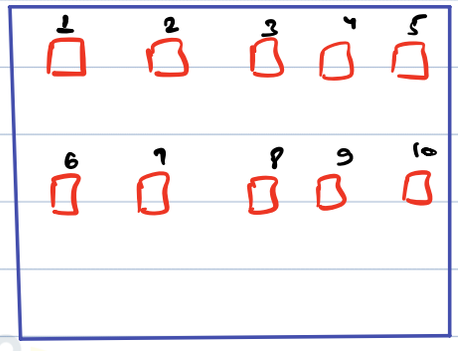
Today's agenda

- HashMap Intro
- frequency of each query
- first non-repeating elements
- HashSet
- Number of distinct elements
- Pair sum == K → O(N)

// HashMap Intro

① Mohit:



true : occupied
false : empty

boolean n1 :

n2 :

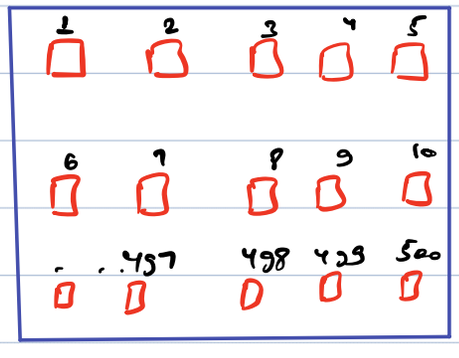
n3 : false (true)

.

.

n : 10 :

② Adarsh



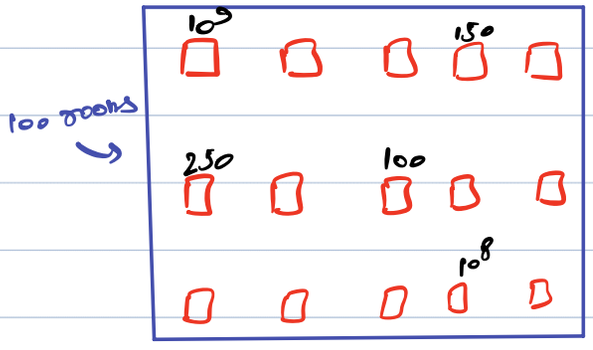
boolean arr[501]

arr[500] : true

```
if (arr[350] == false){
    Print ("Room given");
    arr[350] = true;
}
```

(iii) Cranesham

→ {1, 10^9}

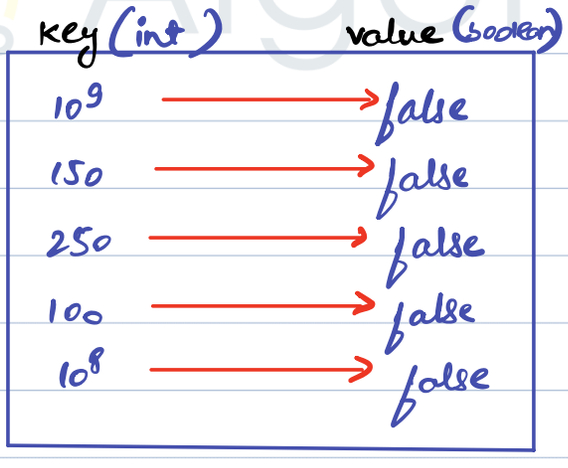


boolean arr[10^9+1]

100 rooms → 10^9 size

↳ Hashmap solves the issue

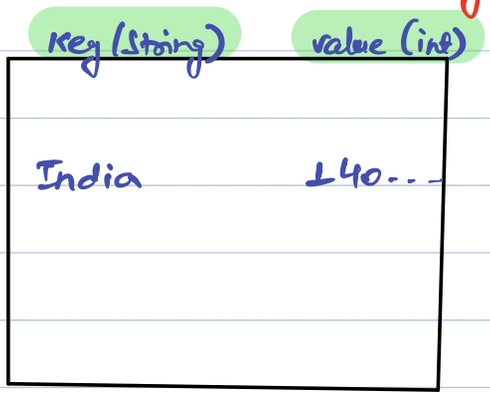
100 size + 0.99



↳ (key, value) Pair

Q) Store Population of every Country:

Key: Country name: String
value: Population : int/long



Q) Store All the School with their Principal name.

key: String
value: String

facts:

(i) Keys can be only of following data type:

wrapper class

boolean	→	Boolean
int	→	Integer
long	→	Long
char	→	Character
double	→	Double
String	→	String

Not applicable: arrays, objects, null

(ii) values can be of any type.

Syntax:

key type ↙ value type ↙ name

```
HashMap<Integer, Integer> hm =
        new HashMap<>();
```

//add

```
hm.put(10, 50);
hm.put(20, 60);
hm.put(30, 60);
hm.put(20, 70); → replace
```

T.C : O(1)

hm

Integer	Integer
~~10~~	~~50~~
20	~~60~~ 70
30	60

//get

```
hm.get(20); → 70
hm.get(40); → null
```

T.C: O(1)

//size

```
hm.size()
```

// Contains key → check if key exists

```
hm.containsKey(20); → True
hm.containsKey(40); → false
```

T.C: O(1)

// remove

hm.remove(10); → it will remove (10, 50)

T.C: O(1)

int arr[n];

→ for(int i=0; i<n; i++){

System.out.println(arr[i]);

}

int arr[n];

for(int val: arr){

System.out.println(val);

}

iterating on keys

↳ for(int key: hm.keySet()){

}

Q) find frequency

Ls Given N array elements & Q queries. for every query find frequency of element in array.
↳ no. of occurrence

Ex: arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10)

queries[4] = { 2 8 3 5 }

//idea:

Ls iterate and count for every query.

//Pseudo code

```
void PrintFrequency (int arr[N], int queries[M]){

    for (int i=0; i<M; i++){
        int val = queries[i];
        int count = 0;
        for (int j=0; j<n; j++){
            if (arr[j] == val){
                count++;
            }
        }
        System.out.println(count);
    }
}
```

T.C: O(M*N)

```
void PrintFrequency (int arr[n], int queries[m]){
    for (int i=0; i<m; i++){
        int val = queries[i];
        int count = 0;
        for (int j=0; j<n; j++){
            if (arr[j] == val){
                count++;
            }
        }
        System.out.println(count);
    }
}
```

Eg: arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10)

queries[4] = { 2 8 3 5 }

val = 8 count = 0

3

AlgoPrep

Idea 2

arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10)

queries[4] = { 2 8 3 5 }

hm: Integer(key) → Integer(val) → frequency

Integer(key)	Integer(val) → frequency
2	3
6	2
3	2
8	3
10	1

3 3 2 0

T.C: O(N) + O(M)

//Pseudo Code

```
void PrintFrequency (int arr[N], int queries[M]){

    HashMap<Integer, Integer> hm = new HashMap<>();

    for (int i=0; i<N; i++) {
        if (hm.containsKey(arr[i]) == true) {
            int temp = hm.get(arr[i]);
            hm.put(arr[i], temp+1);
        }
        else{
            hm.put(arr[i], 1);
        }
    }

    for (int i=0; i<M; i++) {
        int val= queries[i];
        if (hm.containsKey(val) == true) {
            Print(hm.get(val));
        }
        else{
            Print(0);
        }
    }
}
```

T.C: O(N+M)

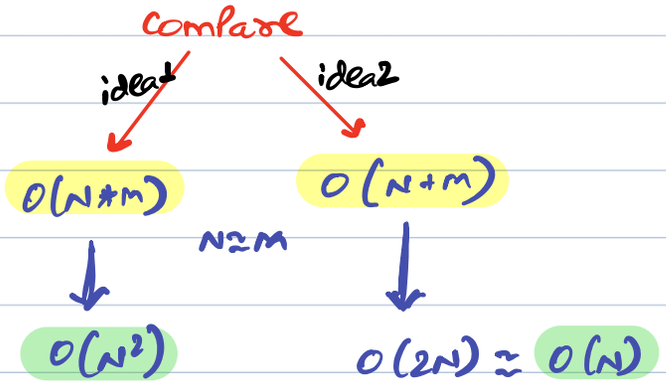
arr[11]: { 2 6 3 8 2 8 2 3 8 10 6 }
(0 1 2 3 4 5 6 7 8 9 10)

```
for (int i=0; i<N; i++) {
    if (hm.containsKey(arr[i]) == true) {
        int temp = hm.get(arr[i]);
        hm.put(arr[i], temp+1);
    }
    else {
        hm.put(arr[i], 1);
    }
}
```

hm

Integer (key)	→	Integer (val)
2	—	~~X~~ 3
6	—	~~X~~ 2
3	—	~~X~~ 2
8	—	~~X~~ 3
10	—	1

i	arr[i]	hm.containsKey(arr[i])	temp
0	2	F	
1	6	F	
2	3	F	
3	8	F	
4	2	T	1
5	8	T	1
6	2	T	2



Break till 9:50 Pm

Q) Find the first non-repeating elements

↳return -1 if all the elements are repeating.

Ex1: arr[6]: {1 2 3 1 2 5} →3

arr[8]: {5 4 4 3 6 7 5 6} →3

//idea

arr[8]: {5 4 4 3 6 7 5 6}

hm

5	x2
4	x2
3	1
6	x2
7	1

↳ iterate on array and get the first element with frequency 1.

//Pseudo Code

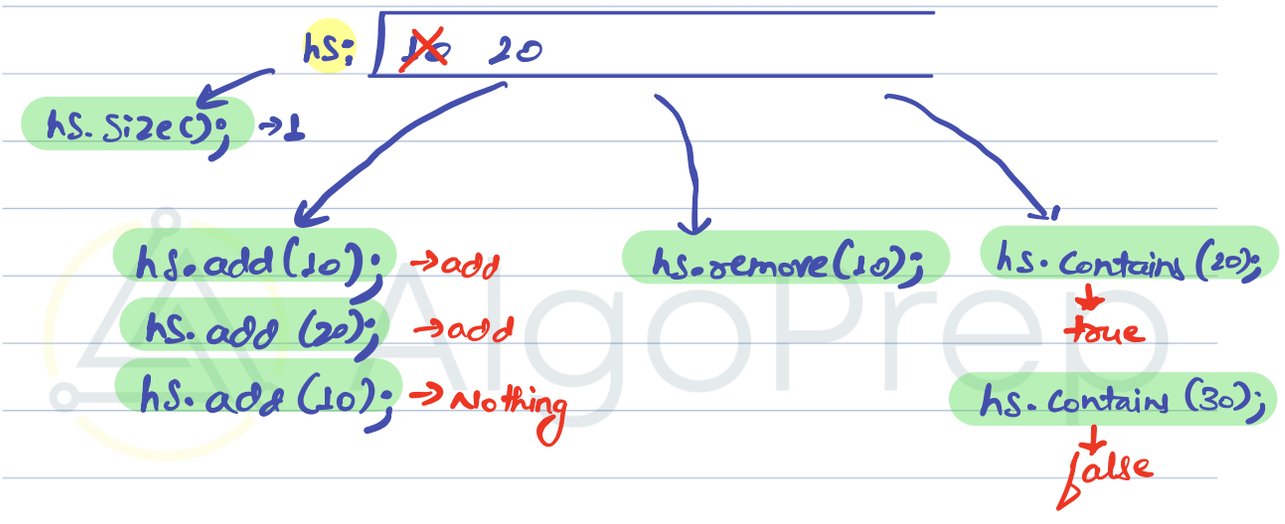
```
P S   int firstNonRepeating(int arr[N]){

    HashMap<Integer, Integer> hm = new HashMap<>();

    for(int i=0; i<N; i++){
        if(hm.containsKey(arr[i]) == true){
            int temp = hm.get(arr[i]);
            hm.put(arr[i], temp+1);
        }else{
            hm.put(arr[i], 1);
        }
    }

T.C: O(2N)
 ≈O(N)

    for(int i=0; i<N; i++){
        if(hm.get(arr[i]) == 1){
            return arr[i];
        }
    }

    return -1;
}
```

// HashSet

↳ only the key Part of hashmap

HashSet < Integer > hs = new HashSet <>();

↳ random order



Q) Given arr[N], find no. of distinct elements.

