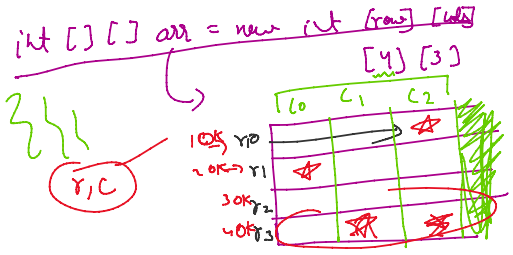
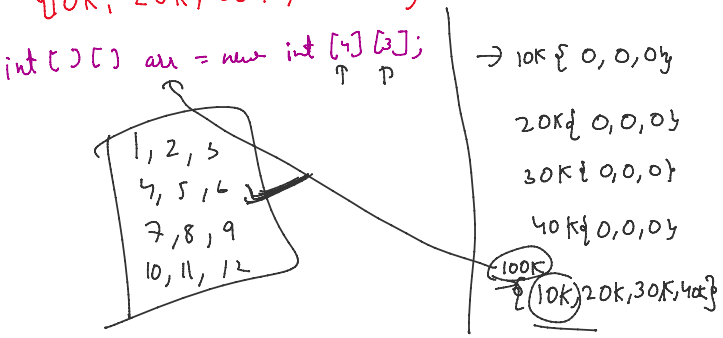
2D array

27 May 2023 12:20



{10K, 20K, 30K, 40K }

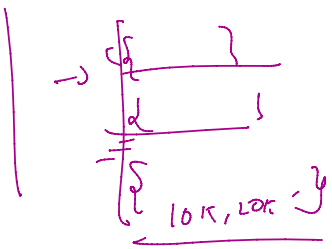


int [] [] arr = new int [n] [m]

47:30

n x m

m + n - 1



Input

11 12 13 14

21 22 23 24

31 32 33 34

41 42 43 44

0,0 1,0 2,0 3,0 | 0,1 1,1 2,1 3,1

==========

Output

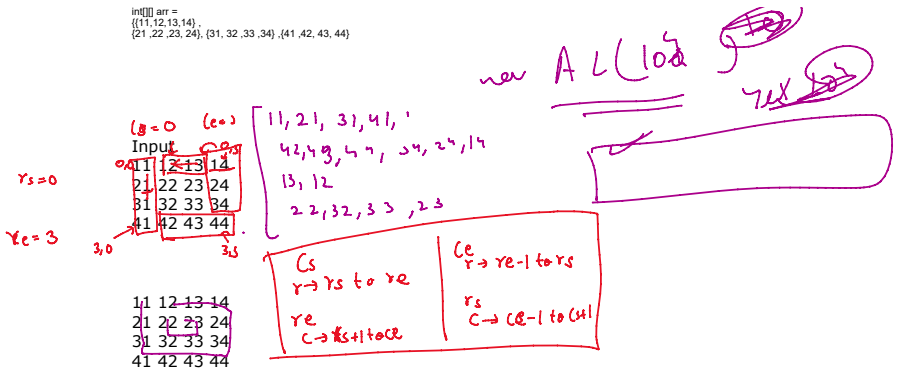
11 21 31 41 ,42 32 22 12 13 23 33 43 44 34 24 14

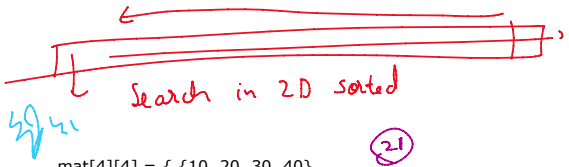
```
int[][] arr =
{{11,12,13,14} ,
{21 ,22 ,23, 24}, {31, 32 ,33 ,34} ,{41 ,42, 43, 44}
```

new ALC(10K)

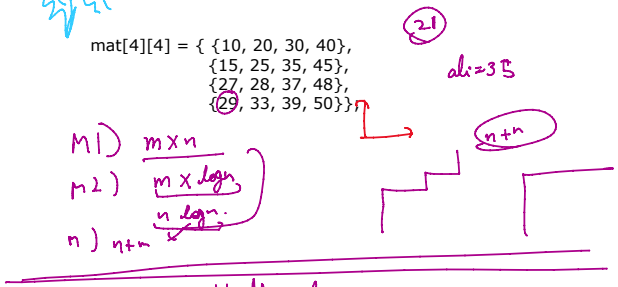
4 2K

```
int[][] arr =
{{11,12,13,14} ,
{21 ,22 ,23, 24}, {31, 32 ,33 ,34} ,{41 ,42, 43, 44}
```






```
mat[4][4] = { {10, 20, 30, 40},
              {15, 25, 35, 45},
              {27, 28, 37, 48},
              {29, 33, 39, 50}};
```



