A 区间异或第k大

如果能获取区间内的线性基，则求第K大是容易的，只需要用线性基贪心地选取即可。

一个容易想到的做法是线段树维护线性基，每次询问将多组线性基合并起来。但是这复杂度为$O(nlog^3)$，无法接受。

我们考虑**前缀线性基**做法，维护前缀区间$a_1 \sim a_i$的线性基和最晚出现时间。则对于询问$[l,r]$，可用的线性基是最晚出现时间大于等于l的，用可用线性基求第K大即可。这样复杂度只有$O(nlogn)$

B 最大popcount

如果$n=0$，$popcount(x)$最大是$x=popcount(x)=0$，输出0即可。

否则满足要求的数字是形如$11...11$的二进制下全1的数字，可以暴力枚举或者用数学方法计算。

C 点对统计

设点i在第一个图里属于x_i联通块，第二个图里属于y_i联通块，x_i,y_i通过并查集求得。

问题转换成有多少个点对i,j满足$i<j$、$x_i==x_j$且$y_i==y_j$，我们将(x_i,y_i)成对地存储在map等形式的桶里，统计出现次数，桶内计算C^2_{sum}即可。

D 万剑归一

首先，a_1必须为1，其他点的a_i需要构成一棵i指向a_i的以1为根的内向树，深度最多为m。

我们抛弃掉a_i，开始计算n个点的树，以1为根，树高$\leq m$的方案数。

设$f[i][j][k]$表示$1\sim i$号点，树高恰好为j，且深度为j的点恰好有k个的方案数。计算$f[i][j][k]$时，枚举上一层用了$k1$个点，则有$f[i][j][k]=\sum f[i-k][j-1][k1]*C^k_{n-i+k}*k1^k$。预处理组合数和快速幂可以做到$O(n^3m)$。

E 最小生成树

如果规定必须使用cnt条白边，可以用wqs二分求出。接下来设$ask(cnt)$表示恰好使用cnt个白边的最小花费。

由于$ask(cnt)$是斜率单调的凹函数，cnt^2也是斜率单调的凹函数，因此$ask(cnt)+cnt^2$也是斜率单调的凹函数，可以用二分求最小值。$ask(cnt)$内部不要暴力排序，而是类似归并排序地合并白边黑边，可以做到$ask(cnt)$复杂度视为$O(m+n)$，总复杂度为$O(mlog^2)$。由于$ask(cnt)$有多次重复计算，使用记忆化可以优化至$1.5s$。

F 传送带

第i条边建立传送带的时间戳是i，则路径i的花费变为0的时间是所经过的边的时间戳的最大值。

将每条边的时间戳设立为边权，则问题转变成树上路径边权最大值，可以用类似倍增lca的做法求之。

每条路径变为0的时间确定后，第i行本质上是在询问有多少条路径变为0的时间小于等于i，用桶+前缀和的思想可以解决。

G 组合数的奇偶

卢卡斯定理告诉我们，将a、b二进制分解为$a_0a_1a_2a_3a_4...$和$b_0b_1b_2b_3b_4...$，则$C_b^a = C_{b_0}^{a_0}C_{b_1}^{a_1}C_{b_2}^{a_2}...$

$\%2$意义下，C_b^a的情况只有四种：$C_0^0 = 1, C_0^1 = 0, C_1^0 = 1, C_1^1 = 1$。因此要想$C_b^a$为奇数，需要不存在$a_i = 1$，$b_i = 0$，也就是说二进制下$a$需要是$b$的子集。

问题转变成区间$[l, r]$内有多少对$a \leq b$满足a是b的子集，为了避免数位dp难写所以调小了t，可以使用数位dp或者非常好写的高维前缀和解决。

H 最小树成本

设$siz[x]$表示以x为根的子树大小，$siz[x] = 1 + \sum siz[y]$，其中y是x的儿子。

设$f[x][i]$表示x在以x为根的子树内部编号排名为i的最小成本

初步思考一下，我们疑似需要枚举儿子们y的排列顺序，花费为$f[x][i] = min(\sum(f[y][j] + c[y] * (x到y的距离))$，其中$x$到$y$的距离受制于儿子们$y$的排列顺序和$j$的情况。

现在考虑儿子们的相对位置

如果确定了a, b在x左边，则如果$a\ b\ x$地排列，多的花费为$c_a * siz_b$；如果$b\ a\ x$地排列，多的花费为$c_b * siz_a$

如果在b左边，这需要满足$c_a * siz_b <= c_b * siz_a$，可以根据这个对儿子进行排序，将$c/siz$更小的放在左边。

反之，如果确定了a, b在x右边，类似地可以证明$x\ a\ b$地排列要$c_b * siz_a <= c_a * siz_b$，发现需要将$c/siz$更大的放在左边。

