Frequency based sorting

{ custom sorting }

Problem Description

Given a string **A** containing lowercase english letters of length **N**, you need to sort it in decreasing order based on the frequency of the characters and return the sorted string.

Note :
1. If two characters have equal frequency, they should be sorted alphabetically.
2. The frequency of a character is the number of times it appears in the string.

s = b c d d e d b c a

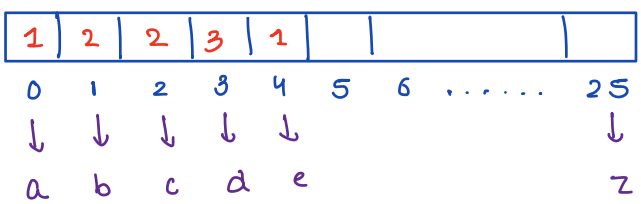
ans = d d d b b c c a e

b → 2
c → 2
d → 3
e → 1
a → 1

i) create count array to calculate freq of every char in s

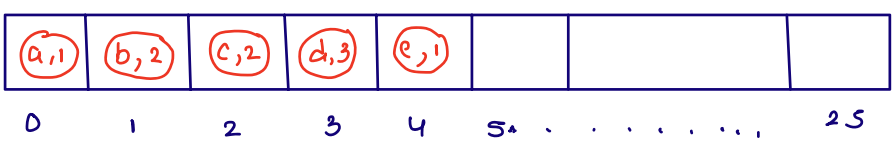
int [ ] 26

s = b c d d e d b c a

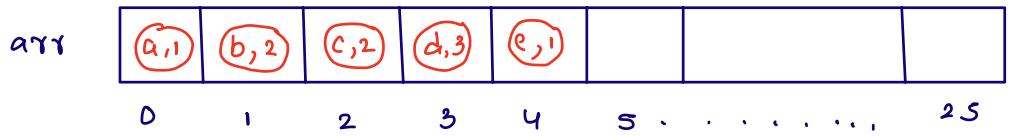


ii) create array of pairs with the help of count array

Pair [ ] arr = new Pair [26];




```
class Pair {
    char ch;
    int freq;
}
```



iii) Sort array of pairs

```
Arrays.sort(arr, new Comparator<Pair>() {
    public int compare(Pair p1, Pair p2) {
        if (p1.freq == p2.freq) {
            return p1.ch - p2.ch;
        }
        else {
            return -(p1.freq - p2.freq);
        }
    }
});
```

{ dec. order of freq }

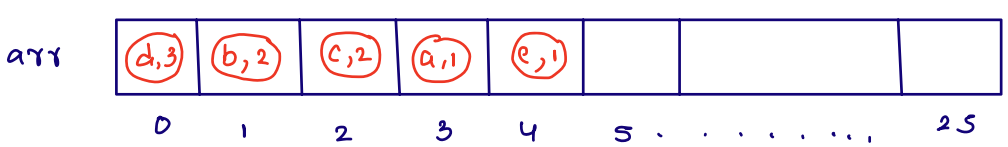
or

```
p2.freq - p1.freq
```

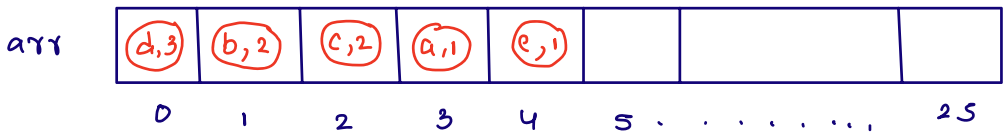
if compare returns -ve: p1 will come first

if compare returns +ve: p2 will come first

after sorting ↓



iv) traverse pair array and calculate final ans.



str : d d d b b c c a e

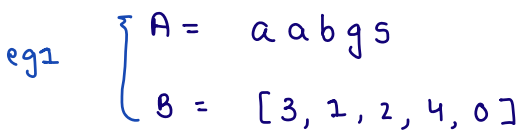
complete code at next page

```java
public class Solution {
    static class Pair {
        char ch;
        int freq;

        Pair(char ch,int freq) {
            this.ch = ch;
            this.freq = freq;
        }
    }

    public String solve(String A) {
        //step1 : create count array
        int[]count = new int[26];

        for(int i=0; i < A.length();i++) {
            char ch = A.charAt(i);
            int idx = ch - 'a';
            count[idx]++;
        }


        //step2 : create and fill data in Pair array using count array
        Pair[]arr = new Pair[26];
        for(int i=0; i < 26;i++) {
            char ch = (char)(i + 'a');
            int freq = count[i];

            Pair p = new Pair(ch,freq);
            arr[i] = p;
        }

        //step3 : sort the Pair array
        Arrays.sort(arr, new Comparator<Pair>(){
           public int compare(Pair p1, Pair p2) {
               if(p1.freq == p2.freq) {
                   return p1.ch - p2.ch;
               }
               else {
                   return -(p1.freq - p2.freq); //or p2.freq - p1.freq;
               }
           }
        });


        //step4 : create final ans using Pair array
        StringBuilder sb = new StringBuilder("");

        for(int i=0; i < 26;i++) {
            Pair p = arr[i];

            char ch = p.ch;
            int freq = p.freq;

            //add ch, freq times in your ans
            for(int k=1; k <= freq;k++) {
                sb.append(ch);
            }

        }
        return sb.toString();
    }
}
```

