# Greedy Algorithms 2

- Priyansh Agarwal

# Fractional Knapsack Problem

Given two arrays named value and weight of size N each, where value[i] represents the value you get if you choose $i^{th}$ item, and weight[i] represents the weight of the $i^{th}$ item.

You want to maximize the value such that the sum of weights of the chosen items is <= W. You are allowed to choose a fraction of an item.

one time can be chosen once

val $\rightarrow$ $\boxed{10}$ $\quad$ 12 $\quad$ 14 $\quad$ $\boxed{8}$ $\quad$ 6

weight $\rightarrow$ 9 $\quad$ $\boxed{18}$ $\quad$ 4 $\quad$ 10 $\quad$ 12

capacity $\rightarrow$ $\boxed{20}$

(w)

$\underline{v}$ / w $\qquad$ $12/2 = 6$ / v

$\underline{19} \rightarrow 18 \rightarrow \boxed{1/8} \times 12 = \frac{3}{2} = \underline{1.5}$

(19.5 , 20)

val → 10 12 14 8 6

weight → 9 8 4 10 12

1kg → $\frac{10}{9}$ $\frac{12}{8}$ $\frac{14}{4}$ $\frac{8}{10}$ $\frac{6}{12}$

$\boxed{\frac{12}{8}}$

→ $\frac{12}{8}$, $\frac{12}{8}$, $\frac{12}{8}$ -- 8 times

| | | | | | |
|---|---|---|---|---|---|
| 10 | 12 | 14 | 8 | 6 | |
| 9 | 8 | 4 | 10 | 12 | |

$\frac{10}{9}, \frac{10}{9}, \frac{10}{9}$ ..... 9 times

$\frac{12}{8}, \frac{12}{8}$ ...... 8 times

$\boxed{val / weight}$ ↑

| | | |
|---|---|---|
| val | 10 | 12 |
| weight | 5 | 2 |

$\frac{10}{5}$ | $\frac{10}{5}$ $\frac{10}{5}$ $\frac{10}{5}$ $\frac{10}{5}$ $\frac{12}{2}$ $\frac{12}{2}$

W = (6) capacity

val $\rightarrow$ 10 12 14 8 6

weight $\rightarrow$ 9 18 4 10 12

Capacity = 15

14, 4 12, 8 10, 9 8, 10 6, 12

ans = 14 + 12 + (3/9) × 10

Capacity = 15 − 4 = 11 − 8 = 3 − 3 = 0

# Fractional Knapsack Problem

Which item should we choose first? The one with maximum ratio of value[i]/weight[i] because that would give us the maximum value of a unit weight.

Which item should we choose next? Same idea

Sort all the items in decreasing order of value[i]/weight[i] and keep taking the maximum amount of weight possible from it ensuring total weight <= W

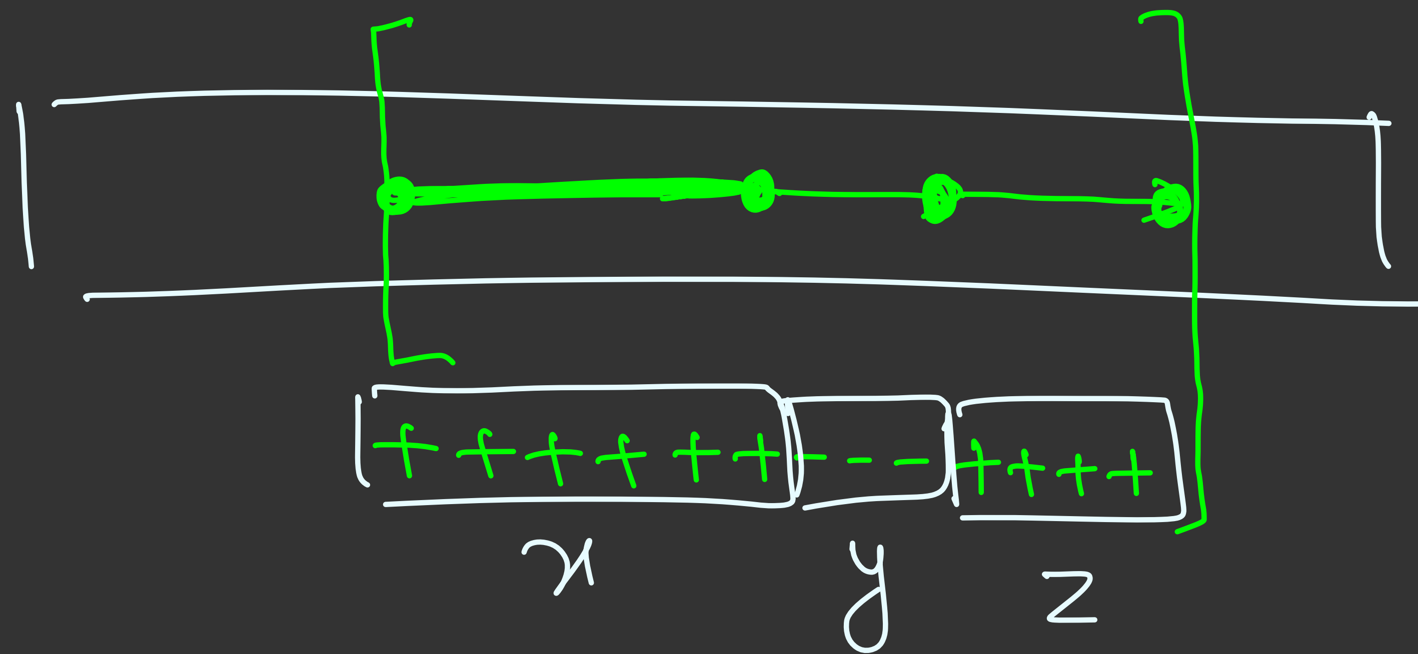
# Kadane’s Algorithm

Given an array of N integers, find the maximum possible sum of a subarray.

Example:

A = [1, 2, -9, 2, 3, -1, 4]

Best subarray = [2, 3, -1, 4], Answer = 8




if ( $x + y < 0$ )

$$0 \quad 0 \quad [\,+ \; + \; + \; + \; + \; - \; - \; - \; - \; - \; + \; + \; + \; +\,] \quad 0 \quad 0$$