Ex: arr[5] = {4 6 7 6 5} → ans: 4

arr[5] = {10 10 10 20 20} → ans: 2

//idea

arr[5] = {10 10 10 20 20}

hs: | 10 20 |

↳ ans = hs.size()

//Pseudo code

```
P S int distinctElements (int arr[N]){

    HashSet<Integer> hs = new HashSet<>();

    for (int i=0; i<n; i++){
        hs.add(arr[i]);
    }

    return hs.size();
}
```

Q) Pair Sum == K

↳ Given arr[N], check if there exists a Pair(i,j) such that arr[i] + arr[j] == K and (i != j).

	0	1	2	3	4	5	6	7	8	9
arr[10]:	8	9	1	-2	4	5	11	-6	7	5

K = 11 → arr[4] + arr[8] → 4 + 7 = 11 → true

K = 6 → arr[0] + arr[3] → 8 - 2 = 6 → true

K = 22 → ~~arr[6] + arr[6]~~ → false

//ideal

↳ Nested loop, check all Pairs for Sum == K.

T.C: O(N²)

Idea 2

	0	1	2	3	4	5	6	7	8	9
arr[10]:	8	9	1	-2	4	5	11	-6	7	5

Step 1: Insert all the elements in hashset.

hs: { 8 9 1 -2 4 5 11 -6 7 }

(I) a + b = 11

a	b	b is present?
8	3	NO
9	2	NO
1	10	NO
-2	13	NO
4	7	yes → true

(II) arr[10]:

	0	1	2	3	4	5	6	7	8	9
arr[10]:	8	9	1	-2	4	5	11	-6	7	5

hs: { 8 9 1 -2 4 5 11 -6 7 }

a + b = -4

a	b	is b present
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	yes → true

//idea3

Ls insert all elements of array in your hashmap with frequency.

	0	1	2	3	4	5	6	7	8	9
arr[10]:	8	9	1	-2	4	5	11	-6	7	5

hm

8 - 1	11 - 1
9 - 1	-6 - 1
1 - 1	7 - 1
-2 - 1	
4 - 1	
5 - 12	

a+b = -4

a	b	is b Present
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	yes, but at same index

//Pseudo code

```
boolean PairSum (int arr[N], int K){

    HashMap<Integer, Integer> hm = new HashMap<>();

    for (int i=0; i<N; i++){
        if (hm.containsKey (arr[i]) == true){
            int temp = hm.get (arr[i]);
            hm.put (arr[i], temp+1);
        }
        else{
            hm.put (arr[i], 1);
        }
    }

T.C: O(N)

    for (int i=0; i<n; i++){
        int a = arr[i];
        int b = K-a;

        if (a != b && hm.containsKey (b) == true){
            return true;
        }
        else if (a == b && hm.get(b) > 1 && hm.containsKey (b) == true){
            return true;
        }
    }

    return false;
```

3

i

arr[10]: 8 9 1 -2 4 5 11 -8 7 5

hm

8 - 1	11 - 1
9 - 1	-8 - 1
1 - 1	7 - 1
-2 - 1	
4 - 1	
5 - 12	

```
for (int i=0; i<n; i++) {
    int a = arr[i];
    int b = k-a;

    if (a!=b && hm.containsKey(b)==true){
        return true;
    }
    else if (a==b && hm.get(b)>1 ){
        return true;
    }
}
```

hm.containsKey(b)==true

a+b = -4

a	b	is b Present
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	NO, same index
4	-8	yes → true

Frequency queries

Java Code:

```
import java.util.*;
import java.util.Map;

class Solution {
    public static void FrequencyQueries(int[] arr,int[] queries) {
        HashMap<Integer, Integer> hm = new HashMap<>();
        for (int num : arr) {
            if(hm.containsKey(num)== true){
                int temp = hm.get(num);
                hm.put(num,temp+1);
            }else{
                hm.put(num,1);
            }
        }

        for (int num : queries) {
            if(hm.containsKey(num)== true){
                System.out.println(hm.get(num));
            }else{
                System.out.println(0);
            }
        }
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int m = scn.nextInt();

        int[]arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        int[] queries = new int[m];
        for(int i=0;i<m;i++){
            queries[i] = scn.nextInt();
        }

        FrequencyQueries(arr, queries); // Output: 3
```

```
    }
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

void FrequencyQueries(vector<int>& arr, vector<int>& queries) {
    unordered_map<int, int> hm;

    for (int num : arr) {
        if (hm.find(num) != hm.end()) {
            int temp = hm[num];
            hm[num] = temp + 1;
        } else {
            hm[num] = 1;
        }
    }

    for (int num : queries) {
        if (hm.find(num) != hm.end()) {
            cout << hm[num] << "\n";
        } else {
            cout << "0\n";
        }
    }
}

int main() {
    int n, m;
    cin >> n >> m;

    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    vector<int> queries(m);
    for (int i = 0; i < m; i++) {
        cin >> queries[i];
    }
```

```
    FrequencyQueries(arr, queries);

    return 0;
}
```

Python Code:

```
def FrequencyQueries(arr, queries):
    hm = {}

    for num in arr:
        if num in hm:
            hm[num] += 1
        else:
            hm[num] = 1

    result = []
    for num in queries:
        if num in hm:
            result.append(hm[num])
        else:
            result.append(0)

    return result

if __name__ == "__main__":
    n, m = map(int, input().split())
    arr = list(map(int, input().split()))
    queries = []

    for _ in range(m):
        queries.append(int(input()))

    results = FrequencyQueries(arr, queries)
    for res in results:
        print(res)
```

Non repeating elements

Java Code:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;


class Solution {
    public static int firstNonRepeating(int[] arr) {
        HashMap<Integer, Integer> hm = new HashMap<>();
        for (int num : arr) {
            if(hm.containsKey(num)== true){
                int temp = hm.get(num);
                hm.put(num,temp+1);
            }else{
                hm.put(num,1);
            }
        }
        for (int num : arr) {
            if (hm.get(num) == 1) {
                return num;
            }
        }
        return -1; // If no non-repeating element is found
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int[]arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }
        System.out.println(firstNonRepeating(arr)); // Output: 3
    }
}
```

C++ Code:

```
#include <iostream>
#include <vector>
```

```
#include <unordered_map>
using namespace std;

int firstNonRepeating(const vector<int>& arr) {
    unordered_map<int, int> hm;
    for (int num : arr) {
        if (hm.count(num) == 1) {
            int temp = hm[num];
            hm[num] = temp + 1;
        } else {
            hm[num] = 1;
        }
    }
    for (int num : arr) {
        if (hm[num] == 1) {
            return num;
        }
    }
    return -1; // If no non-repeating element is found
}

int main() {
    int n;
    cin >> n;
    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    cout << firstNonRepeating(arr) << endl; // Output: 3
    return 0;
}
```

Python Code:

```
def first_non_repeating(arr):
    hm = {}
    for num in arr:
        if num in hm:
            hm[num] += 1
        else:
            hm[num] = 1
    for num in arr:
        if hm[num] == 1:
            return num
    return -1  # If no non-repeating element is found
```

```
def main():
    n = int(input())
    arr = list(map(int, input().split()))
    print(first_non_repeating(arr))  # Output: 3

if __name__ == "__main__":
    main()
```

Distinct element in an array

Java Code:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class
should be named Solution. */
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int[]arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        System.out.println(countDistinct(arr));
    }
    public static int countDistinct(int[] arr){
        int n = arr.length;
        HashSet<Integer> hs = new HashSet<Integer>();

        for (int i = 0; i < n; i++) {
            // add all the elements to the HashSet
            hs.add(arr[i]);
        }

        // return the size of hashset as
        // it consists of all Unique elements
        return hs.size();
    }

}
```

C++ Code:

```
#include <iostream>
#include <unordered_set>
using namespace std;
```

```
int countDistinct(int arr[], int n) {
    unordered_set<int> us;

    for (int i = 0; i < n; i++) {
        us.insert(arr[i]);
    }

    return us.size();
}

int main() {
    int n;
    cin >> n;

    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cout << countDistinct(arr, n) << endl;

    return 0;
}
```

Python Code:

```
def countDistinct(arr):
    return len(set(arr))

n = int(input())
arr = list(map(int, input().split()))

print(countDistinct(arr))
```

Check Pair Sum

Java Code:

```
import java.io.*;
import java.util.*;
import java.text.*;
```

```
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int k = scn.nextInt();
        int[]arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        if(PairSum(arr,k) == true){
            System.out.println("Y");
        }else{
            System.out.println("N");
        }

    }

     public static boolean PairSum(int[] arr, int k) {
         int n = arr.length;
        HashMap<Integer,Integer> map = new HashMap<>();
        for (int num : arr) {
            if(map.containsKey(num) == true){
                int temp = map.get(num);
                map.put(num,temp+1);
            }else{
                map.put(num,1);
            }
        }

        for(int i=0;i<n;i++){
            int a = arr[i];
            int b = k - a;

            if((a != b) && (map.containsKey(b)==true)){
                return true;
            }else if((a == b) && (map.get(b) > 1)){
                return true;
            }
        }

         return false;
    }
```

