I - 寄蒜几盒 X

题目描述

小 A 想给小 B 寄几盒蒜。

小 A 还没有落实行动，因为他还没有想好究竟给小 B 寄多少盒蒜。

小 B 告诉小 A：「我会给你一道答案是整数的题目，当你算出答案之后，就以答案为盒数给我寄蒜吧！」

小 A 欣然同意，但是小 A 很快发现这道题他并不会做。小 B 给出的题目是这样的：

小 B 在 $\mathbb{R}^2$ 平面上标记了 n 个整点 $P_1, P_2, \dots, P_n$，第 i 个点的坐标为 (x_i, y_i)，任意三点不共线。

小 B 会等概率选取点集 $S = \{P_1, P_2, \dots, P_n\}$ 的一个子集 $S' \subseteq S$。记 S' 的凸包为 $C(S')$，并记该凸包的面积为 $\mathcal{L}(C(S'))$，求 $\mathcal{L}(C(S'))$ 的期望对 $998\,244\,353$ 取模的结果。特殊地，当 $S' = \emptyset$ 时，$C(S') = \emptyset$，$\mathcal{L}(C(S')) = 0$。

形式化地说，小 A 需要求出 $\mathrm{E}[\mathcal{L}(C(S'))] = \frac{1}{2^n} \sum_{S' \subseteq \{P_1, P_2, \dots, P_n\}} \mathcal{L}(C(S'))$ 对 $998\,244\,353$ 取模的结果。

小 A 自然把这个问题交给了你。

对任一凸多边形$\{P_0, P_1, P_2, \dots, P_{m-1}\}$（顶点按逆时针排序）

其面积为$\frac{1}{2}\Sigma_{i=0}^{m-1} OP_i \times OP_{(i+1)\%m}$

可以先约定每个凸包的点都按逆时针方向排序

可以考虑每个有序点对P, Q对总期望的贡献为$OP \times OQ * \frac{\text{以}\overrightarrow{PQ}\text{作为凸包中的一条边的方案数}}{2^n}$

由于之前规定了方向，

$$\overrightarrow{PQ}\text{作为凸包中的一条边的方案数} = 2^{\text{在}\overrightarrow{PQ}\text{左侧的点数}} - 1$$

```
ll pts = 0;
for (int k = 0; k < n; ++k) {
    if (k == i || k == j) continue;
    if (sgn(cross(x[k] - x[i], y[k] - y[i], x[j] - x[i], y[j] - y[i])) == -1) {
        ++pts;
    }
}

ll den = 1ll << (n + 1);
tot += (((cross(x[i], y[i], x[j], y[j]) % mod + mod) % mod) * modInverse(den,
mod) % mod) * (((1ll << pts) - 1) % mod) % mod;
tot %= mod;
```

计算过程中需要注意取模，由于就998244353是质数，2^k的模可以用$(2^k)^{m-2}$来计算，使用快速幂即可。

时间复杂度为$O(n^3)$

代码：

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

```

```
int n;
ll x[50], y[50];
ll mod = 998244353;

ll cross(ll x1, ll y1, ll x2, ll y2) {
    return x1 * y2 - x2 * y1;
}

ll sgn(ll x) {
    if (x < 0) return -1;
    if (x > 0) return 1;
    return 0;
}

ll binpow(ll x, ll b) {
    ll ret = 1;
    x %= mod;
    while (b) {
        if (b & 1) {
            ret = ret * x % mod;
        }
        x = x * x % mod;
        b >>= 1;
    }
    return ret;
}

ll modInverse(ll a, ll m) {
    return binpow(a, m - 2);
}

int main() {
    cin.tie(NULL);
    ios::sync_with_stdio(false);

    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> x[i] >> y[i];
    }

    ll tot = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i == j) continue;

            ll pts = 0;
            for (int k = 0; k < n; ++k) {
                if (k == i || k == j) continue;
                if (sgn(cross(x[k] - x[i], y[k] - y[i], x[j] - x[i], y[j] -
y[i])) == -1) {
                    ++pts;
                }
            }

            ll den = 1ll << (n + 1);
```

```
            tot += (((cross(x[i], y[i], x[j], y[j]) % mod + mod) % mod) *
modInverse(den, mod) % mod) * (((1ll << pts) - 1) % mod) % mod;
            tot %= mod;
        }
    }

    cout << tot << endl;
    return 0;
}
```