bnu弱校联萌10.3J +2017百度之星复赛1002

```
6
1-2+3
(1-2+3)#(3)
(1)#(3)
(1+)#(2)1
(2*3+1)#(2)
(2)#(2)1+1(2)#(2)
```

表达式求值矩阵 (a)#(b)表示a^b 也可以直接求

```
int main()
{
    e.E();
    mule.v[0][0] = mule.v[1][2] = mule.v[3][3] = 1, mule.v[1][0] = Z - 1;
    adde.v[0][0] = adde.v[3][2] = adde.v[3][3] = 1;
    sube.v[0][0] = sube.v[3][3] = 1, sube.v[3][2] = Z - 1;
    nume.v[0][0] = nume.v[2][2] = nume.v[3][3] = 1, nume.v[1][0] = 9, nume.v[1]
[1] = 10;
    int n;
    while(~ scanf("%d",&n)){
        matrix ans = e;
        while(n --){
            int m;
            scanf("%d%s", &m, s);
            matrix tmp = e;
            for(int j = 0; s[j]; j ++){
                if(s[j] == '*') tmp = tmp * mule;
                else if(s[j] == '+') tmp = tmp * adde;
                else if(s[j] == '-') tmp = tmp * sube;
                else{
                    nume.v[2][0] = nume.v[2][1] = s[j] - '0';
                    tmp = tmp * nume;
                }
            }
            ans = ans * (tmp ^ m);
        }
        printf("%d\n", (ans.v[2][0] + ans.v[3][0]) % Z);
    }
    return 0;
}
```

百度之星复赛1003

众所周知，度度熊最近沉迷于 Pokémon GO。

今天它决定要抓住所有的精灵球!

为了不让度度熊失望，精灵球已经被事先放置在一个2*N的格子上，每一个格子上都有一个精灵球。度度熊可以选择任意一个格子开始游戏，抓捕格子上的精灵球，然后移动到一个相邻的至少有一个公共点的格子上继续抓捕。例如，(2, 2) 的相邻格子有(1, 1), (2, 1) 和 (1, 2) 等等。

现在度度熊希望知道将所有精灵球都抓到并且步数最少的方案数目。两个方案被认为是不同，当且仅当两个方案至少有一步所在的格子是不同的。

```
f[1] = 2; f[2] = 24; f[3] = 96; f[4] = 416;
f[5] = 1536; f[6] = 5504; f[7] = 18944; f[8] = 64000;
f[9] = 212992; f[10] = 702464; f[11] = 2301952; f[12] = 7512064;
for(int i = 13; i <= 10000; i ++) {
    f[i] = f[i - 1] * 6 - f[i - 2] * 8 - f[i - 3] * 8 + f[i - 4] * 16;
    f[i] = (f[i] % Z + Z) % Z;
}
```

从n(3<=n<=200)个点中选择k(3<=k<=min(n,50))个点，使得这k个点所组成的凸包内不包含其他所有点，且面积最大，不存在方案则输出0。保证不存在三点共线的情况，输出保留两位小数。

```
#include<stdio.h>
#include<memory.h>
#include<map>
#include<cmath>
#include<stdlib.h>
#include<algorithm>
#include<set>
#include<vector>
#include<queue>

using namespace std;

typedef pair<int,int> pii;
typedef long long ll;

const int MX = 100005;
const int MM = 1000000007;

pii operator-(const pii &l, const pii &r){
	return pii(l.first - r.first, l.second - r.second);
}

ll operator/(const pii &l, const pii &r){
	return (ll)l.first * r.second - (ll)l.second * r.first;
}

pii D[MX];

int sign(pii x){ return x > pii(0, 0)? 0 : 1; }

auto cmp = [](pii l, pii r){
	return sign(l) != sign(r) ? sign(l) < sign(r) : l/r < 0;
};

struct EV{
	pii d;
	int a, b; // a -> b;
	EV(pii d, int a, int b):d(d), a(a), b(b){}
	bool operator<(const EV &l)const{
		return cmp(d, l.d);
	}
};

int N, K;
ll solve(vector<pii> &L){
	sort(L.begin()+1, L.end(), cmp);
	int N = L.size();

	vector<EV> E;
	E.reserve(N*N);
	for(int i = 1; i < N; i++){
		E.emplace_back(L[i] - L[0], 0, i);
		E.emplace_back(L[0] - L[i], i, 0);
	}
	for(int i = 1; i < N; i++){
		for(int j = i+1; j < N; j++){
			int ch = 1;
			for(int k = i+1; k < j; k++){
				if( (L[j] - L[i]) / (L[k] - L[i]) < 0 ){
					ch = 0;
```

```
                                    break;
                              }
                        }
                        if( !ch ) continue;
                        E.emplace_back(L[j] - L[i], i, j);
                        E.emplace_back(L[i] - L[j], j, i);
                  }
            }
            sort(E.begin(), E.end());
            static ll area[205][55] = {};
            memset(area, -1, sizeof(area));

            area[0][0] = 0;
            static int cnt = 0;
//          printf("%d\n", ++cnt);
//          for(pii c : L) printf("%d,%d\n", c.first, c.second);
            for(EV e : E){
//                printf("%d -> %d, (%d,%d)\n", e.a, e.b, e.d.first, e.d.second);
                  int a = e.a, b = e.b;
                  for(int i = 0; i < K; i++){
                        if( area[a][i] == -1 ) continue;
                        area[b][i+1] = max(area[b][i+1], area[a][i] + L[b] / L[a]);
                  }
            }
            return area[0][K];
}

int main()
{
            scanf("%d%d", &N, &K);
            for(int i = 1; i <= N; i++){
                  scanf("%d%d", &D[i].first, &D[i].second);
            }
            sort(D+1, D+N+1);
            ll ans = 0;
            for(int i = 1; i <= N; i++){
                  vector<pii> L;
                  L.push_back(pii(0, 0));
                  for(int j = i+1; j <= N; j++) L.push_back(D[j] - D[i]);
                  ans = max(ans, solve(L));
            }
            if( ans%2 == 0 ) printf("%lld.00\n", ans/2);
            else printf("%lld.50\n", ans/2);
}
```

