Sorting - 3

Question: Maximum Gap between consecutive elements

Given an unsorted constant array which cannot be modified, find the maximum gap between any 2 integers

min = 1
max = 10^9

Ex: A : 7 8 3 1 6

Sort(A) : 1 3 6 7 8

2 $\boxed{3}$ 1 1

Gap: Difference between consecutive elements of sorted Array

A : 3 8 1 5

A = 1 3 5 8

2 3 3

$\boxed{\text{Ans} = 3}$

Approach 1:

- Copy it to a new array → O(nlogn)
- Sort the array
- Traverse and find max gap O(n)

T.C : O(n logn)

S.C : O(n)

Approach 2:

1) Max value of max-gap :

2 adjacent

You know max(A), min(A), N

A = 2 5 2 5 2 2 5

max = 5
min = 2
N = 7

Max Gap = 5 - 2 = $\boxed{3}$

MAX VALUE = $\boxed{\text{MAX} - \text{MIN}}$

2) Min value of Max-Gap

You know max, min, N (max = min)

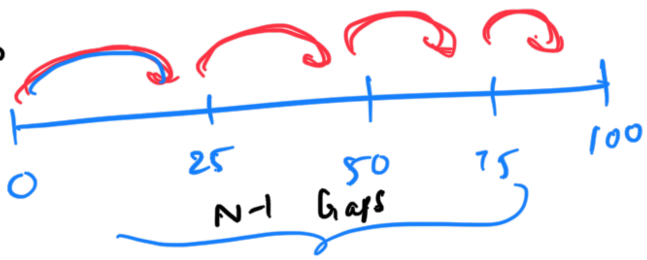
→ Arrange 5 numbers between 0 and 100 such that max-gap is minimum

→ If all elements are at equal distance

N = 5
max = 100
min = 0

0 1 50 75 100

$\boxed{1}$ $\boxed{50-1=49}$



$\left[\frac{100-0}{4}\right]$

(Min value) = $\boxed{\frac{MAX-MIN}{N-1}}$ → $\boxed{O(n)}$

g

A = 8 14 6 12 4 10 N = 6

4 6 8 10 12 14

Sorting ... max = ... min = 4

$\underbrace{\quad}_{2}$ $\underbrace{\quad}_{2}$ $\underbrace{\quad}_{2}$ 2

$\lceil 12 \rceil = \frac{14-4}{5} = \frac{max-min}{N-1}$

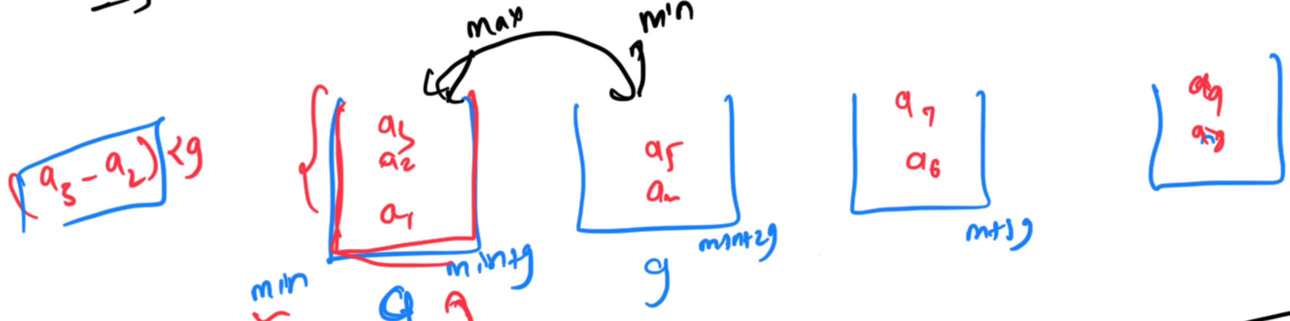
$\boxed{Ans \geq g} = \frac{max-min}{g}$

Bucketisations!

