A. Vanya and Fence

time limit per test: 1 second
memory limit per test: 256 megabytes

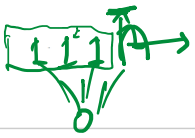
Vanya and his friends are walking along the fence of height h and they do not want the guard to notice them. In order to achieve this the height of each of the friends should not exceed h. If the height of some person is greater than h he can bend down and then he surely won't be noticed by the guard. The height of the i-th person is equal to a_i.

Consider the width of the person walking as usual to be equal to 1, while the width of the bent person is equal to 2. Friends want to talk to each other while walking, so they would like to walk in a single row. What is the minimum width of the road, such that friends can walk in a row and remain unattended by the guard?

Input
The first line of the input contains two integers n and h ($1 \le n \le 1000$, $1 \le h \le 1000$) — the number of friends and the height of the fence, respectively.

The second line contains n integers a_i ($1 \le a_i \le 2h$), the i-th of them is equal to the height of the i-th person.

Output
Print a single integer — the minimum possible valid width of the road.



Output
Print a single integer — the minimum possible valid width of the road.

Examples

input	output
3 7 4 5 14	4
6 1 1 1 1 1 1 1	6
6 5 7 6 8 9 10 5	11

Note
In the first sample, only person number 3 must bend down, so the required width is equal to $1 + 1 + 2 = 4$.

In the second sample, all friends are short enough and no one has to bend, so the width $1 + 1 + 1 + 1 + 1 + 1 = 6$ is enough.

In the third sample, all the persons have to bend, except the last one. The required minimum width of the road is equal to $2 + 2 + 2 + 2 + 2 + 1 = 11$.

Q)

n , h
friends minimum

bend → 1 widm
bends → 2

count = 0
for (Element → array) {
if (Element > h) count += 2
else count ++;

B. Little Nikita

time limit per test: 1 second
memory limit per test: 256 megabytes

The little boy Nikita was given some cubes as a present. He decided to build a tower out of them.

Initially, the tower doesn't have any cubes. In one move, Nikita either puts exactly 1 cube on top of the tower or removes exactly 1 cube from the top of the tower. Is it possible that after n moves, the resulting tower has exactly m cubes?

Input
Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains two integers n and m ($1 \le n, m \le 100$).

Output
For each test case, output "Yes" (without quotes) if Nikita can obtain a tower with m cubes, and "No" (without quotes) otherwise.

You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

Example

input	output
3 3 3 2 4 5 3	Yes No Yes

h → +

n , m

moves , cubes height

Output
For each test case, output "Yes" (without quotes) if Nikita can obtain a tower with m cubes, and "No" (without quotes) otherwise.

```
Yes
No
Yes
```

Output

For each test case, output "Yes" (without quotes) if Nikita can obtain a tower with m cubes, and "No" (without quotes) otherwise.

You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

Example

input
3 3 3 2 4 5 3

output
Yes No Yes

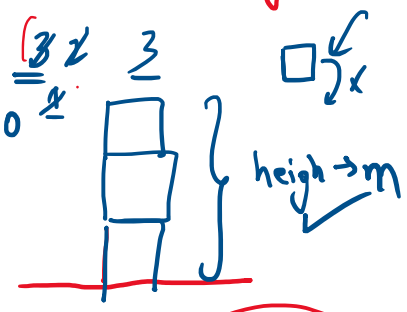
Note

In the first test case, Nikita can put 1 cube on top of the tower 3 times in a row, so the answer is "Yes".

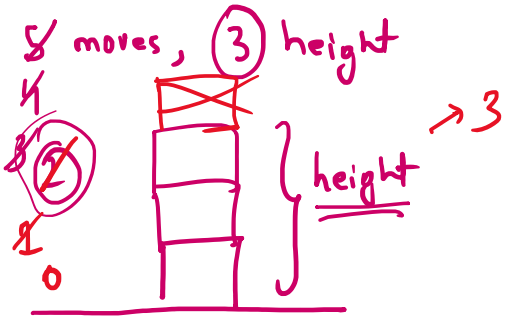
In the second test case, Nikita can only end up with either a tower with no blocks or a tower with 2 blocks, so the answer is "No".

