11. Rearrange characters

Medium Accuracy: 49.6% Submissions: 2940 Points: 4

Given a string S such that it may contain repeated lowercase alphabets. Rearrange the characters in the string such that no two adjacent characters are same.

Example 1:

```
Input:
S = geeksforgeeks
Output: 1
Explanation: egeskerskegof can be one way of
rearranging the letters.
```

2. Steps involved:
1. Build a Priority Queue or max heap, **pq** that stores characters and their frequencies.
   (Priority_queue or max_heap is built on the bases of frequency of character.)
2. Create a temporary Key that will used as the previous visited element (previous element in resultant string. Initialize it { char – '#' , freq – '-1' }
3. While **pq** is not empty.
   Pop an element and add it to result.
   Decrease frequency of the popped element by '1'
   Push the previous element back into the pq if it's frequency > '0'
   Make the current element as previous element for the next iteration.
4. If length of resultant string and original, print "not possible". Else print result.

Expected Time Complexity : O(n log n)
Expected Auxilliary Space : O(n)

Constraints:
1 <= length of string <= 10^4
String has only lowercase English alphabets.

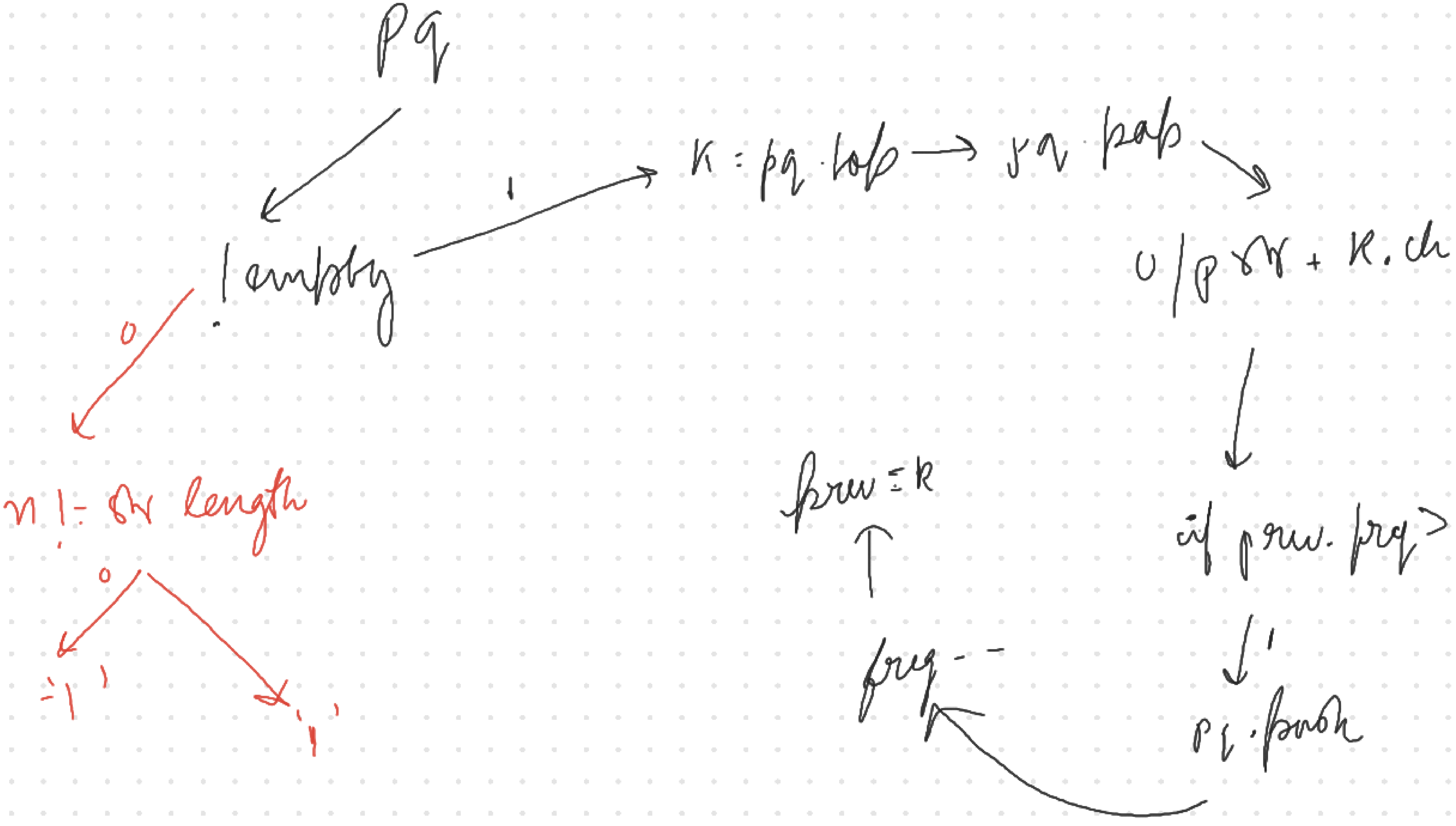
```
class Solution{
   const int MAX_CHAR=26;
   public:

   struct Key
   {
    int freq; // store frequency of character
    char ch;
    bool operator<(const Key &k) const
    {
     return freq < k.freq;
    }
   };

   string rearrangeString(string str)
   {
    int n = str.length();
    int count[MAX_CHAR]={0};//cout<<"MArker"<<"\n";
    //for(int i=0;i<MAX_CHAR;i++)cout<<count[i]<<" "<<i<<" ";
    for (int i = 0 ; i < n ; i++) // storing count of character in array
     count[str[i]-'a']++;

    priority_queue< Key > pq;

    for (char c = 'a' ; c <= 'z' ; c++)
     if (count[c-'a'])
      pq.push( Key { count[c-'a'], c} );
    // storing pair of no. of characters and character
    str = "" ;
    Key prev {-1, '#'};

    while (!pq.empty())
    {
     Key k = pq.top();
     pq.pop();
     str = str + k.ch; // store the most remaining character in string
     if (prev.freq > 0)
      pq.push(prev); // pushing the elements with remaining freq.
     (k.freq)--;
     prev = k;
    }
    if (n != str.length())
     return "-1";
    else
     return str;
   }
};
```

Objective → Given a string, rearrange such that no 2 consecutive characters are repeating.

$O(n \log n)$ & $O(n)$ space

Algorithm:

1. Make Priority Queue → with Key → operator< / cnt freq / character
2. Store freq. of ch in an array with {index} → 26
3. Push all the (ch + freq) [Key] in pq.
4. while pq != empty → add top of pq to string
   ↓
   pop it → freq --
   → if freq > 0 push again


pq
!empty
1
k = pq.top
pq.pop
o/p str + k.ch
if prev.frq>0
1
pq.push
frq--
prev = k
0
n != str length
0

k = pq.top → pq.pop → str + k.ch

↓

if any freq && freq > 0

push in pq

↓

k.freq - -

prev = k ←

① Concurrently adds and removes max freq elements.

② Case This would return NO

↓

aaaab