→ x, y

$\boxed{\text{if } (x-y) < g}$

⇒ Create buckets of size g



(min, min+g)

$\boxed{\frac{A[i] - min}{g}}$

A = 10 20 8 15 37

$g = \frac{37-8}{5-1} = \frac{29}{4} = 7$

ans = 17

37-20 = $\boxed{17}$

max = 37

min = 8

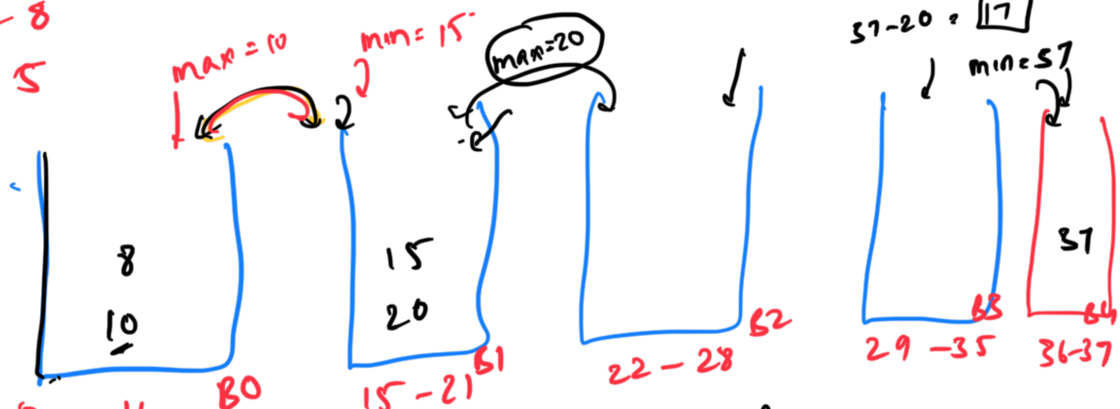
N = 5

6

$\boxed{\frac{20-8}{7} = \frac{12}{7} = 1}$

$\frac{15-8}{7} = \frac{7}{7} = 1$

$\frac{37-8}{7} = \frac{29}{7} = 4$



[8, 8+7)

[8+7, 8+2·7]

[8+2·7, 8+3·7)

[8, 8+g)

[8+g, 8+2g]

[8+2g, 8+3g)

A[i] → k th bucket

$\frac{10-8}{7} = \frac{2}{7}$

A[i]

$8 + k \cdot g = A[i]$

$k \cdot g = A[i] - 8$

$k = \frac{A[i] - 8}{g} = \frac{A[i] - min}{g}$

$\frac{15-8}{4} = \frac{7}{4} [1.75] = \lceil 1 \rceil$

A = 10 20 8 15 37 30 28 23

(Min Ans) $g = \frac{37-8}{8-1} = \frac{29}{7} = 4.1 : \boxed{4}$

Max = 37
Min = 8
N = 8

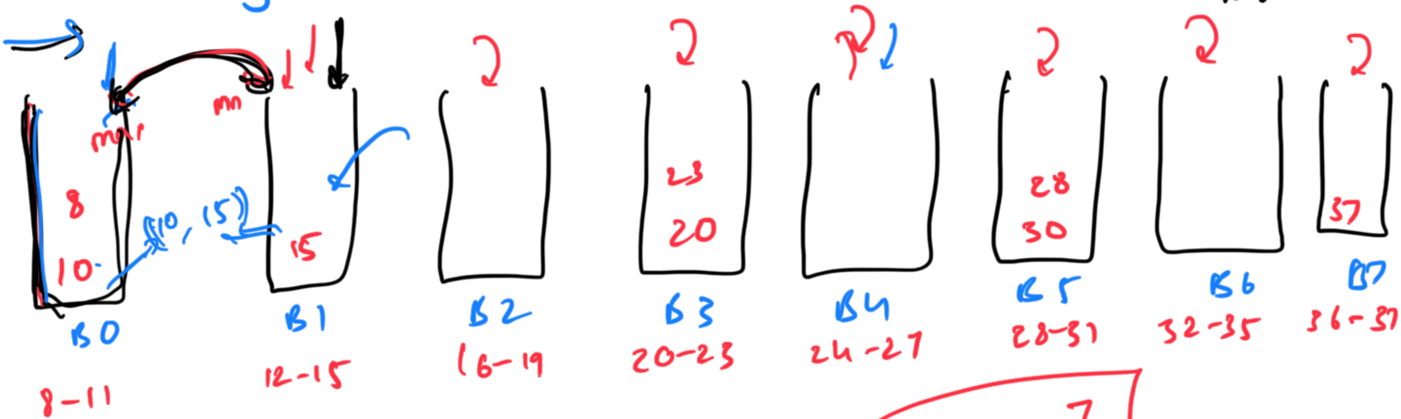
$b\_num \Rightarrow \frac{A[i] - min}{g} = \frac{A[i] - 8}{4}$

