2024acm暑期团队赛3

A - a = b + c

可以先选择一个常数M，对能量小于等于M的中子用dp算出能量为i的中子的最小能量$dp[i]$,转移方程见代码，对于大于M能量的中子,最

优策略应该是选择一个1~n的中子循环num次不断将其能量分解至$K \leq M$以下后用$dp[k] + num * a_i$求得，此处M选择的是$1e5$

B - kkkeyboardd

分别记录两个字符串中字母出现次数，相减不为0就输出

C - biubiubiu

思路：这题可以明确环内任意两个点都是顶点对，同时连向环的链上的每个点与环内的点和链上在他之前的点都是顶点对。

所以这道题本质就是找环和环外的链。

对于找环，可以使用两次dfs解锁，第一个dfs找环上的点，第二个dfs找整个环上的点并计数，具体方案如下。

- 第一次dfs通过状态数组找到每个环上的一个点，同时对他们进行编号$1, 2, 3 \ldots n$。
- 第二个dfs从第一个dfs找到的环上的点开始遍历，若能到达环上的点（编号不为0），那么就是该点便是此时遍历的环的点，进行编号，同时该编号环的点个数$+1$。

至此，所有环和所有环上的点以及环的大小全部确定，剩下的就是找图上的链，由于链是连向环的，通过环不好找链，所以我们可以**建反图**，这样通过找到的环上的点，判断是否有向外的边即可。

D - do you like xor?

思路：不难看出，在$\lceil \frac{r-l+1}{2} \rceil$的时间代价下，我们本质上只有选择单独一个元素和相邻两个元素与x进行异或的操作。

何时才能通过相邻两个元素的操作来减少多次操作?

我们可以考虑一个动态规划。$dp[i]$ 为前i个数变成0的最小代价。

- 如果$a[i]$为0，则$dp[i] = dp[i-1]$。
- 否则，要使得$a[i]$变成0，它必然要异或一个$a[i]$。并且以这个位置终止的操作次数为1次（一定存在一个最优操作方案，使得$a[i]$为右端点的次数为1）。
- 因此只有两种选择方案
    - 选择操作1，将这个数置零，此时$dp[i] = dp[i-1] + 1$

- 连续选择若干个操作2，如果这一段操作后，不全变成0，那跟全部进行炒作1没有区别，因此不用考虑，只需考虑操作后变成0的情况。而要变成0，需要 $a[j] \oplus a[j+1] \oplus \ldots \oplus a[i-1] \oplus a[i] = 0$，必须这一段$[j+1, i-1]$的异或和为0。所以我们可以记录前缀异或和，当$s[j] = s[i]$时即可。此时$dp[i] = dp[j] + (i-j-1)$。所以我们可以用$map$记录前缀异或和为$x$的$min(dp[j] - j)$

所以可以列出转移方程$dp[i] = min(dp[i-1] + 1, mp[dp_{s[j]=s[i]}] + i - 1)$

E - do you like sorting?

找到最小的 u，满足 $u \neq a_u$（若没有这样的 u，则序列已经有序），然后找到最大的 v, 满足 $a_v < a_u$ ，排序区间 $[u, v]$。 这样的操作最多只需要 $\lfloor \frac{n}{2} \rfloor$ 次循环就能排 序原序列。

F - 你补题了吗?

思路：二分第k小的雪球的体积，分析可知，对于x如果 $x \geq v_l + v_r$ 那么 $v_i (l \leq i \leq r-1) + v_r$ 都小于x。check函数判断当前二分的体积在该数组中位于第几小，$\geq$ k则左边界更新，反之右边界更新。

G - im 扫雷 master

考虑贪心。假设第 i 轮购买一个地雷探测器所需的扫雷币 c_i 最少，那么第 i 轮及之前轮次获得的扫雷币留到第 i 轮购买最优。 剩下的扫雷币与第 $i+1$ 轮开始的 $n-i$ 轮构成子问题，重复贪心 直到所有轮次均考虑完即可。

H - 收集龙珠召唤神龙

思路：对于每个点先询问$(0,0)$点，得到$z = x^2 + y^2$，不难发现在$1e12$范围内满足条件的x和y的取值很少，可以直接暴力枚举后询问，时间复杂度为$o(n \times 1e6)$

