第六周直播课（上午）作业讲评

洛谷网校

基础-提高衔接计划

2025-08

disangan233

洛谷

www.luogu.com.cn

【第六周】阻拦乘积

- T654609
- 给定长度为 n 的正整数序列 a，q 组询问
- 每次给定 x，令 y 为 x 右边第一个满足 $a_y \geq a_x$ 的位置 y，没有则为 $n+1$，求

$$\left(\prod_{i=x}^{y-1} a_i\right) \bmod 10000 = a_x \times \ldots \times a_{y-1} \bmod 10000$$

- $n, q \leq 2 \times 10^5$，$a_i \leq 10^9$

【第六周】阻拦乘积

- 对于每个询问 x，找到被拦住的山 y，那么只要求出 x 往后连续 $y-x$ 座山高度的乘积 $\bmod 10000$ 即可
- 以 $f_{k,x}$ 表示从第 x 座山开始连续 2^k 座山 $[x, x+2^k-1]$ 高度的乘积 $\bmod 10000$ 的结果
- 倍增预处理，每组询问对 $y-x$ 进行二进制分解即可
- 使用单调栈来预处理每一个 x 对应的 y
- 也可以用倍增预处理区间 $\max$ 来找 y
- 时间复杂度 $O((n+q)\log n)$

【第六周】阻拦乘积

```
for(int i=1;i<=n;i++)
    mx[0][i]=a[i],mul[0][i]=a[i]%mod;
for(int k=1;k<=J;k++)
for(int i=1;i+(1<<k)-1<=n;i++)
{
    mx[k][i]=max(mx[k-1][i],mx[k-1][i+(1<<(k-1))]);
    mul[k][i]=mul[k-1][i]*mul[k-1][i+(1<<(k-1))]%mod;
}
```

【第六周】阻拦乘积

```
scanf("%d",&x);
int pos=x+1;
int ans=1;//已经确认不会被[x,pos-1]拦住
for(int k=J;k>=0;k--)
{
    if(pos+(1<<k)-1<=n&&mx[k][pos]<a[x])
    //[pos,pos+(2^k)-1]<a[x]
    //[x,pos-1]不会拦住
    //[pos,pos+(2^k)-1]不会拦住
    //-> [x,pos+(2^k)-1] 不会拦住
    {
        // ans=ans*mul[k][pos]%mod;
        pos+=(1<<k);
    }
}
int d=pos-x;//从x往后d座山的乘积
for(int k=J;k>=0;k--)
if(d&(1<<k)) ans=ans*mul[k][x]%mod,x+=(1<<k);
printf("%d\n",ans);
```

【第六周】树上阻拦乘积

- T654608
- T654609 【第六周】阻拦乘积的树上版本
- 给定一棵 n 个点的树，点 i 权值为 a，q 组询问
- 每次给定 x，令 y 为 x 所有祖先中第一个满足 $a_z \geq a_x$ 的位置 z 对应的子节点，y 不存在则为 1，求

$$\left(\prod_{u \in \mathsf{path(x,y)}} a_u \right) \bmod 10000$$

- 也就是 u 开始跳到下一个点权值大于等于 a_u 时经过的所有点的点权之积
- $n, q \leq 2 \times 10^5$，$a_i \leq 10^9$

【第六周】树上阻拦乘积

- 同前一题，我们先考虑求，从 x 向上跳多少步会被拦住
- 以 $mx_{k,x}$ 表示 x 向上跳 2^k 步内 a_i 的最大值，$mul_{k,x}$ 表示 x 向上跳 2^k 步内 a_i 的乘积
- 求出会被 fa_{pos} 拦住，就对 $dep_x - dep_{pos}$ 二进制分解

```
int pos=fa[0][x];//从x节点跳到pos之前不会被拦住
int ans=1;
for(int k=J;k>=0;k--) {
    if(dep[pos]>=(1<<k)&&mx[k][pos]<a[x]) {
        pos=fa[k][pos];
    }
}
int d=dep[x]-dep[pos];
for(int k=J;k>=0;k--) {
    if(d&(1<<k))
        ans=ans*mul[k][x]%mod,x=fa[k][x];
}
```

