Today's content:

→ find pair (i, j) where a[i] = a[j] and (j - i) is minimum.

→ Longest sub-array with sum = 0

→ Longest sequence of consecutive elements

→ {Implementation idea}

2-pointers.

→ count the pair for given sum

Q) Count of pairs such that sum = 10.

a → [1 3 4 5 6 7 10] ans → 2.

a → [1 3 4 4 5 6 7 10] ans → 3.

a → [1 3 4 4 5 6 6 6 7 10] ans → 7

a → [1 3 4 4 5 5 5 5 6 6 6 7 10]

i j

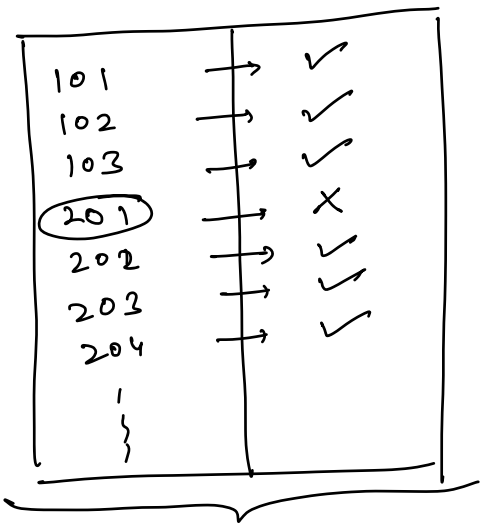
5+5 = 10.

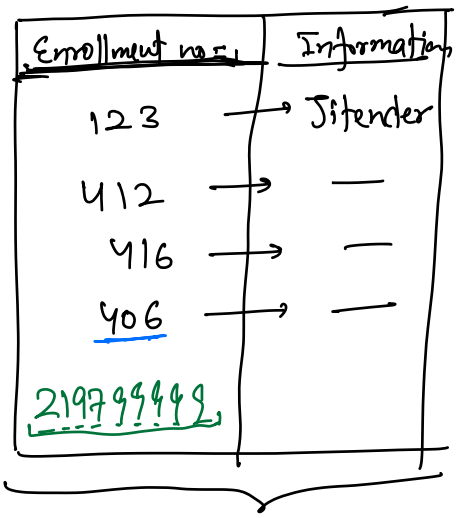
$1 + ({}^{2}C_1 * {}^{3}C_1) + {}^{4}C_2 = 1 + 6 + 6 = 13$

$\frac{4 \times 3}{2}$

```
if (arr[i] + arr[j] = tar) {
    if (arr[i] == arr[j]) {
        count = j - i + 1;
        ans += count * (count-1) / 2;
    }
    else {
        left = arr[i],   => lcount
        right = arr[j].  => rcount
        ans += lcount * rcount;
    }
}
```

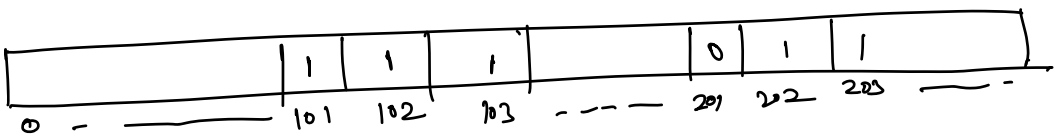
Hashing.





Array?

Index → room no.



loop-holes? memory. → wastage.
→ size ≈ 10^7

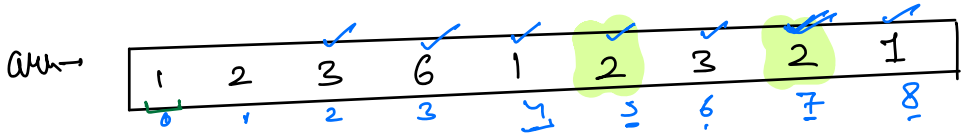
Java	C++	Python / C#
HashMap	unordered_map	Dictionary.
HashSet	ordered_map	Set.
TreeMap	map	
TreeSet	set	

(key, value) → HashMap
Keys ← HashSet

insert, delete, access. → O(1)

Q1 Given an array, find any pair (i, j) such that $A[i] == A[j]$ and $(j - i)$ is minimum.

We are looking closest duplicates.



ans = 7 - 5 = 2

B.F. → Consider all pairs → $O(N^2)$

A2: Fix one element.
↳ You need to find the last occurrence of that element.

A2.: (HashMap)

element	last occurrence
1	8
2	7
3	6
6	3

ans → ~~10~~ 4, 2

ans = Min(ans, i - lo[a[i]]);

T.C → O(n)
S.C → O(n)

Q: Given an array. Find length of largest subarray with sum = 0.

a:

0	1	2	3	4	5	6	7	8	9	10
2	2	1	-3	4	3	1	-8	6	-2	-1

(4-7)
(1-7)
(5-9)
(3-10)
(1-3)

ans = 8.

B.f. Consider all the sub-arrays $\Rightarrow \frac{n(n+1)}{2}$.

T.C $\rightarrow O(n^2)$

trading

10 A.m ⇓ 10 K

11 A.m: +1K → 11K

12 p.m: +4K → 15K

1 p.m: -3K → 12K

2 p.m: -2K → 10K

3 p.m

prefixSum →

0	1	2	3	4	5	6	7	8	9	10
2	4	5	2	6	9	10	2	8	6	5

→ Consider first occurence.

