简单题

B

问题简化为寻找一条从起点到终点的路径，该路径包含的边权的最小值最大

方法有很多，这里提供一种

按照边权从大到小对边进行排序，依次加入图中，直至起点和终点联通，判断起点和终点是否联通可以使用并查集

C

现只考虑左上角（1，1），右下角为（n，n）的子矩阵a，其他位置的子矩阵可以通过复制到一个新的数组，变换后，覆盖回去进行处理

- 顺时针旋转90° 相当于对于每一个元素$a_{i,j}$变换至$a_{j,n-i+1}$
- 左右翻转 交换$a_{i,j}$和$a_{i,n-j+1}$
- 上下翻转 交换$a_{i,j}$和$a_{n-i+1,j}$
- 延左上至右下对角线 交换$a_{i,j}$和$a_{j,i}$
- 延右上至左下对角线 交换$a_{i,j}$和$a_{n-i+1,n-j+1}$

注：第一种操作是针对所有元素进行操作，后面四种针对一半元素进行操作

E

将匹配完 $x_A^{(i)}$ 和 $x_B^{(j)}$ 的状态记作 (i,j)。稍加分析，易知该问题符合动态规划的条件。

- 重复子问题：(i,j) 状态可以由 $(i-1,j)$、$(i,j-1)$、$(i-1,j-1)$ 三个状态转化过来，他们的问题是相同的。
- 无后效性：在 (i,j) 状态下，之前做的连线对后边没有影响。
- 最优子结构：(i,j) 应该由其子问题的最优解转移而来。

为了叙述方便，记录 $dp[i][j]$ 为匹配完 $x_A^{(i)}$ 和 $x_B^{(j)}$ 时，得到的最小距离之和。设距离矩阵 $\text{dist}[i][j]=\left|x_A^{(i)}-x_B^{(j)}\right|$，该问题的求解公式为

$$dp[i][j]=\text{dist}[i][j]+\min(dp[i-1][j],dp[i][j-1],dp[i][j])$$

时间复杂度：$O(n^2)$；空间复杂度：$O(n^2)$。

H

对于这道题我们有两种普通的dp思路：

- n^2做法： f_i表示以第i柱香为结尾的方案数，不难发现f_i可以通过前面递推而来。

```
for(int i = 1;i<=n;i++)
    for(int j = 1; j < i; j++)
        if(abs(a[i] - a[j]) >= k)
            f[i] = (f[j] + f[i]) % mod;
```

- $n \times a_{imax}$做法: 加入s数组，s_i表示以i高度为结尾的方案数。

```
for (int i = 1; i <= n; i++){
    ll sum = 0;
    for(int j = 0; j <= a[i] - k; j++)
        sum = (sum + s[j]) % mod;
    for(int j  = a[i] + k; j <= max_ai; j++)
        sum = (sum + s[j]) % mod;
    s[a[i]] = (s[a[i]] + sum) % mod;
}
```

显然这两种做法都无法在给定时限内通过题目，考虑优化：

- nk 做法：题面中给定了k的范围，不妨直接计算 $\sum\limits_{a_i-k<j<a_i+k} s_j$，在转移时将其减去，就可以将复杂度降为 nk
- 树状数组优化：可以发现在$n \times a_{imax}$做法中出现的单点修改区间求和可以使用树状数组优化，将复杂度降为 $n \log a_{imax}$

I

根号暴力分解每个数字，判断其质因子是否在数列中（开一个数组 `st`，`st[a]==1` 表示a在数组中，可以 $O(1)$ 判断）

J

记 $p_{i,j} = \frac{a_{i,j}}{100}$

先赢R7，概率 $p_{5,6}$

G3的对手是GG的概率为 $p_{7,8}$，BLG获胜概率 $p_{5,7}$

G3的对手是GAM的概率为 $p_{8,7}$，BLG获胜概率 $p_{5,8}$

所以答案为 $p_{5,6} * (p_{7,8} * p_{5,7} + p_{8,7} * p_{5,8})$

M

定位为简单模拟题

不难发现每个数字都由

* * * * *

或

...

或

....*

或

*....