【第六周】树上阻拦乘积

```
1  void dfs(int u) {
2      mx[0][u]=a[u];
3      mul[0][u]=a[u]%mod;
4      for(auto v:e[u]) {
5          if(v==fa[0][u]) continue;
6          dep[v]=dep[u]+1;
7          fa[0][v]=u;
8          dfs(v);
9      }
10 }
11 void pre() {
12     for(int k=1;k<=J;k++)
13     for(int i=1;i<=n;i++) {
14         fa[k][i]=fa[k-1][fa[k-1][i]];
15         mx[k][i]=max(mx[k-1][i],mx[k-1][fa[k-1][i]]);
16         mul[k][i]=mul[k-1][i]*mul[k-1][fa[k-1][i]]%mod;
17     }
18 }
```

【第六周】或或和之和

- T654611
- 给定长度为 n 的序列 a，定义 $f(l,r) = a_l \text{ or } a_{l+1} \text{ or } \ldots \text{ or } a_r$，求

$$\sum_{1 \le l \le r \le n} f(l,r)$$

- $n \le 5 \times 10^5$，$0 \le a_i < 2^{20}$

【第六周】或或和之和

- 拆位考虑，假设当前在考虑 2^k 的这一位
- 那么就是要计算有多少个区间或的结果在这一位为 1
- 枚举右端点 r，考虑有效左端点范围
- $[1,p]$ 其中 p 是最大的满足 $p \le r$ 且 a_p 在 2^k 这一位为 1
- 等价于记录 01 序列当前 $[1,r]$ 最后一个 1 的位置，如果 r 这一位是 1 就更新
- 时间复杂度 $O(n \log a)$

【第六周】或或和之和

```
for(int k=0;k<=20;k++)
{
    int lst=0;//lst表示[1,i]内最后一个第k位为1的数字下标
    for(int i=1;i<=n;i++)//i作为右端点
    {
        if((a[i]>>k)&1)//左端点<=lst时区间或在2^k这位为1
        lst=i;
        ans+=(1ll<<k)*lst;//有效的左端点[1,lst]
        //有lst个区间答案里包括2^k
    }
}
```

【第六周】苹果

- T654612
- 给定一个长度为 n 的 01 字符串。对于每一位，如果是 1 则代表有该位置一个苹果，反则代表为空
- 要在剩余的空位中再放入两个苹果，请你求出所有方案中，最近的两个苹果的距离的最大值
- $n \leq 10^5$

【第六周】苹果

- 答案“最小值的最大值”具有单调性，考虑二分答案
- 如何去 check 一个距离 d 是否能放入两个苹果？

【第六周】苹果

- 答案“最小值的最大值”具有单调性，考虑二分答案
- 如何去 check 一个距离 d 是否能放入两个苹果？
- 贪心检查每一个 0 的位置，如果距离上一个苹果距离为 d 就放
- 如果下一个 1 发现此时距离小于 d，撤回这个苹果
- 最后能放下至少两个苹果就返回 true

【第六周】苹果

```
int m=0;
int last=-1e9;
for(int i=0;i<n;i++){
    if(s[i]=='1'&&i-last<mid) return 0;//放前就不满足条件
    if(s[i]=='1'){
        last=i;
    }
}
last=-1e9;//上一个
for(int i=0;i<n;i++){
    if(s[i]=='1'&&i-last<mid) m--;//不满足条件
    if((i-last>=mid)&&s[i]!='1'){//可放
        last=i;
        m++;//计数器++
    }
    if(s[i]=='1') last=i;
}
return m>=2;
```

【第六周】合成

- T654613
- 有 n 种金属，第 i 种金属有 a_i 个
- 有 k 个合成表，一个合成表形如 $x, m, b_1, \ldots, b_m$，代表用 $b_1, \ldots, b_m$ 这 m 个金属可以合成一个金属 x，对于任意的 b_i 满足 $x > b_i$
- 计算经过一系列转化后，最多可以合成多少个第 n 种金属
- $1 \leq k < n \leq 100$，$0 \leq a_i \leq 10^4$
- 对于一个金属至多存在一个合成表

【第六周】合成

- 由于合成的金属编号一定大于所有原料，可以用 $x \to b_i$ 来建图
- 得到一个 DAG 合成关系
- 如果我们需要金属 x，此时有就直接提供，没有则遍历出边的每一个原料
- 这样得到了一个 dfs 的思路，`dfs(x)` 代表是否能提供金属 x
- 一直 `while(dfs(n)) ans++;`
- 复杂度是多少？