$x \quad < 0 \quad y \qquad z$

$(x+y+z)$ is my answer

when $\boxed{x+y>0}$

0 0 0 + | + + + + − − − + + + | 0 0 0

+ve

any start from a positive element

$x \quad y \quad z$

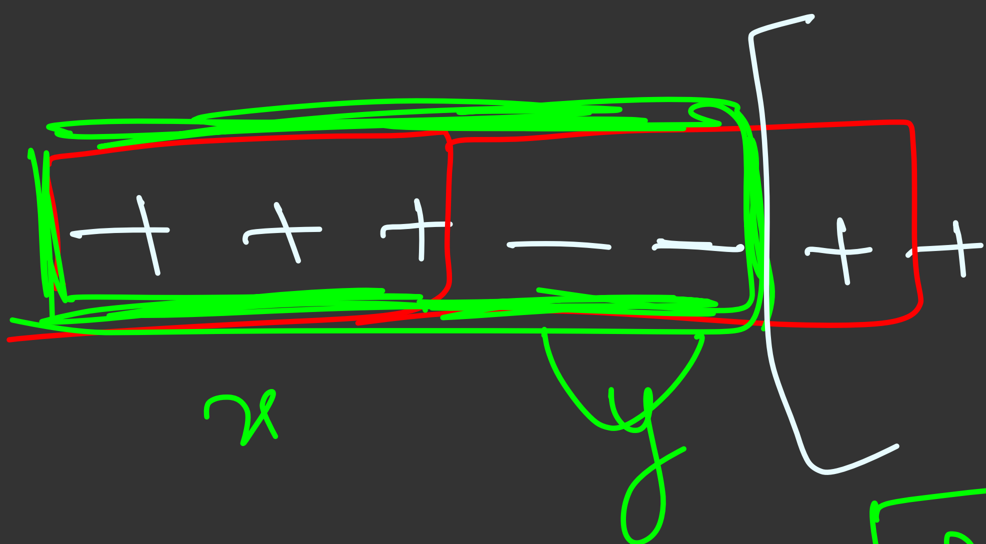
− − | + + | − − + | + + − − − − +

1 2 | −5 −6 100

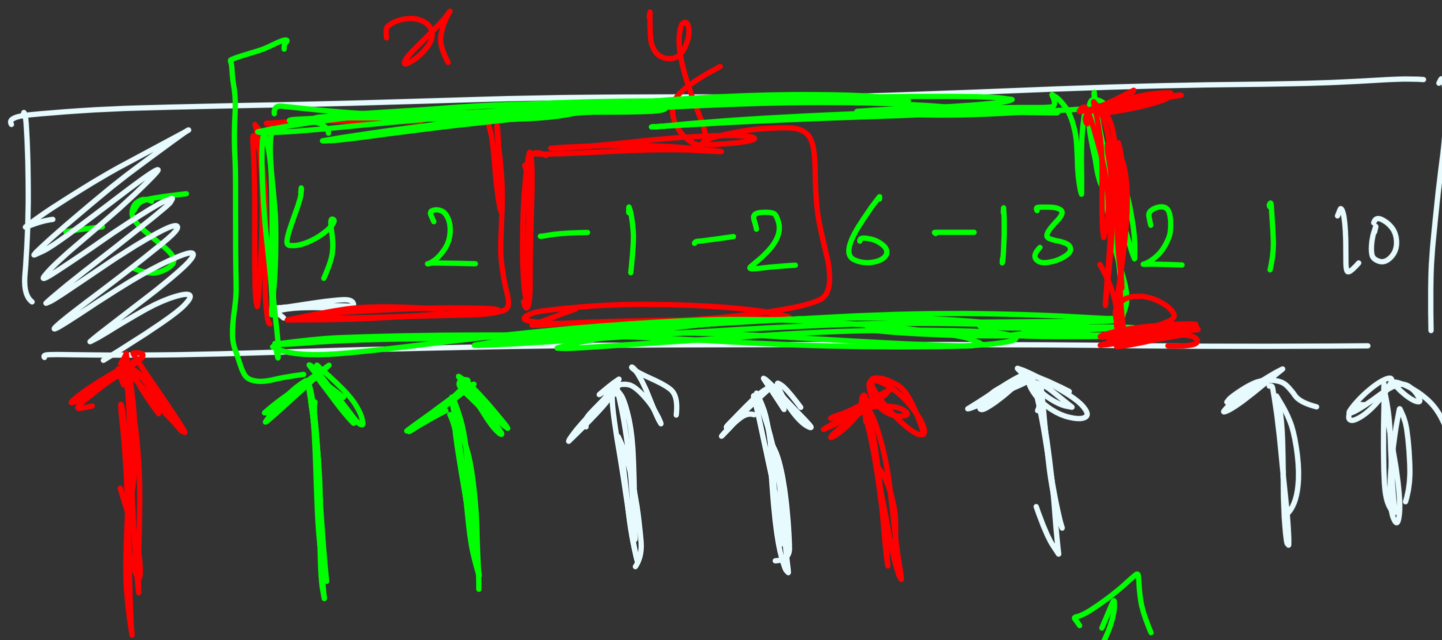
$x + y < 0$

① ans starts from the positive

② ans cannot end on negative element

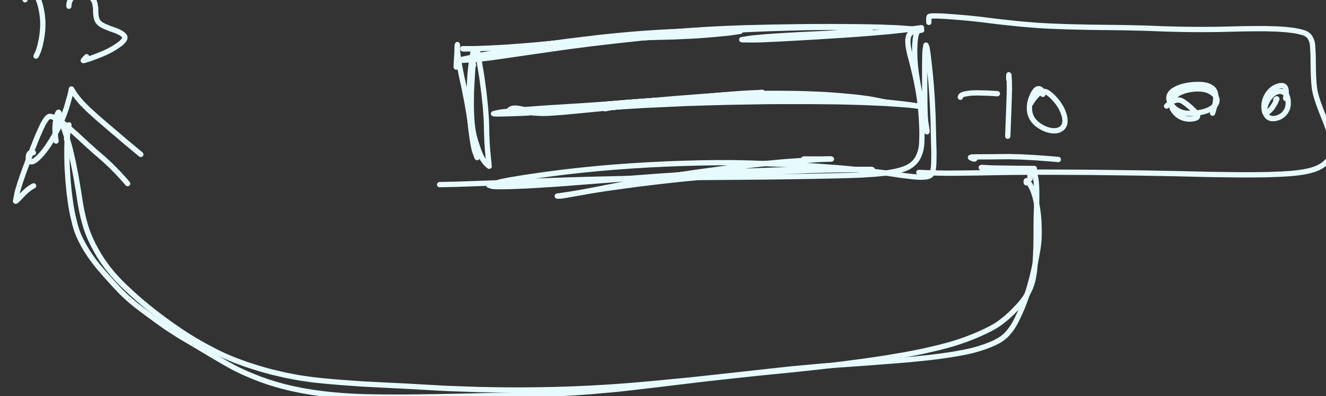



$\boxed{x+y<0}$




max_ans = 13

Sum = 13



```
{
int ans = 0, sum = 0

for (int i=0; i<n; i++) {
    sum += arr[i]
    if (sum < 0)
        sum = 0
    ans = max(ans, sum)
}
```

O(n)

$\boxed{\text{empty subarray} \to 0}$

$a_1 \quad a_2 \quad a_3 \quad - - - - \quad a_n$

min sum subarray

10 $\quad$ -5

$-a_1 \quad -a_2 \quad -a_3 \quad - - - \quad a_n$

max sum subarray

return $-$ans;

# Kadane's Algorithm

Is it every feasible to include a negative integer in the subarray?

When should we include a negative integer?

How can we simulate the process greedily?