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <queue>
#include <ctime>
#include <cassert>
#include <complex>
#include <string>
#include <cstring>
using namespace std;
```

```
#ifdef LOCAL
	#define eprintf(...) fprintf(stderr, __VA_ARGS__)
#else
	#define eprintf(...) 42
#endif

typedef long long ll;
typedef pair<int, int> pii;
#define mp make_pair

struct Point {
	ll x, y;

	Point() : x(), y() {}
	Point(ll _x, ll _y) : x(_x), y(_y) {}

	void scan() {
		scanf("%lld%lld", &x, &y);
	}
	void print() {
		printf("(%lld %lld)\n", x, y);
	}
	Point operator + (const Point &a) const {
		return Point(x + a.x, y + a.y);
	}
	Point operator - (const Point &a) const {
		return Point(x - a.x, y - a.y);
	}
	ll operator % (const Point &a) const {
		return x * a.x + y * a.y;
	}
	ll operator * (const Point &a) const {
		return x * a.y - y * a.x;
	}

	bool operator < (const Point &a) const {
		return *this * a > 0;
	}
};

bool cmp(const Point &a, const Point &b) {
	if (a.x != b.x) return a.x < b.x;
	return a.y < b.y;
}

const int N = 202;
const int K = 50;
ll ANS = 0;
int k;
Point a[N];
Point z[N];
ll dp[N][N][K];
ll inner[K];
bool good[N][N];
pair<Point, int> b[N];

ll solve(int n) {
	sort(a, a + n);
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			for (int h = 0; h <= k; h++)
				dp[i][j][h] = -1;
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			good[i][j] = false;
```

```
	for (int i = 0; i < n - 1; i++) {
		good[i][i + 1] = true;
		Point P = a[i + 1];
		for (int j = i + 2; j < n; j++) {
			Point Q = a[j];
			if ((Q - a[i]) * (P - a[i]) > 0) continue;
			P = Q;
			good[i][j] = true;
		}
	}
	for (int i = 0; i < n; i++)
		for (int j = i + 1; j < n; j++)
			if (good[i][j])
				dp[i][j][0] = a[i] * a[j];
	for (int i = 1; i < n - 1; i++) {
		int m = 0;
		for (int j = 0; j < i; j++) {
			if (good[j][i])
				b[m++] = mp(a[i] - a[j], j);
		}
		for (int j = i + 1; j < n; j++)
			if (good[i][j])
				b[m++] = mp(a[j] - a[i], j);
		sort(b, b + m);
		for (int j = 0; j <= k; j++)
			inner[j] = -1;
		for (int j = 0; j < m; j++) {
			int id = b[j].second;
			if (id < i) {
				for (int h = 0; h <= k; h++)
					inner[h] = max(inner[h], dp[id][i][h]);
			} else {
				ll S = a[i] * a[id];
				for (int h = 0; h < k; h++)
					if (inner[h] != -1)
						dp[i][id][h + 1] = max(dp[i][id][h + 1],
inner[h] + S);
			}
		}
	}
	ll ans = 0;
	for (int i = 0; i < n; i++)
		for (int j = i + 1; j < n; j++)
			ans = max(ans, dp[i][j][k]);
	return ans;
}