【第六周】合成

- 由于合成的金属编号一定大于所有原料，可以用 $x \to b_i$ 来建图
- 得到一个 DAG 合成关系
- 如果我们需要金属 x，此时有就直接提供，没有则遍历出边的每一个原料
- 这样得到了一个 dfs 的思路，`dfs(x)` 代表是否能提供金属 x
- 一直 `while(dfs(n)) ans++;`
- 复杂度是多少？
- 原料是 10^4 范围，那么至多合成 $10^4 \times n$ 个
- dfs 一次复杂度 $O(n)$，总复杂度 $O(n^2 \max a_i)$

【第六周】合成

```
bool dfs(int u){
    if(a[u]){
        a[u]--;
        return 1;
    }
    if(e[u].empty()) return 0;
    for(auto v:e[u])
        if(!dfs(v))
            return 0;
    return 1;
}
while(k--){
    cin>>l>>m;
    while(m--){
        int u; cin>>u;
        e[l].push_back(u);
    }
}
while(dfs(n)) ans++;
```

[JOI 2022 Final] 星际蛋糕

- P8160
- 有一个长度为 n 的数组 a，你需要一直操作直到这个数组没有偶数
- 每次操作找到这个数组中最靠右的偶数 k，将其分成两个相同的数 $\frac{k}{2}$，不改变其它数的位置
- 接下来有 q 个询问，每次询问 a_x 的值
- $n, q \leq 2 \times 10^5$，$a_i \leq 10^9$，$x_i \leq 10^{15}$

[JOI 2022 Final] 星际蛋糕

- 整段考虑，从初始第 i 块蛋糕切割出的所有蛋糕长度都是相等的
- 预处理出初始第 i 块蛋糕切割出的蛋糕长度和个数，并将个数用前缀和进行处理
- 询问时二分一下第 x 块蛋糕在初始第几块蛋糕中，直接输出之前预处理好的从这一块蛋糕切割出的每一块蛋糕的长度即可
- $O(n \log a + q \log n)$

[JOI 2022 Final] 星际蛋糕

```
read(n);
for(int i = 1; i <= n; ++i) {
    read(a[i]);
    long long cnt = 1;
    while(!(a[i] & 1)) {
        a[i] >>= 1;
        cnt <<= 1;
    }
    s[i] = s[i - 1] + cnt;
}
read(q);
while(q--) {
    read(x);
    int id = lower_bound(s + 1, s + n + 1, x) - s;
    write(a[id]);
    puts("");
}
```

[蓝桥杯 2017 省 B] k 倍区间

- P8649
- 给定长度为 n 的一段序列 a，求此序列中有多少子区间和为 k 的倍数
- $n, k, a_i \leq 10^5$

[蓝桥杯 2017 省 B] k 倍区间

- P8649
- 给定长度为 n 的一段序列 a，求此序列中有多少子区间和为 k 的倍数
- $n, k, a_i \leq 10^5$
- 求 a_i 的前缀和 s_i，那么 $[l, r]$ 的区间和等于 $s_r - s_{l-1}$
- 现在要区间和是 k 的倍数，那么 $s_r - s_{l-1} \equiv 0 \pmod{k}$
- 等价于 $s_r \equiv s_{l-1} \pmod{k}$
- 也就是说，这两个前缀和对 k 取模的结果是相等的

[蓝桥杯 2017 省 B] k 倍区间

- P8649
- 给定长度为 n 的一段序列 a，求此序列中有多少子区间和为 k 的倍数
- $n, k, a_i \leq 10^5$
- 求 a_i 的前缀和 s_i，那么 $[l, r]$ 的区间和等于 $s_r - s_{l-1}$
- 现在要区间和是 k 的倍数，那么 $s_r - s_{l-1} \equiv 0 \pmod{k}$
- 等价于 $s_r \equiv s_{l-1} \pmod{k}$
- 也就是说，这两个前缀和对 k 取模的结果是相等的
- 那么对于 s_i，能贡献出的答案就是 $s_0, \ldots, s_{i-1}$ 中模 k 同余的前缀和数
- 只需要在循环枚举的时候记录余数为 i 的前缀和个数即可

[蓝桥杯 2017 省 B] k 倍区间

```
cin>>n>>k;
for(int i=1,t;i<=n;i++)
    cin>>t,s[i]=s[i-1]+t;
for(int i=0;i<=n;i++) // 从 0 开始
    ans+=b[s[i]%k]++;
    // 统计答案时也要加上当前余数，方便为以后的 i 服务
cout<<ans;
```

