HASHING

1. DIRECT ADDRESS TABLE

```
#include <bits/stdc++.h>
using namespace std;

struct DAT{
    int table[1000]={0};

    void insert(int i){
        table[i]=1;
    }

    void del(int i){
        table[i]=0;
    }

    int search(int i){
        return table[i];
    }
};

int main()
{
    DAT dat;
    dat.insert(10);
    dat.insert(20);
    dat.insert(119);
    cout<<dat.search(10)<<endl;
    cout<<dat.search(20)<<endl;
    dat.del(119);
    cout<<dat.search(119)<<endl;
```

```
    return 0;
}
```

2. IMPLEMENTATION OF CHAINING

```
// C++ program to demonstrate implementation of our
// own hash table with chaining for collision detection
#include<bits/stdc++.h>
using namespace std;

struct MyHash
{
    int BUCKET;
    list<int> *table;
    MyHash(int b)
    {
        BUCKET = b;
        table = new list<int>[BUCKET];
    }
    void insert(int k)
    {
        int i = k % BUCKET;
        table[i].push_back(k);
    }
    bool search(int k)
    {
        int i = k % BUCKET;
        for (auto x : table[i])
          if (x == k)
            return true;
        return false;
    }
    void remove(int k)
```

```
    {
        int i = k % BUCKET;
        table[i].remove(k);
    }
};

    // Driver method to test Map class
    int main()
    {
        MyHash mh(7);
        mh.insert(10);
        mh.insert(20);
        mh.insert(15);
        mh.insert(7);
        cout << mh.search(10) << endl;
        mh.remove(15);
        cout << mh.search(15);
    }
```

3. IMPLEMENTATION OF OPEN ADDRESSING

```
#include<bits/stdc++.h>
using namespace std;

struct MyHash
{
    int *arr;
    int cap,size;

    MyHash(int c)
    {
        cap=c;
        size=0;
        arr=new int[cap];
```

```
    for(int i=0;i<cap;i++)
        arr[i]=-1;
}

int hash(int key){
    return key%cap;
}
bool insert(int key)
{
    if(size==cap)
        return false;
    int i=hash(key);
    while(arr[i]!=-1 && arr[i]!=-2 && arr[i]!=key)
        i=(i+1)%cap;
    if(arr[i]==key)
        return false;
    else{
        arr[i]=key;
        size++;
        return true;
    }
}
bool search(int key)
{
    int h=hash(key);
    int i=h;
    while(arr[i]!=-1){
        if(arr[i]==key)
            return true;
        i=(i+1)%cap;
        if(i==h)
            return false;
    }
    return false;
}
```

```
    bool erase(int key)
    {
        int h=hash(key);
        int i=h;
        while(arr[i]!=-1){
            if(arr[i]==key){
                arr[i]=-2;
                return true;
            }
            i=(i+1)%cap;
            if(i==h)
                return false;
        }
        return false;
    }
};

     int main()
     {
         MyHash mh(7);
         mh.insert(49);
         mh.insert(56);
         mh.insert(72);
         if(mh.search(56)==true)
             cout<<"Yes"<<endl;
         else
             cout<<"No"<<endl;
         mh.erase(56);
         if(mh.search(56)==true)
             cout<<"Yes"<<endl;
         else
             cout<<"No"<<endl;
     }
```

4. COUNT DISTINCT ELEMENTS

```
#include <bits/stdc++.h>
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 arr[] = {15, 16, 27, 27, 28, 15};
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << countDistinct(arr, n);

    return 0;
}
```

5. FREQUENCIES OF ARRAY ELEMENTS

```
/*package whatever //do not write package name here */

import java.io.*;
import java.lang.*;
import java.util.*;
import java.math.*;

class GFG {
        public static void main (String[] args) throws IOException{
```

```
        BufferedReader inp=new BufferedReader(new
InputStreamReader(System.in));
        String s[] = inp.readLine().split(" ");
        int n = Integer.parseInt(s[0]);
        int w = Integer.parseInt(s[1]);
        int l = Integer.parseInt(s[2]);
        int h[] = new int[n];
        int r[] = new int[n];
        int index=0;
        int n1=n;
        while(n-->0)
        {
            String s1[] = inp.readLine().split(" ");
            h[index] = Integer.parseInt(s1[0]);
            r[index] = Integer.parseInt(s1[1]);
            index++;
        }
        long mon=0;

        while(mon>=0)
        {
          //  System.out.println(mon);
            long sum=0;
            long height=0;
            for(long i=0;i<n1;i=i+1){
            height=h[i]+r[i]*mon;
            if(height>=l)
            {
                sum+=height;
              //  System.out.println(sum);
            }}
            if(sum>=w){
            System.out.println(mon);
            break;
            }
```

```
                mon=mon+1;
            }
        }
}
```