O(n)

bucket = $\boxed{8}$

N or N-1 bkt

min-prev



[8-12]

[12, 8]

prev-max = ~~23~~ 30

$\boxed{ans = 7}$

15 - 10 = 5

37 - 30 = $\boxed{7}$

20 - 15 = 5

28 - 23 = 5

x x+4

[x, x+4]

$\underline{min}$ value = $\left(\frac{max - min}{N-1}\right)$

$g = \frac{max - min}{N-1}$

$[min, min+g)$, $[min+g, min+2g)$, $[m+2g, min+3g)$ . . . .

Bucket No := k_1

$A[i] = [min + k_1 g, \; min + k_2 g]$

$$\boxed{min + k_1 g \le A[i]} < min + k_2 g$$

$$k_1 g \le A[i] - min$$

$$k_1 \le \frac{A[i] - min}{g}$$

$$\boxed{k_1 = \left\lfloor \frac{A[i] - min}{g} \right\rfloor}$$

Algo

1) Find g

2) Place elements into buckets

Any Bucket's max and min

2) Iterate the buckets and find the max gap

{O(n)}

// Compute $g = \frac{max - min}{N - 1}$

buckets[N].max = {INT_MIN}

buckets[N].min = {INT_MAX}

$[\{MIN, MAX\}, \{MIN, MAX\} . \; . \; . \; . \; . \;]_N$

O(N) {

for (i = 0; i < N; i++){

b_num = A[i] - min / g;

buckets[b_num].max = max(A[i], buckets[b_num].max)

buckets[b_num].min = min(A[i], buckets[b_num].min)

}

prev_max = bucket[0].max;

O(N)

for(i = 1; i < n; i++) {

if(bucket[i].min != INT_MAX) {

ans = max(ans, bucket[i].min - prev_max);

prev_max = bucket[i].max;

}

}

N = 10^6

T.C: O(N)

S.C: O(N)

A = 2 6 10 14

max = 14

min = 2

N = 4

$\frac{max - min}{N-1} = \frac{12}{3} = \boxed{4}$

$\boxed{g = 4}$

[2, 6] [7, 11] [12 16]

6 - 2 =

(min + min + g - 1)

(min + g, min + 2g - 1)

Question: Max Chunks to Make array Sorted

Ans = 1

A = 2 0 1 2

0 2 1 2

2 0 1 2

2 0 1 2 ⇒ X

2 0 1 2 $\boxed{Ans = 2}$

0 1 2 2

Exa

[3 2 3 4 0]

0 2 3 3 4 $\boxed{Ans = 1}$

A = 2 1 3 9 8 15 11 14 13

1 2 3 8 9 11 13 14 15

$\boxed{Ans = 4}$

A = 2 3 1 9 8 15 11 14 13

1 2 3 8 9 11 13 14 15

$\boxed{Ans = 3}$

✓ A = 2 3 1 9 8 15 11 14 13

✓ B = 1 2 3 8 9 11 13 14 15

Task: Count no. of contiguous subarrays which have same elements but in different order

2 2

HashA = { }

HashB = { }

( ) O(n)

ans = ~~1~~ 2

2 Hashmaps → 1 Hashmap

A =	2	3	1	9	8	15	11	14	13
B =	1	2	5	8	9	11	15	14	15

ans++;

ans = 2

HashMap: { }

T.C: O(nlogn) + O(n)

[0...N-1]

```
for(i=0; i<n; i++) {
    hash[A[i]]++;
    hash[B[i]]--;
    if(hash[A[i]] == 0) remove A[i];
    if(hash[B[i]] == 0) remove B[i];
    if(hash is empty) ans++;
}
ans
```

T.C: O(nlogn)

S.C: O(n)

Question: Sum the difference.

subsequences = $2^n - 1$

A = [1, 2]

$2^2 - 1 = 3$

max = 1 → 1 - 1 = 0

{ 1 } — min = 1

{ 2 } → max = 2, min = 2 → 2 - 2 = 0

{ 1, 2 } → max = 2, min = 1 → 2 - 1 = 1

$\boxed{0 + 0 + 1 = 1}$

A = [2, 3, 1]

7 non-empty subsequences

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

$1 + 1 + 2 + 2 = \boxed{6}$

Brute Force :

Generate all subsequences

T.C: $O(2^n)$?