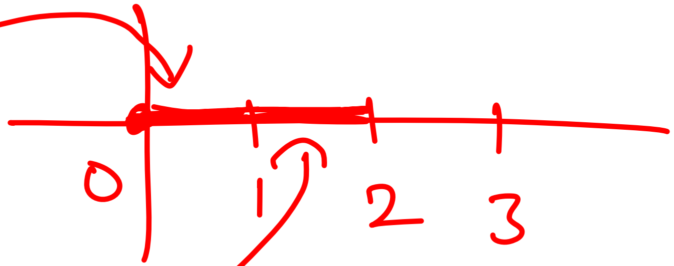
Try to include elements in current subarray until the subarray sum > 0

# Job Sequencing Problem

Given an array of jobs where every job has a deadline and associated profit, if the job is finished before the deadline. It is also given that every job takes a single unit of time. Maximize the profit if only 1 job can be done at any time.

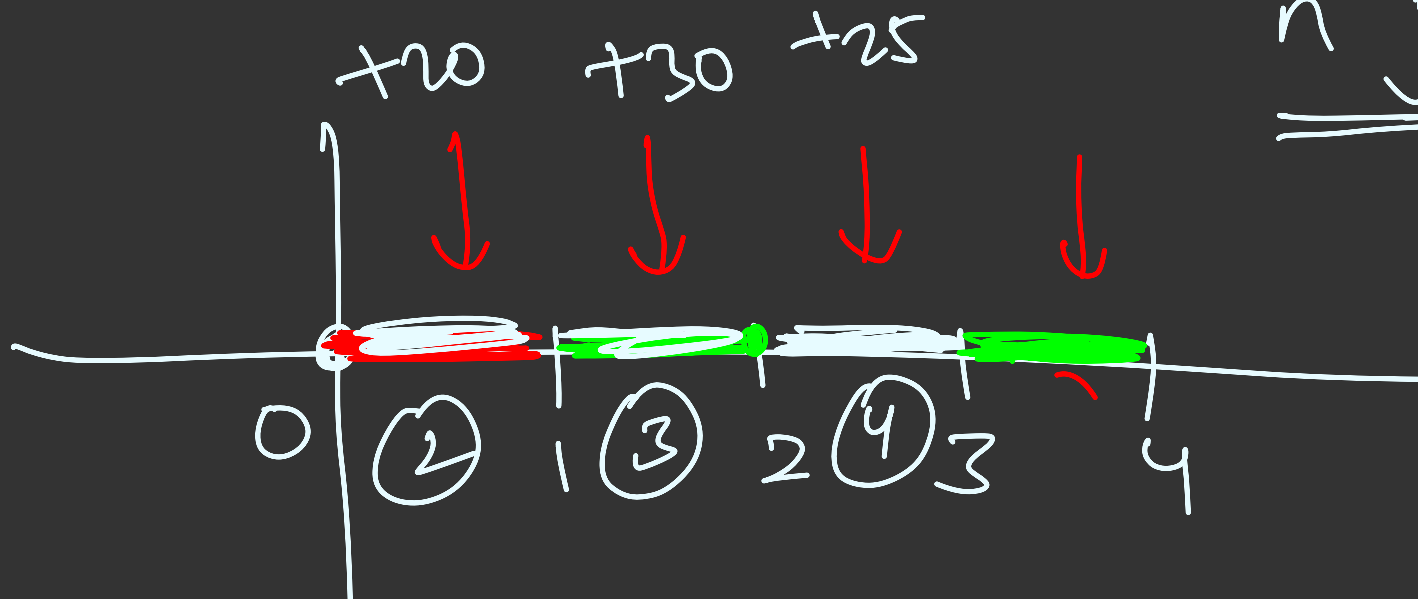
Example: 10

Time = [1, 2, 2], Profit = [10, 20, 30]




The best combination includes picking up 2nd and 3rd one -> Profit = 50

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| deadline | 1 | 2 | 2 | 4 | $\rightarrow$ 2n |
| profit | 10 | 20 | 30 | 25 | |