n, m

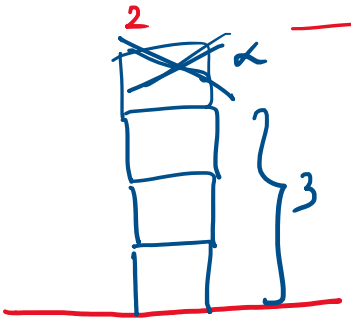
moves, ~~cubes~~ height

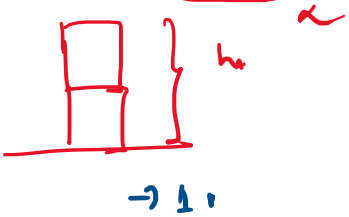


2 moves, 4-height



* NO,





1 moves, 3 height →

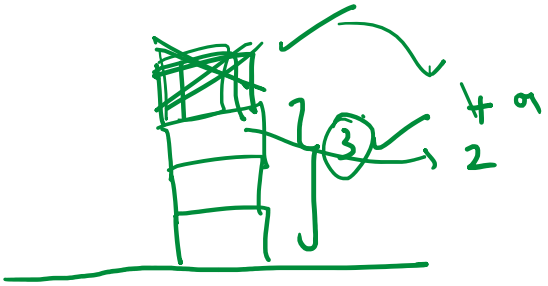
moves - height should be even

100 moves, 3 height →

97 moves

- 2

1 move ✓



1) moves >= height

2) (moves - height % 2) == 0 (should be even) ✓

C)

C. Spell Check

time limit per test: 1 second
memory limit per test: 256 megabytes

→ clear ✓

Timur likes his name. As a spelling of his name, he allows any permutation of the letters of the name. For example, the following strings are valid spellings of his name: `Timur`, `miurT`, `Trumi`, `mriTu`. Note that the correct spelling must have uppercased `T` and lowercased other letters.

Today he wrote string s of length n consisting only of uppercase or lowercase Latin letters. He asks you to check if s is the correct spelling of his name.

Input

The first line of the input contains an integer t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each test case contains an integer n ($1 \le n \le 10$) — the length of string s.

The second line of each test case contains a string s consisting of only uppercase or lowercase Latin characters.

→ $10^5, 10^6$

* permutation

SAT
STA
TSA
TSA

Timur

Timur
↓
5 letters

4 ✗ →
TOK ✓

Input

The first line of the input contains an integer t ($1 \le t \le 10^3$) — the number of test cases. → $10^5, 10^6$

The first line of each test case contains an integer n ($1 \le n \le 10$) — the length of string s.

The second line of each test case contains a string s consisting of only uppercase or lowercase Latin characters.

Output

For each test case, output "YES" (without quotes) if s satisfies the condition, and "NO" (without quotes) otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

Example

input Copy

```
10
5
Timur
5
miurT
5
Trumi
5
mriTu
5
timur
4
Timr
6
Timuur
10
codeforces
10
TimurTimur
5
TIMUR
```