6. INTERSECTION OF 2 ARRAYS

```
#include <bits/stdc++.h>
using namespace std;

int intersection(int arr1[], int m, int arr2[], int n)
{
    unordered_set<int> us;
    for(int i = 0; i < m; i++)
      us.insert(arr1[i]);

    int res = 0;
    for(int i = 0; i < n; i++)
    {
        if(us.find(arr2[i]) != us.end())
        {
            res++;
            us.erase(arr2[i]);
        }
    }
    return res;
}

int main()
{
    int arr1[] = {15, 17, 27, 27, 28, 15};
    int arr2[] = {16, 17, 28, 17, 31, 17};
    int m = sizeof(arr1)/sizeof(arr1[0]);
```

```
    int n = sizeof(arr2)/sizeof(arr2[0]);

    cout << intersection(arr1, m, arr2, n);

    return 0;
}
```

7. UNION OF 2 UNSORTED ARRAYS

```
#include <bits/stdc++.h>
using namespace std;

int unionSize(int arr1[], int m, int arr2[], int n)
{
    unordered_set<int> us;
    for(int i = 0; i < m; i++)
     us.insert(arr1[i]);
    for(int i = 0; i < n; i++)
     us.insert(arr2[i]);

    return us.size();
}

int main()
{
    int arr1[] = {2, 8, 3, 5, 6};
    int arr2[] = {4, 8, 3, 6, 1};
    int m = sizeof(arr1)/sizeof(arr1[0]);
    int n = sizeof(arr2)/sizeof(arr2[0]);

    cout << unionSize(arr1, m, arr2, n);
}
```

8. PAIR WITH GIVEN SUM IN UNSORTED ARRAY

```
#include <bits/stdc++.h>
using namespace std;


int pairWithSumX(int arr[],int n, int X)
{
    unordered_set<int> us;
    for(int i = 0; i < n; i++)
    {
        if(us.find(X - arr[i]) != us.end())
          return 1;

        us.insert(arr[i]);
    }
    return 0;

}

int main()
{
    int arr[] = {3, 8, 4, 7, 6, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    int X = 14;

    cout << pairWithSumX(arr, n, X);
    return 0;
}
```

9. SUBARRAY WITH ZERO-SUM

```
#include <bits/stdc++.h>
using namespace std;
```

```
int ZeroSumSubarray(int arr[], int n)
{
    unordered_set<int> us;
    int prefix_sum = 0;
    us.insert(0);
    for(int i = 0; i < n; i++)
    {
        prefix_sum += arr[i];
        if(us.find(prefix_sum) != us.end())
          return 1;
        us.insert(prefix_sum);
    }
    return 0;

}

int main()
{
    int arr[] = {5, 3, 9, -4, -6, 7, -1};
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << ZeroSumSubarray(arr, n);
}
```

10. SUBARRAY WITH GIVEN SUM

```
#include <bits/stdc++.h>
using namespace std;

bool isSum(int arr[], int n, int sum)
{
    unordered_set<int> s;
    int pre_sum = 0;
    for(int i = 0; i < n; i++)
```

```
    {
        if(pre_sum==sum)
            return true;
        pre_sum += arr[i];
        if(s.find(pre_sum-sum) != s.end())
          return true;
        s.insert(pre_sum);
    }
    return false;
}

int main()
{
    int arr[] = {5, 8, 6, 13, 3, -1};
    int sum=22;
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << isSum(arr, n, sum);
}
```

11. LONGEST SUBARRAY WITH GIVEN SUM

```
#include <bits/stdc++.h>
using namespace std;

int largestSubarrayWithSumX(int arr[], int n, int sum)
{
    int prefix_sum = 0;
    unordered_set<int> us;
    us.insert(0);
    for(int i = 0; i < n; i++)
    {
        prefix_sum += arr[i];
        if(us.find(prefix_sum - sum) != us.end())
```

```
            return 1;
        us.insert(prefix_sum);
    }
    return 0;
}

int main()
{
    int arr[] = {8, 3, -7, -4, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    int sum = -8;

    cout << largestSubarrayWithSumX(arr, n, sum);

}
```

12. LONGEST SUBARRAY WITH EQUAL NUMBER OF 0s AND 1s

```
#include <bits/stdc++.h>
using namespace std;

int largestZeroSubarray(int arr[], int n)
{
    for(int i = 0; i < n; i++)
     arr[i] = (arr[i] == 0) ? -1 : 1;

    unordered_map<int, int> ump;
    int sum = 0, maxLen = 0;
    for(int i = 0; i < n; i++)
    {
        sum += arr[i];
        if(sum == 0)
         maxLen = i+1;
```

```
        if(ump.find(sum+n) != ump.end())
        {
            if(maxLen < i - ump[sum+n])
             maxLen = i - ump[sum+n];

        }
        else ump[sum + n] = i;
    }

    return maxLen;
}



int main()
{
    int arr[] = {1, 1, 1, 0, 1, 0, 1, 1, 1};
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << largestZeroSubarray(arr, n);
}
```

