20分：

考虑枚举左端点l，向右维护哪些m是可行的，以及对应的$a_i \mod m$是多少即可。

复杂度$O(n^2a)$。

40分：

我们考虑先枚举m是多少，把所有数字变成$a_i \mod m$，接下来的问题就变成：给你一个序列，问最长的全相同的序列有多长了。

随便怎么做，复杂度$O(na)$。

70分：

考虑a和b在什么情况下可以模m相同?

我们换种表示：$a = k_a m + t, b = k_b m + t$。

换句话说，$a - b = (k_a - k_b)m$一定是m的倍数。

也就是说，只要选择的m，是a, b差的倍数，就一定能满足条件。（其实也可以打表看出来）。

那么我们维护一个差分数组b_i。

问题就变成：我最多能选多长一段b，使得这一段的gcd非1了。

预处理$g_{i,j}$表示从i出发，向右2^j个元素的gcd，则这个问题可以通过枚举左端点l，倍增右端点r来完成。

```
    ll len=1;
    for (int i=1;i<=n-1;i++)
    {
        ll gg=g[i][0],now=i;
        for (int j=21;j>=0;j--)
        if (__gcd(gg,g[now][j])!=1)
        {
            gg=__gcd(gg,g[now][j]);
            now=now+(1<<j);
            if (now>=n) {now=n; break;}
        }
        len=max(len,now-i+1);
    }
```

时间复杂度$O(n \log n \log a)$，跑的非常慢。

100分：

我们很想双指针，但是双指针不好删元素。

一个脑洞大开的方法是，我对于当前维护的区间$[l,r]$，我找一个位置mid。

我来维护$a_i(l \leq i \leq mid)$表示$\gcd(a_i,\ldots,a_{mid})$。

维护一个$b_i(mid \leq i \leq r)$表示$gcd(b_{mid},\ldots,b_r)$。

这样，对于当前区间$[l,r]$，他们的$\gcd$就是$\gcd(a_l,b_r)$。

对于添加$r+1$，只需要$O(loga)$的速度。

对于删除l，只需要$O(1)$的速度。

当l超过我们的mid时，我们就重新选定mid为r，然后重构数组a,b。

复杂度非常好分析，当mid从x变成y的时候，我只需要重构$[x,y]$内的信息，所以总复杂度是$O(n\log a)$的。非常快。

```
//洛谷上复制来的
#include<cstdio>
typedef unsigned ui;
typedef unsigned long long ull;
const ui M=2e6+5;
ui T,n;ull a[M],f[M];
inline ull gcd(const ull&a,const ull&b){
  return b?gcd(b,a%b):a;
}
signed main(){
  T=1;
  ui i,j,l,mid,ans;
  while(T--){
    scanf("%u",&n);l=mid=1;ans=0;
    for(i=1;i<=n;++i)scanf("%llu",a+i),a[i-1]=a[i-1]>a[i]?a[i-1]-
a[i]:a[i]-a[i-1];
    for(i=1;i<n;++i){
      f[i]=i==1||i-1==mid?a[i]:gcd(f[i-1],a[i]);
      while(l<=mid&&gcd(f[l],f[i])==1)++l;
      if(l>mid){
        f[i]=a[mid=i];
        for(j=i-1;j>=l;--j)f[j]=gcd(a[j],f[j+1]);
        while(l<=i&&f[l]==1)++l;
      }
      if(i-l+1>ans)ans=i-l+1;
    }
    printf("%u\n",++ans);
  }
}
```