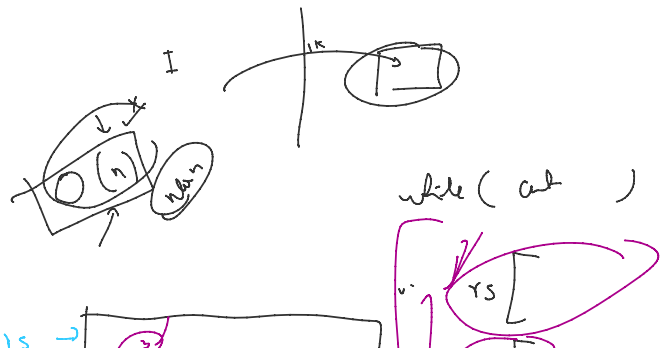
Wrapper class

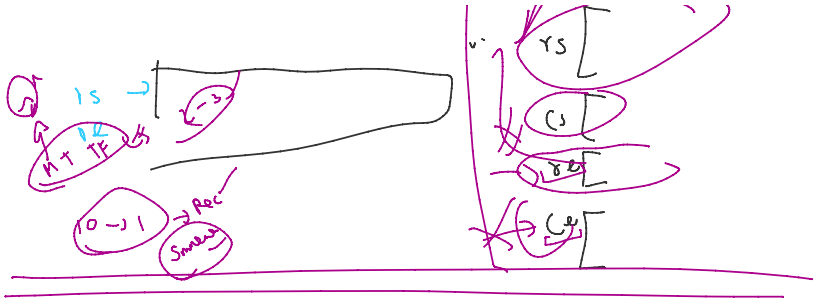
Primitive → non primitive

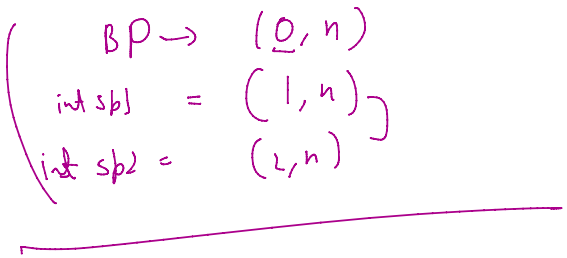
heap → stack → ∴ true

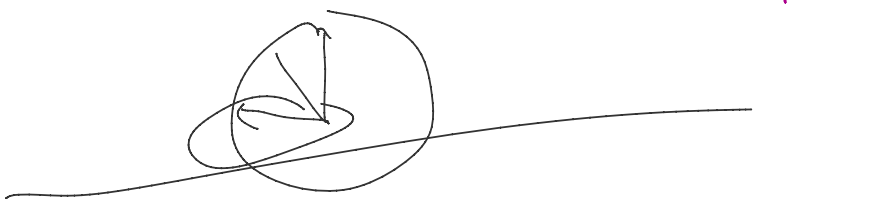
primitive → byte, short, int, long, float, double, boolean, char

Wrapper class → Byte, Short, Integer, Long, Float, Double, Boolean, Character









The coding blocks members went to the success party of their first ever online boot-camp at Murthal. They ordered **P** number of paranthas. The stall has **L** cooks and each cook has a rank **R**. A cook with a rank **R** can cook 1 parantha in the first R minutes 1 more parantha in the next 2R minutes, 1 more parantha in 3R minutes and so on(he can only cook a complete parantha) (For example if a cook is ranked 2.. he will cook one parantha in 2 minutes one more parantha in the next 4 mins and one more in the next 6 minutes hence in total 12 minutes he cooks 3 paranthas. In 13 minutes also he can cook only 3 paranthas as he does not have enough time for the 4th parantha). Calculate the minimum time needed to cook all the paranthas.

Input Format

First line contains P, the number of pratha ordered. In the next line the first integer denotes the number of cooks L and L integers follow in the Next line each denoting the rank of a cook.

Output Format

Print an integer which tells the number of minutes needed to get the order done.

Sample Input

10

4

1 2 3 4

Sample Output

12

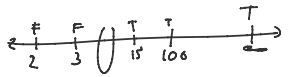
Explanation

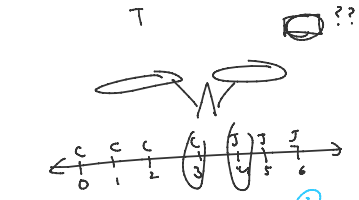
First cook with rank 1 cooks 4 paranthas in 10 minutes (1+2+3+4).

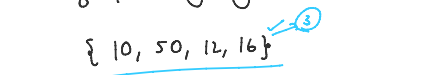
Second cook with rank 2 cooks 3 paranthas in 12 minutes (2+4+6)

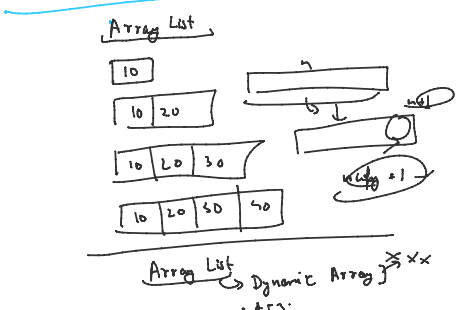
Third cook with rank 3 cooks 2 paranthas in 9 minutes (3+6) Fourth cook with rank 4 only needs to cook one last remaining parantha. He can do that in 4 minutes.

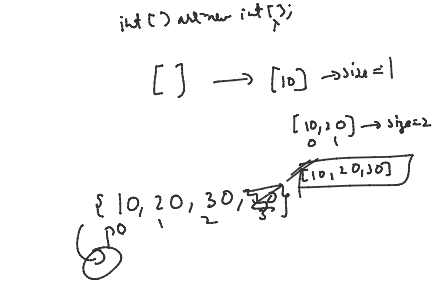
Since these cooks cook parallely, the total time taken will be the maximum of the four i.e. 12 minutes.





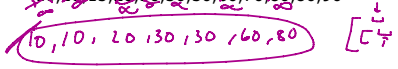


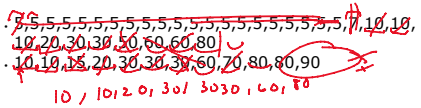


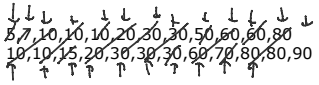


2 sorted arrays. find elements which are present in both arrays

- 5,7,10,10,10,20,30,30,50,60,60,80
- 10,10,15,20,30,30,30,60,70,80,80,90





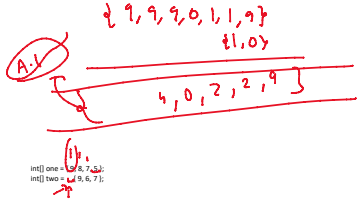


- one: [9,8,7,5]
- two: [9,6,7]
- [1,0,8,4,2]

