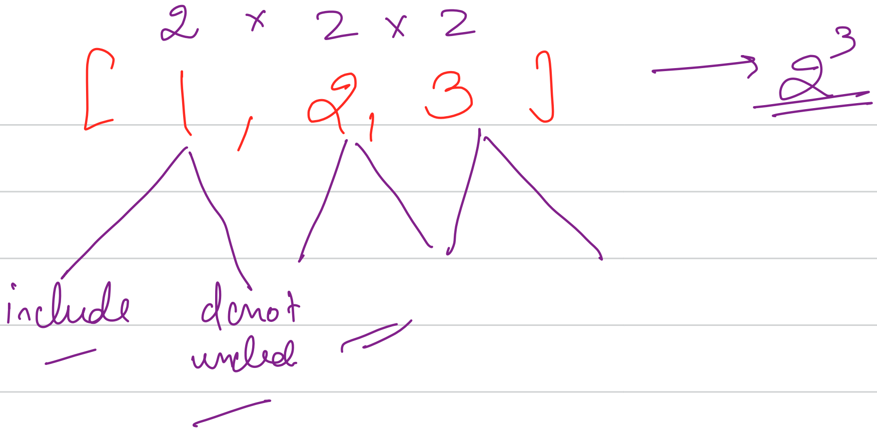
[1, 2, 3]