int main()
{
//	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);

	int n;
	scanf("%d%d", &n, &k);
	k -= 3;
	for (int i = 0; i < n; i++)
		z[i].scan();
	sort(z, z + n, cmp);
	for (int i = 0; i < n; i++) {
		int m = 0;
		for (int j = i + 1; j < n; j++)
			a[m++] = z[j] - z[i];
		ANS = max(ANS, solve(m));
	}
	printf("%lld.", ANS / 2);
	if (ANS & 1)
```

```
            printf("50\n");
      else
            printf("00\n");

      return 0;
}
```

多边形加权面积并

```
typedef double flt;
const flt inf = 1e18;
inline flt  sqr(flt x) {return x * x;}
inline int sgn(flt x) {return x < -eps ? -1 : (x > eps);}
inline flt fix(flt x) {return sgn(x) == 0 ? 0 : x;}

struct point {
    flt x, y;
    point(flt x = 0, flt y = 0): x(x), y(y) {}
    bool operator < (const point &rhs) const {
        return sgn(x - rhs.x) < 0 || (sgn(x - rhs.x) == 0 && sgn(y - rhs.y) <
0);
    }
    bool operator == (const point &rhs) const {
        return sgn(x - rhs.x) == 0 && sgn(y - rhs.y) == 0;
    }
    point operator + (const point &rhs) const {
        return point(x + rhs.x, y + rhs.y);
    }
    point operator - (const point &rhs) const {
        return point(x - rhs.x, y - rhs.y);
    }
    point operator * (const flt k) const {
        return point(x * k, y * k);
    }
    point operator / (const flt k) const {
        return point(x / k, y / k);
    }
    flt dot(const point &rhs) const {
        return x * rhs.x + y * rhs.y;
    }
    flt det(const point &rhs) const {
        return x * rhs.y - y * rhs.x;
    }
    flt norm2() const {
        return x * x + y * y;
    }
    flt norm() const {
        return hypot(x, y);
    }
};

typedef std::vector<point> poly_t;
poly_t polygon[N];

std::vector<point> convex_hull(std::vector<point> u) {
    std::sort(u.begin(), u.end());
    u.erase(std::unique(u.begin(), u.end()), u.end());
    if (u.size() < 3u) return u;
    std::vector<point> ret;
    for (size_t i = 0, o = 1, m = 1; ~i; i += o) {
        while (ret.size() > m) {
            point A = ret.back() - ret[ret.size() - 2];
            point B = ret.back() - u[i];
            if (sgn(A.det(B)) < 0) break;
            ret.pop_back();
```

```
        }
        ret.push_back(u[i]);
        if (i + 1 == u.size()) m = ret.size(), o = -1;
    }
    ret.pop_back();
    return ret;
}

inline flt rati(const point &A, const point &B, const point &O) {
    if (sgn(A.x - B.x) == 0) return (O.y - A.y) / (B.y - A.y);
    else return (O.x - A.x) / (B.x - A.x);
}

flt polygon_union(poly_t poly[], int n) {
    flt ret = 0;
    for (int i = 0; i < n; ++i) {
        for (size_t v = 0; v < poly[i].size(); ++v) {
            point A = poly[i][v], B = poly[i][(v + 1) % poly[i].size()];
            std::vector<std::pair<flt, int> > segs;
            segs.push_back(std::make_pair(0.0, 0));
            segs.push_back(std::make_pair(1.0, 0));
            for (int j = 0; j < n; ++j) if (i != j) {
                for (size_t u = 0; u < poly[j].size(); ++u) {
                    point C = poly[j][u], D = poly[j][(u + 1) % poly[j].size()];
                    int sc = sgn((B - A).det(C - A)), sd = sgn((B - A).det(D -
A));
                    if (sc == 0 && sd == 0) {
                        if (sgn((B - A).dot(D - C)) > 0 && i > j) {
                            segs.push_back(std::make_pair(rati(A, B, C), +1));
                            segs.push_back(std::make_pair(rati(A, B, D), -1));
                        }
                    } else {
                        flt sa = (D - C).det(A - C), sb = (D - C).det(B - C);
                        if (sc >= 0 && sd < 0)
segs.push_back(std::make_pair(sa / (sa - sb), 1));
                        else if (sc < 0 && sd >= 0)
segs.push_back(std::make_pair(sa / (sa - sb), -1));
                    }
                }
            }
            std::sort(segs.begin(), segs.end());
            flt pre = std::min(std::max(segs[0].first, 0.0), 1.0), now, sum = 0;
            int cnt = segs[0].second;
            for (size_t j = 1; j < segs.size(); ++j) {
                now = std::min(std::max(segs[j].first, 0.0), 1.0);
                if (!cnt) sum += now - pre;
                cnt += segs[j].second;
                pre = now;
            }
            ret += A.det(B) * sum;
        }
    }
    return ret / 2;
}