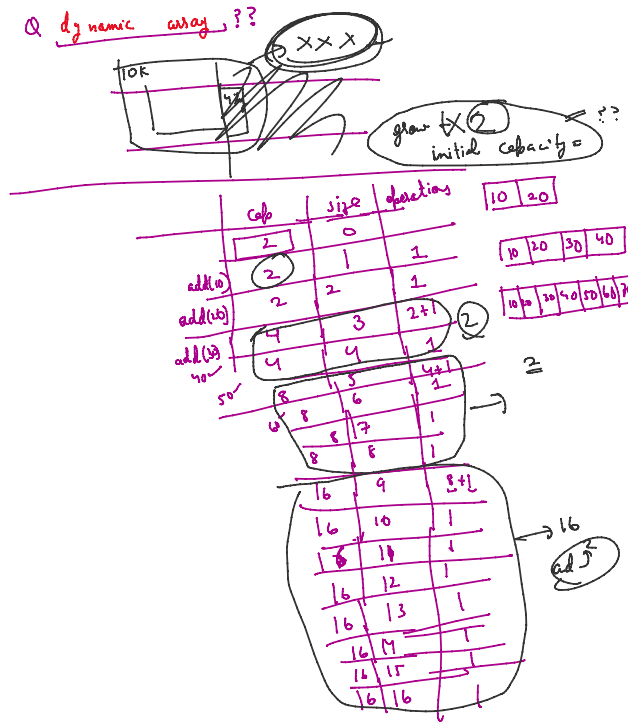
3min

one: [9,8,7,5]
two: [9,6,7]
[1,0,8,4,2]

3min



{2,4,8,0}



min Cap = 1
old = 10

(10, 1, 5)

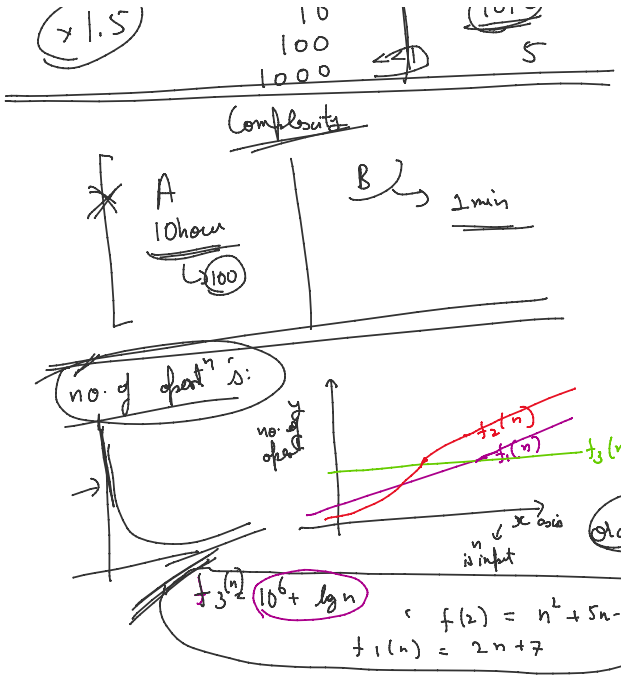
×1.5

1
10
100
1000

1010 >>1

101

5



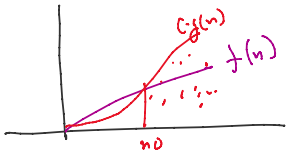
1 GB → 1000 MB → 10^6 KB → 10^9 B

1 GH$_2$ → 10^9 H$_2$

$f(n)$ → large values of n worst case

Big O → O

$c \cdot g(n) \geq f(n) \quad \forall n \geq n_0$



$O(g(n))$

↳ large values of n converge $f(n)$ worst case

$f(n) = 5n + 7$ → $O(n)$

$5n + 7 \leq 12n \quad \forall n \geq 2$

$f(n) = 2n^2 - 5n - 500 \leq 2n^2 + 5n^2 + 500n^2 \quad \forall n \geq 100$

$\leq 510 n^2$

$f(n) = O(n^2)$

$f(n) = a_x n^x + a_{x-1} n^{x-1} + a_{x-2} n^{x-2} + \cdots a_0 n^0$

$O(n^2)$

$a_2 n^2 + a_{x-1} n^{x-1} + a_{x-2} n^{x-2} \dots a_0 n^x$

$n^x \quad n \quad \geq f(n) \quad n \geq n_{00}$

1

$O(k_2 n^x)$

→ Maths cost
→ Variable cost
→ Print cost

```
i=0                    1
while (i<n){           1
  System.out.println   1
  i++                  1
}
```

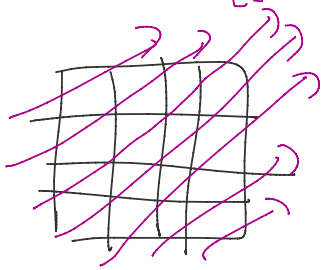
$f(n) = 1+3l$

$f(n) = 1+3n \hookrightarrow O(n)$

```
i = 1
while (i<n){
  Syso(??)
  i = i*2;
}
```