output Copy

```
YES
YES
YES
YES
NO
NO
NO
NO
NO
NO
```

3 - letters

STA
TSA
AST
ATS
⋮

TSA
ATS

Timur →
miurT
Trumi
* T should be

timur ✗
T

Length, T, i

Timur ✓

TIMUR → ✗
↳ LC

→ length → 5 ✓

i → ✗ ↑ NO
m → ✓
u → ✓
r → ✓

* Length → 5 ✓

Alphabetical → CAT → ACT
TCA → ACT
Yes
NO

↓
sort → A String

Ctt, sort ✓

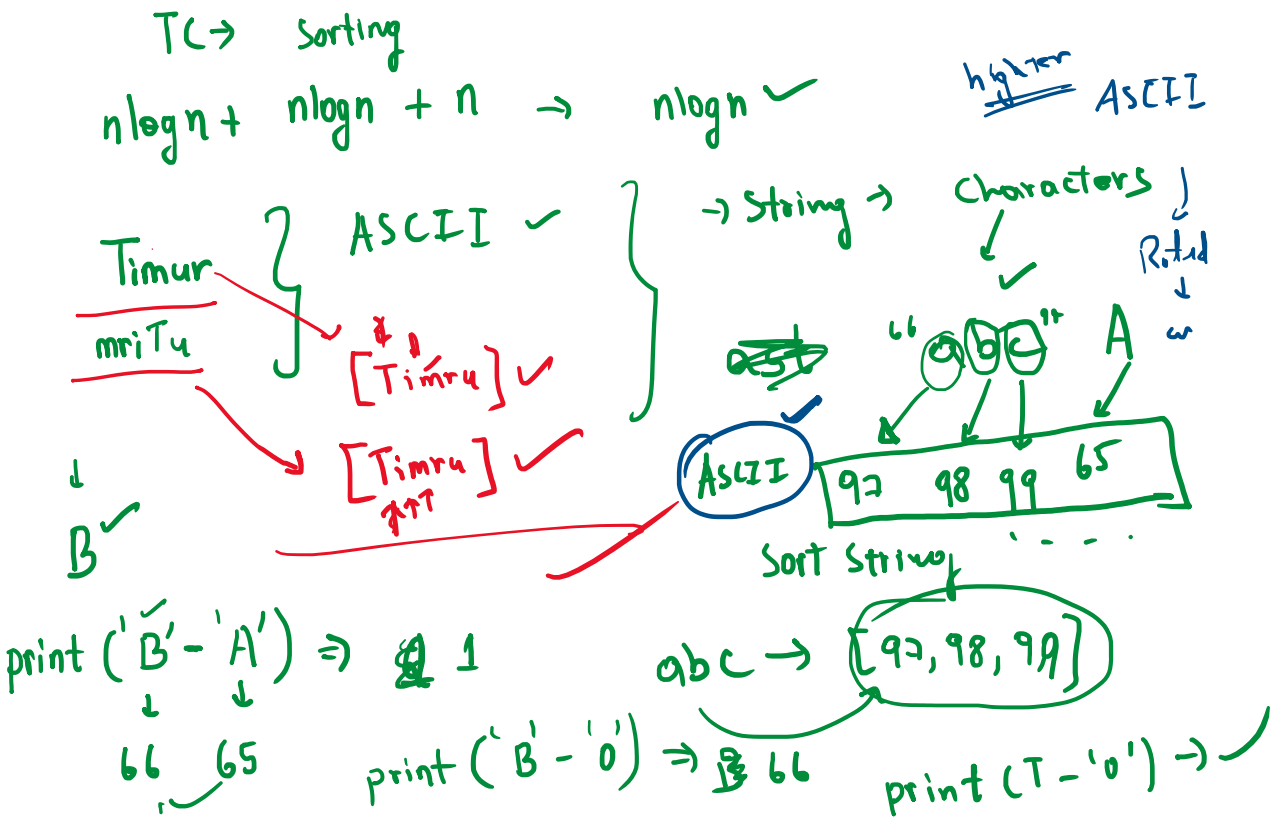
Pytho, sort

Java, → charArray → MTIMR → [M,T,I,M,R]

Timur

↓

[Alphabet]



D. Powering the Hero (easy version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes

This is an easy version of the problem. It differs from the hard one only by constraints on n and t.

There is a deck of n cards, each of which is characterized by its power. There are two types of cards:

- a hero card, the power of such a card is always equal to 0;
- a bonus card, the power of such a card is always positive.

You can do the following with the deck:

- take a card from the top of the deck;
- if this card is a bonus card, you can put it **on top** of your bonus deck or discard;
- if this card is a hero card, then the power of **the top** card from your bonus deck is added to his power (if it is not empty), after that the hero is added to your army, and the used bonus discards.

Your task is to use such actions to gather an army with the maximum possible total power.

Input

The first line of input data contains single integer t ($1 \le t \le 1000$) — the number of test cases in the test.

The first line of each test case contains one integer n ($1 \le n \le 5000$) — the number of cards in the deck.

The second line of each test case contains n integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le 10^9$) — card powers in top-down order.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

Output t numbers, each of which is the answer to the corresponding test case — the maximum possible total power of the army that can be achieved.

Output

Output t numbers, each of which is the answer to the corresponding test case — the maximum possible total power of the army that can be achieved.