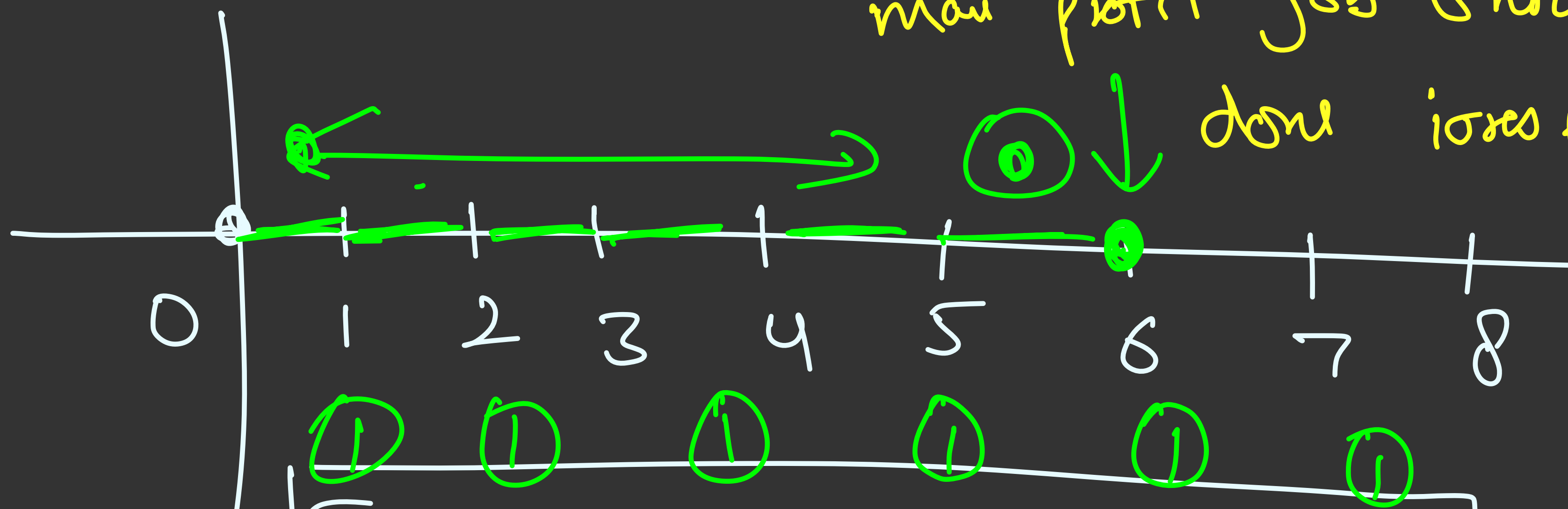



n jobs

job $d_1$ $d_2$ $d_3$ $d_4$ ..... $d_n$

$p_1$ $p_2$ $p_3$ $p_4$ ..... $p_n$

max profit job should be done irrespective of its deadline




5, 10, 20, 30, 40, 50, $\boxed{70}$

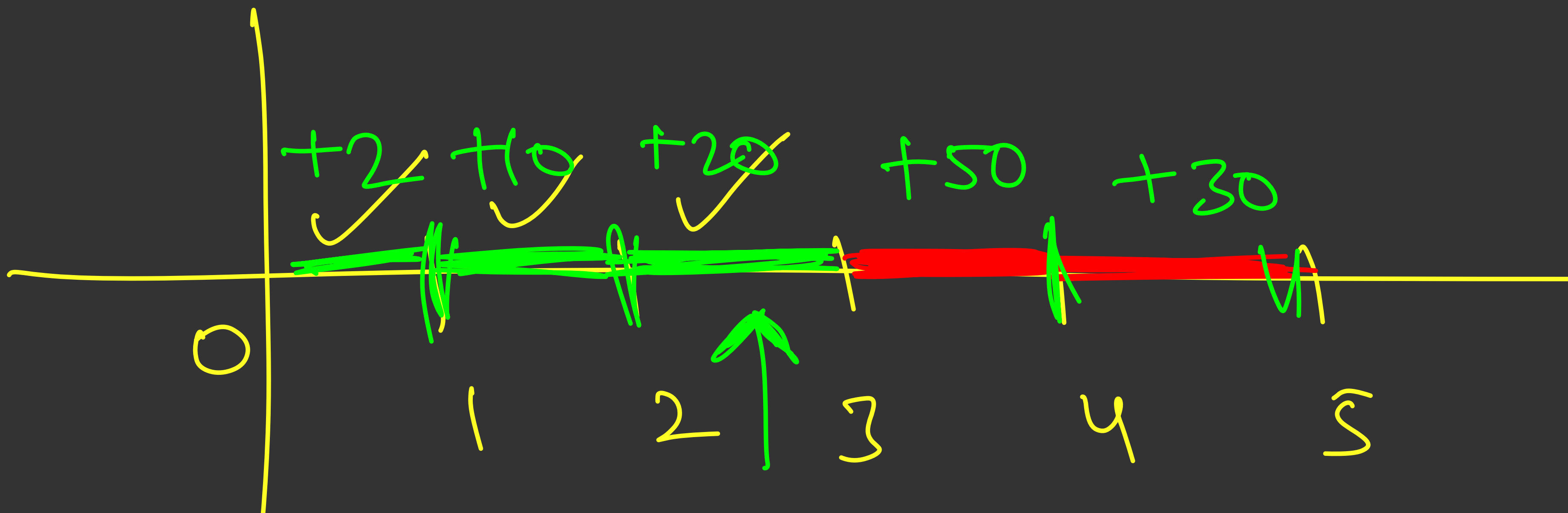
6

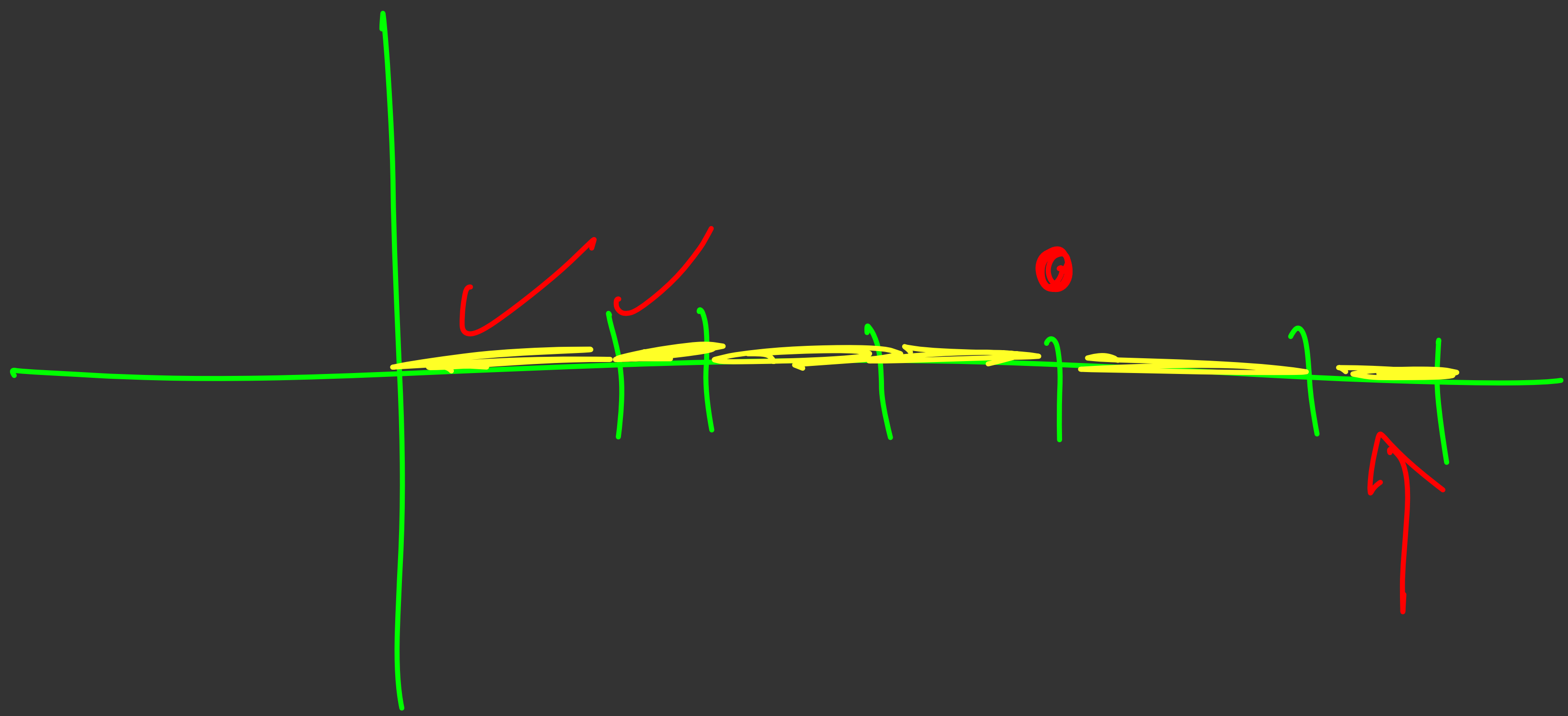
max profit job $\boxed{M}$

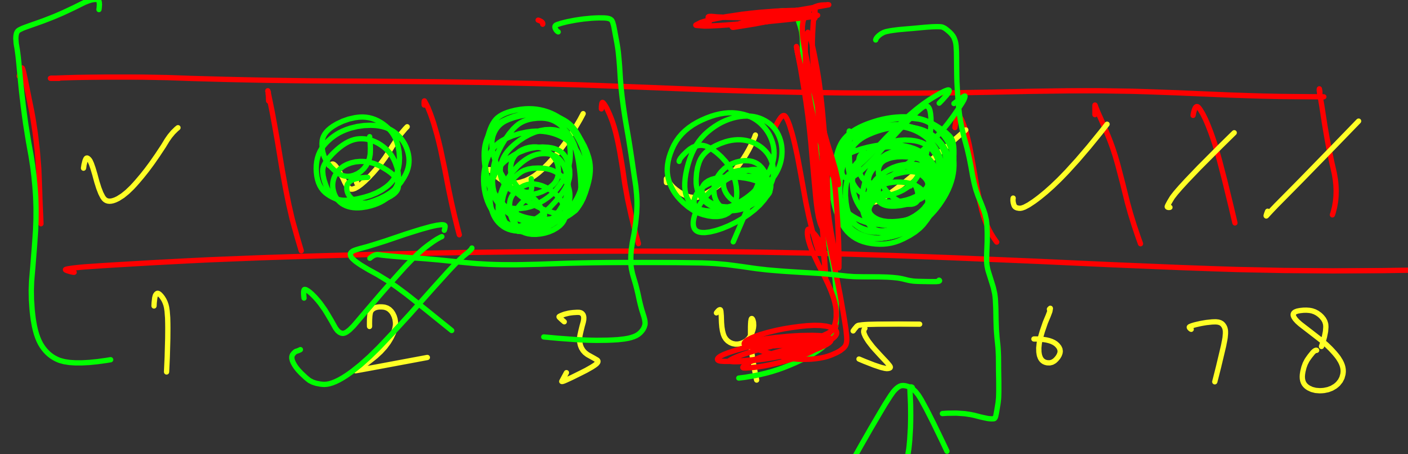

D slots

max profit $\rightarrow$ last available slot

| | | | | | |
|---|---|---|---|---|---|
| deadline → | 5 | 5 | 5 | 4 | 1 |
| profit → | 10 | 20 | 30 | 50 | 2 |
| | 1 | 2 | 3 | 4 | 5 |








2n slots

job

① ② ③ ④ ②

n log n

O(n) time for every job

$O(n^2)$

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 |
| NA | A | NA | NA | NA | A |
| 1 | 2 | 3 | 4 | 5 | 6 |

