Max value of equation

1499. Max Value of Equation

Hard 1196 47 Add to List Share

You are given an array `points` containing the coordinates of points on a 2D plane, sorted by the x-values, where `points[i] = [xi, yi]` such that `xi < xj` for all `1 <= i < j <= points.length`. You are also given an integer `k`.

Return *the maximum value of the equation* `yi + yj + |xi - xj|` where `|xi - xj| <= k` and `1 <= i < j <= points.length`.

It is guaranteed that there exists at least one pair of points that satisfy the constraint `|xi - xj| <= k`.

Example 1:

```
Input: points = [[1,3],[2,0],[5,10],[6,-10]], k = 1
Output: 4
Explanation: The first two points satisfy the condition |xi - xj| <= 1 and if we calculate the equation we get 3 + 0 + |1 - 2| = 4. Third and fourth points also satisfy the condition and give a value of 10 + -10 + |5 - 6| = 1.
No other pairs satisfy the condition, so we return the max of 4 and 1.
```

Given: arr [[1, 3], [2, 0], [5, 10], [6, -10]]

$k = 1$

where value of x coordinates is sorted

[1, 2, 5, 6]

$x_1\ x_2\ x_3\ x_4$

$x_i < x_j$ & $1 \le i < j \le$ length

Aim: Maximise $y_i + y_j + |x_i - x_j|$

where $|x_i < x_j| \le k$, $i < j$

Given: Array which is sorted according to x-coordinates

$T = y_i + y_j + |x_i - x_j|$

we know $j > i \Rightarrow x[j] > x[i]$

$|A - B| = -(A - B)$ if $B > A$ i.e $-(x_i - x_j)$

↳ abs

↓ Always positive

Always positive → ↓ (1)

Using ① in T

$$y_i + y_j - (x_i - x_j)$$

$$y_i + y_j - x_i + x_j$$

$$(x_j + y_j) + (y_i - x_i)$$

If we know $x_j + y_j$ then we need to find max value of

$$(y_i - x_i)$$

⇒ we need to calculate $y_i - x_i$ and store it in priority queue along with x_i; → we need priority queue with pairs

we are using maxheap because on top we will have max value

⇒ Iterate over all element if if the top x in priority queue

i.e (pq.top().second) and current x have difference greater then k then remove it from pq since it is not useful (points[i][0] − second) > k

pq.pop()

Calculate the next equation

q.top().first + points[i][0] + points[i][1]

$y_i - x_i + y_j + x_j$

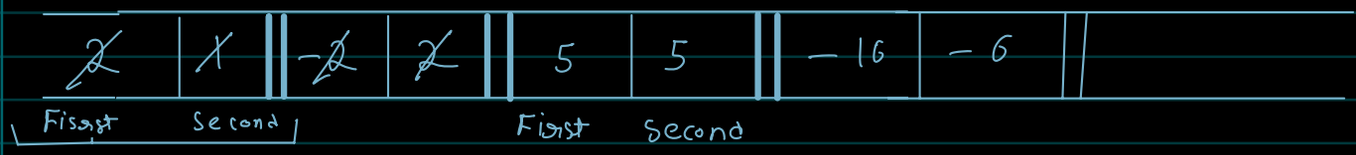
store it max

q.top().first + points[i][0] + points[i][1]

Dry Run

points = [[1, 3], [2, 0], [5, 10], [6, -10]]
(0) (1) (2) (3)