Example

input

```
5
5
3 3 3 0 0
6
0 3 3 0 0 3
7
1 2 3 0 4 5 0
7
1 2 5 0 4 3 0
5
3 1 0 0 4
```

output

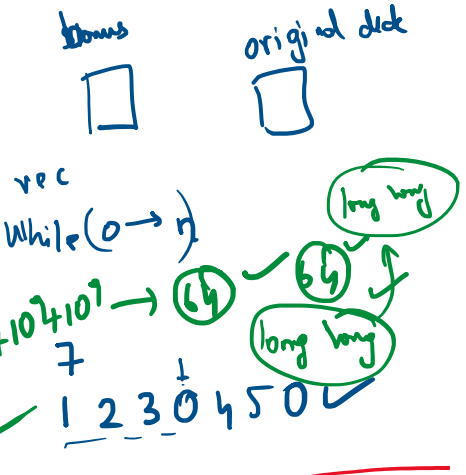
```
6
6
8
9
4
```

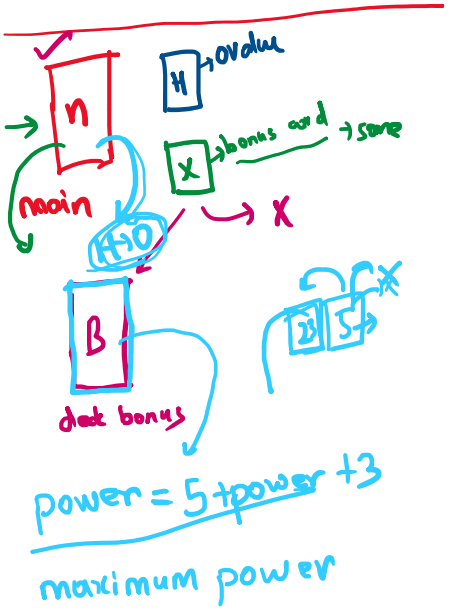
Note

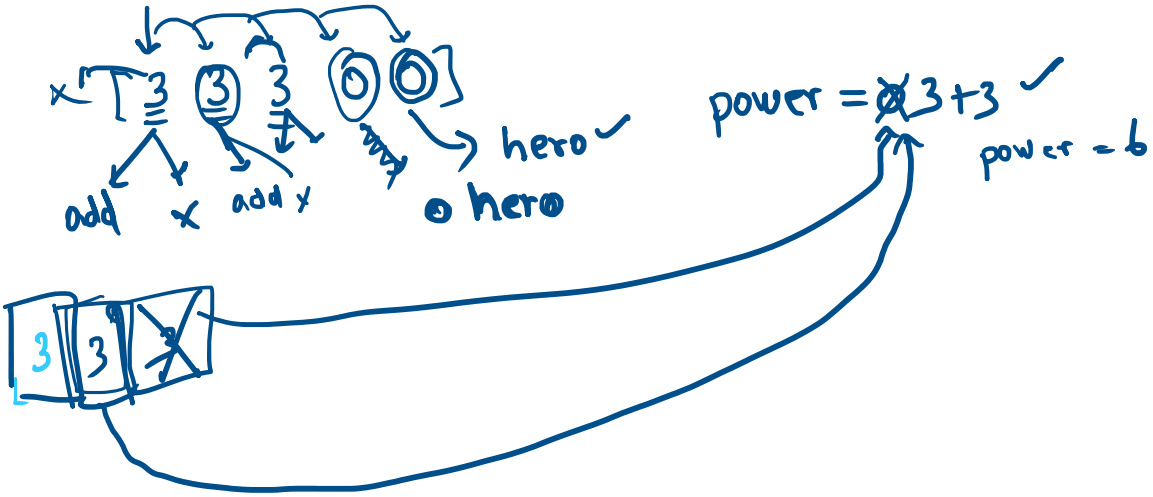
In the first sample, you can take bonuses 1 and 2. Both hero cards will receive 3 power. If you take all the bonuses, one of them will remain unused.