(2, 6)

Set → { 1, 2, 3, 7, 9, 10 }

available slots

job's deadline = $\boxed{6}$

$\boxed{\log n}$

upperbound

$\boxed{O(n \log n)}$

deadlines → $\boxed{10^6}$ $10^9$ $10^8$ $10^7$

job → 100 200 10 20

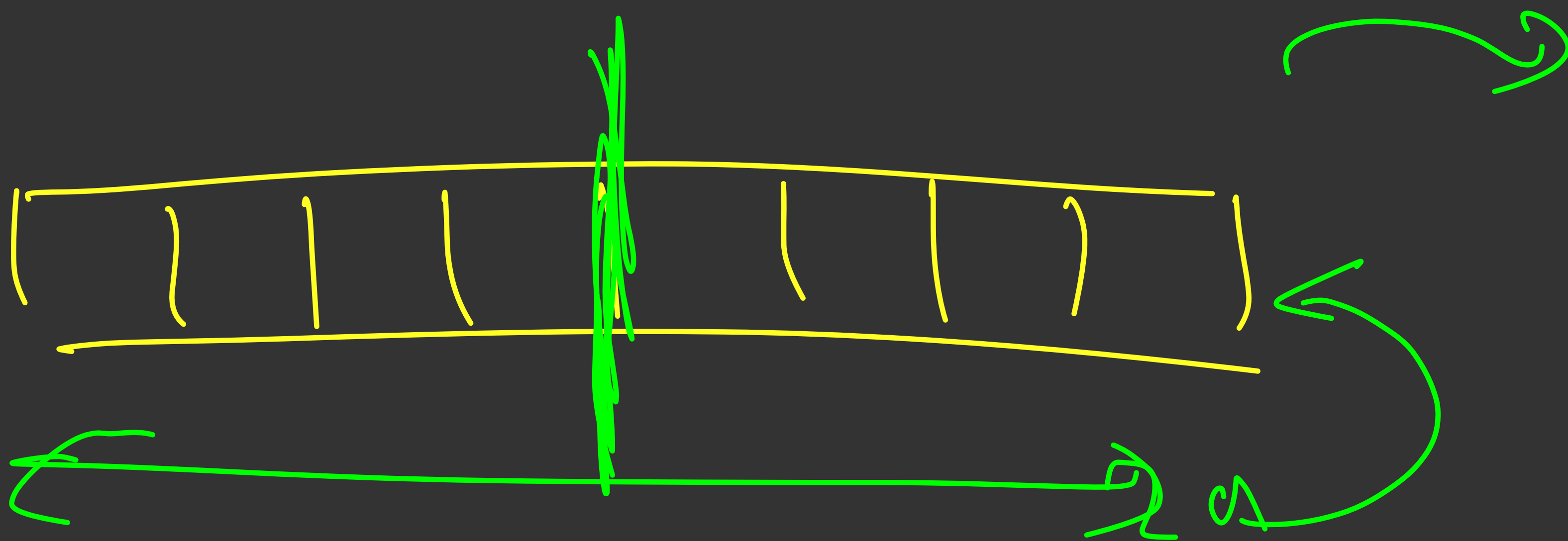
$\boxed{2n}$

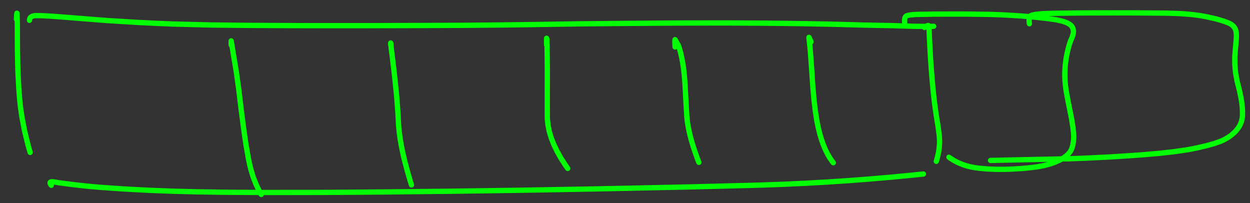
deadlines $\longrightarrow$ $10^6$ $10^7$ $10^8$ $10^9$

$n = 10^5$

$\boxed{2 \times 10^5}$

| | | | | |
|---|---|---|---|---|
| job → | ~~100~~ 8 | ~~200~~ 8 | ~~300~~ 8 | ~~400~~ 8 |
| profits → | 10 | 2 | 3 | 4 |





1 $\leq d$

2 $\geq d$

$n$ jobs

CP

safe side

Time spent

How much optimize

$10^9$

$10^8$

$\boxed{10 \times n}$

$\boxed{12 \times n}$

1. complete every job before deadline
2. complete " " after deadline
3. < start , end >