$f(n) = 1 + 3l.$

i	l
1	1
2	2
2^2	3
2^3	4
2^{l-1}	l



$2^{l-1} = n$

$l-1 = \log_2 n$

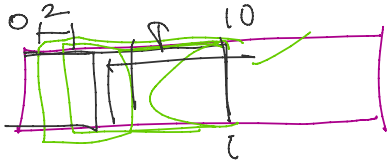
$l = \log_2 n + 1$

$2^? = 4 \quad ? = \log_2 4$

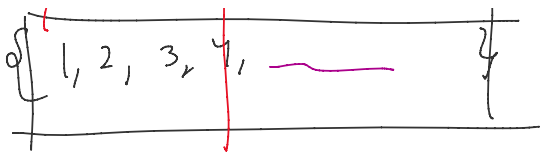
$2^? = 8 \quad ? = \log_2 8$

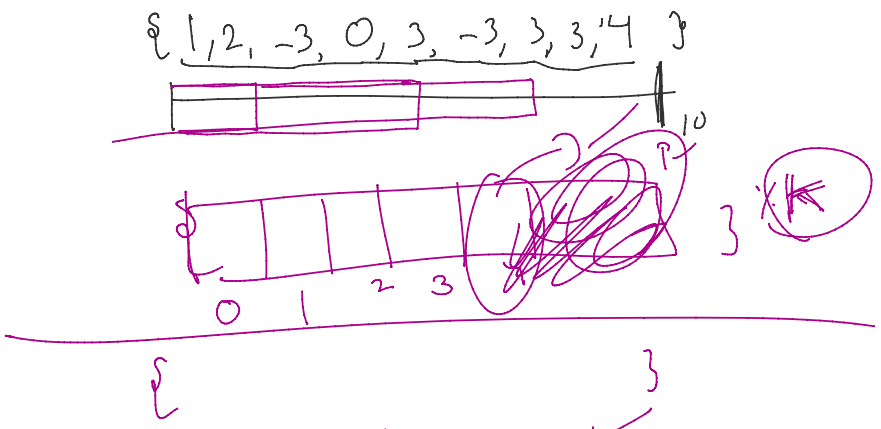
$f(n) = 1 + (\log_2 n + 1) \cdot 3$

$O(\log n)$



2





678 , 6905

A B

678 6905

6905 678

l	i
1	0
2	1
3	2

```
for( i=0 ; i*i<=n ; i++) {
    Syso (i);
}
```

l	i
1	0
2	1
3	2
.	
l	l-1

$(l-1)^2 <= n$

$l = \sqrt{n} + 1$

---

i → 1 to n | i++

j → 1 to i | j++

k → 1 to 1000 k++

i=1	i=2	i=3	i=4	i=5	i=6	i=n
j=1	j=1, 2					
1000c	1000c, 1000c	3·1000c	4Kc	5kc	6Kc	nKc

1000c + 2·1000c + 3·1000c . . . n·1000c

1000c (1+2+3 . . . n)

1000c · $\frac{n\cdot(n+1)}{2}$ → n → n^2

```
for ( i= 1 ; i <= n ; i++){
    for ( j=1; j<= i^2 ; j++ ) {
        for ( k= 1 ; k <= n/2 ; k++ ) {
            Syso( " Ques Karo" );
        }
    }
}
```

i=1	i=2	i=3	i=n
j= 1	1, 2, 3, 4		
n/2	$2^2 \cdot n/2$	$3^2 \cdot n/2$	$n^2 \cdot n/2$
		i=3	i=n
		$n^2 \cdot n/2$	$5^2 \cdot n/2$

$n/2 \left(1^2 + 2^2 + 3^2 \ldots n^2 \right)$

$n/2 \cdot \frac{n\cdot(n+1)(2n+1)}{6}$ → n^3

$\frac{n \cdot (n+1)(2n+1)}{6}$ $\quad i = n$ $\quad n^3 \cdot n/1$

$\Rightarrow n^4$

$\Rightarrow \sum \frac{1}{n} = \log n$

```
for ( i = 1 ; i <= n ; i++ ) {
    for (j = 1 ; j <= n ; j = j + i ) {
        Sysо ("Copy Par Karo");
    }
}
```

$i = 1$

$j = 1, n, j = j+1$ $\quad n$

$i = 2$ $\quad j = j+2$ $\quad j = 1, n$ $\quad n/2$

$i = 3$ $\quad i = j+3$ $\quad n/3$

$i = n$ $\quad n/n$

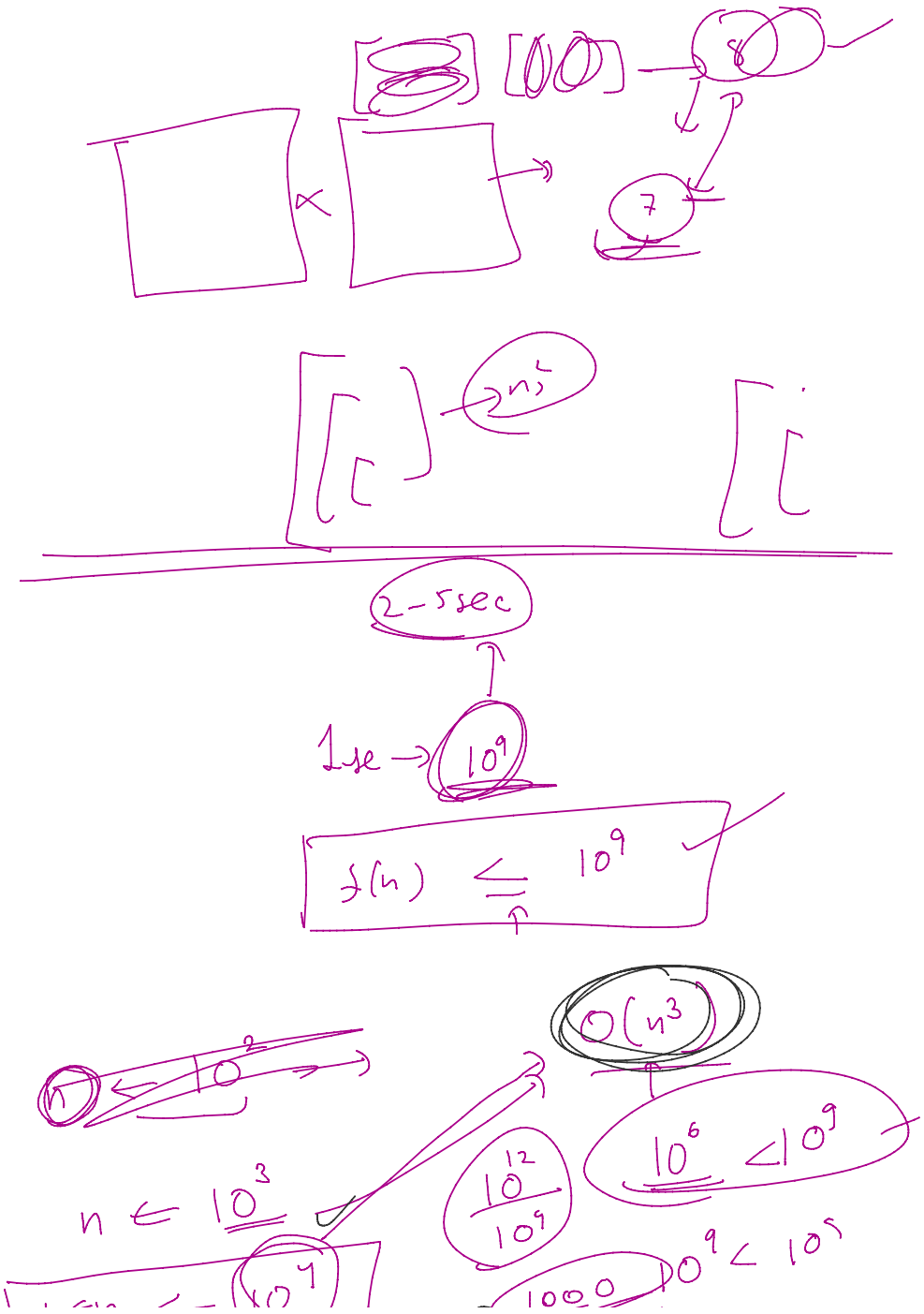
$\sum \frac{1}{n} < \log n + n/2 + n/3 + \cdots n/n$