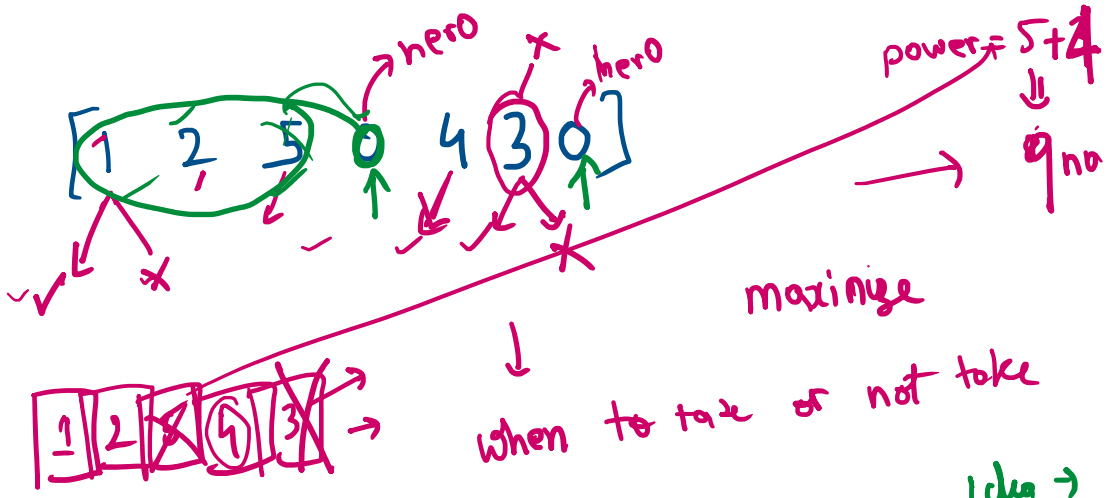
In the second sample, the hero's card on top of the deck cannot be powered up, and the rest can be powered up with 2 and 3 bonuses and get 6 total power.

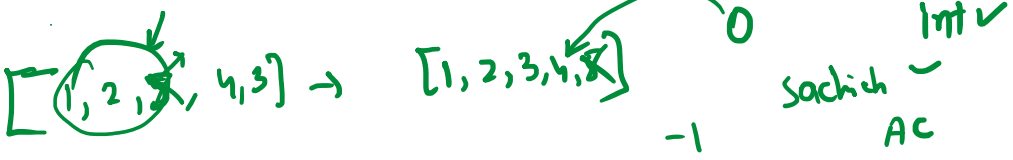
In the fourth sample, you can take bonuses 1, 2, 3, 5 and skip the bonus 6, then the hero 4 will be enhanced with a bonus 3 by 5, and the hero 7 with a bonus 5 by 4. $4 + 5 = 9$.



