Explanation
Hashing

Int -
Table - a[n][2]
1 2 3 2 4 1 4 5 count sort -- c
C[3] = 1
C[key] = value
cout<<c[key];

keys	values
1	2
2	2
3	1
4	2
5	1

Stl - unordered_map< string , vector<int> > arr;

Arr[1] = 2;
Arr[3] =2;
…
arr.insert(make_pair(3,2));

Keys - int, long int, char, string
Values - int, long int, char, string, vector, pair

Size - table_size

Abc bcd abcd abcd asd

Keys must be unique

Hashmaps -
Search
Insert
Delete -
O(1) best , avg

Worst - O(n)   O(1)

Insertion
H[key] = value
h.insert(make_pair(key,value))

Deletion
h.erase(key)

Search
H[key]

h.count(key)	1 or 0
h.find(key)	h.end() - if not present

How to find the index of key
Hash function - input key - return index

Table size = n = 10
Int,int
19,20
23,56
34,54
73, 76

key%n = 19%10 = 9
23 == 3
34 ==   4
73 ==

Collision -
Open hashing, chaining, open addressing, linear probing, double hashing

Hash function
Modulo with prime number - 11

2 primes -   p1, p2
P2 = nearest to table size
P1 = nearest to data/input size

Basic implementation
Hash function, double hashing example

Hashmaps - unordered_map array O(1)
Map - bst O(logn)

2 sum

Find the pairs which has their sum as target

Basic = O(n2)
Sorting = O(nlogn)
Hashmaps = O(n)

Sum = 8
1 3 5 4 2 1 6 7

```
Int c=0;
unordered_map<int,int> h;
for(int i=0;i<n;i++)
{
        Int x = sum - a[i];
        if(h.count(x) >0)
                C += h[x];
        H[a[i]]  += 1;
}
C = 4           i = 7
```

1	2
3	1
5	1
4	1
2	1

Count subarrays with 0 sum

2 -4 2 4 -6 -3 2

2 -2 0 4 -2 -5 -3

Intersection of 2 arrays

Basic = O(n2)
Sorting = O(nlogn) 2 pointers = i and j
Hashmap = O(n)

Heaps -
Complete binary tree

0 based indexing
Curr = i
Parent = floor((i-1)/2)
Child = 2*i+1, 2*i+2

1 based indexing
Curr = i
Parent = floor(i/2)
Child = 2*i, 2*i+1

```
10 15 30 40 50 100 40
0   1   2    3   4   5      6
```

priority_queue = priority - max element   max-heap
priority_queue = priority - min element   min-heap

Explanation :
Heaps

Insert - add new node to end and up heapify   O(logn)
Delete - delete top node and down heapify   O(logn)
Space - O(n)

```
Priority-queue <int> pq;                              // max heap
pq.push(0);
pq.pop();
Pq.top;

Class comp
{
Public:
        Bool operator()(int a,int b)
        {
                //comp
        }
};
Priority-queue <int, vector<int>, greater<int>> pq;           // min heap
Priority-queue <int, vector<int>, comp> pq;                   // custom heap
```

Priority-queue <pair<int,int>, vector<pair<int,int>>, comp> pq; // custom heap

sort(a,a+n, greater<int>())

Heap sort
Push all elements in min heap
Pop one by one and add to array

K th max min element in array
Time - O(nlogn + klogn)
Space - O(n)

Time - O(nlogk + (n-k+1)logk)
Space - O(k)
Max element - min heap

K elements
1 pop - kth max
2nd element - k-1 th max

1, 2, 3, …, n-k-1,n-k,n-k+1,... n

Running stream
Input size - very large

Kth Max element in running stream

Merge k arrays in sorted arrays of same size
Brute force - add all to single array and sort

Min heap of k size

Value, ind, array ind
pair<int,pair<int,int>>
Struct