$n\left(1 + 1/2 + \frac{1}{3} + \frac{1}{n}\right)$

$n \log n$

$K = 5$

$n = 50$


n^3
2-5sec
1sec
10^9
f(n) ≤ 10^9
O(n^3)
n^2
n ≤ 10^3
10^6 < 10^9
10^12 / 10^9
10^9 < 10^9
1000

$1 \le n \le 10^7$

10^7

$O(n^2)$ 1000 $10^9 < 10^9$

$2^{100} = (2^{10})^{10} = (1000)^{*10} = 10^{30}$

$n = 25$

2, 3, 5, 7, 11, 13, 17, 19, 23,

num 1 to n

div → 1 to num

div → 2 to num-1

div → 2 to num/2

div → 2 to $\sqrt{num}$

$n\sqrt{n}$

→

4 to 12 12 to 1

p q = 12

2 to 12/2 12/2 to 2

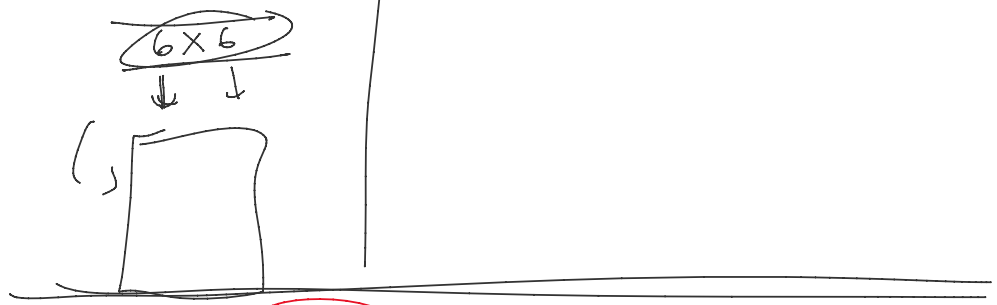
$= 12$

1 x 12
2 x 6
3 x 4
4 x 3
6 x 2
12 x 1

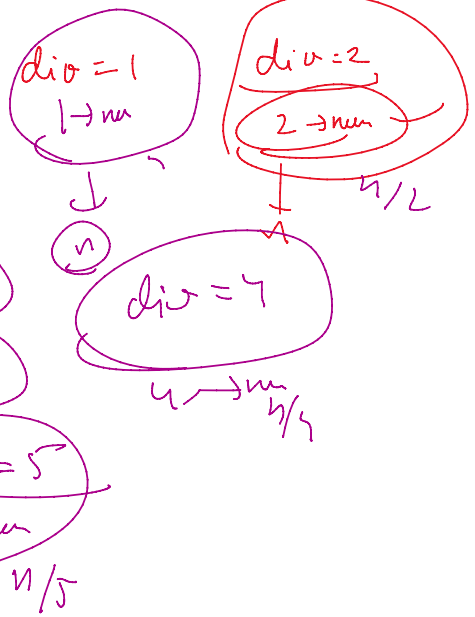