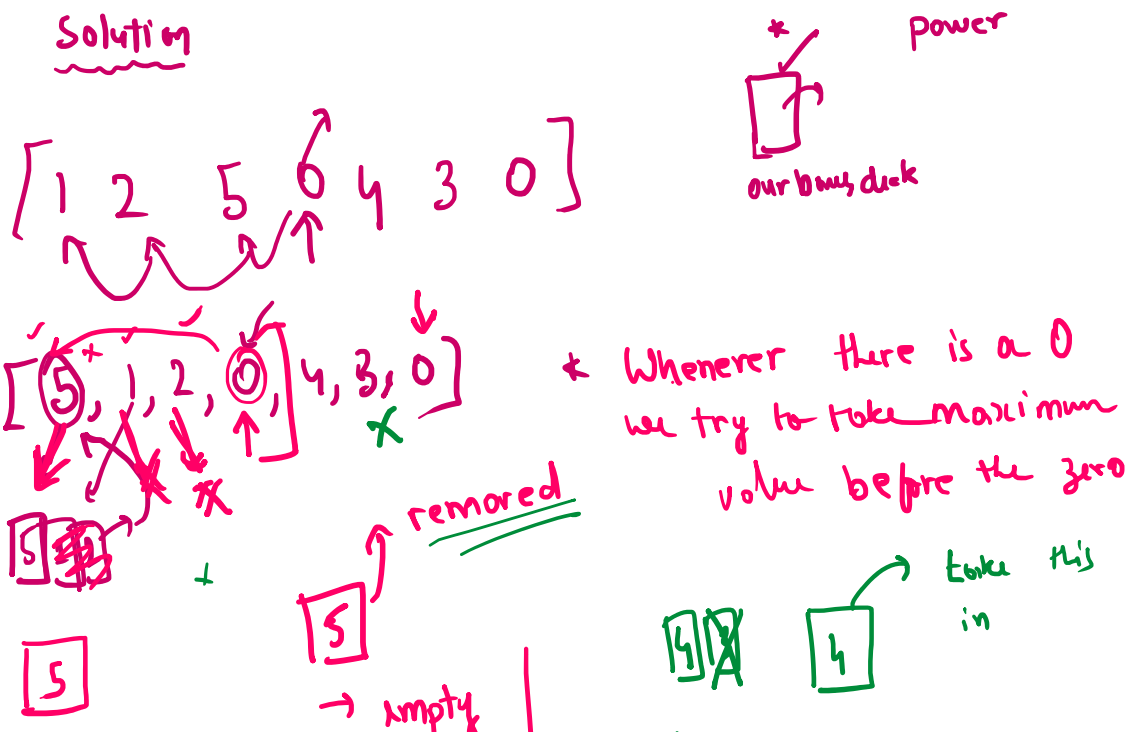



power = 3+3
power = 6
hero
hero
add
add


hero
hero
power = 5+4
no
maximize
when to take or not take


idea →
int
sachith
AC
-1

Solution


power
our boss duck
* Whenever there is a 0 we try to take maximum value before the zero
removed
→ empty
Take this in

[5]

→ empty

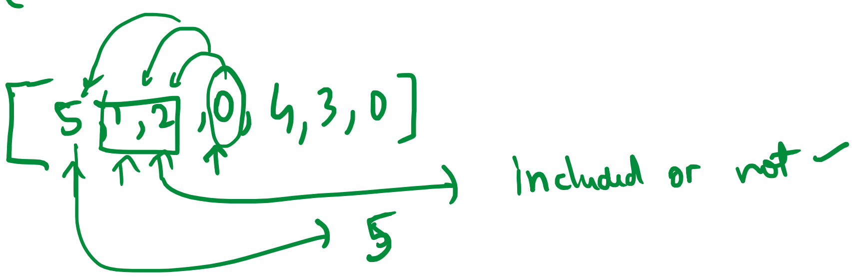
[4] A   [4]

kidea → intin

→ Is this clear

*optimal

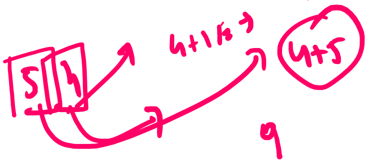
→ D   ('



* included → [5, 4, 3, 0, 1, 2, 0]

5+2 = 7, 5+4 = 9

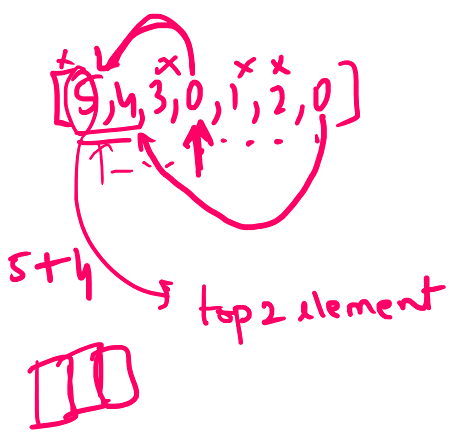
[5, 4, 3, 0, 1, 2, 0]



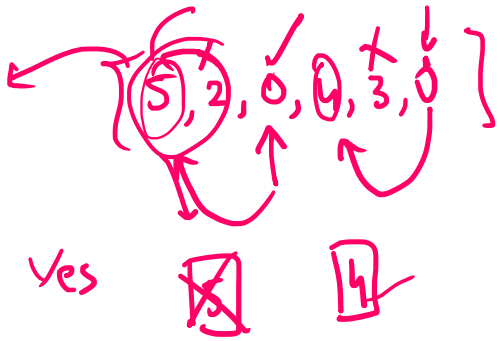
sort and do

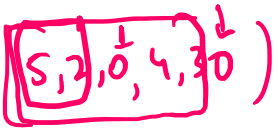
[0, 2, 3, 0] → Ans 3

[2, 3] → Ans (5)

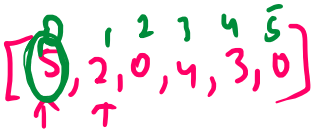


[5][4]