$N = 10^6$

$\boxed{2^{10^6}}$

2^{32}

Efficient Approach:

A = [2, 3, 1]

3 is max

[1, 3]
[2, 3]
[1, 2, 3]

} = 4

[3] -

2 is max : [1, 2] [2] } 2

{0}

A = [1 2 3 4 5, 6, 7]

lesser elem = 2

[3 (2) (2)]

$\underbrace{\qquad}_{2 \times 2 = 4}$

[3]

No. of Subsequence in which 3 is the min ele :

2^4 sub

$(2^4$

A = 1 2 3 4 5

1 => Min => 2^4
Max => 1

Ans = $1 \times (1 - 2^4)$

-2

(2×8)

2×2^1

$\boxed{2}$ => Min => 2^3
Max => 2^1

$+ 2(2^1 - 2^3)$

3 => Min => 2^2
Max => 2^2

$+ 3(2^2 - 2^2)$

4 => Min => 2^1
Max = 2^3

$+ 4(2^3 - 2)$

2^1

$4 \times 8 - 4 \cdot 2$

5 => Min = 2^0
Max = 2^4

$+ 5(2^4 - 2^0)$

$(-4) \times 2$

min = 2^1 = $\boxed{2}$

$(+4) \times 8 = 32$

$2^3 = 8$

max = 2

$\boxed{T.C: O(n \log n)}$

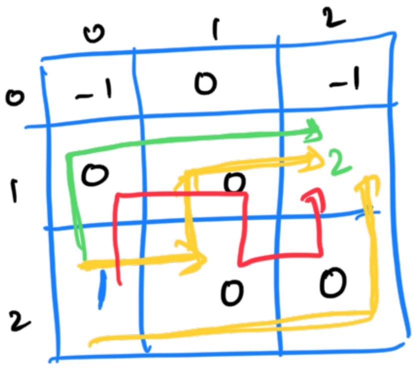
$O(n \log n)$ $\rightarrow$ $O(n \log n)$

$O(n)$ $\rightarrow$ $O(n)$

Question: Unique Path.

2D matrix which contains [0, 1, 2, -1]