$p = q$ $\sqrt{n}$

36

4 X 3
6 X 2
12 X 1

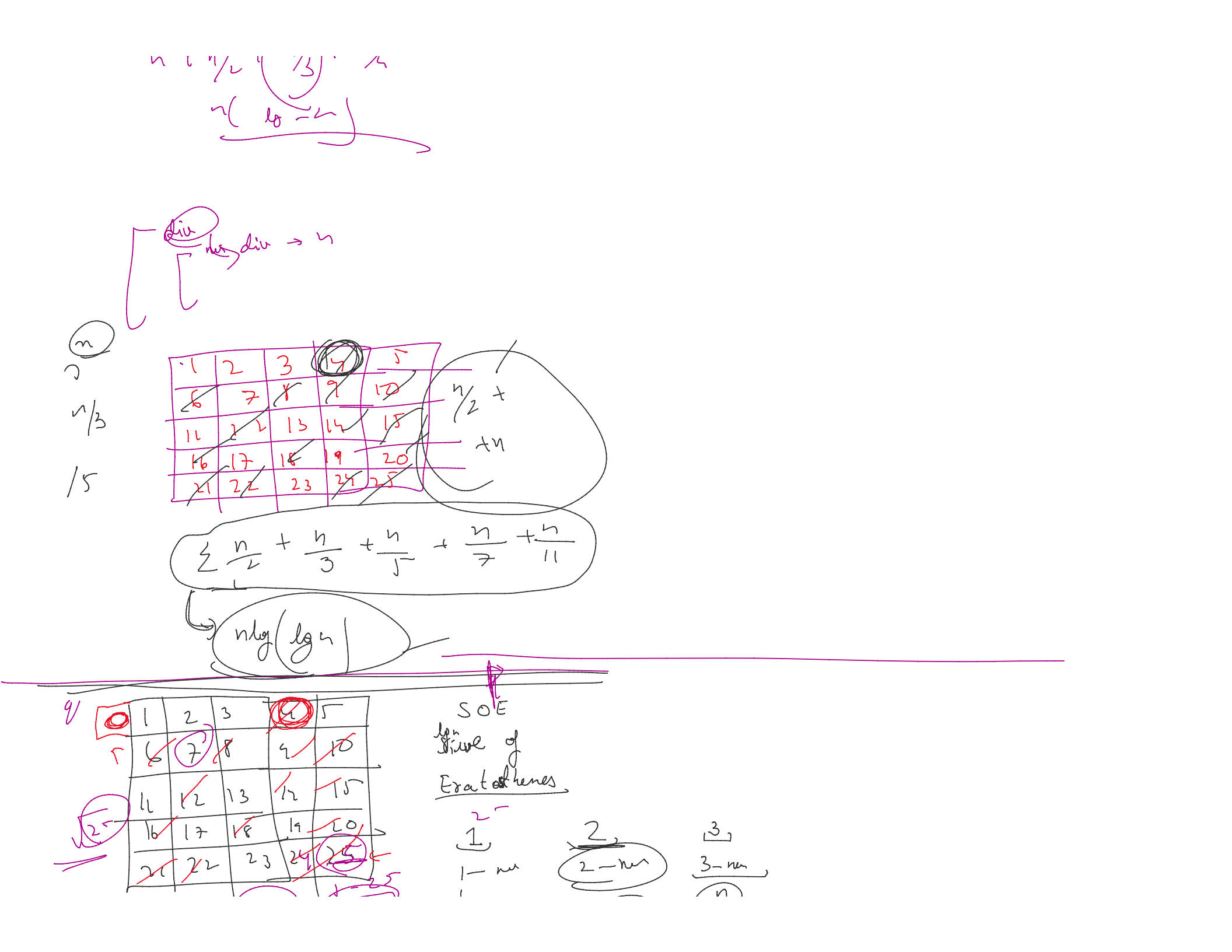


num → 1 to n
div → 1 to num.
→ [2



$n + n + n + 2$

$n + n/2 + n/3 + n/4 + n/5$


n
n/3
/5
n/2 +
+n
nlog(lg n)
SOE
Sieve of
Eratosthenes
1
2
3
1— nu
2— nu
3— nu
n

$2 \rightarrow n/2$

$3 \rightarrow n/3$

$5 \rightarrow n/5$

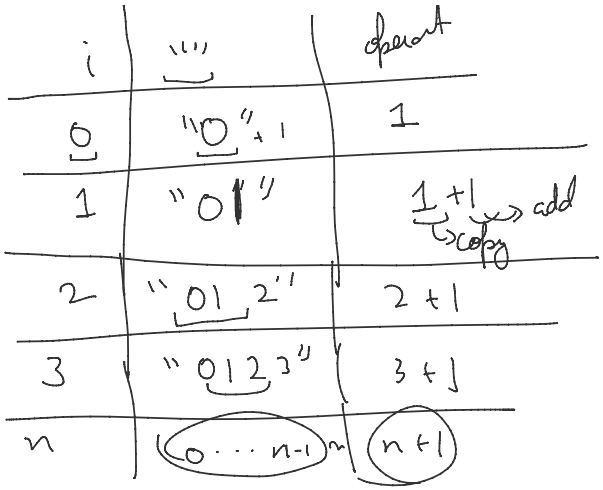
$$\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} \cdots \frac{n}{\sqrt{n}}$$

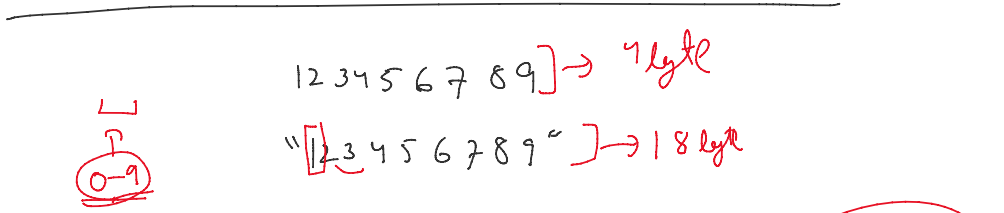
$$n\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} \cdots \frac{1}{\sqrt{n}}\right)$$