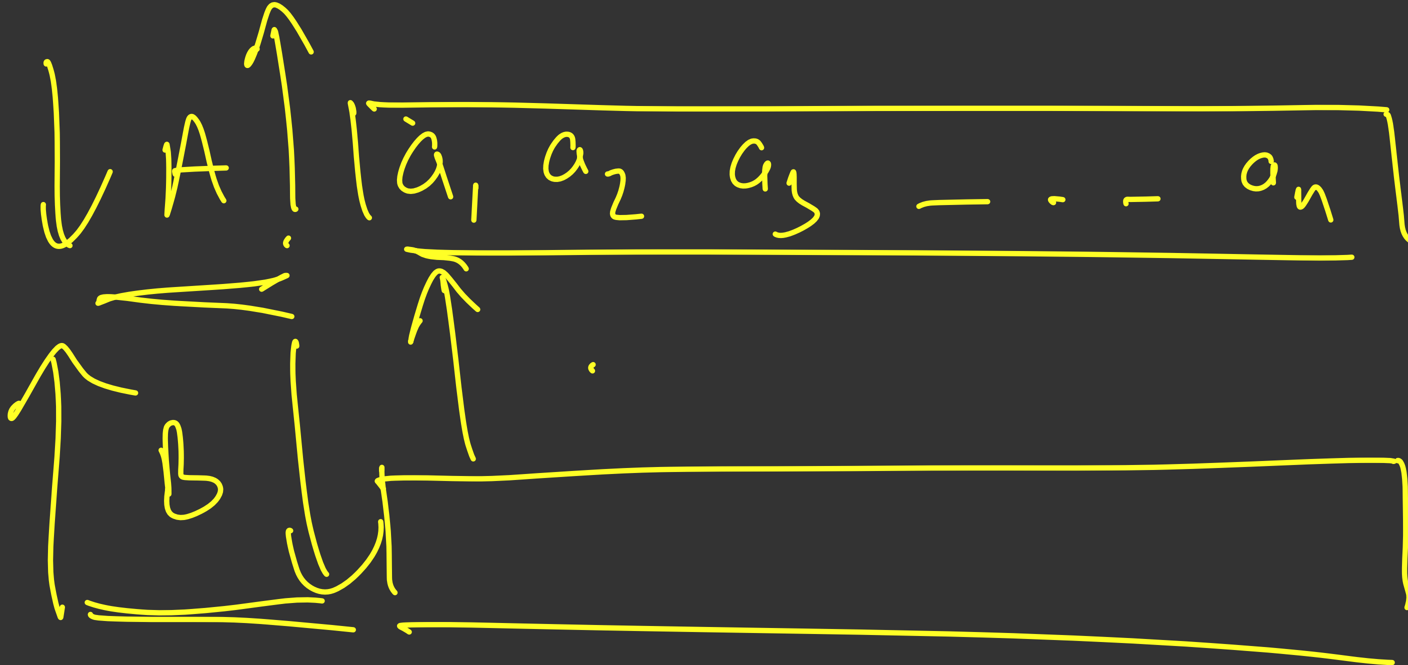
# Minimum Dot Product

Given 2 vectors A [a1, a2, a3, … vn] and B [b1, b2, b3, … bn], rearrange the elements in the vectors so that the dot product is minimized.

Example:

A = [-1, 3, -2], B = [-10, 1, 5] => [-1 * -10 + 3 * 1 + -2 * 5] = [10 + 3 + -10] = 3

A = [-2, -1, 3], B = [5, 1, -10] => [-2 * 5 + -1 * 1 + -10 * 3] = [-10 + -1 + -30] = -41




$a_1 \leq a_2 \ldots \leq a_n$

$b_1 \quad b_2 \quad \ldots \quad b_n$

$b_1 \leq b_2 \ldots \leq b_n$

A $\quad a_1 \le a_2 \le a_3 \ldots \le a_n$

B $\quad b_1 \le b_2 \le b_3 \ldots \le b_n$

1

A $\rightarrow$ $a_1 \quad a_2 \quad a_3 \quad a_j \ldots a_{n-1} \quad a_n$

B $\rightarrow$ $b_j \qquad\qquad b_n$

2

A $\rightarrow$ $a_1 \quad a_2 \quad a_3 \ldots a_j \ldots a_{n-1} \quad a_n$

B $\rightarrow$ $b_n \qquad\qquad b_j$

① − ②

$$(a_1 \times b_i + a_j + b_n) - (a_1 \times b_n + a_j \times b_i)$$

$$\boxed{a_1 \times (b_i - b_n)} - \boxed{a_j \times (b_i - b_n)}$$

$$\downarrow$$

$$\boxed{a_1 - a_j} \times \boxed{b_i - b_n}$$

$$\leq 0 \quad \times \quad \leq 0$$

$$\geq 0$$

$$① - ② > 0$$

$$① > ②$$

# Minimum Dot Product

Can we try out for 2 elements? Assuming we have 2 sorted vectors:

$a_1 \leq a_2$

$b_2 \geq b_1$

A = [a1, a2] | B = [b1, b2]

X = a1 * b1 + a2 * b2 | Y = a1 * b2 + a2 * b1

X = Y + a1 * b1 - a2 * b2 + a2 * b2 - a2 * b1

X = Y + (b2 - b1) * (a1 - a2)

$\underline{\underline{X \leq Y}}$

Since b2 - b1 >= 0 & a1 - a2 <= 0, this means that X is <= Y

# Minimum Dot Product

Shouldn't we try to multiple bigger numbers in A with smaller numbers in B and smaller numbers in A with bigger numbers in B to get the most optimal answer?

But how can we prove that this actually works?

# Minimum Dot Product

$A \;=\; a_1 \;<=\; a_2 \;<=\; a_3 \;\ldots\ldots\; <=\; a_{n-2} \;<=\; a_{n-1} \;<=\; a_n$

$B \;=\; b_1 \;>=\; b_2 \;>=\; b_3 \;\ldots\ldots\; >=\; b_{n-2} \;>=\; b_{n-1} \;>=\; b_n$

Optimal Ans

What if we swap 2 adjacent elements?

$A \;=\; a_1 \;\; a_2 \;\; a_3 \;\ldots\ldots\; a_k \;\; a_{k+1} \;\ldots\ldots\; a_{n-2} \;\; a_{n-1} \;\; a_n$

$B \;=\; b_1 \;\; b_2 \;\; b_3 \;\ldots\ldots\; b_{k+1} \;\; b_k \;\ldots\ldots\; a_{n-2} \;\; a_{n-1} \;\; a_n$

What is the change in the total sum => $[a_k - a_{k+1}] * [b_{k+1} - b_k]$ which is $>= 0$

# Maximum Perimeter Triangle $x, y, z$

Given a list of N positive integers, pick up 3 of them which form a valid triangle with the maximum perimeter possible. $x, y, z$

$x + y > z$

$z + x > y$

$z + y > x$

Example:

A = [2, 3, 15, 5, 1, 7]

