Hash Map

key, value → unique

Rohini	→ 12
Noida	→ 15
Gurugram	→ 20
Janakpuri	→ 34
Dwarka	→ 17

Syntax

```
HashMap<String, Integer> map = new HashMap<>();
```

functions → O(1)

add
```
map.put("Rohini", 12);   map.put("Noida", 17);
```

remove → O(1)
```
map.remove("Noida");
```

Contains key → O(1)
```
map.containsKey("Noida")
```
→ true
→ false

size → O(1)
```
map.size();
```

value get → O(1)
```
map.get("Rohini") = 12
```

O(1) → exact
→ average

print
```
System.out.println(map);
```

```
Arraylist<String> al = map.keySet();
{"Rohini", "Noida" _ _ _ _ _}
```

HashSet

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

add — set.add(s)

remove — set.remove(s)

contains — set.contains(7) → true / → false

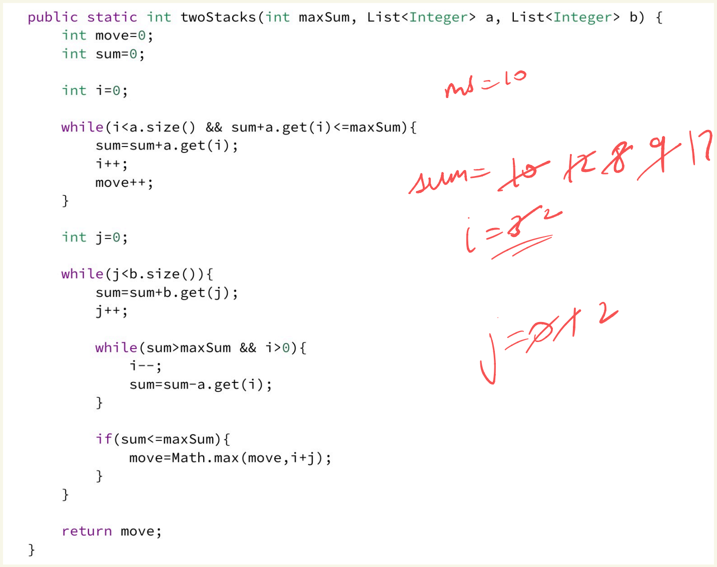
size — set.size();

{ set.add(s)
{ set.add(s)

set.size();

1 , 2 , 4 , 5

(1, 2, 4, 5)



```
public static int twoStacks(int maxSum, List<Integer> a, List<Integer> b) {
    int move=0;
    int sum=0;

    int i=0;

    while(i<a.size() && sum+a.get(i)<=maxSum){
        sum=sum+a.get(i);
        i++;
        move++;
    }

    int j=0;

    while(j<b.size()){
        sum=sum+b.get(j);
        j++;

        while(sum>maxSum && i>0){
            i--;
            sum=sum-a.get(i);
        }

        if(sum<=maxSum){
            move=Math.max(move,i+j);
        }
    }

    return move;
}
```

ms = 10

sum = ~~10~~ ~~12~~ ~~8~~ ~~9~~ 17

i = ~~3~~ 2

j = ~~0~~ ~~1~~ 2

$\boxed{10}$

0	4	0	2
1	2	1	1
2	4	2	8
3	6	3	5
4	1		

tree

→ t → 1

→ r → 1

e → 2

eetr

ababaccdf

a → 3

b → 2

c → 2

d → 1

f → 1

aaabbccfd

$\overset{0}{t}\ \overset{1}{e}\ \overset{2}{r}\ \overset{3}{e}\ \overset{4}{f}\ \overset{5}{e}\ \overset{6}{t}$

ch = ~~t~~ ~~e~~ ~~r~~ ~~e~~ ~~f~~ ~~e~~ t

fre = ~~1~~ ~~2~~ 1

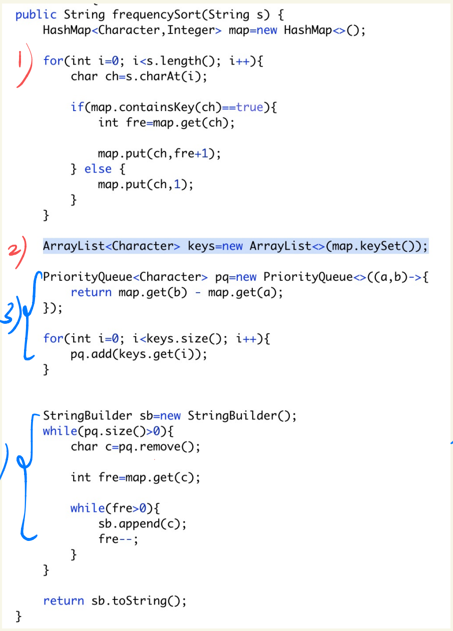
1) ⇒

t → 2

e → 3

r → 1

f → 1



```
public String frequencySort(String s) {
    HashMap<Character,Integer> map=new HashMap<>();

1)  for(int i=0; i<s.length(); i++){
        char ch=s.charAt(i);

        if(map.containsKey(ch)==true){
            int fre=map.get(ch);

            map.put(ch,fre+1);
        } else {
            map.put(ch,1);
        }
    }

2)  ArrayList<Character> keys=new ArrayList<>(map.keySet());

3)  PriorityQueue<Character> pq=new PriorityQueue<>((a,b)->{
        return map.get(b) - map.get(a);
    });

    for(int i=0; i<keys.size(); i++){
        pq.add(keys.get(i));
    }


4)  StringBuilder sb=new StringBuilder();
    while(pq.size()>0){
        char c=pq.remove();

        int fre=map.get(c);

        while(fre>0){
            sb.append(c);
            fre--;
        }
    }

    return sb.toString();
}
```

2) → keys = { 't', 'e', 'r', 'f' }

3) → pq ⇒ 't' 'e' 'r' 'f'

4) sb = eeettfr

c = r

fre = 1