[JOI 2021 Final] 有趣的家庭菜园 4

- P7404
- 给定长度为 n 的一段序列 a，每次可选择区间 $[L, R]$ 加一
- 求最少操作次数使得存在 k 满足
- $a_1 < a_2 < \ldots < a_k$，$a_k > a_{k+1} > \ldots > a_n$
- $n \leq 2 \times 10^5$，$a_i \leq 10^9$

[JOI 2021 Final] 有趣的家庭菜园 4

- 对原数组做差分，令差分数组为 b
- $a_1 < a_2 < \ldots < a_k$，$a_k > a_{k+1} > \ldots > a_n$ 等价于
- $b_1, b_2, \ldots, b_k > 0$，$b_{k+1}, b_{k+2}, \ldots, b_n < 0$
- 一次区间 $+1$ 相当 $b_l \leftarrow b_l + 1, b_{r+1} \leftarrow b_{r+1} - 1$
- $r = n$ 时此时只有一个位置 $+1$
- 那么总操作数等于需要 $+1$ 的次数和需要 -1 的次数的最大值

[JOI 2021 Final] 有趣的家庭菜园 4

- 定义 pr_i 为将 b_i 及其之前所有数变成正数的次数
- 定义 nx_i 为将 b_i 及其之后所有数变成负数的次数
- 答案为 $\min_{1\le i\le n}\{\max(pr_i, nx_{i+1})\}$
- 做一次前缀和、一次后缀和即可

[JOI 2021 Final] 有趣的家庭菜园 4

- 定义 pr_i 为将 b_i 及其之前所有数变成正数的次数
- 定义 nx_i 为将 b_i 及其之后所有数变成负数的次数
- 答案为 $\min_{1\le i\le n}\{\max(pr_i, nx_{i+1})\}$
- 做一次前缀和、一次后缀和即可

```
cin>>n;
for(int i=1;i<=n;i++)
    cin>>a[i],sub[i]=a[i]-a[i-1];
for(int i=1;i<=n;i++)
    if(sub[i]>0)pr[i]=pr[i-1];
    else pr[i]=pr[i-1]-sub[i]+1;
for(int i=n;i>=1;i--)
    if(sub[i]<0)nx[i]=nx[i+1];
    else nx[i]=nx[i+1]+sub[i]+1;
for(int i=1;i<=n;i++)
    mini=min(mini,max(pr[i],nx[i+1]));
cout<<mini;
```

[CCC 2007] Snowflake Snow Snowflakes

- P10467
- 给定 n 个雪花，问是否有两个雪花是相同的
- 一个雪花原始表示为一个长度为 6 的环，可以选取一个地方断开然后顺时针或逆时针给出
- 例如雪花 $1\ 2\ 3\ 4\ 5\ 6$ 也可以表示为 $3\ 4\ 5\ 6\ 1\ 2$ 或 $4\ 3\ 2\ 1\ 6\ 5$
- $n \leq 10^5$

[CCC 2007] Snowflake Snow Snowflakes

- P10467
- 给定 n 个雪花，问是否有两个雪花是相同的
- 一个雪花原始表示为一个长度为 6 的环，可以选取一个地方断开然后顺时针或逆时针给出
- 例如雪花 $1\ 2\ 3\ 4\ 5\ 6$ 也可以表示为 $3\ 4\ 5\ 6\ 1\ 2$ 或 $4\ 3\ 2\ 1\ 6\ 5$
- $n \leq 10^5$
- 枚举 6 个位置依次作为第一个位置，处理顺时针和逆时针
- 这样能得到一个雪花的 12 种表示
- 把每种表示处理为一个字符串，存到 map 里统计出现次数即可

[CCC 2007] Snowflake Snow Snowflakes

```
unordered_map <string, int> mp;
string a[7];
int n;
void add(int st, int dis) {
    string s = ""; int i = st;
    do {
        s += a[i] + ',';
        i = (i + dis + 5) % 6 + 1;
    } while (i != st);
    if (mp[s] ++) { // 之前有相同的雪花
        flag = 1;
    }
}
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= 6; j++) cin >> a[j];
    for (int j = 1; j <= 6; j++) add(j, -1), add(j, 1);
}
```

[NOISG 2024 Prelim] School Photo

- P10710
- 有所学校有 n 个班，每个班有 s 名同学。第 i 个班中的第 j 名同学的身高是 $a_{i,j}$
- 现在从每个班上选出一名同学，使得这 n 名同学中最高的同学和最低的同学的身高差最小
- 求这个最小值
- $n, s \leq 1000$，$a_i \leq 10^9$