这四个选取组合而成，因此只需要找到每个数字的排列方式就行

分别用数字1，2，3，4来表示 上面四行，如下：

```
map<string, int> row_to_num;
row_to_num["*****"] = 1;
row_to_num["*...*"] = 2;
row_to_num["....*"] = 3;
row_to_num["*...."] = 4;
```

那么每个数字都可以用一个五位数来表示，如下：

```
map<int, int> num_to_res;
num_to_res[12221] = 0;
num_to_res[33333] = 1;
num_to_res[13141] = 2;
num_to_res[13131] = 3;
num_to_res[22133] = 4;
num_to_res[14131] = 5;
num_to_res[14121] = 6;
num_to_res[13333] = 7;
num_to_res[12121] = 8;
num_to_res[12131] = 9;
```

接下来处理输入就行了，可以一行一行看，每一行都输入 了三个字符串，分别记录

```
int res1 = 0, res2 = 0, res3 = 0;
for(int i = 1; i <= 5; ++i) {
    string str1, str2, str3;
    cin >> str1 >> str2 >> str3;
    res1 = res1 * 10 + row_to_num[str1];
    res2 = res2 * 10 + row_to_num[str2];
    res3 = res3 * 10 + row_to_num[str3];
}
```

最后对于res1，res2，res3用num_to_res输出结果即可！

中档题

A

1.异或和的计数问题，考虑数位DP

2.按照二进制从高往低位考虑，设$f[d][res]$为m元组中的每个数已经确定了第$mx, mx-1, \ldots, d+2, d+1$位（$mx$为最高位），$n$减去$m$个数已经确定的位的和的总和后，还剩下$res$（也就是$m$个数剩下的低位和的总和），在这样的条件下，后面低位的所有填数方案的异或和的总和

3.转移考虑当前第d位（位权位2^d），肯定是选取m个数的某个子集填1，剩下的为0；那么枚举m个数中填1的数的个数，就可以用后面的dp值转移过来，具体地，枚举有i个数填1：

$$f[d][res] += C_m^i\{f[d-1][res-2^di]+[i\text{为奇数}]2^dg[d-1][res-2^di]\}$$

其中g表示在相应f对应的条件下的填数方案数，用于辅助f的转移（计算当前位的贡献）

4.以上是经典的计数DP方式，现在考虑其效率：

(a).如果直接枚举每个状态，状态数$O(nlogn)$，每次转移$O(m)$，效率是$O(mnlogn)$

(b).发现第二维状态为d时，第一维只能是$n-k\times 2^{d+1}$（因为前面减去的数都是2^{d+1}的倍数），所以第一维的状态数是$\frac{n}{2^{d+1}}$，这样枚举d从1到mx，总和是$\frac{n}{2^1}+\frac{n}{2^2}+\ldots+\frac{n}{2^{mx}}=O(n)$，效率是$O(mn)$

(c).考虑还有哪些状态是没有用的：因为题目的数据范围中，n很大而m很小，所以在d枚举到较低位而res较大时，即使后面每一位都是m个数全填1，也无法填到res，这样的状态是无效的

(d).具体来说，第二维状态为d时，res的上界为$m\times 2^{d+1}$(即后面每一位都是m个数全填1，比这个大的状态dp值必然都是0)，而由(b)中观察到的性质，每间隔2^{d+1}才有一个状态有效，所以有效的状态数是$O(m)$的，那么总的状态数为$O(mlogn)$,总复杂度为$O(m^2logn)$
实现可以直接比较暴力地用$(unordered)map$记录状态（常数较大），也可以把状态除掉2^{d+1}的间隔，每个d就只要开$O(m)$的数组记录了。

F

对每一个下标$i\in[0,n)$，考虑这个下标是ODT中某一结点对应区间的左端点的概率。每次执行操作时，如果i不是左端点，那么有$p_0=\frac{n}{\binom{n+1}{2}}$的概率变成左端点；如果$i$是左端点，那么有$p_1=\frac{i(n-i)}{\binom{n+1}{2}}$的概率变成不是左端点。无穷多次操作之后，设$i$是左端点的占比是$x$，那么有方程$(1-p_1)x+p_0(1-x)=x$，解得$x=\frac{n}{n+ni-i^2}$。一个左端点对应ODT内一个结点，由期望的线性性，最终答案就是所有下标的占比之和，即

$$ans_n=\sum_{i=0}^{n-1}\frac{n}{n+ni-i^2}$$

这样就能$O(n)$处理单次询问了。

观察ans_n和$2\ln n$的差值。当n逐渐增大，该差值趋于一个常数。当$n=10^9$时，该差值约为0.154431。在$n\geq 50000$时，直接输出$2\ln n+0.154431$即可保证绝对误差小于10^{-3}。