k = 1



n = 4

Ans = ~~Int_min~~, 4

for(i : 0 → n) // n = 4

At i = 0

while (!q.empty()) false

if (!q.empty()) False

q.push (points[0][1] - points[0][0], points[0][0])
// 3 - 1
// 2 , 1

At i = 1

while (!q.empty()) True && point[1][0] - q.top().second >= 1

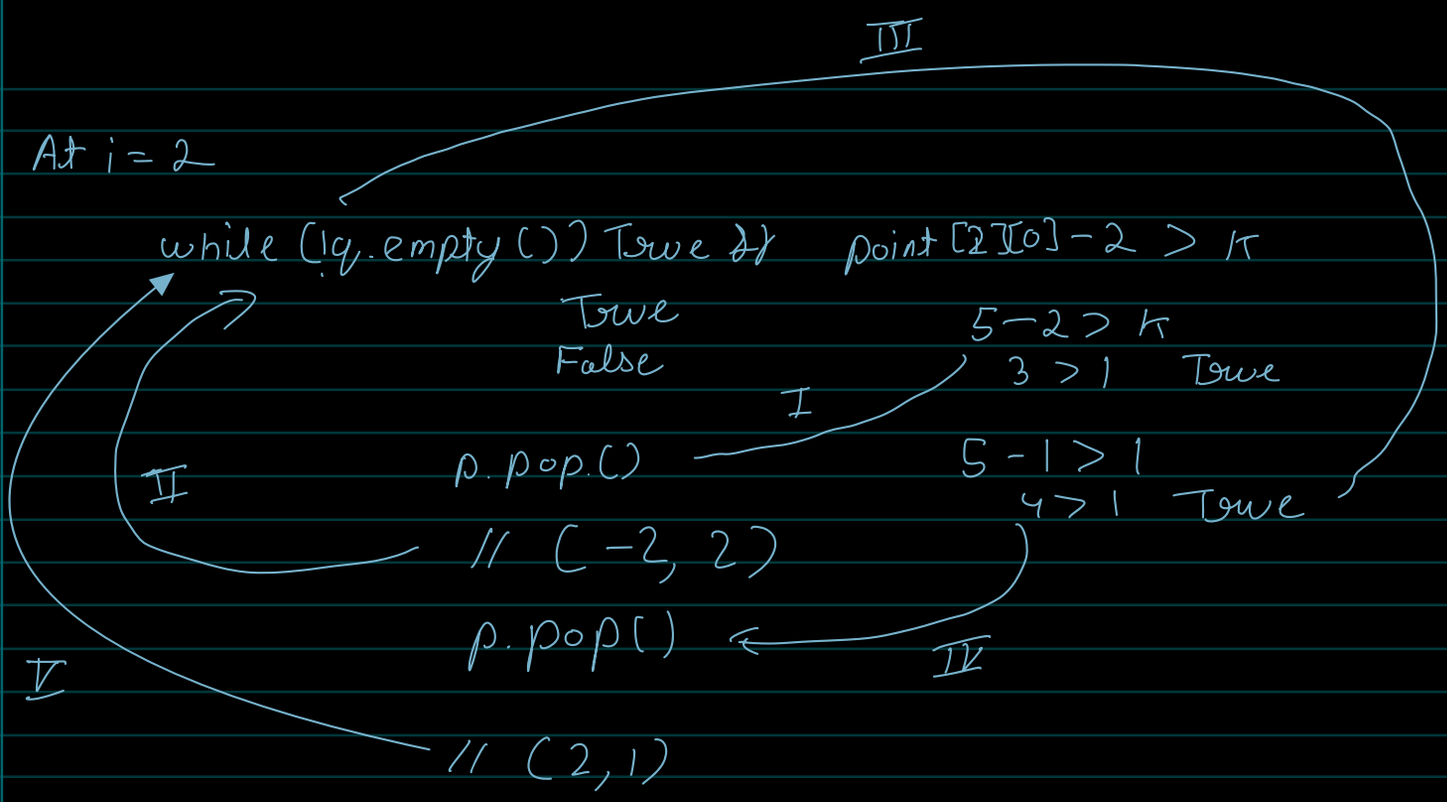
// 2 - 1
// 1 >= 1 False

if (!q.empty()) // True

ans = max(ans, p.top().second + points[1][0] + points[1][1])
// 2 + 2 + 0
// 4

ans = 4

q.push (points[1][1] - points[1][0], points[1][0])
// 0 - 2, 2
// -2, 2



if (!q.empty()) False

p.push(10 - 5, 5)

(5, 5)

At i = 3

while(!q.empty()) && 5 - 6 > 1

True -1 > 1 False

// possible Answer

if (!q.empty()) True

Ans = max(4, 5 + points[3][0] + points[3][1])

// 5 + 6 - 10

// 1

p.push(points[3][1] - points[3][0], points[3,0])

// -10 - 6

// -16, -6

i++ => 4 < 4 False

return ans

Since we have to get maximum value from the equation

$y_j + x_j + y_i - x_i$ following condition $i<j$ and $|x_i - x_j| \le k$

always positive ← $|x_i - x_j| \le k$ and $x_i \le x_j$

$x_j - x_i \ge k$

↓

always positive

points = [[1, 3], [2, 0], [5, 10], [6, -10]]

0 1 2 3

i = 0

put in priority queue

i = 1

Now for this case

[1, 3] [2, 0]

↓

priority queue

3 - 1	1
↓	↓
$y_i - x_i$	x_i

already done in pq

Check with previously inserted element

↓

If condition statisfy possible answer don't pop

i.e $x_j - x_i \ge k$

↓ → Priority queue top - x

current loop x

If possible ans [2, 0]

get max [ans, pq.top().first + 2 + , 0]

$x_j + y_j$

```cpp
class Solution {
public:
    int findMaxValueOfEquation(vector<vector<int>>& point, int k) {
        int n=point.size(),ans=INT_MIN;
        priority_queue < pair<int, int> > p;
        for(int i=0;i<n;i++){

            while((!p.empty()) && (point[i][0] - p.top().second >k)){
                p.pop();
            }

            if(!p.empty()){
                ans=max(ans,(p.top().first + point[i][0] + point[i][1]));
            }


            p.push(make_pair( point[i][1] - point[i][0], point[i][0]));
        }
        return ans;
    }
};
```

T.C - O(n log n)

↓ loop ↓ Priority queue.

S.C - O(n)