C K-Beautiful Strings

Description

You are given a string s consisting of lowercase English letters and a number k. Let's call a string consisting of lowercase English letters beautiful if the number of occurrences of each letter in that string is divisible by k. You are asked to find the lexicographically smallest beautiful string of length n, which is lexicographically greater or equal to string s. If such a string does not exist, output −1.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

The first line contains a single integer T $(1 \leq T \leq 10000)$ — the number of test cases.

The next $2 \cdot T$ lines contain the description of test cases. The description of each test case consists of two lines.

The first line of the description contains two integers n and k $(1 \leq k \leq n \leq 10^5)$ — the length of string s and number k respectively.

The second line contains string s consisting of lowercase English letters.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case output in a separate line lexicographically smallest beautiful string of length n, which is greater or equal to string s, or −1 if such a string does not exist.

Example

input

```
4
4 2
abcd
3 1
abc
4 3
aaaa
9 3
abaabaaaa
```

output

```
acac
abc
-1
abaabaaab
```

Method

- 首先,这是一道处理字符串的经典例题,我们看到这个字符串的题目是要输出字典序最小的那个答案,因此,我们也是需要按照字典序来进行枚举
- 这道题用到的核心方法就是采用了一种贪心的思想.我们在这道题目之中研究的是一个$prefix$,即当字符串前面的字符都已经满足要求之后,我们后面的字符要怎么来做.基本的思路就是一个贪心的从后往前枚举,直到找到那个不用动的位置,在这个位置之前的可以原封不动进行输出,而在这个位置之后的则可以依据字典序来进行输出
- 其实这道题还有一个方法值得留意,就是那一步的get.一开始我就完全没有想到可以用这种方法来调整得出剩余步数,因此在这道题上面就跪了

Solution

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

int cnt[26];


//演算为了达到题目所给要求,所需要的步数
int get(int x,int k)
{
    return (k-x%k)%k;
```

```
}



int n,k;

int main()
{
    int t;
    cin>>t;

    while(t--)
    {
        cin>>n>>k;
        string s;
        cin>>s;
        memset(cnt,0,sizeof(cnt));

        for(int i=0;i<s.size();i++)
        {
            cnt[s[i]-'a']++;
        }

        int sum=0;

        for(int i=0;i<26;i++)
        {
            sum+=get(cnt[i],k);
        }
        if(sum==0)
        {
            cout<<s<<endl;
            continue;
        }
        if(n%k!=0)
        {
            cout<<-1<<endl;
            continue;
        }
        int flag=1;
        for(int i=n-1;i>=0;i--)
        {
            sum-=get(cnt[s[i]-'a'],k);//维护该位,从该位开始枚举
            cnt[s[i]-'a']--;
            sum+=get(cnt[s[i]-'a'],k);
            for(int j=s[i]-'a'+1;j<26;j++)
            {
                //枚举看看每个字典序大的行不行
                int tmp=sum;
                sum-=get(cnt[j],k);
                cnt[j]++;
                sum+=get(cnt[j],k);
                //这里ok了
```

```
                if(i+sum+1<=n)
                {
                    for(int pos=0;pos<i;pos++)
                    {
                        cout<<s[pos];
                    }
                    char ch='a'+j;
                    cout<<ch;
                    string ans;
                    for(int w=0;w<26;w++)
                    {
                        int f=get(cnt[w],k);
                        while(f)//完成接下来的目标
                        {
                            f--;
                            ans+='a'+w;
                        }
                    }
                    while(ans.size()+i+1<n)
                    {
                        ans+='a';
                    }
                    sort(ans.begin(),ans.end());
                    cout<<ans<<endl;
                    flag=0;
                    break;
                }
                cnt[j]--;
                sum=tmp;
            }
            if(!flag) break;
        }
    }


}
```

D GCD of an Array

Description

You are given an array a of length n. You are asked to process q queries of the following format: given integers i and x, multiply a_i by x.

After processing each query you need to output the greatest common divisor (GCD) of all elements of the array a.

Since the answer can be too large, you are asked to output it modulo $10^9 + 7$.

Input

The first line contains two integers — n and q $(1 \le n, q \le 2 \cdot 10^5)$.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 2 \cdot 10^5)$ — the elements of the array a before the changes.

The next q lines contain queries in the following format: each line contains two integers i and x $(1 \le i \le n, 1 \le x \le 2 \cdot 10^5)$.

Output

Print q lines: after processing each query output the GCD of all elements modulo $10^9 + 7$ on a separate line.

Example

input

```
4 3
1 6 8 12
1 12
2 3
3 3
```

output

```
2
2
6
```

Method

- 首先要注意到,这个答案肯定是非递减的
- 因此,我们只需要维护那些改变了的质因数,并且这道题目的关键是使用$multiset$来对于那些质因数来进行维护,然后还要用筛法来得到质因数分解
- 这道题目算是一道使用map和$multiset$来维护质因数唯一分解的题目,模板比较清晰易懂,并且以后遇到类似的题目也会有所帮助,是一道好的题目

Solution

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

const int maxn=2e5+5;
ll mod =1e9+7,ans=1;
int nxt[maxn],n;
multiset <int> cnt[maxn];
```

```
map<int,int> cnt_div[maxn];

void add(int i,int x)
{
    while(x!=1)
    {
        //找到的是最小质因数
        int div=nxt[x];
        int tmp=0;
        //类似唯一分解定理那样子来分解这个东西
        while(nxt[x]==div)
        {
            tmp++;
            x=x/nxt[x];
        }
        //map来维护每一个数的唯一分解
        int lst=cnt_div[i][div];
        cnt_div[i][div]+=tmp;
        int lst_min=0;
        //multiset来维护质因数之和,并且因为其有序,总是能找到一个最小的质因数的幂
        if(cnt[div].size()==n)
        {
            lst_min= (*cnt[div].begin());
        }
        if(lst!=0)
        {
            //维护multiset中总的质因数
            cnt[div].erase(cnt[div].find(lst));
        }
        cnt[div].insert(lst+tmp);
        //如果到n个的话,就要开始弄
        if(cnt[div].size()==n)
        {
            for(int j=lst_min+1;j<=(*cnt[div].begin());j++)
            {
                ans= ans*(ll)div%mod;
            }
        }
    }
}


int q,l,x;

int main()
{
    cin>>n>>q;
    //相当于一个质因数的筛,把每一个数的最小质因数给筛出来
    for(int i=2;i<maxn;i++)
    {
        if(nxt[i]==0)
        {
            nxt[i]=i;
            if(i>10000) continue;
```

```
            for(int j=i*i;j<maxn;j+=i)
            {
                if(nxt[j]==0) nxt[j]=i;
            }
        }
    }

    for(int i=1;i<=n;i++)
    {
        cin>>x;
        add(i,x);
    }
    for(int i=1;i<=q;i++)
    {
        cin>>l>>x;
        add(l,x);
        cout<<ans<<endl;
    }



}
```