事实上，

$$ans_n=\sqrt{\frac{n}{n+4}}(2\psi(\frac{\sqrt{n(n+4)}+n}{2})-2\psi(\frac{\sqrt{n(n+4)}-n}{2})-1)=2\ln n+2\gamma-1+o(1)$$

其中ψ是digamma函数，γ是Euler–Mascheroni常数。这里不做推导，有兴趣可以自行研究。

G

计算几何+交互

题意

在平面上确定一个圆的圆心，可以询问一个点是否在圆内，最多询问三百次。

题解

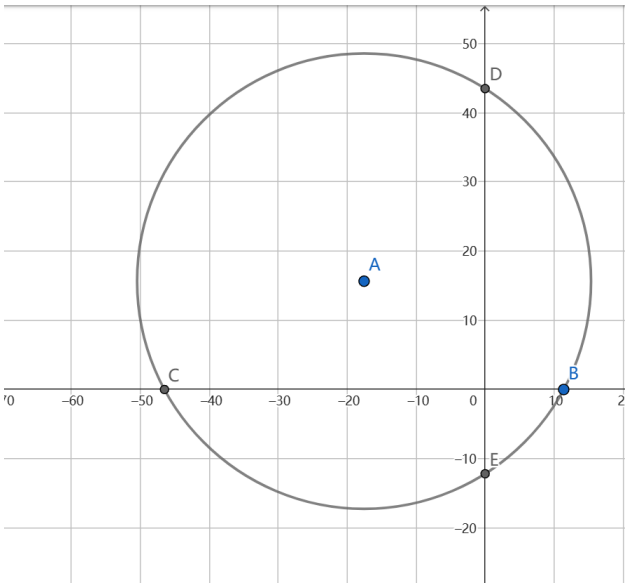
考虑是否能找到圆边界上的点，可以用二分的方法，让边界上点的范围不断缩小。

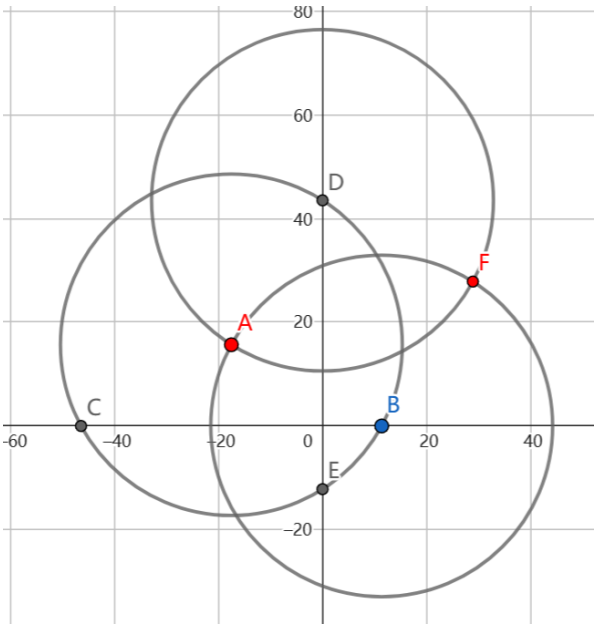
如下图在x正半轴一定有B点在圆边界上，可以二分横坐标，不断询问正半轴上的点确定B的位置 。

1.如果确定了三个边界上的点可以三点确定圆心位置。

2.确定B，C，D，E四点圆心横坐标是B，C两点的横坐标，纵坐标是D，E两点的纵坐标。

3.也可以只确定边界上两个点，通过r算出圆心可能的两个坐标，再通过一次询问确定。





K

字典树+BFS

题意

问两组字符串，求是否能分别从第一组和第二组选出任意个拼接，使两组拼成的字符串相同，并求最短的相同字符串长度。

题解

如果我们把一组字符串建成一颗字典树，那么能拼接成的字符串就是从字典树的根不断走到某个字符串结束节点再回到根节点的过程。

把小H和小W的两组字符串分别建两颗字典树，在两颗字典树上同时从根节点走，记录d(i,j) 为第一棵树走到i节点，第二棵树走到j节点的最短距离，如果能同时走回根节点则游戏成功，走回根节点的最短距离为答案。

d(i,j)可以由BFS得到，可以证明每个d(i,j)最多被搜索到一次，时间复杂度为O(n*n),n为字典树的节点个数，字典树的节点个数最多5000个，可以通过本题。