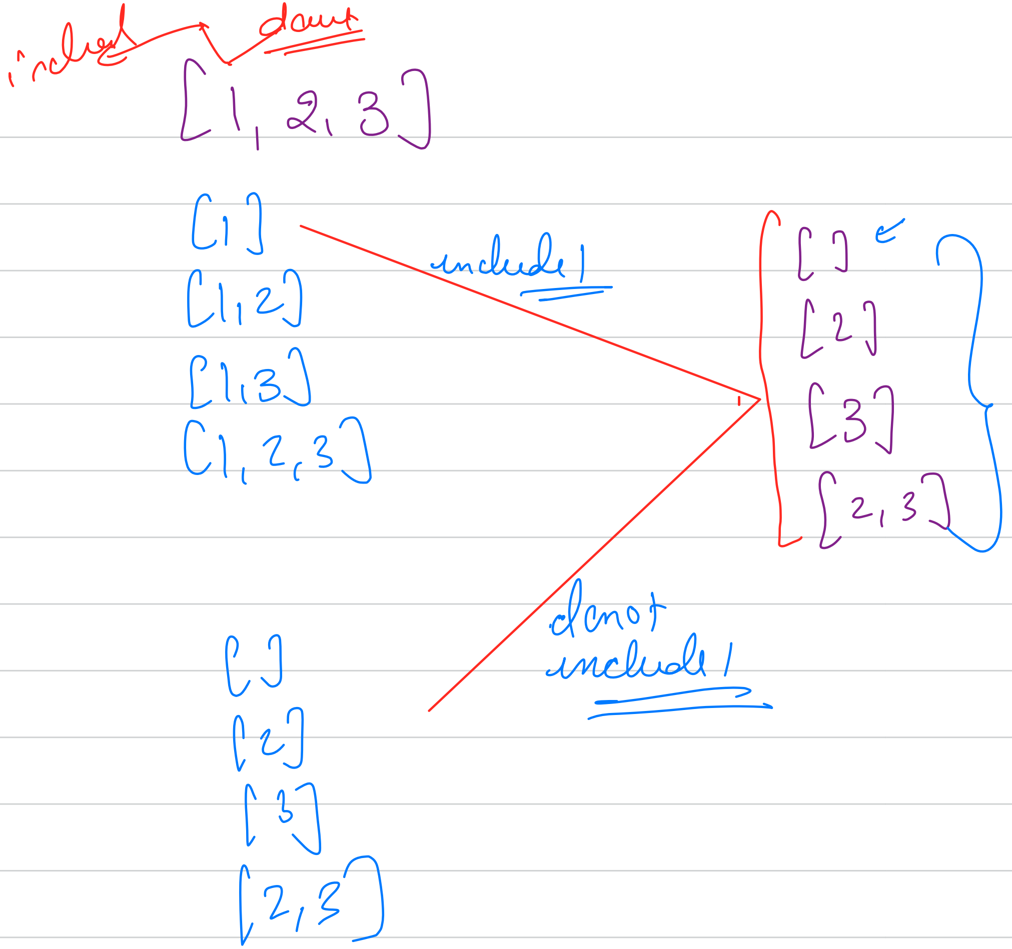
2^n

Recursively

$n \rightarrow$ size of list

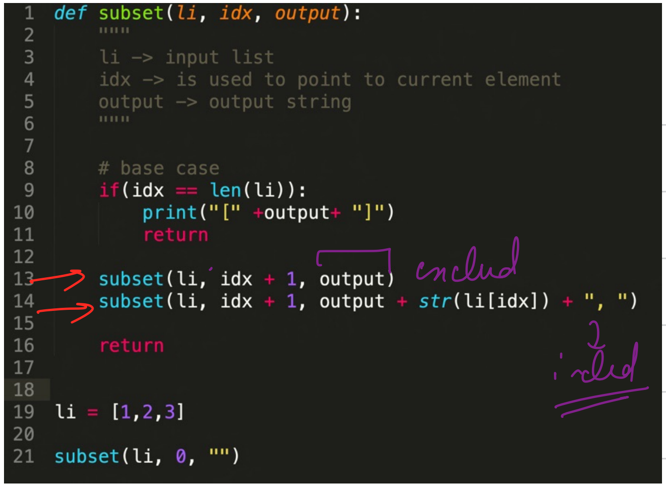
$n \leq 20$


2 × 2 × 2
[1, 2, 3]
$\rightarrow 2^3$
include
donot include


include
dont
[1, 2, 3]
[1]
[1, 2]
[1, 3]
[1, 2, 3]
include 1
[ ]
[2]
[3]
[2, 3]
donot include 1
[ ]
[2]
[3]
[2, 3]