Shuffling String

Problem Description

Given a string **A** containing only lowercase english letters and an integer array **B** of same length **N**.
You need to shuffle the given string such that the character at the **ith** position moves to **B[i]** in the shuffled string and return the shuffled string.




```java
public class Solution {
    public String solve(String A, int[] B) {
        char[]arr = new char[A.length()];

        //travel A and shuffle
        for(int i=0; i < A.length();i++) {
            char ch = A.charAt(i);
            int pos = B[i];

            //put ch at pos in ans
            arr[pos] = ch;
        }


        String ans = new String(arr);
        return ans;
    }
}
```

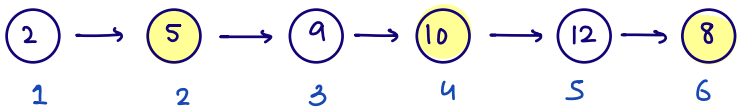
Even reverse

Q. Given head of a LinkedList, reverse the order of all nodes at even position.

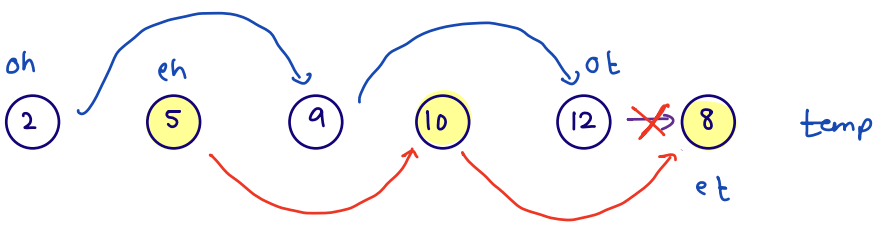
i) with space → easy

ii) without space → medium

eg1



i) Segregate given LL into two halves



idx = 0 1 0 1 ...

oh → odd head

ot → odd tail

eh → even head

et → even tail

et.next = null

ot.next = null

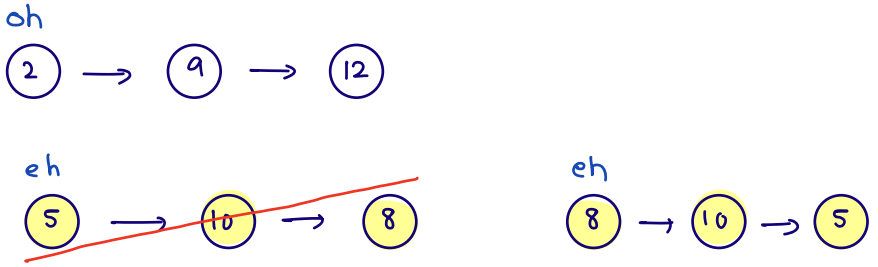
→ if idx is odd join temp in odd LL

ot.next = temp;

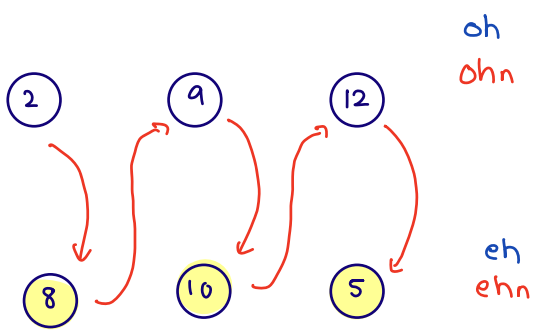
→ else join temp in even LL

et.next = temp;

ii) step 2 is : eh = reverse (eh)



iii) join these two to create final ans.