L

观察：对于一个美丽的子序列，其最大的数的值只可能为（这个序列的）MEX+1或MEX-1

设$dp_{i,j,0/1}$为只考虑前i个数，MEX为j，序列最大值为j+1/j-1的美丽的子序列的数目，初始$dp_{0,0,0}=1$

假设已经求出了dp_{i-1}，考虑怎么求dp_i：

不以a_i结尾的子序列，数目为$dp_{i-1,j,k}$；

以a_i结尾的子序列，讨论前面的数的MEX：

小于a_i-1，不符合条件；

等于a_i-1，把a_i接上去之后MEX不变，序列最大值变为序列的MEX+1，即$dp_{i,a[i]-1,1}+=dp_{i-1,a[i]-1,0/1}$；

等于a_i，把a_i接上去之后MEX增加1或2（取决于原本序列的最大值），即$dp_{i,a[i]+1,0}+=dp_{i-1,a[i],0}$，$dp_{i,a[i]+2,0}+=dp_{i-1,a[i],1}$；

大于a_i，把a_i接上去之后对原序列的状态无影响，即$dp_{i,j,0/1}+=dp_{i-1,j,0/1},j>a_i$；

第一维可以滚动，剩下两维可以用区间乘单点加线段树维护。

注意这样算出来的答案是包含了空串的，最后输出的时候要减去1。

O

计算几何

题意

求过边界过原点的一圆能覆盖的最大点权值和。

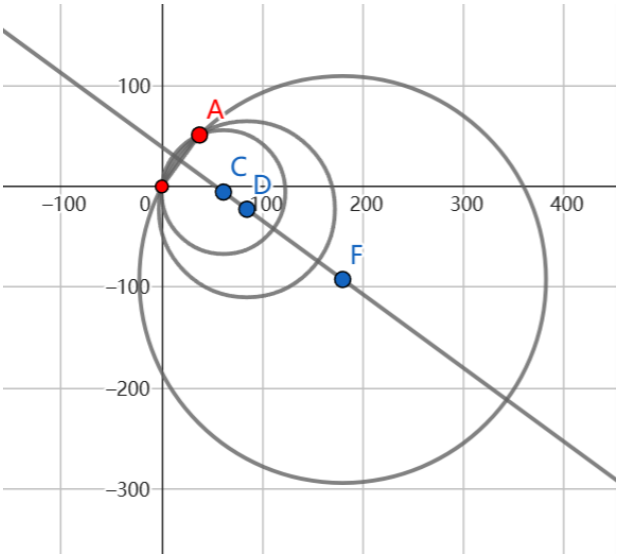
题解1

一种可能的最优解，原点和某一个其他点一定在圆边界上。

考虑枚举这个点，所有过这点与原点的圆，圆心都在一条线上。这条线是原点和枚举点的中垂线。

考虑能覆盖某个点的圆的圆心位置，在中垂线上是一个区间。

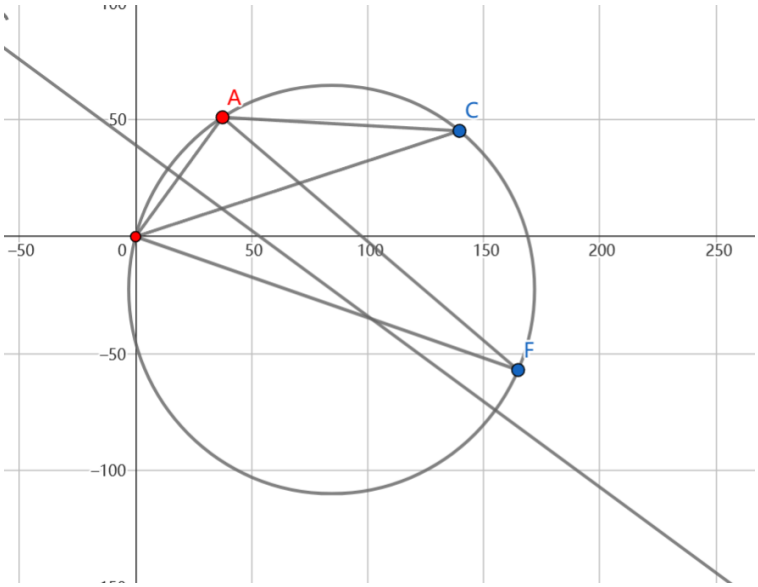
可以对所有区间排序之后用扫描线解决。



Note

四点共圆问题，如果用浮点数坐标距离<eps=1e-8判断。

也可用圆上弦所对应的圆心角判断。圆心角可以用余弦表示，余弦可以用点积除以模长。平方后比较可以完全消除浮点误差。



题解2

考虑对原点的圆心圆反演。位置关系不变的情况下，把一个圆变成一条直线。

转换为选择一条线，让线的一侧的点权和最大。

枚举一点，其他点对这个点极角排序。也可以用扫描线解决。

难题

D

我们先跑一遍双指针，对于每个点 i，求 f_i 为以 i 为右端点而使得区间和大于等于 v 的最靠后左端点，同理也可求得以 i 左端点的对应的 g_i。

对于确定的区间 $[L,R]$，我们有不满意度的计算方式：

$$\sum_{i=L}^{R}\frac{[i-\max(l-1,f_i)]^k-[i\neq f_i][i-1-\max(l-1,f_i)]^k}{(R-L+1)^k}$$

只考虑 $R\geq g_L$ 的区间。进行简化，有

$$\frac{(g_L-L)^k+\sum_{i=g_L}^{R}(i-f_i)^k-[i\neq f_i](i-1-f_i)^k}{(R-L+1)^k}$$