Best choice -> [3, 5, 7] forms a valid triangle with perimeter = 15

$\boxed{x \leq y \leq z}$ $\boxed{x + y > z}$

$a, b, c$

$$a + b > c$$
$$b + c > a$$
$$a + c > b$$

c is the highest among $(a, b, c)$

$$\boxed{a + b > c}$$

sum of 2 smaller sides > bigger side

point →

| f | | T | | f | f | f | f |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 5 | 6 | 10 | | | |

a b c

a + b < c

3 4 10

T T T T f f f f f T T T T f f f f

$a \quad , \quad b \quad , \quad c$

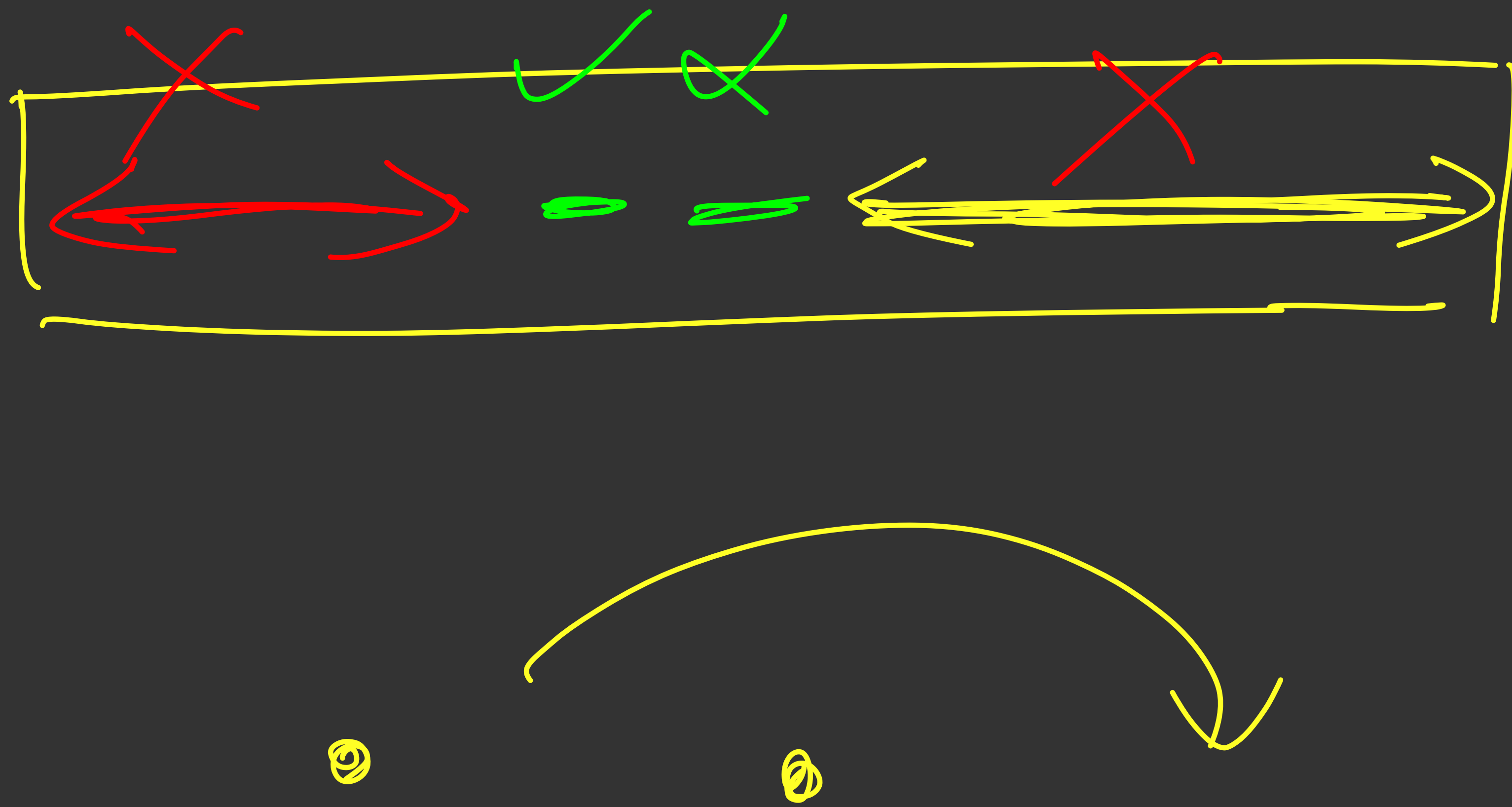
$$\boxed{a + b < c}$$

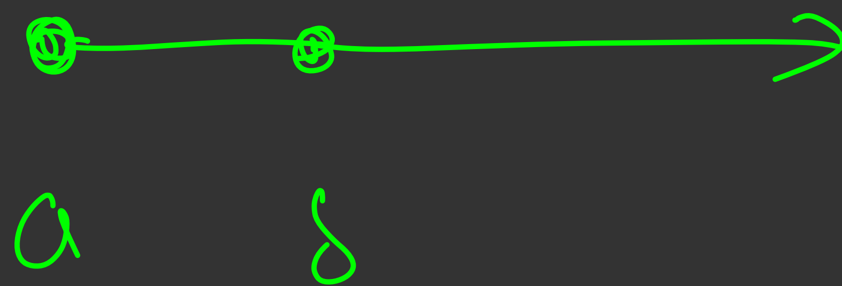
9, 10, 20

$9 + 10 \leq 20$

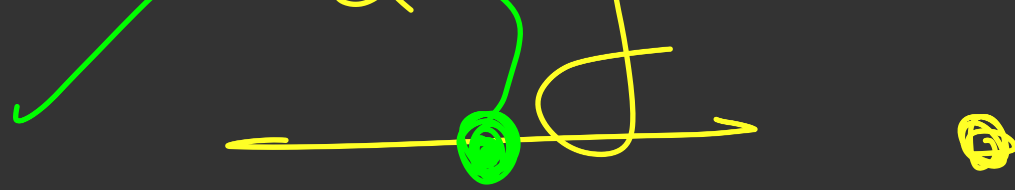
$9 + 10 \leq 20$

$9 + 12 > 20$




a
δ

No. of valid △s in the array



$a_1 \le a_2$ - - - - $\le a_n$

$\boxed{C}$

$O(n^2 \log n) \longrightarrow O(n^2)$

$$\boxed{a_1 \quad a_2 \quad a_3 \; - - - \; a_n}$$

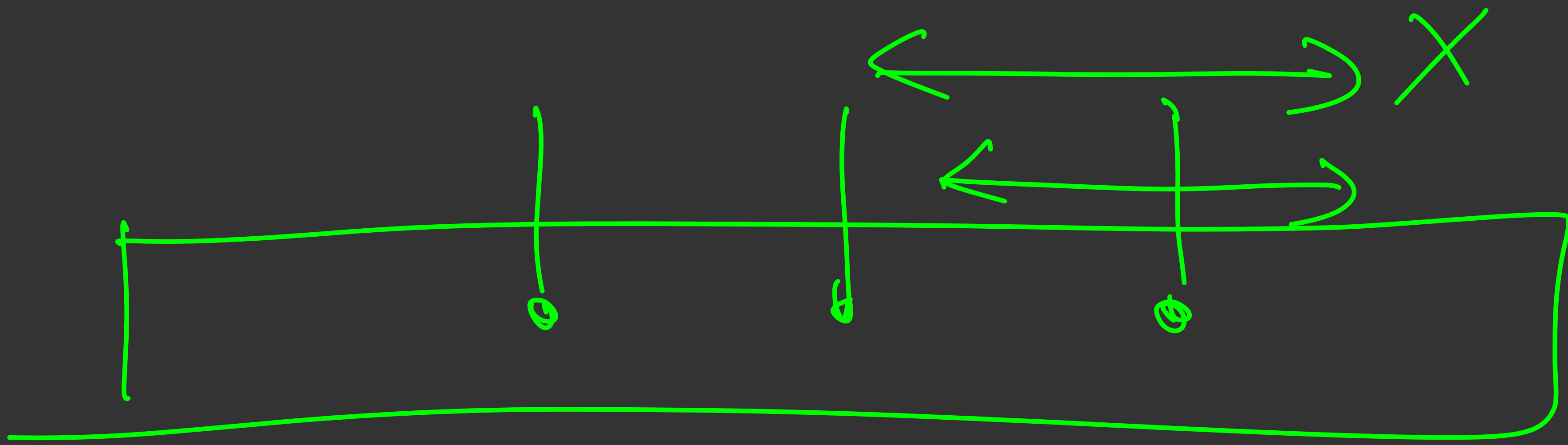
$$a_1 \leq a_2 \leq a_3 - - - \leq a_n$$

$$a_i \quad , \quad a_j \quad , \longrightarrow \boxed{a_k}$$

$$\boxed{k > j > i}$$

$$a_i + a_j = x$$

$$a_k < x$$



$$\underbrace{a_i + a_j}_{=x}$$

$$a_i \le a_k < X$$

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

# Maximum Perimeter Triangle

What if we fix the longest side of the triangle?

Then the other 2 sides must add up to a number >= longest side

