The Frequency Map

Vanilla Version: Given N integers $A_1, A_2, \cdots, A_N$ print all the distinct integers appearing and the number of times in which they appear

$$\begin{cases} 1 \leq N \leq 2 \cdot 10^5 \\ 1 \leq A_i \leq 10^3 \end{cases}$$

1, 2, 3, 7, 3, 9, 2

O(N): int freq[1001];
for i from 1 to 1001
freq[i] = 0;
for i from 1 to N
cin >> x;
freq[x]++;

Given N integers $A_1, A_2, \cdots, A_N$ print all the distinct integers appearing and the number of times in which they appear

$1 \leq N \leq 2 \cdot 10^5$

$1 \leq A_i \leq 10^{18}$

```
map<long long, int> freq;
for(int i=1; i<=N; ++i){
    cin >> x;
    freq[x]++;
}
```

$O(\log N)$

$O(N \log N)$

Problem: Select Training Set

https://www.codechef.com/ICPCIN19/problems/TRAINSET

ICPC 2019 online round

abc 0

abc 1

ab c 0

abc 0

efg 1

efg 0

efg 0

ccc 1

$1 \le N \le 25,000$

$1 \le |w_i| \le 5$

```
map<string, int> zero, one;
set<string> S;
for(i=1; i<=N; ++i){
    string w; bool t;
    cin >> w >> t; S.insert(w);
    t? one[w]++ : zero[w]++;
}
int max_sz=0;
for(string w: S) max_sz += max(one[w], zero[w]);
```

{zero[abc] = 3, one[abc] = 1} 3

{zero[efg] = 2, one[efg] = 1} 2

{one[ccc] = 1} 1

Exercise: Bitwise Equalized (https://www.hackerrank.com/contests/goc-cdc-series-15/challenges/bitwise-equalized)

Monotonic Stack :: Next Greater Element:

Given N integers $A_1, A_2, \cdots A_N$ for all i find the smallest index $j > i$ such that $A_i < A_j$

$1 \le N \le 10^6$
$1 \le A_i \le 10^{18}$

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4,	5,	8,	2,	8,	5,	7,	9,	3,	5,	8,	7,	2,	6
2	3	8	5	8	7	8	-1	10	11	-1	-1	14	-1

index of the next greater element →

Brute Force: $O(N^2)$

```
for i from 1 to N {
    for j from i+1 to N {
        if (A[j] > A[i]) {
            print j; found = true;
            break;
        }
    } if(!found) print -1; }
```

Using a Set:

$(-9,8), (-8,3), (-8,5), (-8,11), (-7,7), (-7,12), (-6,14)$

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4,	5,	8,	2,	8,	5,	7,	9,	3,	5,	8,	7,	2,	6
	3	8		8		8	-1			-1	-1		-1

S: 2 3 5 7 8 11 12 14

→ V = { −A[i], i } ; sort V;

(sort in dec. order. If there is a tie prefer the element the occurs to the left)

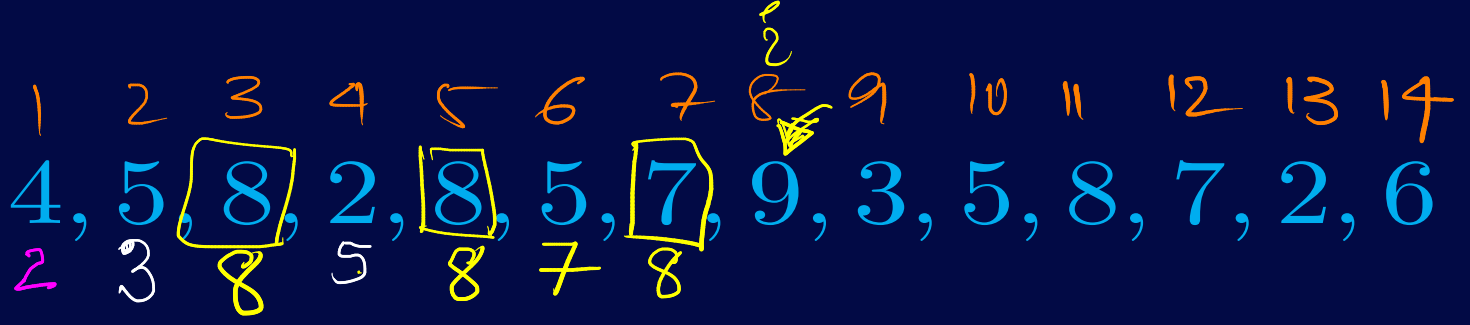
```
set<int> S; // set of indices
for i from 0 to N-1 {
    S.insert(V[i].second);
    auto it = S.upper_bound(V[i].second);
    nxt[V[i].second] = (it == S.end()) ? -1 : *it;
}
```

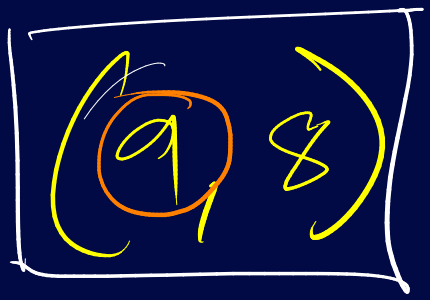
O(N log N)

Using a Stack:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4,	5,	8,	2,	8,	5,	7,	9,	3,	5,	8,	7,	2,	6

```
stack<pair<long long, int>> S; // <value, index>
for i from 1 to N{
    while(!S.empty() && S.top.first < A[i]){
        nxt[S.top().second] = i;
        S.pop();
    }
    S.push({A[i], i});
}
while(!S.empty()){nxt[S.top().second] = -1; S.pop();}
```