记作

$$\frac{(g_L-L)^k+\sum_{i=g_L}^{R}v_i}{(R-L+1)^k}$$

我们尝试对这个东西进行分数规划，也就是求最小的 λ 使得存在 $F(L,R)\leq\lambda$，即

$$(g_L-L)^k+\sum_{i=g_L}^{R}v_i-\lambda(R-L+1)^k\leq 0$$

倒着枚举 L，过程中维护所有在 g_L 及之后的，可能使得式子取得最小值的 R。两个 R 的价值的相对大小关系的改变仅涉及最后一项 $-\lambda(R-L+1)^k$，可知随着 L 向前移动，靠后的 R 的值相对前面的会越来越小。

所以我们维护一个（从左往右）值递增的序列，在过程中，对序列中每个点维护他踢掉他左边紧挨着的点的时间。L 每移动一次，针对 g_L 的变化加入新值，然后再一顿踢，随后队首对应值即为当前最小。

N

引理：在模$p=931135487$意义下，记$\mathbf{G}=\{f(G)\mid f(x)$取遍所有多项式$\}$，则$(\mathbf{G},+,\times)$构成一域，阶为$p^2$.

证明：由Cayley–Hamilton定理，$G^2+4G+7I=\mathbf{0}$，所以所有多项式$f(x)$都可以模上一个x^2+4x+7变成一次函数，因此$\mathbf{G}$内所有矩阵具有形式

$$f(G)=aI+bG$$

可以证明(a,b)和$aI+bG$一一对应，即$|\mathbf{G}|=p^2$。$\mathbf{G}$上的加法、乘法的交换律、结合律等性质显然，这里只证明乘法逆元。考虑矩阵$P=p_0I+p_1G$并假设它的逆$Q=q_0I+q_1G$，由待定系数法得

$$\begin{pmatrix}q_0\\q_1\end{pmatrix}=\frac{1}{p_0^2-4p_0p_1+7p_1^2}\begin{pmatrix}p_0-4p_1\\-p_1\end{pmatrix}$$

注意到分母$p_0^2-4p_0p_1+7p_1^2=|P|$，所以$P^{-1}\in\mathbf{G}$当且仅当P可逆。尝试令$|P|=0$，解得$7p_1=(2\pm\sqrt{-3})p_0$，因为-3不是模p下的二次剩余，所以该方程只有唯一解$(p_0,p_1)=(0,0)$，因此$\mathbf{G}$中非$\mathbf{0}$矩阵均存在乘法逆元。证毕。

可以证明有限域内的乘法群是循环群，所以存在一个矩阵使其在乘法上的阶为$|\mathbf{G}|-1=2^{24}\times3\times37\times465567743$。可以验证$G$的阶就是$|\mathbf{G}|-1$，也可以用类似于找原根的方法找到其他满足条件的矩阵。剩下的事情就和普通的多项式求逆一样了，在$\mathbf{G}$上构造多项式并用NTT计算卷积即可。有一个优化是用(a,b)表示$aI+bG$，计算矩阵相乘时只需要4次乘法。