

Codeforces Round #641 (Div. 2)

A. Orac and Factors

大致题意: $f(n)$ 为 n 除 1 外的最小约数, 令 $g(n) = f(n) + n$ 求 $g_k(n)$

易证 $f(n)$ 与 n 同奇偶, 所以 $f(n) + n$ 必为偶数。又 $f(2k) = 2, k \in \mathbb{N}$ 所以我们只需先计算 $f(n) + n$ 然后加 $k-1$ 个 2 即可。

代码:

```
#include <stdio.h>
int main()
{
    int t;
    scanf("%d",&t);
    for (int kkk=1; kkk<=t; kkk++)
    {
        int n,k;
        scanf("%d%d",&n,&k);
        if (n%2==0)
            printf("%d\n",n+k*2);
        else
        {
            int i;
            for (i=2; i<=n; i++)
                if (n%i==0)
                    break;
            printf("%d\n",n+i+(k-1)*2);
        }
    }
}
```

B. Orac and Models

大致题意: 给定一个序列 a_n 求满足下面条件的递增子列 s_n 的最大长度:

$$1. s_i \bmod s_{i-1} = 0, \text{ for all } 1 < i \leq n$$

$$2. a_{s_{i-1}} < a_{s_i}, \text{ for all } 1 < i \leq n$$

设 $dp[i]$ 代表 s_i 的最后一项为 i 时的最大长度, 可得

$$dp[j] = \max(1, dp[i]), j > i \text{ 且 } j \bmod i = 0$$

代码:

```
#include <stdio.h>
#include <string.h>
#include <iostream>
using namespace std;
int dp[100001];
int a[100001];
int main()
{
    int T;
    scanf("%d",&T);
    while (T--)
    {
        memset(dp,0,sizeof(dp));
        int n;
        scanf("%d",&n);
        for (int i=1;i<=n;i++)
        {
            scanf("%d",&a[i]);
            dp[i]=1;
        }
        for (int i=1;i<=n;i++)
        for (int j=2*i;j<=n;j+=i)
        if (a[j]>a[i])
        {
            dp[j]=max(dp[j],dp[i]+1);
        }
        int ans=0;
        for (int i=1;i<=n;i++)
        ans=max(ans,dp[i]);
        printf("%d\n",ans);
    }
    return 0;
}
```

C. Orac and LCM

大致题意：求 $\gcd(\{lcm(\{a_i, a_j\}) | i < j\})$

当数列的长度等于2时，答案即为 $lcm(a_1, a_2)$

当数列的长度大于2时

注意到 $1 \leq a_i \leq 200000$ ，可以先把1到200000的素数筛出来，下面考虑每一个素数 p 对结果的影响：

1. 假如该素数是数列 a_n 中至多一项的因子，则这个素数对答案不做贡献

2. 否则，由于该题求的是最大公约数，只需求出 $lcm(a_i, a_j)$ 中含有 p 的因子个数最小的那个数，可以证明，最小因子数等于数列 a_n 中含有 p 的次小因子数。

可是这样会超时，我们考虑缩小枚举范围。可以先用假的算法求出一个扩大了范围的答案，对这个答案分解质因数，只需枚举这些素数即可。

代码

```
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#include <vector>
using namespace std;
long long a[200001];
int pow(int a,int b)
{
    int an=1;
    for (int i=1;i<=b;i++)
        an*=a;
    return an;
}
int main()
{
    srand(2);
    int n;
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
        scanf("%lld",&a[i]);
    if (n==2)
    {
        printf("%lld",a[1]*a[2]/__gcd(a[1],a[2]));
        return 0;
    }
    long long an=0;
    for (int i=2;i<=n;i++)
        an=__gcd(an,(a[i]*a[i-1])/__gcd(a[i],a[i-1]));
    int time=0;
    vector<int> fact;
    int tmp=2;
    while(an>=tmp)
    {
        if (an%tmp==0)
            fact.push_back(tmp);
        while(an%tmp==0)
            an/=tmp;
        tmp++;
    }
    int ans=1;
    for (auto factor:fact)
    {
        vector<int> sorts;
        int min1=1e9,min2=1e9;
        for (int i=1;i<=n;i++)
        {
```

```
    int now=0;
    int p=a[i];
    while (p%factor==0)
        p/=factor,now++;
    sorts.push_back(now);
}
sort(sorts.begin(),sorts.end(),[](int a,int b)
{
    return a<b;
}));
if (sorts.size()<=1)
    min2=0;
else min2=sorts[1];
ans*=pow(factor,min2);
}
printf("%d",ans);
}
```

From:

<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:cf641

Last update: **2020/05/13 17:55**