[NOISG 2024 Prelim] School Photo

- 我们如果把这 $n \times m$ 个数升序排起来，要使得差尽可能小，所选的那些数一定是靠得近的
- 使用队列 (或者双指针) 维护所选区间，队首就是最小的，队尾是最大的
- 如果某一个班级在队尾已经入队，同时队首也是这个班级，那么就可以把队首出队，同时维护当前不同班级的数量，当所有班级都有时更新答案

[NOISG 2024 Prelim] School Photo

```
for(int i=1;i<=n;i++)
for(int j=1,val;j<=s;j++){
    cin>>val;
    vec.push_back({val,i});
}
sort(vec.begin(),vec.end());
for(int i=0;i<vec.size();i++){
    if(!vis[vec[i].second])
        cnt++;//记录颜色数
    vis[vec[i].second]++;
    q.push(vec[i]);
    while(vis[q.front().second]>1){
        vis[q.front().second]--;//删除多余的
        q.pop();
    }
    if(cnt==n)//符合条件
    ans=min(ans,q.back().first-q.front().first);
}
```

【MX-J15-T3】叉叉学习与自我和解

- P12683
- 给定一棵以 0 号节点为根的树，表示从起点 0 到所有其他节点的最短路树（BFS 树）
- 问在所有可能的原图中（边权均为 1），有多少种不同的图能生成这棵最短路树？
- 答案对 10^9+7 取模
- $n \le 10^6$

【MX-J15-T3】叉叉学习与自我和解

- 因为根是 0，所以我们可以首先从根开始做一遍 DFS，求出每个点到根的距离，记为 dep_i
- 考虑从树到图新增的一条边 (i,j)，设 $dep_i < dep_j$
- 如果 $dep_j \geq dep_i + 2$，那么在连了 (i,j) 之后，0 到 j 的最短路就变成了 $dep_i + 1$，与题意矛盾
- 所以一个点只能和与自己深度相差不超过 1 的点连边

【MX-J15-T3】叉叉学习与自我和解

- 因为根是 0，所以我们可以首先从根开始做一遍 DFS，求出每个点到根的距离，记为 dep_i
- 考虑从树到图新增的一条边 (i,j)，设 $dep_i < dep_j$
- 如果 $dep_j \geq dep_i + 2$，那么在连了 (i,j) 之后，0 到 j 的最短路就变成了 $dep_i + 1$，与题意矛盾
- 所以一个点只能和与自己深度相差不超过 1 的点连边
- 加法原理可以算出能连多少条边：i 向深度为 $dep_i, dep_i - 1$ 的所有点连边
- 因为这些边都相互独立，所以再用乘法原理，假设有 p 条边可以连，答案就是 2^p

【MX-J15-T3】叉叉学习与自我和解

```cpp
void dfs(int now, int fa) {
    dep[now] = dep[fa] + 1; cnt[dep[now]]++;
    for(auto v: e[now]) {
        if(v == fa) continue;
        dfs(v, now);
    }
}
dep[n + 1] = -1, dfs(0, n+1);
int ans1 = 0, ans2 = 0;
for(int i = 1;i < n;i++) {
    ans1 += cnt[dep[i]-1] - 1,
    ans2 += cnt[dep[i]] - 1; // 同深度每条边会算两次
}
cout << qpow(2, ans1 + ans2 / 2) << endl;
```

[COCI 2019/2020 #4] Spiderman

- P13423
- 给定长度为 n 的一段序列 a 和正整数 k
- 对于每一个 i，求有多少个 $j(i \neq j)$ 满足 $a_i \bmod a_j = k$
- $n \leq 3 \times 10^5$，$k, a_i \leq 10^6$

[COCI 2019/2020 #4] Spiderman

- 考虑到 $a_i \bmod a_j = k$，那么 $a_i = p \times a_j + k (a_j > k)$
- 对于一个 a_j，能贡献的点是 $\frac{N}{a_j}$ 个的
- 用桶存下每个值的个数，从 $k+1$ 开始枚举每个权值，这样的复杂度是

$$\frac{N}{k+1} + \ldots + \frac{N}{N} < O(N \log N)$$

- 也就是离线处理每个点权 + 调和级数

[COCI 2019/2020 #4] Spiderma

```
for (int i = 0; i < N; i++) {
    scanf("%d", &h[i]); f[h[i]]++;
    if (h[i] > cnt) cnt = h[i];
}
for (int i = K + 1; i <= cnt; i++) {
    if (f[i] == 0) continue;
    for (int j = 0; ; j++) {
        int x = j * i + K;
        if (x > cnt) break;
        if (f[x] > 0) {
            ans[x] += f[i];
        }
    }
}
for (int i = 0; i < N; i++) {
    long long p=ans[h[i]];
    if (K == 0) p--;
    printf("%lld␣",p);
}
```