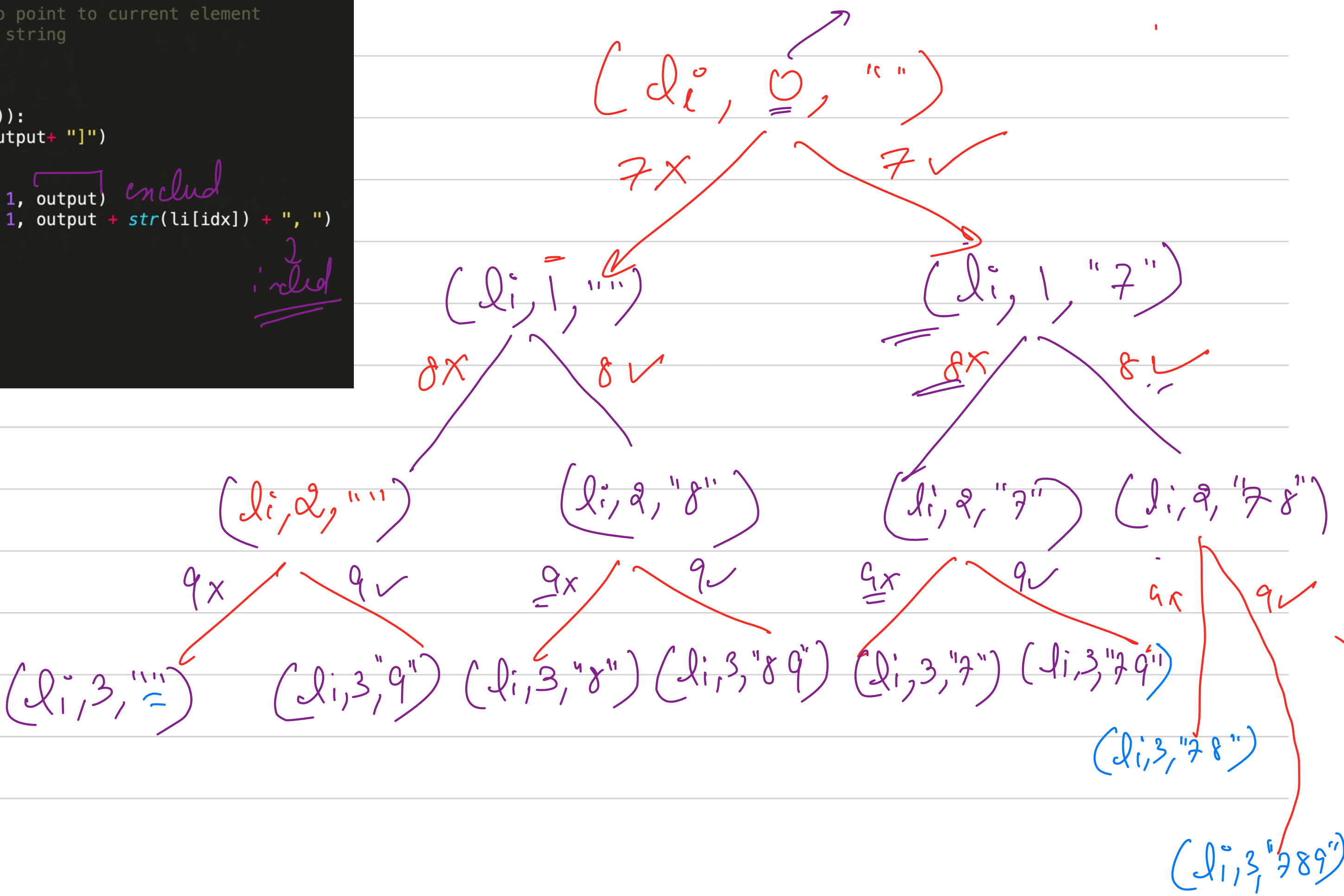
$$\begin{tikzcd}
{[2,3]} & & & \\
{[2]} & & & \left\{\begin{matrix}[3] \\ [\ ]\end{matrix}\right. \arrow[lll, "\text{include } 2"'] \arrow[dlll, "\text{donot include } 2"] \\
{[3]} & & & \\
{[\ ]} & & &
\end{tikzcd}$$

$$\begin{tikzcd}
& & & {[3]} \\
{[\ ]} \arrow[r] & {[\ ]} \arrow[urr, "\checkmark 3"] \arrow[drr, "\times 3"'] & & \\
& & & {[\ ]}
\end{tikzcd}$$



```
def subset(li, idx, output):
    """
    li -> input list
    idx -> is used to point to current element
    output -> output string
    """

    # base case
    if(idx == len(li)):
        print("[" +output+ "]")
        return

    subset(li, idx + 1, output)
    subset(li, idx + 1, output + str(li[idx]) + ", ")

    return


li = [1,2,3]

subset(li, 0, "")
```

li = [7, 8, 9]

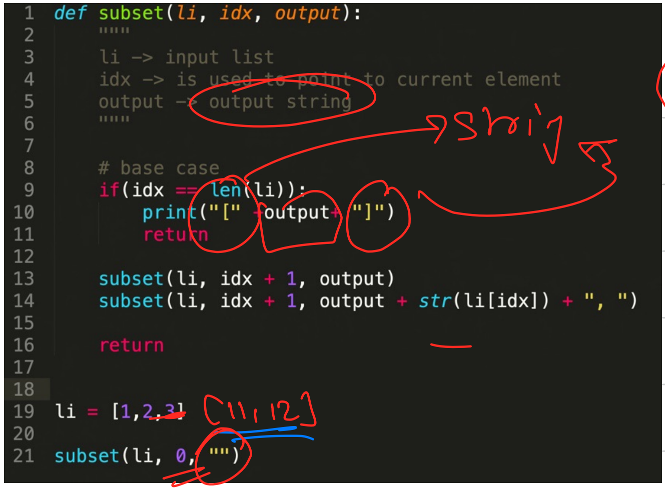


f(li, idx, output)