因此，对儿子们按c/siz从大到小排序，就可以先安排靠近x的儿子，新枚举到的儿子会远离x，放在最左边或者最右边。这样的好处是之前按顺序靠近x安排了不少儿子，新枚举到的儿子要么放在当前块的左端，要么放在当前块的右端，收益是只和c_y、y在自己块内的位置、x之前的块的位置有关的。到这里对于随机的树已经可以通过，而链的情况会退化到n^3，还需要继续优化。

```
    sort(e[x].begin(),e[x].end(),cmp);
    for(auto y:e[x])
    {
        sizz+=siz[y];
        for(int j=sizz;j>=1;j--)
        {
            ll minn=0x3f3f3f3f3f3f3f3f;
            for(int k=1;k<=siz[y];k++)
                if(j>=siz[y])
                    minn=min(minn,min(f[x][j-siz[y]]+f[y][k]+c[y]*(j-k),f[x]
[j]+f[y][k]+c[y]*(sizz-siz[y]-j+k)));
                else
                    minn=min(minn,f[x][j]+f[y][k]+c[y]*(sizz-siz[y]-j+k));
            f[x][j]=minn;
        }
    }
```

可以发现除了要确定了哪些儿子在x左边、哪些在右边，还需要确定x的儿子们在自己的子树内部的位置，这个信息和x、j无关，可以预处理：

若x未来会放在父亲的左边，设$gl[x]=min(f[x][j]-c[x]*j)$表示$x$在子树内部不同位置的最小花费。

当x未来会放在父亲的右边，同样地设$gr[x]=min(f[x][j]+c[x]*j)$表示$x$在子树内部不同位置的最小花费。

确定了儿子们的相对位置和内部位置后，接下来只需要确定每个儿子放在x前面还是后面即可，是类似背包的过程。

代码如下：

```
int cmp(int a,int b)
{
    return c[a]*siz[b]>c[b]*siz[a];
}
void dfs(int x)
{
    siz[x]=1;
    for(auto y:e[x])
    {
        dfs(y);
        siz[x]+=siz[y];
    }
    ll sizz=1;
    f[x][1]=0;
    sort(e[x].begin(),e[x].end(),cmp);
    for(auto y:e[x])
    {
        sizz+=siz[y];
        for(int j=sizz;j>=1;j--)
        {
            if(j>=siz[y])
                f[x][j]=min(f[x][j-siz[y]]+gl[y]+c[y]*j,f[x][j]+gr[y]+c[y]*
(sizz-siz[y]-j));
            else
                f[x][j]=f[x][j]+gr[y]+c[y]*(sizz-siz[y]-j);
        }
    }
    for(int j=1;j<=sizz;j++)
    {
        gl[x]=min(gl[x],f[x][j]-c[x]*j);
        gr[x]=min(gr[x],f[x][j]+c[x]*j);
    }
}
```

I 挡雨布

将点集的**上凸包**处理出来，计算面积即可。

用set维护当前上凸包，每次插入新点时找到这个点左右临近的两个点，判断是否应该插入，或是是否应该将临近的点删去。也有对点排序后单调栈维护上凸包的做法，总而言之是计算几何模板题。

复杂度为$O(nlogn)$

J 输入距离

由于刚开始停留在a字母上，因此只需要贪心地将字符串排序，就可以保证不走回头路，不使用向左的操作。

排序后统计答案即可。

K 圆

由于$n \leq 10$，因此暴力枚举全排列判断是否合法即可，复杂度$O(n! * n^2)$。

L 编辑器

shift操作可以使用一个标记变量，这样小写字母和数字$0 \sim 9$容易解决。

为了处理其他操作，我们可以维护两个字符串a, b，其中a表示光标以前的部分，b表示光标之后的部分。则每个操作如下处理：

`ENTER`：令`a+='\n'`即可

`BACKSPACE`：如果a为空则不操作，否则删掉a字符串的最后一个元素

`DEL`：如果b为空则不操作，否则删掉b字符串的第一个元素

`LEFT`：如果a为空则不操作，否则把a字符串的末尾字符删掉，放在b的前面

`RIGHT`：如果b为空则不操作，否则把b字符串的首元素放在a的最后

`UP`：寻找a字符串的倒数第一个`\n`和倒数第二个`\n`，如果发现找不到则不做任何操作，否则执行若干次`LEFT`操作

`DOWN`：寻找a字符串的最后一个`\n`，b的第一个`\n`和b的第二个`\n`，如果找不到b的`\n`则不做任何操作，否则执行若干次`RIGHT`操作

`HOME`：寻找a字符串的倒数第一个`\n`，执行若干次`LEFT`操作

`END`：寻找b字符串的第一个`\n`，执行若干次`RIGHT`操作

M 内存溢出

分类讨论题，如果是MB则输出$x * 1000\ KB$，否则输出$x * 1024\ KiB$。