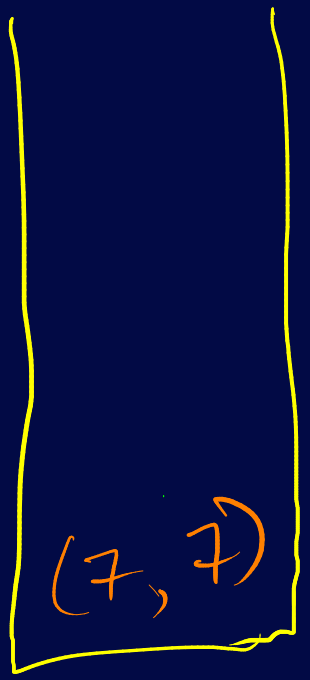




O(N)

1

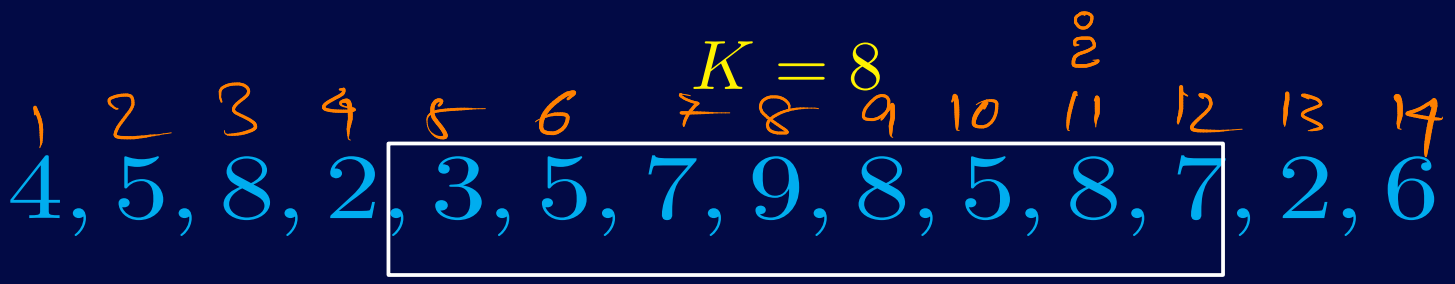
1	2	3	4	5	6	7
6	5	4	3	2	5	7
1	1	1	1	1	2	-1
7	7	6	6	6	7	



Monotonic Queue :: Minimum element of all K-sized windows:

Given N integers $A_1, A_2, \cdots A_N$ for all "windows" (subarrays) of size K, find the minmimum element.

$1 \leq K \leq N \leq 10^6$
$1 \leq A_i \leq 10^{18}$



Brute Force: $O(KN)$

```
for i from 0 to N-K {
    min_element(A+i, A+i+K);
}
```

Using a Set / Multiset :

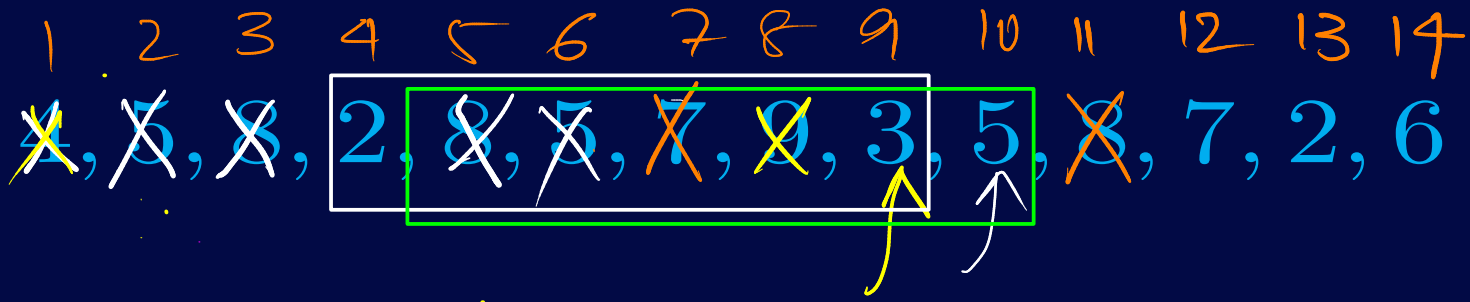
Insert the first K elements in multiset<ll> S

```
for i from K to N :{
    print * S.begin();
    if i < N {
        S.insert(A[i+1]);
        S.erase(S.find(A[i-K+1]));
    }
}
```

$O(N \log K)$

Using a Deque:

(Monotonic Queue)



i, $\Rightarrow$ In the current window, there is no $j > i$ s.t. $A[j] < A[i]$

{4}

4

```
deque<pair<ll, int>> dq; // value, index
for(int i=1 ; i<=K ; ++i){
  while(!dq.empty() && dq.back().first > A[i]){
    dq.pop_back();
  }
  dq.push_back({A[i], i});
}
for(int i=K ; i<=N ; ++i){
  cout << dq.front() << '\n';
  if(dq.front().second == i-K+1){
    dq.pop_front();
  }
}
```

```
if(i<N){ while(!dq.empty() && dq.back().first > A[i+1]){
        dq.pop_back();
    }
    dq.push_back({A[i], i});
}
```

$O(N)$