I - i hate English

大大大大大大模拟，给个代码自己体会吧，情况考虑全了就能ac

```
void solve(){
    string s;
    cin >> s;
    int n = s.size();
    if(s[0] == '-' || s[n - 1] == '-'){
        cout << "invalid" << endl;
        return;
    }
    int cnt = 0;
    vector<int> pos,val;
    for(int i = 0;i < n;i ++){
        if(s[i] == '-'){
            pos.push_back(i);
```

```
        }else{
            if(s[i] == 'X'){
                cnt ++;
                val.push_back(10);
            }else val.push_back(s[i] - '0');
        }
    }
    if(val.back() == 10) cnt --;
    if(cnt >= 1){
        cout << "invalid" << endl;
        return;
    }
    if(val.size() != 10){
        cout << "invalid" << endl;
        return;
    }
    if(pos.size() >= 4){
        cout << "invalid" << endl;
        return;
    }else if(pos.size() != 0){
        for(int i = 1;i < pos.size();i ++){
            if(pos[i] == pos[i - 1] + 1){
                cout << "invalid" << endl;
                return;
            }
        }
        if(pos.size() == 3){
            if(pos.back() != n - 2){
                cout << "invalid" << endl;
                return;
            }
        }
    }
    LL num = 0,idx = 10;
    for(int i = 0;i < val.size() - 1;i ++){
        num += idx * val[i];
        idx --;
    }
    num %= 11;
    if(val.back() != ((11 - num) % 11)){
        cout << "invalid" << endl;
        return;
    }
    num = 38;
    for(int i = 0;i < val.size() - 1;i ++){
        if(i % 2 == 0){
            num += 3 * val[i];
        }else{
            num += val[i];
        }
    }
    val.pop_back();
    num %= 10;
    int res = (10 - num) % 10;
    s.pop_back();
    cout << "978-" << s << res << endl;
```

```
}
```

J - a * b problem

签到题，不过多赘述

K - minmax lcp

考虑从左到右确定答案的每一位。我们举个例子来介绍这一过程。

假设我们已经确定了答案的前三位是 `abc`，接下来要确定第四位。为了让字典序尽量小，我们需要从 `a` 到 `z` 枚举第四位。

此时我们能选m个字符，当我们枚举到了 `d`，我们考虑已知答案为 `abcd*` 时，与已知答案为 `abc*` 时相比，能选择的字符串数量如何变化。

- 所有 `abc[a-d]*` 都能选择，因为它们的最长公共前缀肯定小于等于 `abcd*`。
- 所有 `abc[e-z]*` 的字符串，原来答案是 `abc*` 的时候都能选择，现在每种字母只能选一个，否则比如 `abce*` 选了两个，那答案就至少是 `abce` > `abcd*` 了。
- 剩下的字符串可选情况维持不变。

假如上述过程我们枚举到 `d` 时我们可以选k个字符，因为我们当前一共能选m个字符，若$k \geq m$，那么答案的第四位就是 `d`，否则我们要继续枚举 `e`、`f`、...

确定了答案的第四位以后，我们还要确定答案为 `abcd*` 后还能在 `abcd` 的基础上选几个字符，设选 `d` 前一共能选cnt个字符，一共能选m个字符，则在 `abcd` 的基础上还能选$m - cnt$个字符。

我们可以利用字典树一路枚举每一位即可，复杂度 $O(26 * \sum |s|)$。

L - htam

从M下手，若M第k位为1，将N拆分为大于k位的部分，等于k位的部分，小于k位的部分，即$up = n >> (i + 1LL), middle = n >> i \& 1, down = n \& ((1LL << i) - 1)$此时分两种情况

- 若N的第k位也为1，即$middle = 1$

  ①那么高位有up种选择可以使得当前数一定比N小，那么低位就可以任意选，因为低位此时有$k - 1$位，所以一共有2^k个选择。

  ②此外高位可以与N相同，那么低位就只能选$down + 1$位（考虑0）

  所以一共有$up * (1LL << k) + down + 1$种选择

- 若N的第k位为0，即$middle = 0$

  此时高位不能与N相同，所以结果为上述减去高位与N相同的情况

  所以一共有$up * (1LL << k)$种选择

根据每一位枚举即可

M - Dear Alice and Bob

思路：不难发现对于$Alice$可以枚举第一条命令的跳点后$o(1)$找到到达Bob点的唯一方案，$o(logn)$查询是否存在即可。

本题难点在于如何确定一个位置Bob无法到达或者发现不存在。正常需要$o(n^2)$的枚举复杂度。这显然不达标，关注到本题只有n条命令，考虑随机数生成$Alice$的跳转坐标，对于最坏情况，通过一次找到$Alice$的概率为$\frac{n\times(n-1)/2}{n^2-2}$，约为$\frac{1}{2}$，考虑在时间允许范围内，可以查找$30$余次$o(30\times n\times logn)$，那么$30$次全部找到$Alice$的概率为$\frac{1}{2^{30}}$约为0，那么此时的情况可以近似认为就是$Bob$胜利。