}

C++ Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

bool PairSum(vector<int>& arr, int k) {
    unordered_map<int, int> map;
    for (int num : arr) {
        if (map.count(num) == 1) {
            map[num]++;
        } else {
            map[num] = 1;
        }
    }

    int n = arr.size();
    for (int i = 0; i < n; i++) {
        int a = arr[i];
        int b = k - a;

        if ((a != b) && (map.count(b) == 1)) {
            return true;
        } else if ((a == b) && (map[b] > 1)) {
            return true;
        }
    }

    return false;
}

int main() {
    int n, k;
    cin >> n >> k;

    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    if (PairSum(arr, k)) {
        cout << "Y" << endl;
```

```
    } else {
        cout << "N" << endl;
    }

    return 0;
}
```

Python Code:

```
def pair_sum(arr, k):
    freq_map = {}
    for num in arr:
        if num in freq_map:
            freq_map[num] += 1
        else:
            freq_map[num] = 1

    n = len(arr)
    for i in range(n):
        a = arr[i]
        b = k - a

        if (a != b) and (b in freq_map):
            return True
        elif (a == b) and (freq_map[b] > 1):
            return True

    return False

def main():
    n, k = map(int, input().split())
    arr = list(map(int, input().split()))

    if pair_sum(arr, k):
        print("Y")
    else:
        print("N")

if __name__ == "__main__":
    main()
```

Count of right triangles_HW

Solution Vid:

https://youtu.be/219ae5GyKJA

Java Code:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int[][] a = new int[n][2];

        for(int i=0;i<n;i++){
            for(int j=0;j<2;j++){
                a[i][j] = scn.nextInt();
            }
        }

        System.out.println(RightAngled(a,n));
    }

    static int RightAngled(int a[][], int n){

    HashMap<Integer, Integer> xpoints = new HashMap<>();
    HashMap<Integer, Integer> ypoints = new HashMap<>();

    for (int i = 0; i < n; i++){
        if(xpoints.containsKey(a[i][0])){
            xpoints.put(a[i][0], xpoints.get(a[i][0]) + 1);
        }else{
            xpoints.put(a[i][0], 1);
            }
        if(ypoints.containsKey(a[i][1])){
             ypoints.put(a[i][1], ypoints.get(a[i][1]) + 1);
        }
        else{
             ypoints.put(a[i][1], 1);
```

```
        }
    }

        int count = 0;

        for (int i = 0; i < n; i++){
            if (xpoints.get(a[i][0]) >= 1 &&
                ypoints.get(a[i][1]) >= 1){

                count += (xpoints.get(a[i][0]) - 1) *
                    (ypoints.get(a[i][1]) - 1);
            }
        }


        return count;
    }

}
```

C++ Code:

```
#include <iostream>
#include <vector>
#include <map>

using namespace std;

int RightAngled(vector<vector<int>>& a, int n) {
    map<int, int> xpoints;
    map<int, int> ypoints;

    for (int i = 0; i < n; i++) {
        if (xpoints.count(a[i][0])) {
            xpoints[a[i][0]] = xpoints[a[i][0]] + 1;
        } else {
            xpoints[a[i][0]] = 1;
        }
        if (ypoints.count(a[i][1])) {
            ypoints[a[i][1]] = ypoints[a[i][1]] + 1;
        } else {
            ypoints[a[i][1]] = 1;
        }
    }

    int count = 0;
```

```
    for (int i = 0; i < n; i++) {
        if (xpoints[a[i][0]] >= 1 && ypoints[a[i][1]] >= 1) {
            count += (xpoints[a[i][0]] - 1) * (ypoints[a[i][1]] - 1);
        }
    }

    return count;
}

int main() {
    int n;
    cin >> n;
    vector<vector<int>> a(n, vector<int>(2));

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 2; j++) {
            cin >> a[i][j];
        }
    }

    cout << RightAngled(a, n) << endl;

    return 0;
}
```

Python Code:

```
def RightAngled(a, n):
    xpoints = {}
    ypoints = {}

    for i in range(n):
        if a[i][0] in xpoints:
            xpoints[a[i][0]] += 1
        else:
            xpoints[a[i][0]] = 1

        if a[i][1] in ypoints:
            ypoints[a[i][1]] += 1
        else:
            ypoints[a[i][1]] = 1

    count = 0
```

```
    for i in range(n):
        if xpoints[a[i][0]] >= 1 and ypoints[a[i][1]] >= 1:
            count += (xpoints[a[i][0]] - 1) * (ypoints[a[i][1]] - 1)

    return count

if __name__ == "__main__":
    n = int(input())
    a = []

    for i in range(n):
        row = list(map(int, input().split()))
        a.append(row)

    print(RightAngled(a, n))
```

Distinct Points

Java Code:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args){
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        int[] x = new int[n];
        int[] y = new int[n];

        for(int i=0;i<n;i++){
            x[i] = scn.nextInt();
        }

        for(int i=0;i<n;i++){
            y[i] = scn.nextInt();
        }
```

```
        System.out.println(Max_Points(x,y));
    }


    static long Max_Points(int[] X, int[] Y){
        HashSet<String> hs = new HashSet<>();

        for(int i=0;i<X.length;i++){
            hs.add(X[i]+" "+Y[i]);
        }

        return hs.size();
    }

}
```

C++ Code:

```
#include <iostream>
#include <vector>
#include <unordered_set>

using namespace std;

long Max_Points(vector<int>& X, vector<int>& Y) {
    unordered_set<string> hs;

    for (int i = 0; i < X.size(); i++) {
        hs.insert(to_string(X[i]) + " " + to_string(Y[i]));
    }

    return hs.size();
}

int main() {
    int n;
    cin >> n;

    vector<int> x(n);
    vector<int> y(n);

    for (int i = 0; i < n; i++) {
        cin >> x[i];
    }

    for (int i = 0; i < n; i++) {
```

```
        cin >> y[i];
    }

    cout << Max_Points(x, y) << endl;

    return 0;
}
```

Python Code:

```
def Max_Points(X, Y):
    hs = set()

    for i in range(len(X)):
        hs.add(str(X[i]) + " " + str(Y[i]))

    return len(hs)

if __name__ == "__main__":
    n = int(input())
    x = list(map(int, input().split()))
    y = list(map(int, input().split()))

    print(Max_Points(x, y))
```

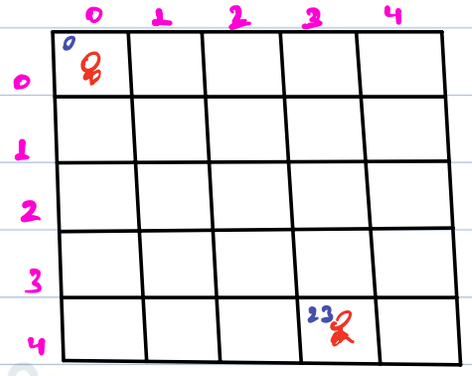
Today's agenda

- Grid illumination
- Brick wall

Leetcode 1001 [Hard]

Q) Grid illumination

Ques Given N×N matrix, lamps[i] = [xi, yi] indicates that grid[xi][yi] is turned on. On lamp illuminates the cell like queen. you will be given queries and answer accordingly.

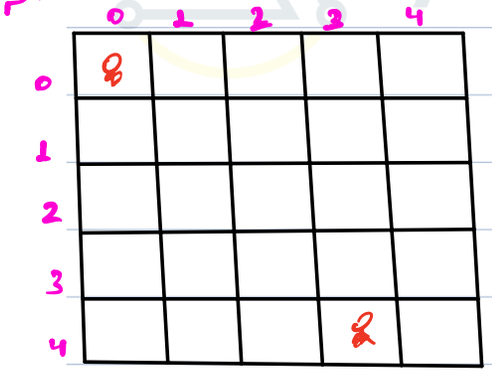


lamps : {(0,0) (4,3)}

queries : {(1,1) (0,0)}
↓ true ↓ false

Idea 1

Brute force



lamps : {(0,0) (4,3)}

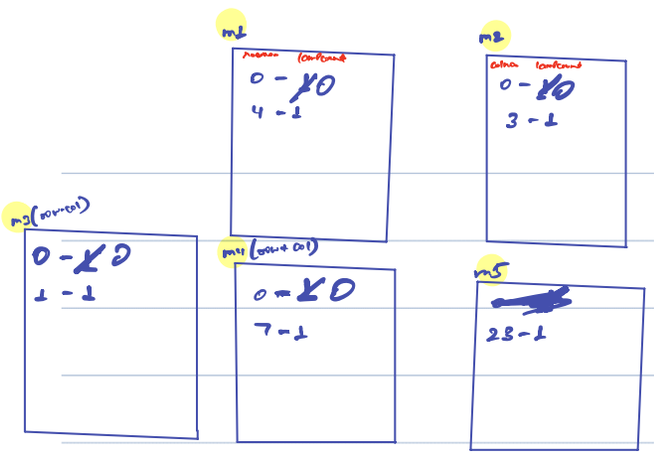
m1
row lampCount
0 - 1
4 - 1

m2
col lampCount
0 - 1
3 - 1

m3 (row-col)
0 - 1
1 - 1

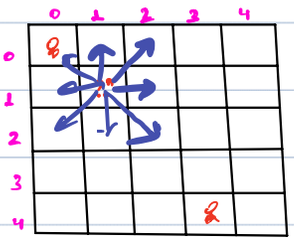
m4 (row+col)
0 - 1
7 - 1

m5
0 - 1
23 - 1



queries = {(1,1) (0,0)}

ans : [1 |]



n = 1
y = 1

(0,0)

```
int [][] dirs = {{-1,0},
                 {-1,1},
                 {0,1},
                 {1,1},
                 {1,0},
                 {1,-1},
                 {0,-1},
                 {-1,-1},
                 {0,0}};

for(d=0; d<dirs.length; d++){
    int n1 = n + dirs[d][0];
    int y1 = y + dirs[d][1];
    if(n1>=0 && y1>=0 && n1<n && y1<n &&
        m5.containsKey(n1*n+y1) == true){
        int times = m5.get(n1*n+y1);
        m1.put(n1, m1.getOrDefault(n1,0) - times);
        m2.put(y1, m2.getOrDefault(y1,0) - times);
        m3.put(n1-y1, m3.getOrDefault(n1-y1,0) - times);
        m4.put(n1+y1, m4.getOrDefault(n1+y1,0) - times);
        m5.remove(n1*n+y1);
    }
}
```

d=0 → n1 = 0, y1 = 1

d=1 → n1 = 0, y1 = 1+1 = 2

d=2 → n1 = 1+0 = 1, y1 = 1+1 = 2

d=3 → n1 = 1+1 = 2, y1 = 1+1 = 2

|

|

|

d=7 → n1 = 1-1 = 0, y1 = 1-1 = 0

times = 1

//Pseudo code