A =



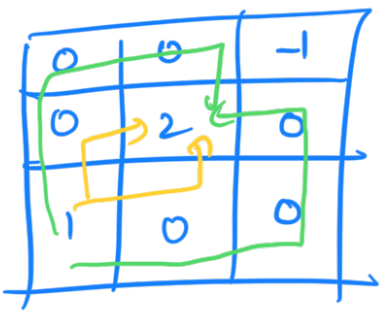
Ans = ~~3~~ $\boxed{4}$

3 x 3

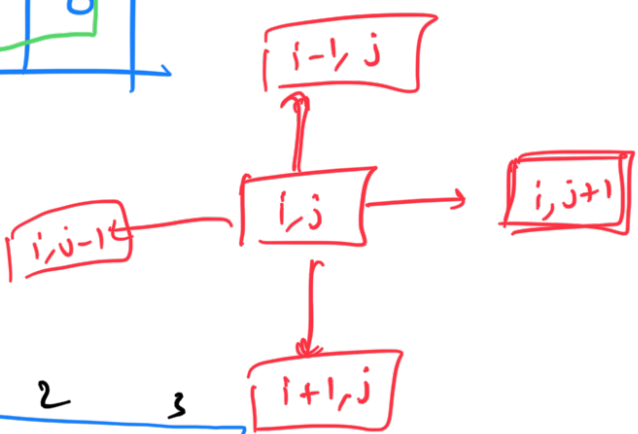
-> You can visit a cell at max once

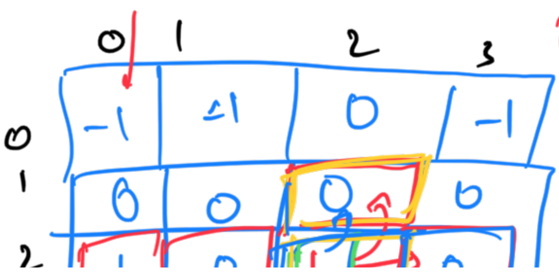
-> You can move up, down left or right

X diagonally



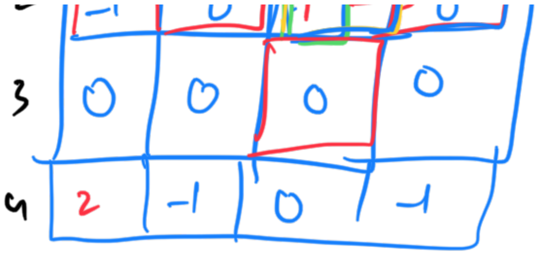
Ans = 4





ways(i, j) =

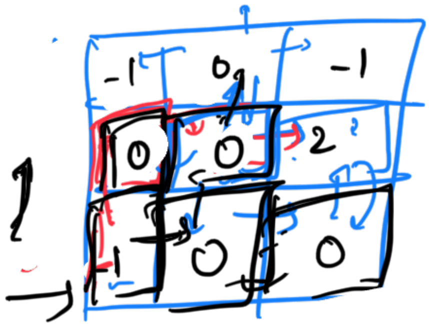
ways(i-1, j) + ways(i, j+1)

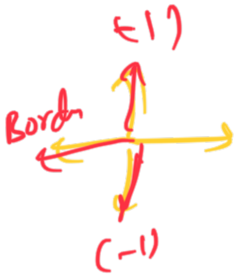


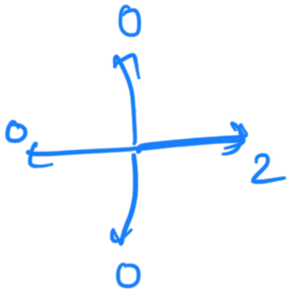
+ ways(i+1, j) + ways(i, j ...

Issues

1) if A[i][j] == -1, ways(i, j) = 0
2) Take care of border
3) Do not visit any cell more than once.







A[i][j] = -1
↓
call All children
ways(i+1, j)
ways(i-1, j)
ways(i, j-1)
ways(i, j+1)

if(A[i][j] == 2)
ways(i, j) = 1

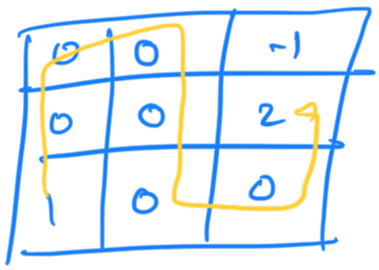
A[i][j] = 0

```
int paths(i, j, jumps) {
    if (i < 0 || i ≥ n || j < 0 || j ≥ n) return 0;
    if (A[i][j] == -1) return 0;
    if (A[i][j] == 2) return 1;
        && jumps == count_zeros + 1
```

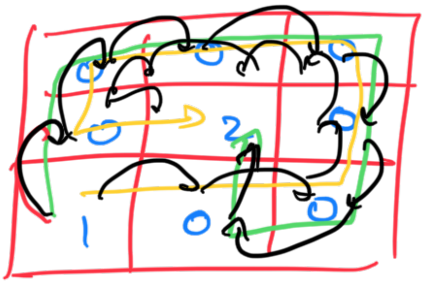
ways = 0;
A[i][j] = -1;
ways += paths(i-1, j, jumps+1);
ways += paths(i, j+1, jumps+1);
ways += paths(i+1, j, jumps+1);
ways += paths(i, j-1, jumps+1);
way +=
A[i][j] = 0;
} return ways;

Question: visit all the 0's

A =



A =



Ans = 2

jumps = 8

zeroes 7

Approach 1:

when you reach the destination, check if you visited all the zeroes

Approach 2:

No. of jumps = count_0 + 1