Reorganizing String :-

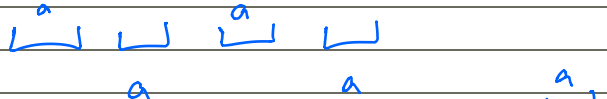
given a string convert it such that no 2 consecutive character are same;

i/p = "aabbc" → o/p = "ababc"

* catch 1 :- if freq[char] > (n+1)/2 → not possible.

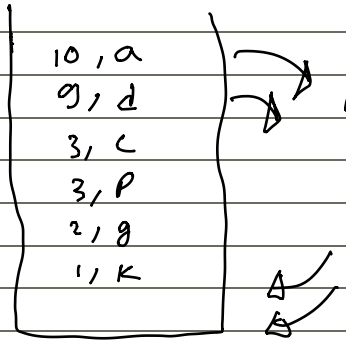
a _ a _ a _ a } maximum case where freq = (n+1)/2
_ a _ a _ a

	$\frac{n+1}{2}$	$\frac{n}{2}+1$
3	2	2
→ 4	2	3
5	3	3
→ 6	3	4
7	4	4



Approach 1 :- using Priority Queue;

```
priority_queue<pr, vector<pr>> pq;
typedef pr = pair<int, char>;
```



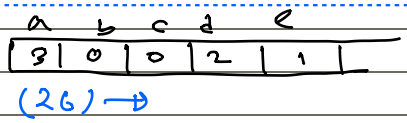
pop top 2 element → ans.push_back(a d)

a d - - - - - ...

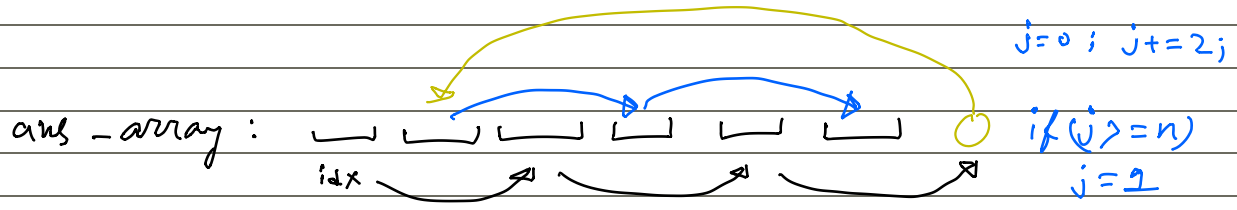
push again (9, a)
(8, d)

at the end push last remaining element.

Approach 2 :- freq vector



#Trick



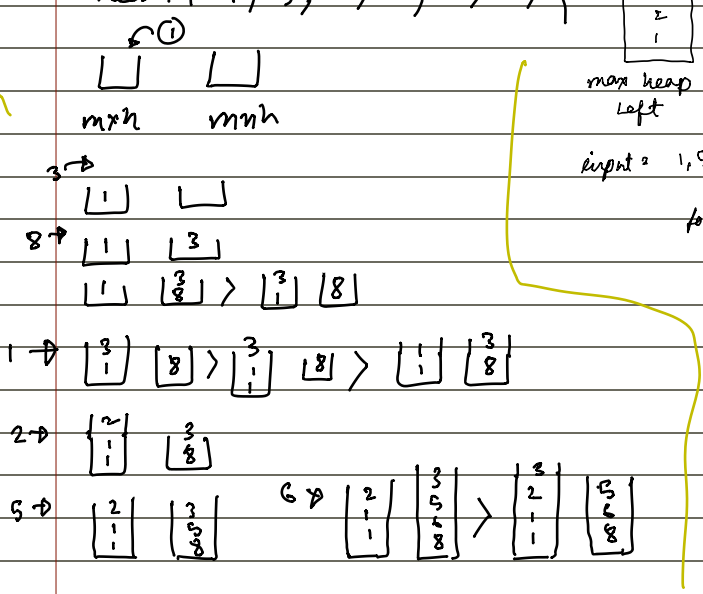
freq array's loop : while(freq[i]) i++;

Median of Sorted array

input = Unsorted array.

Stream : 1, 3, 8, 1, 2, 5, 6

Dry Run



★ at any point
left heapsize = right heapsize
OR left heapsize = right " " + 1

* Return left.top() + right.top()/2
or left.top().

input : 1, 5, 3, 2, 9, 8, 15...

```
for ( int n : arr ){
    if ( left.empty() || n < left.top())
        left.push(n);
    else if ( left.size() - right.size() > 1 )
        left -> right
    else if ( right.size() > left.size() )
        right -> left .
}
```

For generating min heap :-

#C++STL

```
typedef pair< string, int > P;
priority_queue < P, vector<P>, λ > pq;
struct λ { bool operator()(P &P1, P &P2){
        return P1.second < P2.second; }
};
```

#⭐

* slight difference from vector sorting this here gives decreasing order (heap)

K th Largest element in Integer Stream

if K = 3 ; take min heap of size 3 constraint.

Find K Pair with Smallest Sums

input : 2 sorted array.

0	1	2	3
1	1	100	120
1	2	3	9

priority queue
< {sum, {i,j}} >

```
while (k > 0 & ! pq.empty()){
    ans.push_back( pq.top().first );
    curri = pq.top().second.first
    currj = pq.top().second.second.
    pq.pop();

    if(currj+1 is valid) pq.push( i , j+1  pair
    if (curri+1  "  " ) pq.push ( i+1 j  th pair
```