```
int[] gridIlumination(int n, int[][] lamps, int[][] queries){
HashMap<Integer, Integer> m1, m2, m3, m4, m5;

    for(int i=0; i<lamps.length; i++){
        int x= lamps[i][0];
        int y= lamps[i][1];
        m1.put(x, m1.getOrDefault(x,0)+1);
        m2.put(y, m2.getOrDefault(y,0)+1);
        m3.put(x-y, m3.getOrDefault(x-y,0)+1);
        m4.put(x+y, m4.getOrDefault(x+y,0)+1);
        m5.put(x*n+y, m5.getOrDefault(x*n+y,0)+1);
    }
    int[] ans= new length[queries.length];

int[][] dirs = {{-1,0},{-1,1},{0,1},{1,1},{1,0},{1,-1},{0,-1},{-1,-1},{0,0}};

    for(int i=0; i<queries.length; i++){
        int x= queries[i][0];
        int y= queries[i][1];
        if(m1.getOrDefault(x,0)>0 || m2.getOrDefault(y,0)>0
        || m3.getOrDefault(x-y,0)>0 || m4.getOrDefault(x+y,0)>0)
            ans[i]=1;
    }
```

T.C: O(L+Q)

S.C: O(L)

```
        for (d=0; d < dirs.length; d++) {
            int x1 = x + dirs[d][0];
            int y1 = y + dirs[d][1];
            if (x1>=0 && y1>=0 && x1<n && y1<n &&
                m5.containsKey(x1*n+y1) == true) {
                int times = m5.get(x1*n+y);
                m1.put(x1, m1.getOrDefault(x1,0) - times);
                m2.put(y1, m2.getOrDefault(y1,0) - times);
                m3.put(x1+y1, m3.getOrDefault(x1+y1,0) - times);
                m4.put(x1-y1, m4.getOrDefault(x1-y1,0) - times);
                m5.remove(x1*n+y1);
            }
        }
    }
    return ans;
}
```

```
for (int i=0; i< lamps.length; i++) {
    int x = lamps[i][0];
    int y = lamps[i][1];
    m1.put(x, m1.getOrDefault(x,0)+1);
    m2.put(y, m2.getOrDefault(y,0)+1);
    m3.put(x-y, m3.getOrDefault(x-y,0)+1);
    m4.put(x+y, m4.getOrDefault(x+y,0)+1);
}
```

	0	1
0	1	1
1	1	2
2	3	1
3	5	3
4	7	1

i →

m3

6 - 1
7 - 1
16 - 1

x = 1
y = 2

m1

row | rowCount

1 - X2

m2

col | colCount

1 - 1
2 - 1

m3 (row-col)

0 - 1
-1 - 1

m4 (row+col)

2 - 1
3 - 1

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19
4	20	21	22	23	24

→ (i,j) → i*N + j

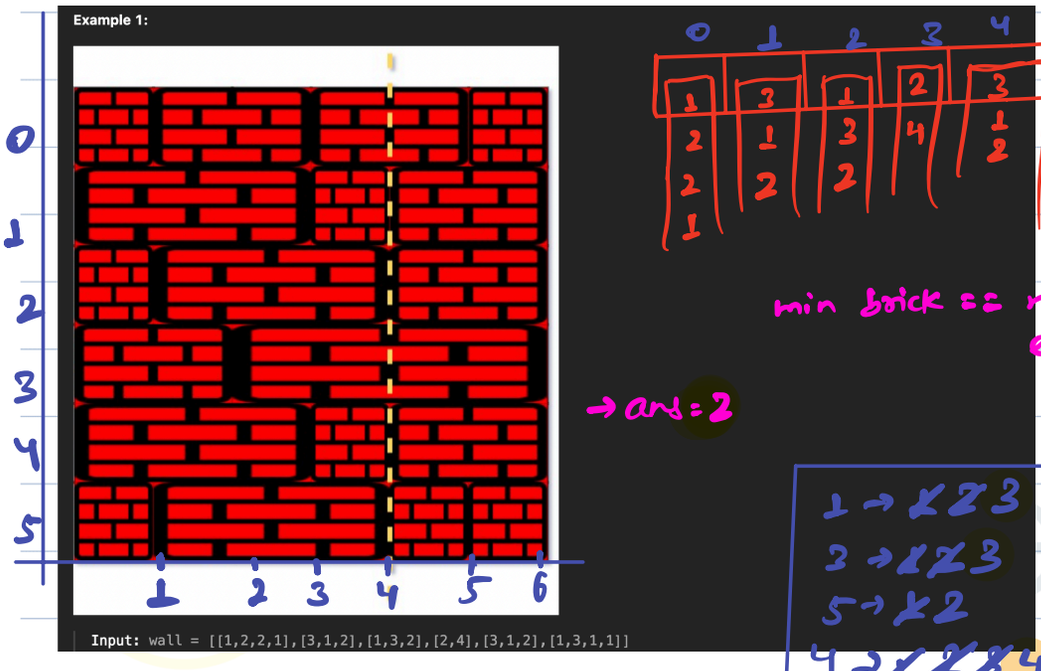
(3,2) → 3*5 + 2 = 17

(1,3) → 1*5 + 3 = 8

leetcode 554 (med)

Q) Brick wall

↳ Given a rectangular brick, Draw a vertical line cutting least bricks.



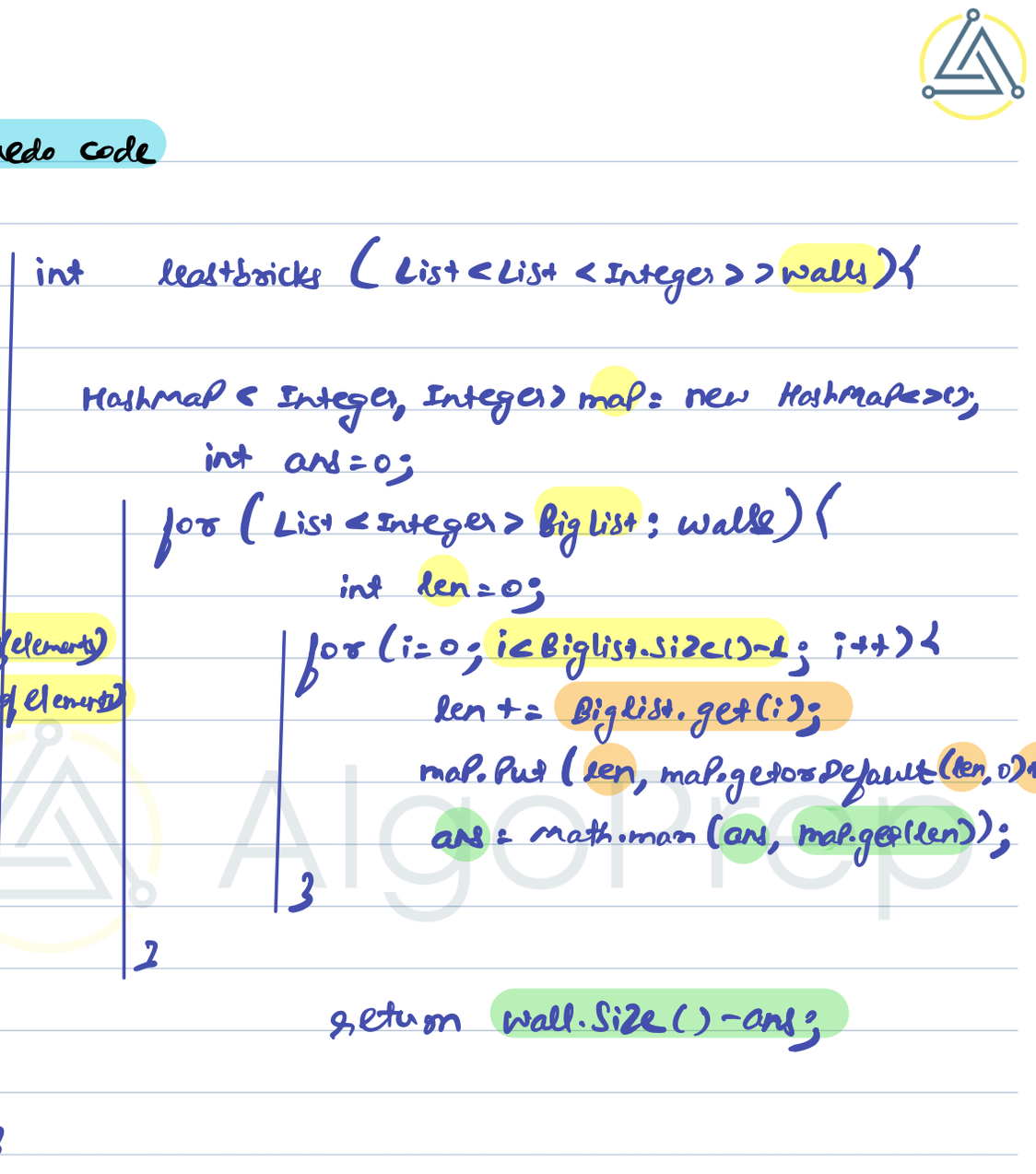
min brick = = max no. of empty space

→ ans = 2

1 → x x 3
3 → x x 3
5 → x 2
4 → x x x 4

2 - 1

ans = 6 - 4 = 2

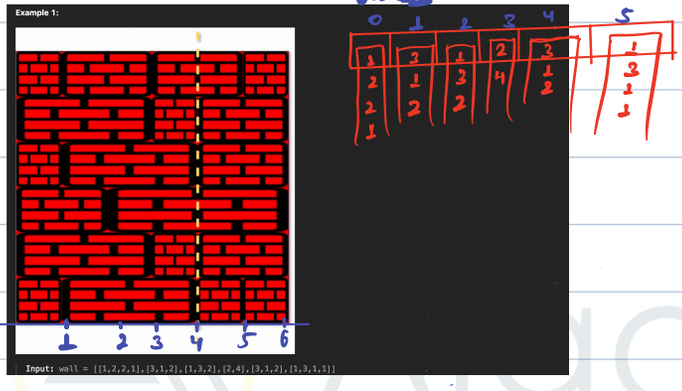

//Pseudo code
int leastbricks (List<List<Integer>> walls){
HashMap<Integer, Integer> map = new HashMap<>();
int ans = 0;
for (List<Integer> Biglist : walls){
int len = 0;
for (i=0; i<Biglist.size()-1; i++){
len += Biglist.get(i);
map.Put (len, map.getorDefault(len, 0)+1);
ans = math.max (ans, map.get(len));
}
}
return wall.size() - ans;
}
T.C: O(no. of elements)
S.C: O(no. of elements)