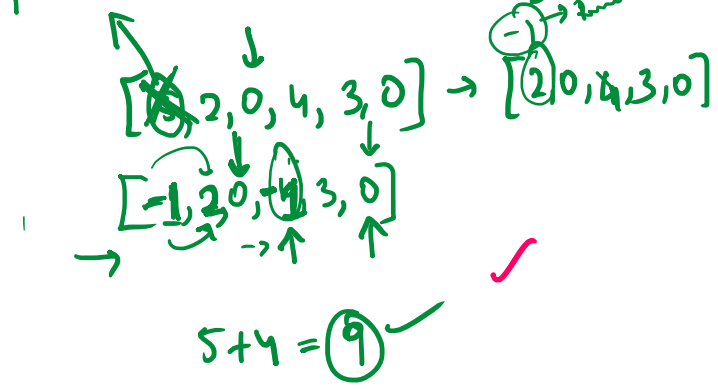


1) Problem D → ✓ , E



max variable = ~~0~~; ~~2~~, 4,

max index = ~~0~~; ~~1~~, 3;



5+4 = 9 ✓

```
for (i in n):
    if i > maxElement :
        max = i          → 5
        maxIndex = index → 0

    if i == 0
        Ans += maxValue
        array[maxIndex] = -1
    index = k = 0, maxValue = 0
    while k < i
        maxElement
        maxIndex
```

Brute force ✓ $O(n^2)$ → D ✗

E

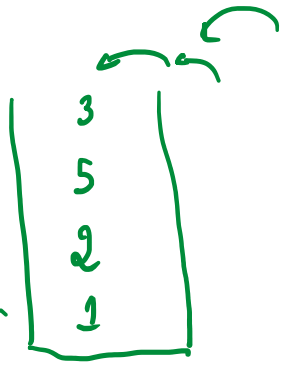
→ $10^5 \times 10^5 \Rightarrow 10^{10} >>> 10^9$ ✗

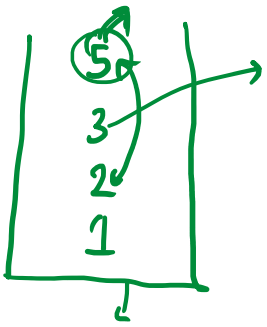
New concept → Picking max K element before

→ DS = Priority Queue → $O(n \log n)$

$O(n^2)$ ↗

→ DS = priority queue / $O(n^2)$



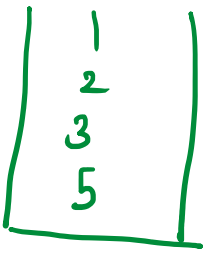


advantage

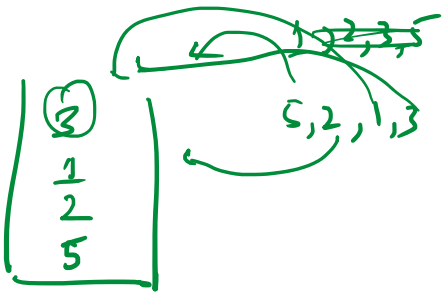
top of que the max is max in

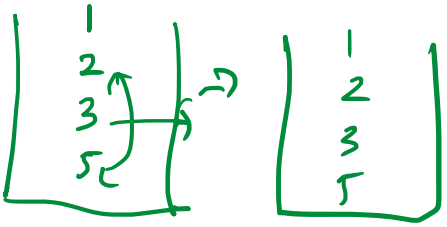
Always get maximum element in O(1)

maxHeap / minHeap



← increasing order

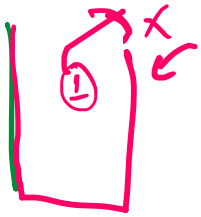




how is it used vs why not sorting

[0,2,3,0] [2,3] 2+3

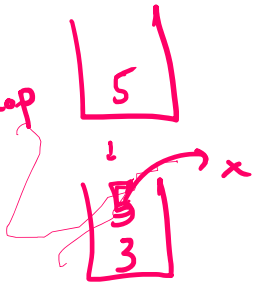
3 Ans



for (i → n):

[5,3,0,4,3,0]

maxheap



if (i == 0):

if (pq is not empty):

pow = pq.peek()

else:

pq.add(element)

O(nlogn)

10^5 TLE

else :

pq.add(element)

4+5=9