ohn = oh.next

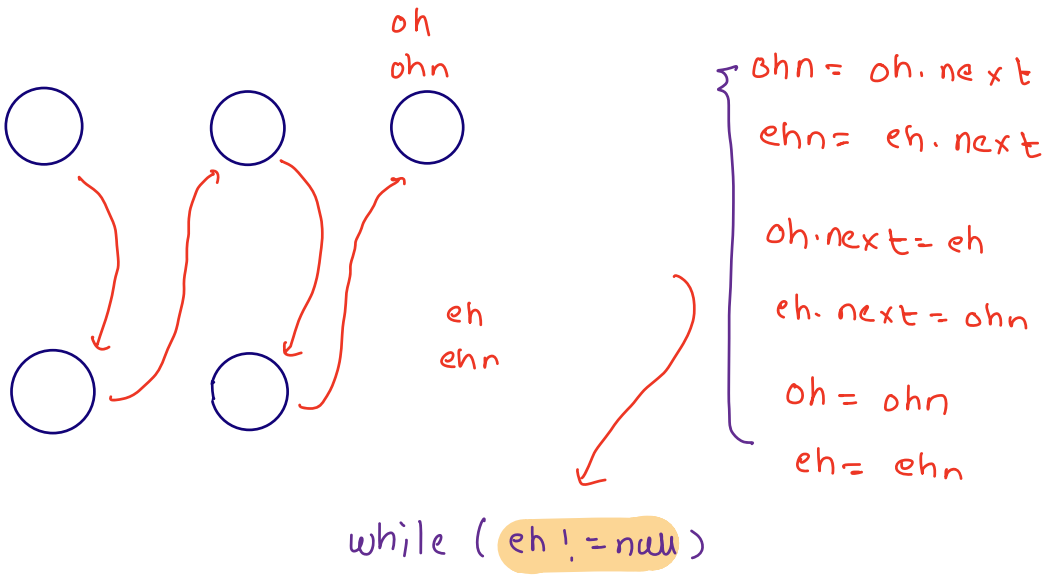
ehn = eh.next

oh.next = eh

eh.next = ohn

oh = ohn

eh = ehn



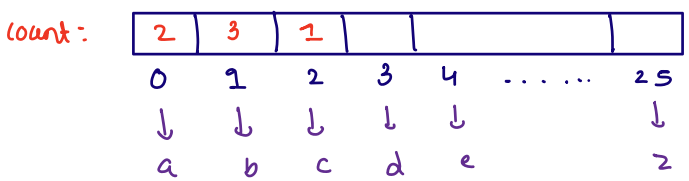
complete code at next page

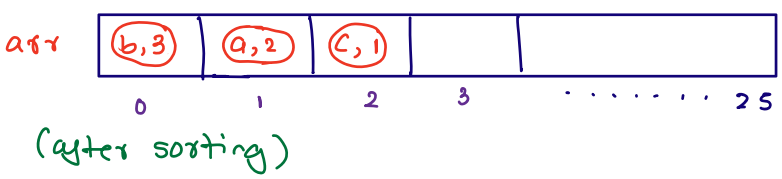
```java
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     public int val;
 *     public ListNode next;
 *     ListNode(int x) { val = x; next = null; }
 * }
 */
public class Solution {

    public ListNode reverse(ListNode head) {
        ListNode p = null, c = head;

        while(c != null) {
            ListNode n = c.next;
            c.next = p;
            p = c;
            c = n;
        }

        return p;
    }

    public ListNode solve(ListNode head) {
        if(head.next == null || head.next.next == null) {
            return head;
        }

        //segregate LL into two halves
        ListNode oh = head;
        ListNode ot = oh;
        ListNode eh = head.next;
        ListNode et = eh;

        ListNode temp = head.next.next;
        int idx = 3;

        while(temp != null) {
            if(idx % 2 == 0) {
                //join temp in even LL
                et.next = temp;
                et = et.next;
            }
            else {
                //join temp in odd LL
                ot.next = temp;
                ot = ot.next;
            }
            temp = temp.next;
            idx++;
        }

        et.next = ot.next = null;


        //reverse even LL
        eh = reverse(eh);


        //merge
        while(eh != null) {
            ListNode ohn = oh.next;
            ListNode ehn = eh.next;

            oh.next = eh;
            eh.next = ohn;

            oh = ohn;
            eh = ehn;
        }
        return head;
    }
}
```

Doubts

```java
public class Solution {
    static class Pair {
        char ch;
        int freq;

        Pair(char ch,int freq) {
            this.ch = ch;
            this.freq = freq;
        }
    }

    public String solve(String A) {
        //step1 : create count array
        int[]count = new int[26];

        for(int i=0; i < A.length();i++) {
            char ch = A.charAt(i);
            int idx = ch - 'a';
            count[idx]++;
        }


        //step2 : create and fill data in Pair array using count array
        Pair[]arr = new Pair[26];
        for(int i=0; i < 26;i++) {
            char ch = (char)(i + 'a');
            int freq = count[i];

            Pair p = new Pair(ch,freq);
            arr[i] = p;
        }

        //step3 : sort the Pair array
        Arrays.sort(arr, new Comparator<Pair>(){
           public int compare(Pair p1, Pair p2) {
               if(p1.freq == p2.freq) {
                   return p1.ch - p2.ch;
               }
               else {
                   return -(p1.freq - p2.freq); //or p2.freq - p1.freq;
               }
           }
        });


        //step4 : create final ans using Pair array
        StringBuilder sb = new StringBuilder("");

        for(int i=0; i < 26;i++) {
            Pair p = arr[i];

            char ch = p.ch;
            int freq = p.freq;

            //add ch, freq times in your ans
            for(int k=1; k <= freq;k++) {
                sb.append(ch);
            }

        }
        return sb.toString();
    }
}
```

S = c a b b a b





(after sorting)

sb => b b b a a c