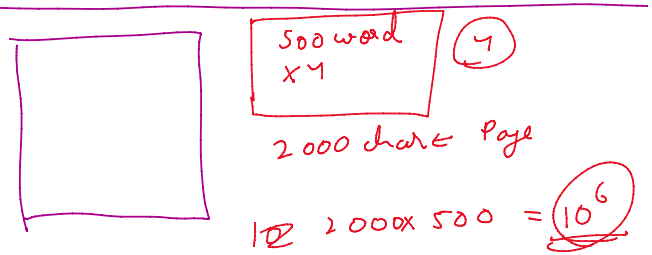
$n \cdot \log(\log n)$

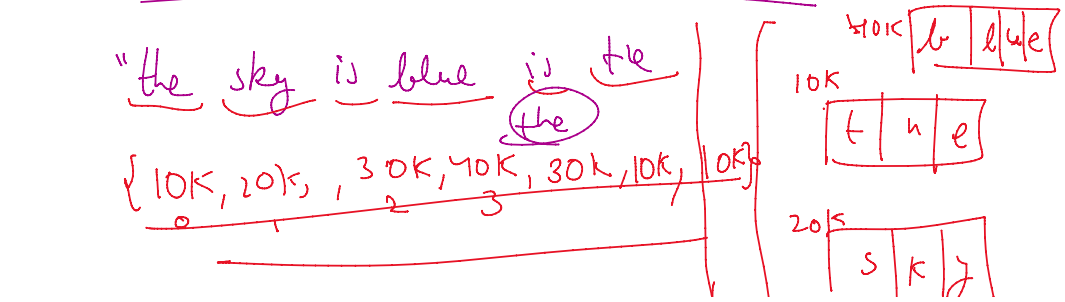
2 X 2	3 X 2	4 X 2	5 X 2
2 X 3	3 X 3	4 X 3	5 X 3
2 X 4	3 X 4	4 X 4	5 X 4
2 X 5	3 X 5	4 X 5	5 X 5
	3 X 6	4 X 6	6 X 2
			6 X 3
			6 X 5
			6 X 6

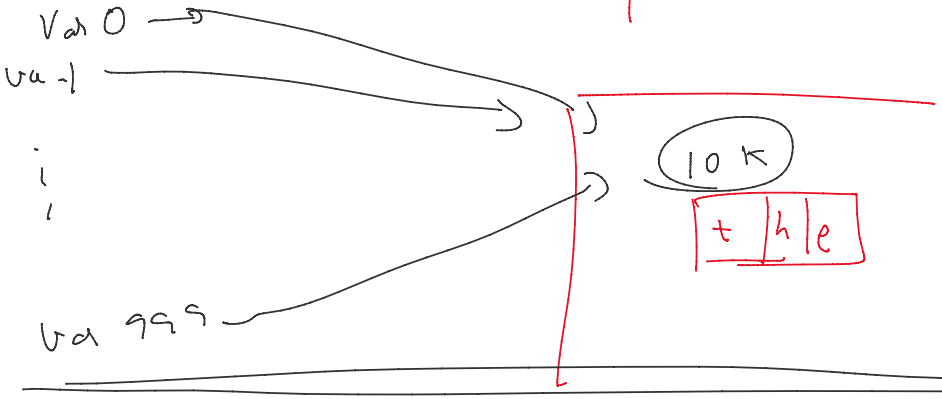
i	...	operation

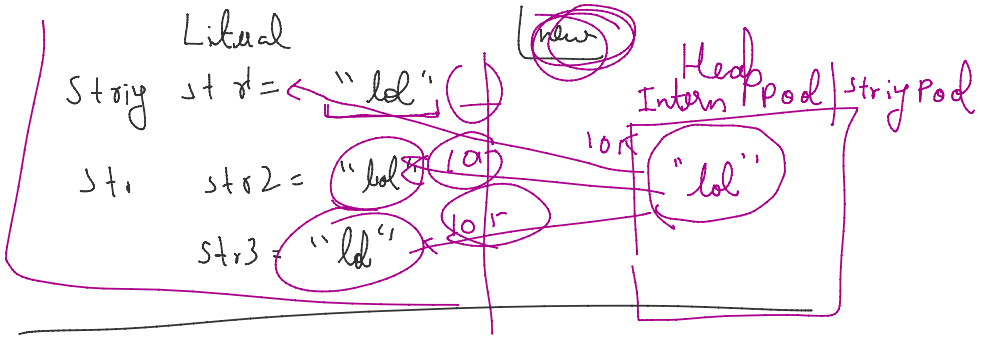

i
""
append
0
"0" + 1
1
1
"01"
1 + 1
copy
add
2
"012"
2 + 1
3
"0123"
3 + 1
n
0 ··· n-1
n
n + 1


12345678 9] → 4 byte
"[1]2345678 9"] → 18 byte
0-9
→ 2 byte → 16 bit
65000


500 word
× 4
4
2000 char ← Page
2000 × 500 = 10^6

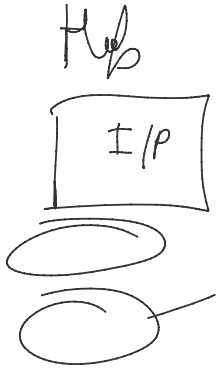

"the sky is blue is the
the
{10K, 20K, , 30K, 40K, 30K, 10K, 10K}
0 1 2 3
40K
b l u e
10K
t h e
20K
s k y