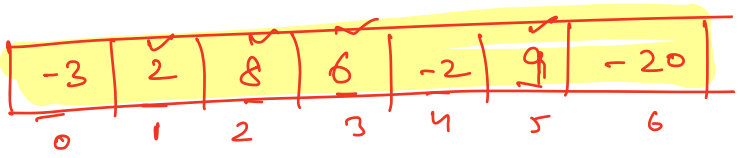
2	→	0
4	→	1
5	→	2
6	→	4
9	→	5
10	→	1
8	→	8

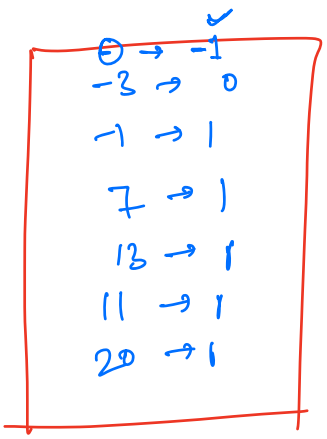
ans → ~~-∞~~ 3 7 8.

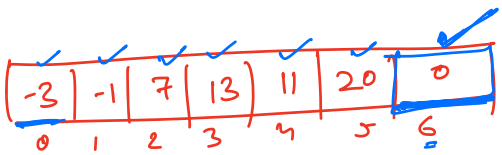
ans = max(ans, i - fo[a[i]]);

[# todo count of sub-arrays with sum = 0]

#edge-case:







ans = 6 - (-1) = 7.

Q: Given an array. find the length of largest sequence which can be re-arranged to get consecutive elements.

a → [3 8 29 4 28 12 10 2 11 9 5 8]

ans → 5

A1: Sort the array.

[2 3 4 5 8 8 9 10 10 11 12 28 29]

ans = 4

len = 1 2 3 4 5

len = 1

curr = prev
curr = prev+1

T.C → O(n log n)

A2.:

3 → [3, 4, 5]

8 → [8, 9, 10, 11, 12]

29 → [29]

4 → [4, 5]

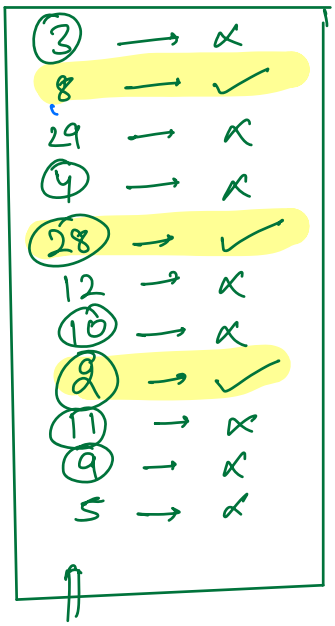
len: 3, 5, 1, 2, 1, 1

max of all.

T.C → O(n³)

using hashmap

O(n²)



unique ⇔ Keys

① Add all elements in the hashmap. → O(n)

② Consider only valid starting points. → O(n)

③ 8 28 2

8 ✓ 9 ✓ 10 ✓ 11 ✓ 12 ✓

28 ✓ 29 ✓

2 ✓ 3 ✓ 4 ✓ 5 ✓

→ O(n)

	8	28	2
len	5	2	4

ans.

T.C → O(n)
S.C → O(n)

implementation.

[17, 20, 27, 30, 10]

M = 17.

⇓

0–16,

0–(M-1)

x % 17. ⇐ hash-value.

arr[17 % 17] = arr[0]

arr[20 % 17] = arr[3]

arr[27 % 17] = arr[10] ← collision

arr[30 % 17] = arr[13]

arr[10 % 17] = arr[10]

To minimise collision

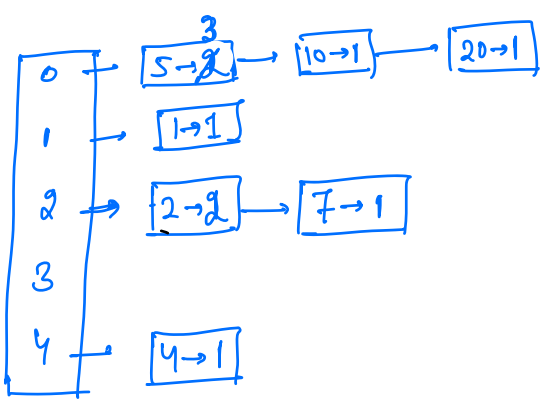
- Chaining
- Open Addressing

arr → | 2 | 4 | 5 | 7 | 10 | 2 | 5 | 20 | 5 | 1 |

(freq of every elem)

x % 5

M = 5.



linked-list.

7 % 5 = 2.

10 % 5 = 0

2 % 5 = 2

5 % 5 = 0

20 % 5 = 0

1, 6, 16, 21

1 % 5 = 1	1 % 10 = 1
6 % 5 = 1	6 % 10 = 6
16 % 5 = 1	16 % 10 = 6
21 % 5 = 1	21 % 10 = 1
M = 5	M = 10

T.C.	Best	Wirst
search	O(1)	O(n)
insert	O(1)	O(n)
update	O(1)	O(n)