```
HashMap<Integer, Integer> map = new HashMap<>();
int ans = 0;
for (List<Integer> Biglist : walls) {
    int len = 0;
    for (i = 0; i < Biglist.size()-1; i++) {
        len += Biglist.get(i);
        map.Put (len, map.getorDefault(len, 0)+1);
        ans = Math.max (ans, mp.get (len))
    }
}
```

1 - 1 2 3 2 - 1
3 - 1 2 3
5 - 1 2
4 - 1 2 3 4

ans = 0 → 4

walls



→ Biglist: 1 3 1 1

len = 0 + 1 + 3 = 4 + 1 = 5

→ Biglist: 1 2 2 1

len = 0 + 1 + 2 = 3 + 2 = 5

→ Biglist: 3 1 2

len = 0 + 2 + 1 = 3

→ Biglist: 1 3 2

len = 0 + 1 = 1 + 3 = 4

→ Biglist: 2 4

len = 0 + 2

→ Biglist: 3 1 2

len = 0 + 3 + 1 = 4

Q) **Longest consecutive sequence**

Given arr[N] ele, find the length of longest sequence which can be rearranged in a strictly increasing by 1.

Ex: arr[]: {-1 8 5 3 10 2 4 9}
(0 1 2 3 4 5 6 7)

arr[]: {3 8 2 1 9 6 5 6 7 2}
(0 1 2 3 4 5 6 7 8 9)

Grid illumination

Java Code:

```
class Solution {
    public int[] gridIllumination(int n, int[][] lamps, int[][] queries) {
        HashMap<Integer,Integer> m1,m2,m3,m4,m5;
         m1 = new HashMap<>();
         m2 = new HashMap<>();
         m3 = new HashMap<>();
         m4 = new HashMap<>();
         m5 = new HashMap<>();
        for(int i =0; i<lamps.length; i++){
            int x = lamps[i][0];
            int y = lamps[i][1];
            m1.put(x , m1.getOrDefault(x,0) + 1);
            m2.put(y , m2.getOrDefault(y,0) + 1);
            m3.put(x-y , m3.getOrDefault(x-y,0) + 1);
            m4.put(x+y , m4.getOrDefault(x+y,0) + 1);
            m5.put(x*n+y , m5.getOrDefault(x*n+y,0) + 1);
        }
        int[] ans = new int[queries.length];
        int[][] dirs = {{-1,0},{-1,1},{0,1},{1,1},{1,0},{1,-1},{0,-1},{-1,-1},{0,0}};

        for(int i = 0; i<queries.length; i++){
            int x = queries[i][0];
            int y = queries[i][1];
            if(m1.getOrDefault(x,0)>0 || m2.getOrDefault(y,0)>0 || m3.getOrDefault(x-y,0)>0 ||
m4.getOrDefault(x+y,0)>0){
                ans[i] = 1;
            }
            for(int d= 0; d<dirs.length; d++){
                int x1 = x+dirs[d][0];
                int y1 = y+dirs[d][1];
                if(x1>=0 && y1>=0 && x1<n && y1<n && m5.containsKey(x1*n+y1)==true){
                    int times = m5.get(x1*n+y1);
                     m1.put(x1 , m1.getOrDefault(x1,0)-times);
                     m2.put(y1 , m2.getOrDefault(y1,0)-times);
                     m3.put(x1-y1 , m3.getOrDefault(x1-y1,0)-times);
                     m4.put(x1+y1 , m4.getOrDefault(x1+y1,0)-times);
                     m5.remove(x1*n+y1);
                }
```

```
        }

      }
      return ans;
   }
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

class Solution {
public:
   vector<int> gridIllumination(int n, vector<vector<int>>& lamps, vector<vector<int>>& queries)
{
      unordered_map<int, int> m1, m2, m3, m4, m5;
      for (int i = 0; i < lamps.size(); i++) {
         int x = lamps[i][0];
         int y = lamps[i][1];
         m1[x]++;
         m2[y]++;
         m3[x - y]++;
         m4[x + y]++;
         m5[x * n + y]++;
      }

      vector<int> ans(queries.size(), 0);
      vector<vector<int>> dirs = {{-1, 0}, {-1, 1}, {0, 1}, {1, 1}, {1, 0}, {1, -1}, {0, -1}, {-1, -1}, {0, 0}};

      for (int i = 0; i < queries.size(); i++) {
         int x = queries[i][0];
         int y = queries[i][1];

         if (m1[x] > 0 || m2[y] > 0 || m3[x - y] > 0 || m4[x + y] > 0) {
            ans[i] = 1;
         }

         for (int d = 0; d < dirs.size(); d++) {
            int x1 = x + dirs[d][0];
            int y1 = y + dirs[d][1];

            if (x1 >= 0 && y1 >= 0 && x1 < n && y1 < n && m5.count(x1 * n + y1) > 0) {
               int times = m5[x1 * n + y1];
```

```
                m1[x1] -= times;
                m2[y1] -= times;
                m3[x1 - y1] -= times;
                m4[x1 + y1] -= times;
                m5.erase(x1 * n + y1);
            }
        }
    }

    return ans;
  }
};

int main() {
  Solution solution;
  int n = /* size of grid */;
  vector<vector<int>> lamps = { /* lamps coordinates */ };
  vector<vector<int>> queries = { /* queries coordinates */ };

  vector<int> result = solution.gridIllumination(n, lamps, queries);

  // Print the result
  for (int res : result) {
    cout << res << " ";
  }

  return 0;
}
```

Python Code:

```
from typing import List

class Solution:
  def gridIllumination(self, n: int, lamps: List[List[int]], queries: List[List[int]]) -> List[int]:
    m1, m2, m3, m4, m5 = {}, {}, {}, {}, {}

    for lamp in lamps:
      x, y = lamp[0], lamp[1]
      m1[x] = m1.get(x, 0) + 1
      m2[y] = m2.get(y, 0) + 1
      m3[x - y] = m3.get(x - y, 0) + 1
      m4[x + y] = m4.get(x + y, 0) + 1
      m5[x * n + y] = m5.get(x * n + y, 0) + 1
```

```
        ans = [0] * len(queries)
        dirs = [[-1, 0], [-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1], [0, 0]]

        for i in range(len(queries)):
            x, y = queries[i][0], queries[i][1]

            if m1.get(x, 0) > 0 or m2.get(y, 0) > 0 or m3.get(x - y, 0) > 0 or m4.get(x + y, 0) > 0:
                ans[i] = 1

            for d in dirs:
                x1, y1 = x + d[0], y + d[1]
                if 0 <= x1 < n and 0 <= y1 < n and (x1 * n + y1) in m5:
                    times = m5[x1 * n + y1]
                    m1[x1] -= times
                    m2[y1] -= times
                    m3[x1 - y1] -= times
                    m4[x1 + y1] -= times
                    del m5[x1 * n + y1]

        return ans

# Example usage:
solution = Solution()
n = /* size of grid */
lamps = [ /* lamps coordinates */ ]
queries = [ /* queries coordinates */ ]

result = solution.gridIllumination(n, lamps, queries)
print(result)
```

Brick wall

Java Code:

```
class Solution {
    public int leastBricks(List<List<Integer>> wall) {
        HashMap<Integer,Integer> map = new HashMap<>();
        int ans = 0;
        for(List<Integer> BigList : wall){
            int len = 0;
            for(int i = 0; i<BigList.size()-1; i++){
                len+=BigList.get(i);
                map.put(len,map.getOrDefault(len,0)+1);
```

```
                ans = Math.max(ans,map.get(len));
            }
        }
        return wall.size()-ans;
    }
}
```

C++ Code:

```
#include <iostream>
#include <vector>
#include <unordered_map>
using namespace std;

class Solution {
public:
    int leastBricks(vector<vector<int>>& wall) {
        unordered_map<int, int> map;
        int ans = 0;

        for (const auto& row : wall) {
            int len = 0;
            for (int i = 0; i < row.size() - 1; i++) {
                len += row[i];
                map[len]++;
                ans = max(ans, map[len]);
            }
        }

        return wall.size() - ans;
    }
};

int main() {
    Solution solution;
    vector<vector<int>> wall = { /* your input wall */ };

    int result = solution.leastBricks(wall);

    cout << result << endl;

    return 0;
}
```

Python Code:

```
from typing import List
from collections import defaultdict

class Solution:
    def least_bricks(self, wall: List[List[int]]) -> int:
        map = defaultdict(int)
        ans = 0

        for row in wall:
            length = 0
            for i in range(len(row) - 1):
                length += row[i]
                map[length] += 1
                ans = max(ans, map[length])

        return len(wall) - ans

# Example usage:
solution = Solution()
wall = [ /* your input wall */ ]
result = solution.least_bricks(wall)
print(result)
```

Longest Consecutive Sequence

Solution vid:

https://youtu.be/pmFNn5ueXxE

Java Code:

```
class Solution {
    public int longestConsecutive(int[] nums) {
        HashMap<Integer , Integer> hm = new HashMap<>();
        int maxlen = 0;
        for(int i =0; i<nums.length; i++){
            int num = nums[i];
            if(!hm.containsKey(num)){
                    int sp = num;
                    int ep = num;
```

```
                if(hm.containsKey(num-1)) sp = sp - hm.get(num-1);

                if(hm.containsKey(num+1)) ep = ep + hm.get(num+1);

                int l = ep-sp+1;
                hm.put(sp, l);
                hm.put(ep, l);
                if(sp != num && ep != num) hm.put(num , 1);
                maxlen = Math.max(maxlen , l);


            }
        }

        return maxlen;
    }
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
        unordered_map<int, int> hm;
        int maxlen = 0;

        for (int i = 0; i < nums.size(); i++) {
            int num = nums[i];
            if (hm.find(num) == hm.end()) {
                int sp = num;
                int ep = num;

                if (hm.find(num - 1) != hm.end()) sp -= hm[num - 1];
                if (hm.find(num + 1) != hm.end()) ep += hm[num + 1];

                int l = ep - sp + 1;
                hm[sp] = l;
                hm[ep] = l;

                if (sp != num && ep != num) hm[num] = 1;
```

```
            maxlen = max(maxlen, l);
        }
    }

    return maxlen;
  }
};

int main() {
  Solution solution;
  vector<int> nums = { /* your input array */ };

  int result = solution.longestConsecutive(nums);

  cout << result << endl;

  return 0;
}
```

Python Code:

```
class Solution:
  def longestConsecutive(self, nums):
    hm = {}
    maxlen = 0

    for num in nums:
      if num not in hm:
        sp = ep = num

        if num - 1 in hm:
          sp -= hm[num - 1]
        if num + 1 in hm:
          ep += hm[num + 1]

        length = ep - sp + 1
        hm[sp] = length
        hm[ep] = length

        if sp != num and ep != num:
          hm[num] = 1

        maxlen = max(maxlen, length)

    return maxlen
```

```
# Example usage:
solution = Solution()
nums = [ /* your input array */ ]
result = solution.longestConsecutive(nums)
print(result)
```

Today's agenda

- Subarray Sum equals K
- Same differences
- Subarray with equal 0 and 1.
- Subarray with equal 0, 1 and 2. -> {Hard}

-> PrepPo.com

Q) Subarray sum equals K

↳ Given arr[n] and an integer K, return total number of continuous subarrays whose sum equals to K.

Ex: arr[3] = {1 1 1} K=2 ans=2

Idea 1

↳ check all subarrays and count if sum == K.

T.C: $O(N^2)$ S.C: $O(N)$

Idea 2 → fix ep and count valid sp in O(1).

K=4

	0	1	2	3	4	5	6	7	8	9	10 (ep)
arr[] =	2	3	-1	4	-5	1	6	-4	3	-1	4
Psum[] =	2	5	4	8	3	4	10	6	9	8	12

Sum(sp, ep) == K

ep=4

Psum[ep] − Psum[sp-1] == K

3 − Psum[sp-1] = 4

−1 = Psum[sp-1]

K=4

	0	1	2	3	4	5	6	7	8	9	10
arr[] =	2	3	-1	4	-5	1	6	-4	3	-1	4
Psum[] =	2	5	4	8	3	4	10	6	9	8	12

sp-1 sp-1 ep

Psum[ep] - Psum[sp-1] == K

Psum[ep] - K = Psum[sp-1]

8 - 4 = Psum[sp-1] = 4

K=4

	0	1	2	3	4	5	6	7	8	9	10
arr[] =	2	3	-1	4	-5	1	6	-4	3	-1	4
Psum[] =	2	5	4	8	3	4	10	6	9	8	12

ep

ep	arr[ep]	Psum[ep] - K = Psum[sp-1]	valid sp
0	2	-2	0
1	5	1	0
2	4	0	1
3	8	4	1
4	3	-1	0
5	4	0	1
6	10	6	0

map

0 - 1	3 - 1
2 - 1	10 - 1
5 - 1	
4 - 2	
8 - 1	

//Pseudo code

```
int subarraySum (int arr[N], int k){

    HashMap<Integers, Integers> map;
    map.put(0,1);
    int ans = 0;

T.C: O(N)
S.C: O(N)
    int[] Psum = func1(arr);

    for(int ep=0; ep<n; ep++){
        int diff = Psum[ep] - k;
        ans = ans + map.getOrDefault(diff,0);

        map.put(Psum[ep], map.getOrDefault(Psum[ep],0)
                                                +1);
    }

    return ans;
}
```

```
int subArraySum (int arr[], int k){

    HashMap<Integer, Integer> map;
    map.put(0,1);
    int ans = 0;

    int[] Psum = func1(arr);

    for(int ep=0; ep<n1; ep++){
        int diff = Psum[ep] - k;
        ans = ans + map.getOrDefault(diff,0);

        map.put(Psum[ep], map.getOrDefault(Psum[ep],0)+1);
    }

    return ans;
}
```

	-1	sp 0	1	ep 2	3	4	5	6	7	8	9	10
arr[] =	0	2	3	-1	4	-5	1	6	-4	3	-1	4
Psum[] =	0	2	5	4	8	3	4	10	6	9	8	12

ep	arr[ep]	Psum[ep] - k = Psum[sp-1]	valid sp
2	4	0	0

map

0-1
2-1
5-1

Codeforces round 719

Q) Same differences

↳ Given arr[N], Count the number of Pairs of indices (i,j) such that $i<j$ and $a_j - a_i = j - i$

	0	1	2	3	4	5
arr[6] = {	3	5	1	4	6	6 }

$a_j - a_i = j - i \Rightarrow a_j - j = a_i - i$

Ideal → for every j, iterate and find valid i.

	0	1	2	3	4	5 (j)
arr[6] = {	3	5	1	4	6	6 }

j	$a_j - j$	valid i
0	3	0
1	4	0
2	-1	0
3	1	0
4	2	0
5	1	1

ans: 1

T.C: $O(N^2)$ S.C: $O(1)$

//idea2

arr[6] = { 3 5 1 4 6 6 }

(0 1 2 3 4 5 — j)

j	aj - j	valid i
0	3	0
1	4	0
2	-1	0
3	1	0
4	2	0
5	1	1

ans = 1

3 - 1
4 - 1
-1 - 1
1 - 2
2 - 1

//Pseudo Code

```
int Samedifferences(int arr[N]){

    HashMap<Integer, Integer> map;
    int ans = 0;
    for (int j=0; j<N; j++) {
        int diff = arr[j] - j;
        ans = ans + map.getOrDefault(diff, 0);

        map.put(diff, map.getOrDefault(diff,0)+1);
    }
    return ans;
}
```

T.C: O(N)

S.C: O(N)

Break till 9:12 PM

Q) Count Subarrays with equal number of 1's and 0's.

↳ Given arr[n] with 0's and 1's only. find the number of subarrays with equal number of 0's and 1's.

Ex: arr[4] = {1 0 0 1} ⇝ ans = 3

Idea 1

↳ Replace 0 with -1 and find number of subarrays with sum == 0.

arr[4] = {1 0 0 1}

arr[4] = {1 -1 -1 1}

T.C: O(n) S.C: O(n)

Idea 2

	0	1 (sP)	2	3	4	5 (eP)	6	7
arr[8]:	0	1	0	0	1	0	0	1
PC0[8]:	1	1	2	3	3	4	5	5
PC1[8]:	0	1	1	1	2	2	2	3

Count 0 (sP, eP) == Count 1 (sP, eP)

PC0[eP] - PC0[sP-1] == PC1[eP] - PC1[sP-1]

PC0[eP] - PC1[eP] == PC0[sP-1] - PC1[sP-1]

$pco[ep] - pci[ep] :: pco[sp-1] - pci[sp-1]$

	0	1	2	3	4	5	6	7 (ep)
arr[8]:	0	1	0	0	1	0	0	1
pco[8]:	1	1	2	3	3	4	5	5
pci[8]:	0	1	1	1	2	2	2	3

ep	pco[ep] - pci[ep]	valid sp-1	
0	1 - 0 = 1	0	
1	1 - 1 = 0	1	
2	2 - 1 = 1	1	
3	3 - 1 = 2	0	ans = 7
4	3 - 2 = 1	2	
5	4 - 2 = 2	1	
6	5 - 2 = 3	0	
7	5 - 3 = 2	2	

Ls if iterate for every ep, T.C : $O(N^2)$

Using HashMap

```
                                    ep
          0  1  2  3  4  5  6  7
arr[8]: { 0  1  0  0  1  0  0  1 }
pc0[8]:   1  1  2  3  3  4  5  5
pc1[8]:   0  1  1  1  2  2  2  3
```

ep	pc0[ep] - pc1[ep]	valid sp-1
0	1	0
1	0	1
2	1	1
3	2	0
4	1	2
5	2	1

map

0 - 2
1 - 2 2 3
2 - 2

//Pseudo Code

```
int countSubarray01(int arr[N]){

    HashMap<Integer, Integer> map;

    int[] pc0 = func0(arr);
    int[] pc1 = func0(arr);
    int ans = 0;
    map.put(0,1);

    for(int ep=0; ep<N; ep++){
        int diff = pc0[ep] - pc1[ep];

        ans = ans + map.getOrDefault(diff, 0);

        map.put(diff, map.getOrDefault(diff,0)+1);

    }

    return ans;

}
```

T.C: O(N)

S.C: O(N)

Q) Count Subarrays with equal number of 0's and 1's and 2's.

↳ Given arr[n] with 0's, 1's and 2's. find the number of subarrays with equal number of 0's, 1's and 2's.

Ex: arr[7]: { 0 1 0 2 0 1 0 } → ans: 2

//idea 1 → incorrect idea

↳ Replace 2 with -1 and find the subarray with sum == 0. ⇝ { -1 0 1 }

arr[3] = { 0 0 0 } ← rep → ans = 0

1 + 2 + 3 = 6

//idea 2

arr[7]: { 0 1 0 2 0 1 0 }

pc0[7]: 1 1 2 2 3 3 4

pc1[7]: 0 1 1 1 1 2 2

pc2[7]: 0 0 0 1 1 1 1

	0	1	2	3	4	5	6
arr[7] : {	0	1	0	2	0	1	0 }
PC0[7] :	1	1	2	2	3	3	4
PC1[7] :	0	1	1	1	1	2	2
PC2[7] :	0	0	0	1	1	1	1

"1@1" "0@1" "1@2" "1@1" "2@1" "1@2" "2@3"

ep	PC0[ep] - PC1[ep]	PC0[ep] - PC2[ep]	valid sp
0	1-0 = 1	1	0
1	0	1	0
2	1	2	0
3	1	1	1
4	2	2	0
5	1	2	1
6	2	3	0

//Pseudo code

```
int countSubarray012 ( int arr[N]){
    HashMap<String, Integer> map;

    int[] pc0 = func0(arr);
    int[] pc1 = func1(arr);
    int[] pc2 = func2(arr);

T.C: O(N)
S.C: O(N)
    int ans = 0;
    map.put("0@0",1);

    for(int ep=0; ep<N; ep++){
        int diff1 = pc0[ep] - pc1[ep];
        int diff2 = pc0[ep] - pc2[ep];
        String diff = diff1 + "@" + diff2;

        ans = ans + map.getOrDefault(diff,0);

        map.put(diff, map.getOrDefault(diff,0)+1);
    }

    return ans;
}
```

```
int countSubarray012 ( int arr[n]){
    HashMap<String, Integer> map;

    int[] pc0 = func0(arr);
    int[] pc1 = func1(arr);
    int[] pc2 = func2(arr);

    int ans = 0;
    map.put("0@0",1);

    for(int ep=0; ep<n; ep++){
        int diff1 = pc0[ep] - pc1[ep];
        int diff2 = pc0[ep] - pc2[ep];
        String diff = diff1 + "@" + diff2;

        ans = ans + map.getOrDefault(diff,0);

        map.put(diff, map.getOrDefault(diff,0)+1);
    }

    return ans;
}
```

ep	0	1	2	3	4	5	6
arr[7] :	0	1	0	2	0	1	0
pc0[7] :	1	1	2	2	3	3	4
pc1[7] :	0	1	1	1	1	2	2
pc2[7] :	0	0	0	1	1	1	1
	"1@1"	"0@1"	"1@2"	"1@1"	"2@1"	"1@2"	"2@2"

ans = 0+0+0+0
+1+0+1
= 2

"0@0" → 1
"1@1" → 2
"0@1" → 1
"1@2" → 2
"2@2" → 1

2@3 → 1

Subarray sum equals k

Java Code:

```
class Solution {
    public int subarraySum(int[] nums, int k) {
        int n = nums.length;
        HashMap<Integer,Integer> map = new HashMap<>();
        map.put(0,1);
        int ans = 0;
        int[] psum = new int[n];
        psum[0] = nums[0];
        for(int i= 1; i<n; i++){
            psum[i] = psum[i-1]+nums[i];
        }

        for(int ep = 0; ep<n; ep++){
            int diff = psum[ep]-k;
            ans = ans + map.getOrDefault(diff,0);
            map.put(psum[ep],map.getOrDefault(psum[ep],0)+1);
        }
        return ans;
    }
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

class Solution {
public:
    int subarraySum(vector<int>& nums, int k) {
        int n = nums.size();
        unordered_map<int, int> map;
        map[0] = 1;
        int ans = 0;
        vector<int> psum(n);
        psum[0] = nums[0];

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

```
            psum[i] = psum[i - 1] + nums[i];
        }

        for (int ep = 0; ep < n; ep++) {
            int diff = psum[ep] - k;
            ans += map[diff];
            map[psum[ep]]++;
        }

        return ans;
    }
};

int main() {
    Solution solution;
    vector<int> nums = { /* your input array */ };
    int k = /* your target sum */;

    int result = solution.subarraySum(nums, k);

    cout << result << endl;

    return 0;
}
```

Python Code:

```
from typing import List
from collections import defaultdict

class Solution:
    def subarraySum(self, nums: List[int], k: int) -> int:
        n = len(nums)
        map = defaultdict(int)
        map[0] = 1
        ans = 0
        psum = [0] * n
        psum[0] = nums[0]

        for i in range(1, n):
            psum[i] = psum[i - 1] + nums[i]

        for ep in range(n):
            diff = psum[ep] - k
            ans += map[diff]
```

```
            map[psum[ep]] += 1

        return ans

# Example usage:
solution = Solution()
nums = [ /* your input array */ ]
k = /* your target sum */
result = solution.subarraySum(nums, k)
print(result)
```

Same Difference

Java Code:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
         Scanner scn = new Scanner(System.in);
        int t = scn.nextInt();
        for(int j = 0; j<t; j++){
            int n = scn.nextInt();
            int[] arr = new int[n];
            for(int i = 0; i<n; i++){
                arr[i] = scn.nextInt();
            }
            System.out.println(sameDifference(arr,n));
        }
    }
    public static int sameDifference(int[] arr,int n){
      HashMap<Integer,Integer> map = new HashMap<>();
      int ans = 0;
        for(int j = 0; j<n; j++){
            int diff = arr[j]-j;
            ans += map.getOrDefault(diff,0);
            map.put(diff,map.getOrDefault(diff,0)+1);
        }
        return ans;

    }
```

```
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
using namespace std;

int sameDifference(int arr[], int n) {
    unordered_map<int, int> map;
    int ans = 0;

    for (int j = 0; j < n; j++) {
        int diff = arr[j] - j;
        ans += map[diff];
        map[diff]++;
    }

    return ans;
}

int main() {
    int t;
    cin >> t;

    for (int j = 0; j < t; j++) {
        int n;
        cin >> n;
        int arr[n];

        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }

        cout << sameDifference(arr, n) << endl;
    }

    return 0;
}
```

Python Code:

```
def same_difference(arr, n):
```

```
    map = {}
    ans = 0

    for j in range(n):
        diff = arr[j] - j
        ans += map.get(diff, 0)
        map[diff] = map.get(diff, 0) + 1

    return ans

def main():
    t = int(input())

    for _ in range(t):
        n = int(input())
        arr = list(map(int, input().split()))
        print(same_difference(arr, n))

if __name__ == "__main__":
    main()
```

Subarrays with equals 0s and 1s

Java Code:

```
class Solution
{
    //Function to count subarrays with 1s and 0s.
    static int countSubarrWithEqualZeroAndOne(int arr[], int n)
    {
        HashMap<Integer,Integer> map= new HashMap<>();
        int[] pc0 = funcc(arr,0);
        int[] pc1 = funcc(arr,1);
        int ans = 0;
        map.put(0,1);

        for(int ep = 0; ep<n; ep++){
            int diff = pc1[ep] - pc0[ep];

            ans = ans + map.getOrDefault(diff,0);
            map.put(diff,map.getOrDefault(diff,0)+1);
        }
```

```
        return ans;
    }
    public static int[] funcc(int[] arr , int num){
        int n = arr.length;
        int[] ans = new int[n];
        int c = 0;
        for(int i = 0; i<n; i++){
            if(arr[i] == num){
                c++;
            }
            ans[i] = c;
        }
        return ans;
    }
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

class Solution {
public:
    static int countSubarrWithEqualZeroAndOne(int arr[], int n) {
        unordered_map<int, int> map;
        vector<int> pc0 = funcc(arr, n, 0);
        vector<int> pc1 = funcc(arr, n, 1);
        int ans = 0;
        map[0] = 1;

        for (int ep = 0; ep < n; ep++) {
            int diff = pc1[ep] - pc0[ep];
            ans += map[diff];
            map[diff]++;
        }

        return ans;
    }

    static vector<int> funcc(int arr[], int n, int num) {
        vector<int> ans(n, 0);
        int c = 0;
```

```
        for (int i = 0; i < n; i++) {
            if (arr[i] == num) {
                c++;
            }
            ans[i] = c;
        }

        return ans;
    }
};

int main() {
    int n;
    cin >> n;
    int arr[n];

    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cout << Solution::countSubarrWithEqualZeroAndOne(arr, n) << endl;

    return 0;
}
```

Python Code:

```
class Solution:
    @staticmethod
    def countSubarrWithEqualZeroAndOne(arr, n):
        def funcc(arr, num):
            ans = [0] * n
            c = 0
            for i in range(n):
                if arr[i] == num:
                    c += 1
                ans[i] = c
            return ans

        map = {}
        pc0 = funcc(arr, 0)
        pc1 = funcc(arr, 1)
        ans = 0
        map[0] = 1
```

```
        for ep in range(n):
            diff = pc1[ep] - pc0[ep]
            ans += map.get(diff, 0)
            map[diff] = map.get(diff, 0) + 1

        return ans


# Example usage:
n = int(input())
arr = list(map(int, input().split()))
print(Solution.countSubarrWithEqualZeroAndOne(arr, n))
```

Equals 0 ,1 and 2

Java Code:

```
//User function Template for Java
class Solution
{
    long getSubstringWithEqual012(String str)
    {
        HashMap<String,Long> map= new HashMap<>();
        int n = str.length();
        int[] pc0 = funcc(str,'0');
        int[] pc1 = funcc(str,'1');
        int[] pc2 = funcc(str,'2');
        long ans = 0;
        map.put("0@0",1L);

        for(int ep = 0; ep<n; ep++){
            int diff1 = pc0[ep] - pc1[ep];
            int diff2 = pc0[ep] - pc2[ep];
            String diff = diff1+"@"+diff2;
            ans = ans + map.getOrDefault(diff,0L);
            map.put(diff,map.getOrDefault(diff,0L)+1);
        }
        return ans;
    }
    public static int[] funcc(String str , char num){
        int n = str.length();
```

```
        int[] ans = new int[n];
        int c = 0;
        for(int i = 0; i<n; i++){
            if(str.charAt(i) == num){
                c++;
            }
            ans[i] = c;
        }
        return ans;
    }
}
```

C++ Code:

```
#include <iostream>
#include <unordered_map>
using namespace std;

class Solution {
public:
    long getSubstringWithEqual012(string str) {
        unordered_map<string, long> map;
        int n = str.length();
        vector<int> pc0 = funcc(str, '0');
        vector<int> pc1 = funcc(str, '1');
        vector<int> pc2 = funcc(str, '2');
        long ans = 0;
        map["0@0"] = 1;

        for (int ep = 0; ep < n; ep++) {
            int diff1 = pc0[ep] - pc1[ep];
            int diff2 = pc0[ep] - pc2[ep];
            string diff = to_string(diff1) + "@" + to_string(diff2);
            ans = ans + map[diff];
            map[diff]++;
        }

        return ans;
    }

    vector<int> funcc(string str, char num) {
        int n = str.length();
        vector<int> ans(n, 0);
        int c = 0;

        for (int i = 0; i < n; i++) {
```

```
            if (str[i] == num) {
                c++;
            }
            ans[i] = c;
        }

        return ans;
    }
};

int main() {
    Solution solution;
    string str;
    cin >> str;

    cout << solution.getSubstringWithEqual012(str) << endl;

    return 0;
}
```

Python Code:

```
class Solution:
    def getSubstringWithEqual012(self, s: str) -> int:
        def funcc(s, num):
            ans = [0] * len(s)
            c = 0
            for i in range(len(s)):
                if s[i] == num:
                    c += 1
                ans[i] = c
            return ans

        map = {}
        n = len(s)
        pc0 = funcc(s, '0')
        pc1 = funcc(s, '1')
        pc2 = funcc(s, '2')
        ans = 0
        map["0@0"] = 1

        for ep in range(n):
            diff1 = pc0[ep] - pc1[ep]
            diff2 = pc0[ep] - pc2[ep]
            diff = f"{diff1}@{diff2}"
```

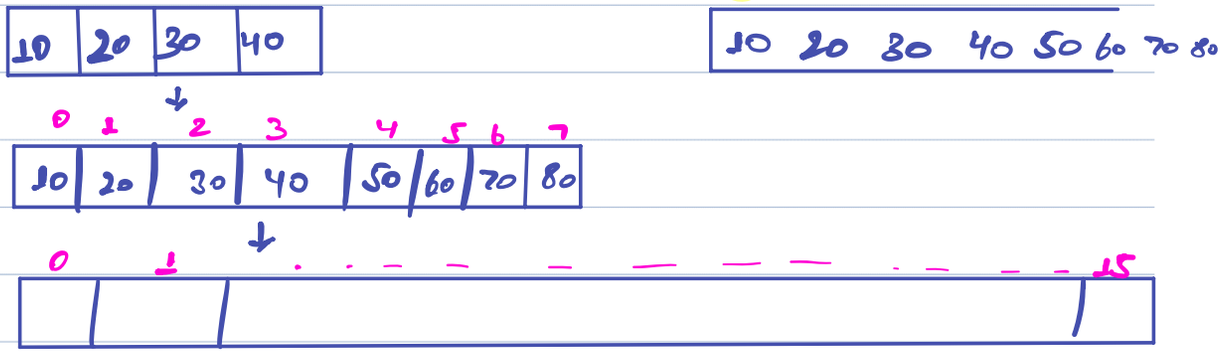
```
            ans += map.get(diff, 0)
            map[diff] = map.get(diff, 0) + 1