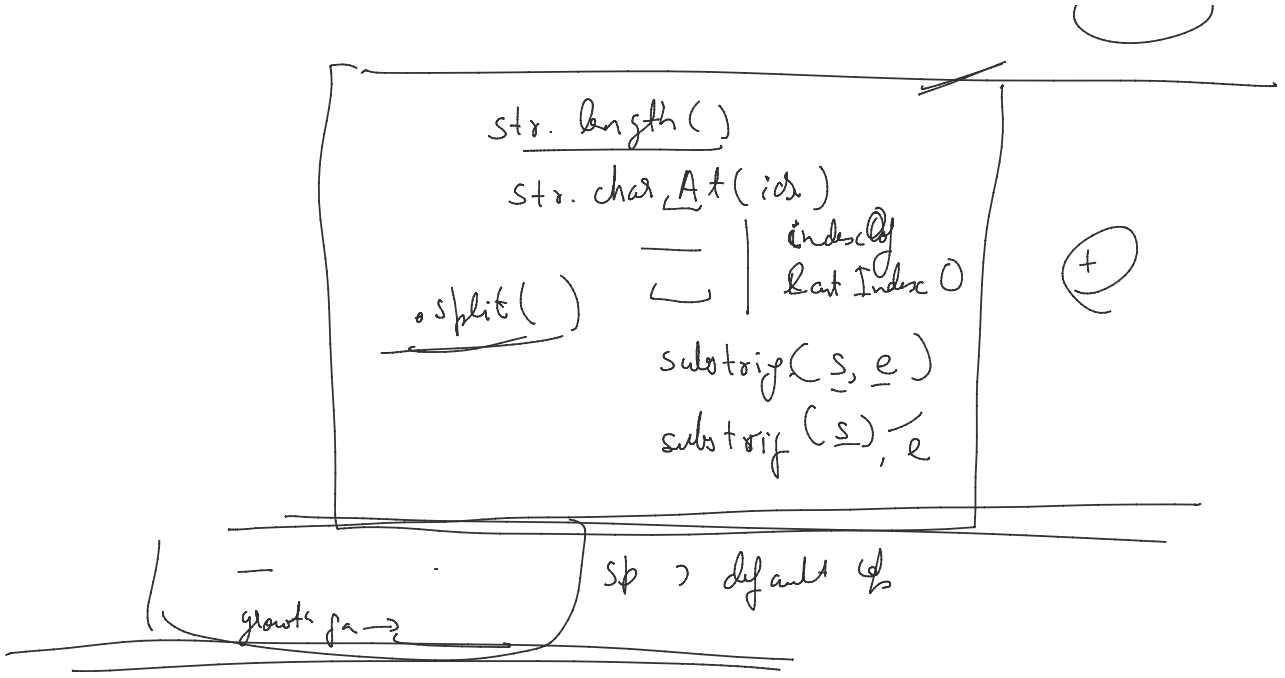




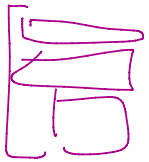
new

String str2 = new ("Hello");





Recursion:



100 MB → 10^8 Pos

1) ↳ Identify B.P.

2) ↳ Identify S.P

3) Assume (S.P) ← choose ki works

4) Create Bigger sol^n using smaller sol^n + self work

Q1) PD(5):

5
4
3
2
1

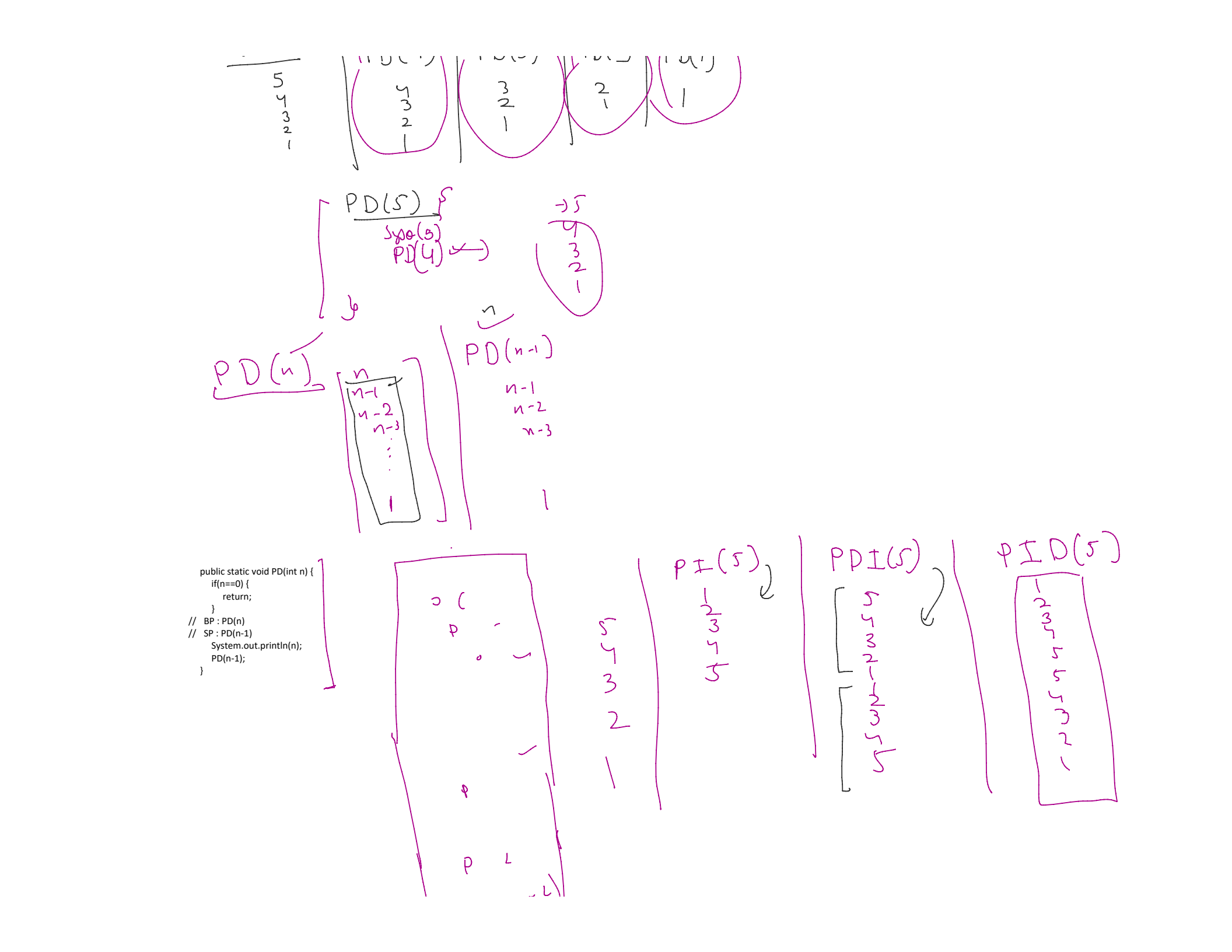
PD(5)
5
4

PD(4)
4
3

PD(3)
3
2

PD(2)
2
1

PD(1)
1


PD(5)
Sysо(5)
PD(4)
PD(n)
PD(n-1)
n
n-1
n-2
n-3
1
PI(5)
PDI(5)
PID(5)
public static void PD(int n) {
if(n==0) {
return;
}
// BP : PD(n)
// SP : PD(n-1)
System.out.println(n);
PD(n-1);
}