题意解释

定义一对整数是“坏的”，当且仅当它们的和是2的幂次。

定义$f(l,r)=[l,r]$ 中最多选出多少个数字，不存在任意一对坏的数对。

有 m 次询问，每次询问$f(l_i,r_i)$的值。 最后还需要输出 $\sum_{1\le l\le n}\sum_{l\le r\le n}f(l,r)$ 的值

知识点提炼

树状数组，动态规划，莫队

核心解题思路

思路一：$n\le 20, m\le 3$

模拟题意即可，期望得分 10 分。

思路二：$n\le 500$

不妨把所有和是2的幂次的数对连一条边，打表/观察发现是一片森林。

这个证明是简单的，只需要把所有边定向为从大指向小，然后不难通过反证法证明一个数只会最多向一个比他小的数连边，这符合森林的特征。

所以要求的东西实际上就是森林的最大独立集，每次询问直接暴力做个树上DP就好，复杂度 $O(n^3)$。期望得分 30 分。

思路三：$l=1$，无需输出子区间和

考虑离线，把右端点从小到大加入森林，每次一定是加入一个叶子。

容易证明森林的最大深度不超过 $\log n$，所以可以每次加入叶子后暴力更新叶子的所有祖先的DP值，复杂度 $O(n\log n)$。期望得分 45 分。

思路四：$n\le 2000$

枚举左端点，每一个左端点重复思路三的过程，复杂度 $O(n^2\log n)$。期望得分 55 分。

思路五：$n\le 20000$, 无需输出子区间和

考虑莫队。删除左端点时，不难发现一定是删除一棵树的根。我们取答案时，取的实际上就是森林每一个根的答案之和。每个点的儿子数量是 $O(\log n)$ 的，所以只要减去原来根的贡献，加上每个儿子的贡献即可，复杂度 $O(n\sqrt{n}\log n)$。期望得分 65 分。

思路六：无特殊限制

将 $[1,n]$ 所有的边建出，求最大独立集，假设以 u 为根节点的答案是 f_u。

思路五可以进一步优化，固定右端点时，移动区间的左端点对于答案的变化值是固定的，都是 $\Delta_u=f_u-\sum_{v\in \mathrm{son}_u}f_v$。

所以可以在所有的右端点处统计答案，只需要单点查询以及查询区间和，树状数组维护即可。看似是 $O(n\log^2 n)$ 的，实际精细计算后是 $O(4n\log n)$ 的计算量。

至于所有的询问的和，也是在右端点处统计 $\sum_l f(l,i)$即可，只需要维护一个前缀的总和，在维护 Δ_u 的同时考虑对答案的影响就可以了，复杂度 $O(n\log n)$，期望得分 100 分。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef double dou;
typedef pair<int,int> pii;
#define fi first
#define se second
#define mapa make_pair
typedef long double ld;
typedef unsigned long long ull;
template <typename T>inline void read(T &x){
    x=0;char c=getchar();bool f=0;
    for(;c<'0'||c>'9';c=getchar()) f|=(c=='-');
    for(;c>='0'&&c<='9';c=getchar())
    x=(x<<1)+(x<<3)+(c^48);
    x=(f?-x:x);
}
const int N=3e5+5;
int n, m, tp;
vector<pii> vec[N];
int res[N];
int lg[N];
int fa[N];
int f[N][2];
int sum[N];
int stk[20], top;
int tr[N];
inline void add(int x, int v){              // 树状数组
    for(; x; x-=(x&-x)) tr[x]+=v;
}
inline int get(int x){
    int ret=0;
    for(; x<=n; x+=(x&-x)) ret+=tr[x];
    return ret;
}
int main(){
    read(n); read(m); read(tp);
    for(int i=1, l, r; i<=m; ++i){
        read(l); read(r);
        vec[r].emplace_back(l, i);                  // 在所有的右端点处统计答案
    }
    for(int i=2; i<=n; ++i){
        lg[i]=lg[i>>1]+1;
    }
    ll ans=0, cur=0;
    for(int i=1; i<=n; ++i){
        if(i==(1<<lg[i])) fa[i]=0;                  // 2的幂次不能连向更小的点，是根节
点
        else fa[i]=(1<<(lg[i]+1))-i;
```

```
        cur+=i;
        add(i, 1);
        stk[top=1]=i;
        while(fa[stk[top]]) stk[top+1]=fa[stk[top]], ++top;
        for(int j=top; j>1; --j) {
            int x=stk[j], y=stk[j-1];
            cur-=(ll)x*(max(f[x][0], f[x][1])-sum[x]);
            add(x, -(max(f[x][0], f[x][1])-sum[x]));
            f[x][0]-=max(f[y][0], f[y][1]);
            f[x][1]-=f[y][0];
            sum[x]-=max(f[y][0], f[y][1]);               // 去除之前的贡献
        }
        f[i][0]=0; f[i][1]=1;
        for(int j=2; j<=top; ++j){
            int x=stk[j], y=stk[j-1];
            sum[x]+=max(f[y][0], f[y][1]);
            f[x][0]+=max(f[y][0], f[y][1]);
            f[x][1]+=f[y][0];
            add(x, max(f[x][0], f[x][1])-sum[x]);
            cur+=(ll)x*(max(f[x][0], f[x][1])-sum[x]);   // 加上新的贡献
        }
        ans+=cur;
        for(auto t:vec[i]) res[t.se]=get(t.fi);
    }
    for(int i=1; i<=m; ++i) printf("%d\n", res[i]);
    printf("%lld\n", ans*tp);
    return 0;
}
```

本题易错点

- 要注意到移动左端点和移动右端点对于答案的影响是不同的