CF Round 665 div2

C

题意：给定一个数组a[]可以交换a_i,a_j当且仅当$gcd(a_i,a_j) = min\{a_i\}$[]问最终是否可以把数组变为单调不减的。

题解：考虑当某个a_i不是最小值的倍数，那么它不可能被交换，只会被留在原地。而剩下的都是最小值的倍数，所以如果想交换两个数字，要么可以直接交换，要么可以用最小值的那个数当作跳板交换。所以只要判断不是最小值的倍数的数字是不是在他们排好序之后应该在的位置上就可以。

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <queue>
#include <vector>
using namespace std;
typedef long long ll;
const int MAXN = 1e5 + 20;
int pic[MAXN];
int tem[MAXN];
int b[MAXN];
int read()
{
    int x=0,flag=1; char c=getchar();
    while((c>'9' || c<'0') && c!='-') c=getchar();
    if(c=='-') flag=0,c=getchar();
    while(c<='9' && c>='0') {x=(x<<3)+(x<<1)+c-'0';c=getchar();}
    return flag?x:-x;
}
int main()
{
    int t = read();
    while (t--)
    {
        int n = read();
        int mmin = 0x7f7f7f7f;
        for (int i = 1; i <= n;i++)
        {
            pic[i] = read();
            tem[i] = pic[i];
            mmin = min(pic[i], mmin);
        }
        sort(tem + 1, tem + 1 + n);
        bool ans = true;
        for (int i = 1; i <= n; i++)
            if (pic[i] % mmin != 0 and tem[i] != pic[i])
            {
```

```
                ans = false;
                break;
            }
        if (ans)
            printf("YES\n");
        else
            printf("NO\n");
    }
    return 0;
}
```

F

题意：给定一个长度为2^n的数组□n不超过18□□q次操作：

- 将a_x改为k
- 将区间从头开始划分为多个连续的长度为2^k的子数组，将其分别反转
- 将区间从头开始划分为多个连续的长度为2^k的子数组，将第一个子数组与第二个交换位置，第三个与第四个交换。。。。
- 查询区间和

题解：一开始想想到平衡树维护区间，但是发现这个长度和操作的方式都是2的幂次，维护区间的数据结构中线段树是天然符合这一划分规则的，并且我们发现，区间反转操作，可以视作进行多次区间交换操作，所以我们给线段树的每层打个异或标记，表示这一层的所有节点，都要与相配对的兄弟节点交换位置，那么对于操作3，我们只需在相应的层打一个标记，对于操作2，则要从对应的层开始，一直到最底层都要打个标记，最后查询的时候如果发现标记为1，则往下递归的时候选择另一个子树即可。由于n不超过18，所以这个区间异或可以直接暴力进行。

```
#include <cstring>
#include <iostream>
#include<cstdio>
#include<algorithm>
#include<vector>
#include<stack>
#define ls (node << 1)
#define rs ((node << 1) | 1)
#define mid ((L + R) >> 1)
using namespace std;
typedef long long  ll;
const int MAX = (1 << 18) + 20;
int revisecnt[64],n,q,tf,pic[MAX];
ll sum[MAX << 2];
int read()
{
    int x=0,flag=1; char c=getchar();
    while((c>'9' || c<'0') && c!='-') c=getchar();
    if(c=='-') flag=0,c=getchar();
    while(c<='9' && c>='0') {x=(x<<3)+(x<<1)+c-'0';c=getchar();}
```

```
    return flag?x:-x;
}
inline void build(ll node, ll l, ll r)
{
    if (l == r)
    {
        sum[node] = pic[l];
        return;
    }
    ll midd = (l + r) >> 1;
    build(node << 1, l, midd);
    build((node << 1) | 1, midd + 1, r);
    sum[node] = (sum[node << 1] + sum[(node << 1) | 1]);
}
void change(int node,int goal,int new_val,int L,int R,int deep)
{
    if (L == R)
    {
        sum[node] = new_val;
        return;
    }
    else
    {
        if (goal <= mid)
            change((ls) ^ revisecnt[deep+1], goal, new_val, L, mid,deep+1);
        else
            change((rs) ^ revisecnt[deep+1], goal, new_val, mid+1,
R,deep+1);
        sum[node] = sum[node << 1] + sum[(node << 1) | 1];
    }
}
ll query_sum(int node,int l,int r,int L,int R,int deep)
{
    if (L == l and R == r)
    {
        return sum[node];
    }
    else
    {
        if (r <= mid)
        {
            int temmm = (ls) ^ revisecnt[deep+1];
            int teee = revisecnt[deep+1];
            return query_sum((ls) ^ revisecnt[deep+1], l, r, L,
mid,deep+1);
        }
        else if (l >= mid + 1)
        {
            return query_sum((rs) ^ revisecnt[deep+1], l, r, mid+1,
R,deep+1);
        }
```

```
        else
            return query_sum((ls) ^ revisecnt[deep+1], l, mid, L,
mid,deep+1) + query_sum((rs) ^ revisecnt[deep+1], mid + 1, r, mid+1,
R,deep+1);
    }
}
int main()
{
    tf = read();
    q = read();
    n = 1 << tf;
    for (int i = 1; i <= n;i++)
        pic[i] = read();
    build(1, 1, n);
    for (int i = 1; i <= q;i++)
    {
        int op = read();
        if (op == 1)
        {
            int x = read(), k = read();
            change(1, x, k, 1, n,0);
        }
        else if (op == 2)
        {
            int k = read();
            if (k == 0)
                continue;
            int tem = tf - k  + 1;
            for (int i = tem; i <= tf;i++)
                revisecnt[i]^=1;
        }
        else if (op == 3)
        {
            int k = read();
            int tem = tf - k;
            revisecnt[tem]^=1;
        }
        else if (op == 4)
        {
            int l = read(), r = read();
            ll ans = query_sum(1, l, r, 1, n,0);
            printf("%lld\n", ans);
        }
    }
    return 0;
}
```