If C is the longest side, A + B must be >= C

What should be the optimal values for A and B?

How can we find the best triangle for every possible C?

# Doubled Array Problem

There is an array with N elements, each of the elements is doubled and appended into the array. After that the array elements are jumbled up. Find the original elements in any order.

Example:

A = [2, 9, 4, 10]   gets transformed to [8, 10, 9, 4, 2, 18, 4, 20]

A = [-1, 2, 4] gets transformed to [-2, -1, 2, 4, 4, 8]

# Doubled Array Problem

Can we separate out the negative and positive numbers?

What about the smallest positive number and biggest negative number?

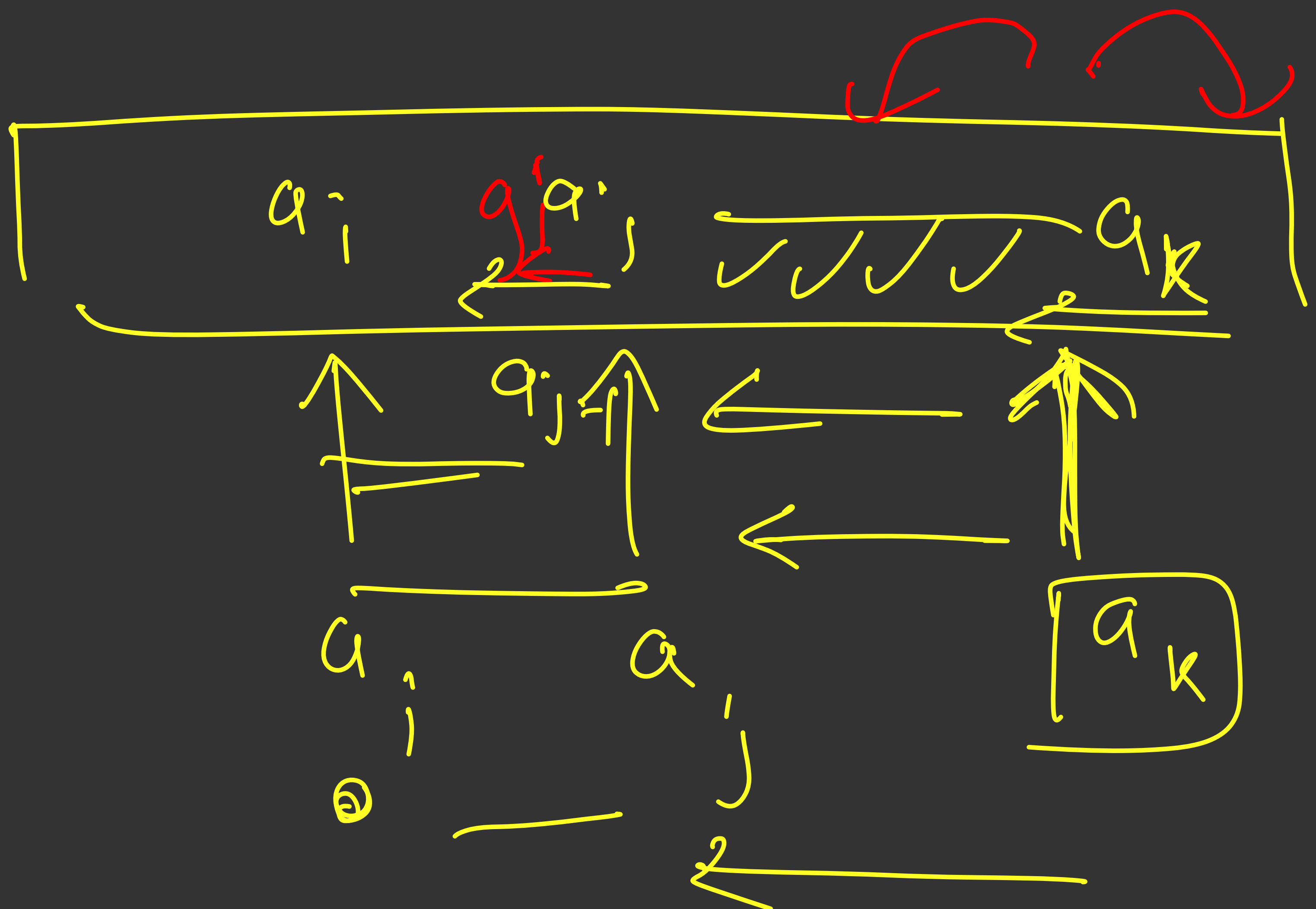
How to simulate this process to get the answer at every step?

# Advantages of Greedy Solutions

- Being able to think about a greedy solution prevents us from trying out all the possibilities, hence saving a lot of runtime
- Greedy solutions are usually very easy to implement, mostly involve some form of sorting.

# Disadvantages of Greedy Solutions

- Greedy solutions lack global awareness
- It is difficult to reason whether or not a greedy solution is correct
- Trying to disprove a greedy solution usually involves lot of hit and trial
- Doesn’t try out all the possibilities



$O(n) \cdot O(n)$ $\quad a_i + a_j > a_k$

$\underline{O(n^2)}$