[蓝桥杯 2022 省 A] 青蛙过河

- P8775
- 有一只青蛙要过河，它需要经过一些石头才能到达距离 n 的对岸
- 有 $n-1$ 个石头，位置 i 的石头高度为 h_i，$h_i=0$ 代表没有石头
- 青蛙的最大跳跃距离为 y，每从一块石头上跳走，这块石头的高度会下降 1
- 当石头的高度下降到 0 时小青蛙不能再跳到这块石头上
- 现在青蛙需要过河 $2x$ 次，求 y 的最小值使得能全部过河
- $n \le 10^5$，$h_i \le 10^4$，$x \le 10^9$

[蓝桥杯 2022 省 A] 青蛙过河

- 题目可以看作有 $2x$ 只青蛙一起从左岸跳到右岸
- 这个题目的答案也是具有单调性的，y 可以 $y+1$ 也可以
- 考虑二分答案如何 check？

[蓝桥杯 2022 省 A] 青蛙过河

- 结论：y 合法等价于所有长度为 y 的区间内石头高度的和 $\geq 2x$
- 证明：容易发现，任意一个长度为 y 的区间都会被每只青蛙踩至少一次，那么 $2x$ 只青蛙就至少会踩 $2x$ 次，所以每个区间内的高度和都必须要 $\geq 2x$
- 在这 $2x$ 只青蛙过河的任意时刻，每个区间的青蛙个数和不会超过 $2x$，所以满足上面的条件后一定可以使所有青蛙过河
- 有了这个性质之后，check 函数就很好写了，使用前缀和快速计算出每个区间的高度和即可

[蓝桥杯 2022 省 A] 青蛙过河

```
bool check(long long y){
    for(int i=y;i<n;i++)
        if(s[i]-s[i-y]<2*x)return 0;
    return 1;
}
for(int i=1;i<n;i++){
    cin>>h[i];
    s[i]=s[i-1]+h[i];
}
l=1,r=n;
while(l<=r){
    long long mid=(l+r)/2;
    if(check(mid))ans=mid,r=mid-1;
    else l=mid+1;
}
```

[COCI 2022/2023 #5] Diskurs

- P9178
- 给一个长度为 n 的整数数组 a，任何一个数都小于 2^m
- 求对于每个 i

$$\max_{1\le j\le n}\{\text{popcount}(a_i \oplus a_j)\}$$

- $\text{popcount}(x)$ 指 x 二进制表示下 1 的个数
- $n \le 2^m$，$m \le 20$

[COCI 2022/2023 #5] Diskurs

- 正着做很难做，但是如果是求 $\mathrm{popcount}(a_i \oplus a_j)$（$a_i$ 和 a_j 的 hamming 距离）的最小值是可以 bfs 的
- 那么我们对每个数取个反，然后去求 $\min\{\mathrm{popcount}(a_i \oplus a_j)\}$
- 注意到这个值不会超过 m（一共就 m 位），枚举 $2^0, \ldots, 2^{m-1}$ 转移 m 次，每一次更新当前的最小值

$$dis_{j \oplus 2^i} = \min(dis_{j \oplus 2^i}, dis_j + 1)$$

- 初始化 $dis_{a_i} = 0$，别的为 inf
- 最后 a_i 对应的答案是 $m - dis_{2^m - 1 - a_i}$

[COCI 2022/2023 #5] Diskurs

```
cin>>n>>m;
for(ll i=0;i<N;i++) dis[i]=INF;
for(ll i=1;i<=n;i++) {
    cin>>a[i];
    dis[a[i]]=0;
}
for(ll i=0;i<m;i++)
for(ll j=0;j<(1<<m);j++)
    dis[j^(1<<i)]=min(dis[j^(1<<i)],dis[j]+1);
for(ll i=1;i<=n;i++) cout<<(m-dis[(1<<m)-1-a[i]])<<'␣';
```