        return ans


# Example usage:
solution = Solution()
s = input()
print(solution.getSubstringWithEqual012(s))
```

Today's agenda

- ↳ ArrayList internal
- ↳ HashMap Construction

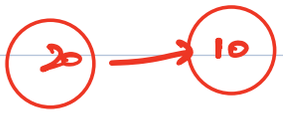
→ Arraylist : Dynamic Array

↓ Array



1st add	→	10	→ O(1)	
2nd add	→	20	→ O(1)	
3rd add	→	30	→ O(1)	
4th add	→	40	→ O(1)	
5th add	→	50	→ O(5)	O(2)
6th add	→	60	→ O(1)	O(2)
7th add	→	70	→ O(1)	O(2)
8th add	→	80	→ O(1)	O(2)
9th add	→	90	→ O(8+1)	→ O(2)
10th add			→ O(1)	→ O(2)
			→ O(1)	→ O(2)
			→ O(1)	
			⋮	⋮
16th add			O(1)	→ O(2)

// LinkedList



↳ LinkedList<Integer> ll = new LinkedList<>();

↳ ll.addLast(10); → ll.add(10);

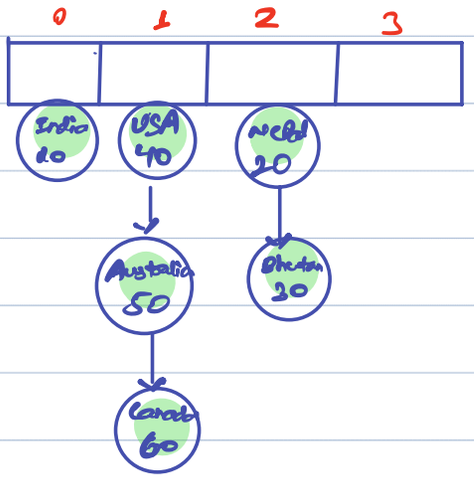
↳ ll.addFirst(20);

```
Public class HMNode{
    int val;
    String s;
    —CONS.
}
```

↳ LinkedList<HMNode> ll = new LinkedList<>();

ll.add(new HMNode(10, "Hello"));

HashMap Construction → Array of LinkedList.



Put →

ContainsKey →

"India" → 10

"Nepal" → 20

"Bhutan" → 30

"USA" → 40

"Australia" → 50

"Canada" → 60

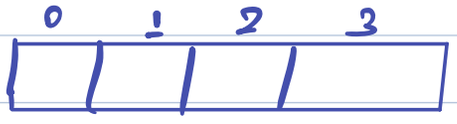
"Canada" → 70

→ λ = load factor = $\frac{\text{No. of elements}}{\text{Size of Array}} = \frac{6}{4}$

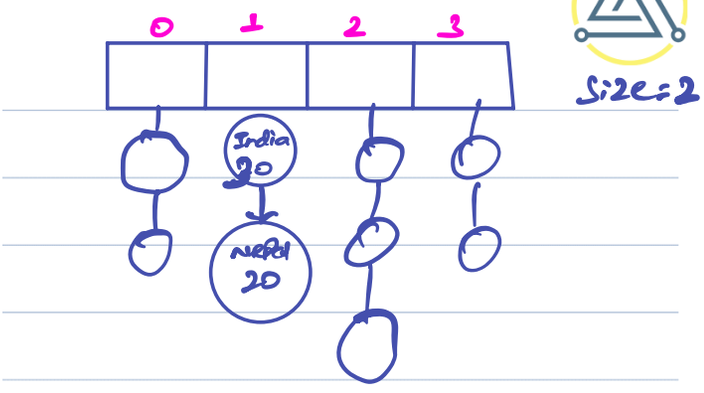
$= 1.5$

$\lambda > 2$ → rehash

→ which index you key belongs → O(1)



```java
public void put(K key, V val) throws Exception {
  int bi = hashFunction(key);
  int di = findInBucket(bi,key);

  if( di == -1){
      HMNode n = new HMNode(key,val);
      buckets[bi].addLast(n);
      size++;
  }else{
      HMNode n = buckets[bi].get(di);
      n.value = val;
  }

  double lambda = size*1.0/buckets.length;

  if(lambda > 2.0){
      rehash();
  }

}
```



size = 2

bi = 1

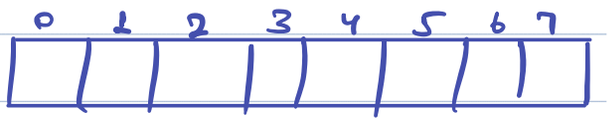
di = 0

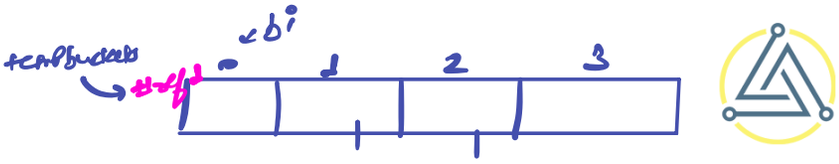
$\frac{9}{4} = 2.25$

put ("India", 10)

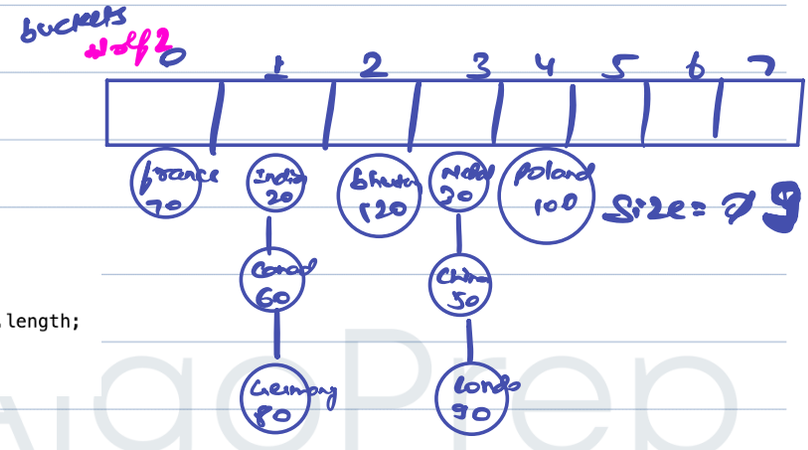
put ("Nepal", 20)

put ("India", 30)






```java
public void rehash(){
    LinkedList<HMNode>[] tempbuckets = buckets;
    initbuckets(2*tempbuckets.length);
    size = 0;

    for(int bi=0;bi<tempbuckets.length;bi++){
        for(HMNode t: tempbuckets[bi]){
            put(t.key,t.value);
        }
    }

}
```



Bhutan 120

```java
public int hashFunction(K key){
    int temp = key.hashCode();
    int bi = Math.abs(temp) % buckets.length;
    return bi;
}
```

HashMap Construction

Java Code:

```
import java.io.*;
import java.util.*;

public class Main {

  public static class HashMap<K, V> {
    private class HMNode {
      K key;
      V value;

      HMNode(K key, V value) {
        this.key = key;
        this.value = value;
      }
    }

    private int size; // n
    private LinkedList<HMNode>[] buckets; // N = buckets.length

    public HashMap() {
      initbuckets(4);
      size = 0;
    }

    public void put(K key, V value) throws Exception {
      int bi = hashFunction(key);
      int di = findInBucket(bi, key);

      if (di == -1) {
        HMNode node = new HMNode(key, value);
        buckets[bi].addLast(node);
        size++;
      } else {
        HMNode node = buckets[bi].get(di);
        node.value = value;
      }

      double lambda = size * 1.0 / buckets.length;
      if (lambda > 2.0) {
        rehash();
      }
```

```
}

public V get(K key) throws Exception {
  int bi = hashFunction(key);
  int di = findInBucket(bi, key);

  if (di == -1) {
    return null;
  } else {
    HMNode node = buckets[bi].get(di);
    return node.value;
  }
}

public boolean containsKey(K key) {
  int bi = hashFunction(key);
  int di = findInBucket(bi, key);

  if (di == -1) {
    return false;
  } else {
    return true;
  }
}

public V remove(K key) throws Exception {
  int bi = hashFunction(key);
  int di = findInBucket(bi, key);

  if (di == -1) {
    return null;
  } else {
    HMNode node = buckets[bi].remove(di);
    size--;
    return node.value;
  }
}

public ArrayList<K> keyset() throws Exception {
  ArrayList<K> set = new ArrayList<>();

  for (int bi = 0; bi < buckets.length; bi++) {
    for (HMNode node : buckets[bi]) {
      set.add(node.key);
    }
  }

  return set;
```

```
}

public int size() {
  return size;
}

public void display() {
  System.out.println("Display Begins");
  for (int bi = 0; bi < buckets.length; bi++) {
    System.out.print("Bucket" + bi + " ");
    for (HMNode node : buckets[bi]) {
      System.out.print( node.key + "@" + node.value + " ");
    }
    System.out.println(".");
  }
  System.out.println("Display Ends");
}
// returns bucket index for a key
private int hashFunction(K key) {
  int hc = key.hashCode();
  int bi = Math.abs(hc) % buckets.length;
  return bi;
}

// return data index for a bucket and key
private int findInBucket(int bi, K key) {
  int di = 0;
  for (HMNode node : buckets[bi]) {
    if (node.key.equals(key)) {
      return di;
    }
    di++;
  }

  return -1;
}

// when lambda crosses a threshold
private void rehash() throws Exception {
  LinkedList<HMNode>[] oba = buckets;
  initbuckets(2 * oba.length);
  size = 0;

  for (int bi = 0; bi < oba.length; bi++) {
    for (HMNode onode : oba[bi]) {
      put(onode.key, onode.value);
    }
  }
```

```
    }

    private void initbuckets(int N) {
      buckets = new LinkedList[N];
      for (int bi = 0; bi < buckets.length; bi++) {
        buckets[bi] = new LinkedList<>();
      }
    }
  }

  public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    HashMap<String, Integer> map = new HashMap();

    String str = br.readLine();
    while (str.equals("quit") == false) {
      if (str.startsWith("put")) {
        String[] parts = str.split(" ");
        String key = parts[1];
        Integer val = Integer.parseInt(parts[2]);
        map.put(key, val);
      } else if (str.startsWith("get")) {
        String[] parts = str.split(" ");
        String key = parts[1];
        System.out.println(map.get(key));
      } else if (str.startsWith("containsKey")) {
        String[] parts = str.split(" ");
        String key = parts[1];
        System.out.println(map.containsKey(key));
      } else if (str.startsWith("remove")) {
        String[] parts = str.split(" ");
        String key = parts[1];
        System.out.println(map.remove(key));
      } else if (str.startsWith("size")) {
        System.out.println(map.size());
      } else if (str.startsWith("keyset")) {
        System.out.println(map.keyset());
      } else if (str.startsWith("display")) {
        map.display();
      }
      str = br.readLine();
    }
  }
}
```