$\downarrow$

have a choice of inclusion & exclusion for element of the idxth <u>index</u> & calculate subset of rest of the elements <u>recursively</u>.

if you don't append anything, that means you didn't choose the current <u>element</u>.



```python
def subset(li, idx, output):
    """
    li -> input list
    idx -> is used to point to current element
    output -> output string
    """

    # base case
    if(idx == len(li)):
        print("[" +output+ "]")
        return

    subset(li, idx + 1, output)
    subset(li, idx + 1, output + str(li[idx]) + ", ")

    return

li = [1,2,3]

subset(li, 0, "")
```

→ string

["+" "+"]

[11, 12]

[11, 12]

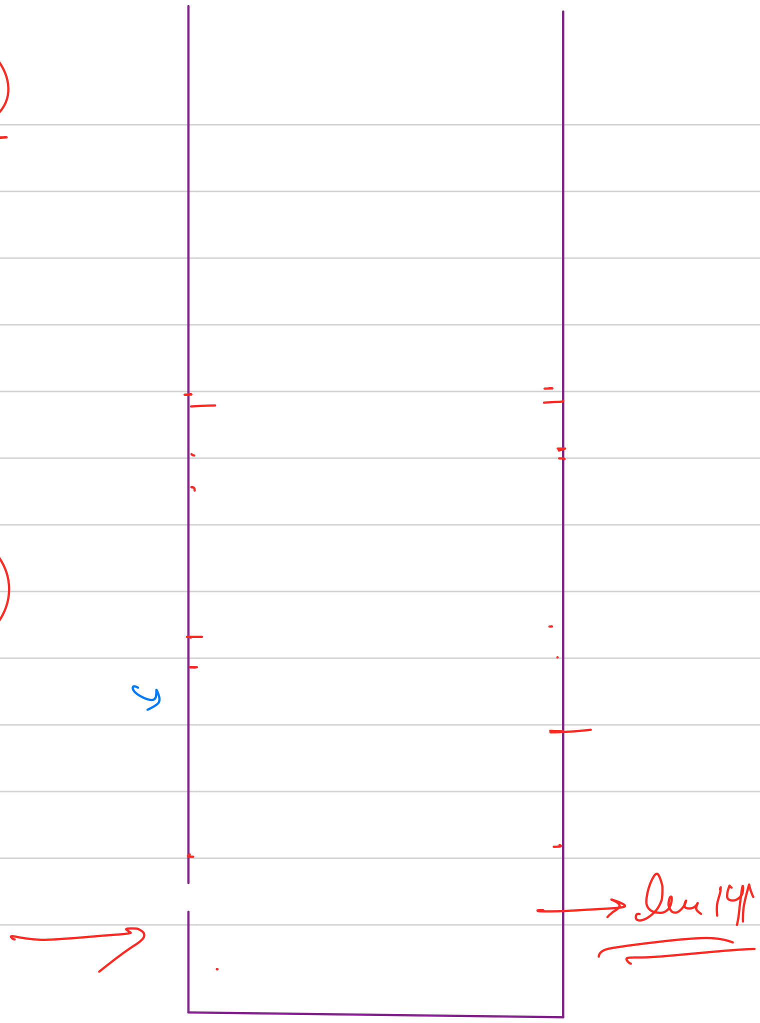
[ ]

[12,]

[11,]

[11, 12,]

"li[idx]"



Bitwise operations

Try to code on your own

[1, 2, 3] → 3 → 2^3

n^{th}

Binary → endian
↘ exclusive

0	0	0	→	[ ]
0	0	1	→	[3]
0	1	0	→	[2]
0	1	1	→	[2, 3]
1	0	0	→	[1]
1	0	1	→	[1, 3]
1	1	0	→	[1, 2]
1	1	1	→	[1, 2, 3]

$0 \rightarrow 2^3 - 1$

i

1 1

S

S & (1 << 2)

S & 1

break → breaks you from nearest loop

→ continue → it goes back for next iteration for the nearest loop.

→ pass →

→

```
if (i%2):
    continue
else
```