13. Longest common span with the same sum in a binary array

```
// C++ program to find largest subarray
// with equal number of 0's and 1's.
#include <bits/stdc++.h>
using namespace std;

// Returns largest common subarray with equal
// number of 0s and 1s in both of t
int longestCommonSum(bool arr1[], bool arr2[], int n)
{
        // Find difference between the two
        int arr[n];
```

```
        for (int i=0; i<n; i++)
        arr[i] = arr1[i] - arr2[i];

        // Creates an empty hashMap hM
        unordered_map<int, int> hM;

        int sum = 0;          // Initialize sum of elements
        int max_len = 0; // Initialize result

        // Traverse through the given array
        for (int i = 0; i < n; i++)
        {
                // Add current element to sum
                sum += arr[i];

                // To handle sum=0 at last index
                if (sum == 0)
                        max_len = i + 1;

                // If this sum is seen before,
                // then update max_len if required
                if (hM.find(sum) != hM.end())
                max_len = max(max_len, i - hM[sum]);

                else // Else put this sum in hash table
                        hM[sum] = i;
        }

        return max_len;
}

// Driver program to test above function
int main()
{
        bool arr1[] = {0, 1, 0, 1, 1, 1, 1};
```

```
        bool arr2[] = {1, 1, 1, 1, 1, 0, 1};
        int n = sizeof(arr1)/sizeof(arr1[0]);
        cout << longestCommonSum(arr1, arr2, n);
        return 0;
}
```

14. LONGEST CONSECUTIVE SUBSEQUENCE

```
#include <bits/stdc++.h>
using namespace std;

int findLongest(int arr[], int n)
{
    unordered_set<int> s;
    int res = 0;

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

    for (int i = 0; i < n; i++) {
        if (s.find(arr[i] - 1) == s.end()) {
            int curr = 1;
            while (s.find(curr+arr[i]) != s.end())
                curr++;

            res = max(res, curr);
        }
    }
    return res;
}

int main()
{
    int arr[] = {1, 9, 3, 4, 2, 10, 13};
```

```
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << findLongest(arr, n);
}
```

15. COUNT DISTINCT ELEMENTS IN EVERY WINDOW

```
#include <bits/stdc++.h>
using namespace std;

void printDistinct(int arr[], int n, int k)
{
    map<int, int> m;

    for (int i = 0; i < k; i++) {
        m[arr[i]] += 1;
    }

    cout << m.size() << " ";

    for (int i = k; i < n; i++) {

        m[arr[i - k]] -= 1;

        if (m[arr[i - k]] == 0) {
            m.erase(arr[i-k]);
        }
        m[arr[i]] += 1;

        cout << m.size() << " ";
    }

}
```

```
int main()
{
    int arr[] = {10, 10, 5, 3, 20, 5};

    int n = sizeof(arr)/sizeof(arr[0]);
    int k=4;
    printDistinct(arr, n, k);
}
```

16. MORE THAN N/K OCCURRENCES

```
#include <bits/stdc++.h>
using namespace std;

void printNByK(int arr[], int n, int k)
{
    unordered_map<int,int> m;

    for(int i=0;i<n;i++)
        m[arr[i]]++;
    for(auto e: m)
        if(e.second>n/k)
            cout<<e.first<<" ";

}

int main()
{
    int arr[] = {10, 10, 20, 30, 20, 10,10};

    int n = sizeof(arr)/sizeof(arr[0]);
    int k=2;
    printNByK(arr, n, k);
```

```
}
```

EFFICIENT SOLUTION

```
#include <bits/stdc++.h>
using namespace std;

void printNByK(int arr[], int n, int k)
{
    unordered_map<int,int> m;

    for(int i=0;i<n;i++){
        if(m.find(arr[i])!=m.end())
            m[arr[i]]++;
        else if(m.size()<k-1)
            m[arr[i]]=1;
        else
            for(auto x:m){
                x.second--;
                if(x.second==0)
                    m.erase(x.first);}
    }
    for(auto x:m){
        int count=0;
        for(int i=0;i<n;i++){
            if(x.first==arr[i])
                count++;

       }
        if(count>n/k)
            cout<<x.first<<" ";
    }
}

int main()
```

```
{
    int arr[] = {30, 10, 20, 20, 20, 10, 40, 30, 30};

    int n = sizeof(arr)/sizeof(arr[0]);
    int k=4;
    printNByK(arr, n, k);
}
```