int n;
flt h, f;

void run() {
    flt x, y, th, ret = 0, sum = 0;
    for (int i = 0; i < n; ++i) {
        flt l, r;
        point rect[4];
        for(int j = 0; j < 4; j ++) scanf("%lf%lf", &rect[j].x, &rect[j].y);
        polygon[i].clear();
        for (int j = 0; j < 4; ++j) {
```

```
            polygon[i].push_back(rect[j]);
        }
        polygon[i] = convex_hull(polygon[i]);
        sum += (rect[1] - rect[0]).norm() * (rect[1] - rect[2]).norm();
    }
    ret = polygon_union(polygon, n);
    printf("%.10f\n", sum / ret);
}


int main()
{
    while(~ scanf("%d", &n)){
        run();
    }
    return 0;
}
```

八面体魔方

```
int a[80];
void rotateL(int b[])
{
    int tmp[9];
    tmp[0] = b[4]; tmp[1] = b[6]; tmp[2] = b[5];
    tmp[3] = b[1]; tmp[4] = b[8]; tmp[5] = b[7];
    tmp[6] = b[3]; tmp[7] = b[2]; tmp[8] = b[0];
    int d[9];
    for(int i = 0; i < 9; i ++) d[i] = a[tmp[i]]; // 原来位置里的数
    for(int i = 0; i < 9; i ++) a[b[i]] = d[i];
}
void moveL(int b[])
{
    int tmp[18];
    for(int i = 0; i < 6; i ++) tmp[i] = b[i + 12];
    for(int i = 6; i < 18; i ++) tmp[i] = b[i - 6];
    int d[18];
    for(int i = 0; i < 18; i ++) d[i] = a[tmp[i]];
    for(int i = 0; i < 18; i ++) a[b[i]] = d[i];
}
void R(int o, int typ)
{
    int b[4][18] = {{37, 39, 38, 42, 41, 9, 14, 15, 16, 17, 18, 23, 63, 62, 58,
57, 55, 64},
        {5, 45, 44, 40, 39, 37, 46, 55, 57, 56, 60, 59, 27, 32, 33, 34, 35, 36},
        {68, 36, 35, 31, 30, 28, 19, 10, 12, 11, 15, 14, 54, 41, 42, 43, 44,
45},
        {5, 6, 7, 8, 9, 14, 54, 53, 49, 48, 46, 55, 64, 66, 65, 69, 68, 36}};
    if(typ) for(int i = 0; i < 9; i ++) swap(b[o][i], b[o][17 - i]);
    int d[4] = {46, 64, 1, 37};
    int c[9]; for(int i = 0; i < 9; i ++) c[i] = i + d[o];
    rotateL(c); if(typ) rotateL(c); moveL(b[o]);
}
void ML(int o, int typ)
{
    int b[4][18] = {{2, 6, 7, 43, 42, 38, 49, 48, 47, 58, 62, 61, 25, 26, 22,
29, 30, 31},
        {40, 44, 43, 7, 8, 4, 11, 12, 13, 20, 24, 25, 61, 60, 56, 67, 66, 65},
        {2, 3, 4, 11, 15, 16, 52, 51, 47, 58, 57, 56, 67, 71, 70, 34, 35, 31},
        {16, 17, 13, 20, 21, 22, 29, 33, 34, 70, 69, 65, 40, 39, 38, 49, 53,
52}};
    if(typ) for(int i = 0; i < 9; i ++) swap(b[o][i], b[o][17 - i]);
```

```
    moveL(b[o]);
}
bool check()
{
    for(int i = 0; i < 8; i ++){
        for(int j = 2; j <= 9; j ++){
            if(a[i * 9 + j] != a[i * 9 + 1]) return 0;
        }
    }
    return 1;
}
void go()
{
    for(int i = 1; i <= 9; i ++) swap(a[i], a[i + 54]);
    for(int i = 64; i <= 72; i ++) swap(a[i], a[i - 36]);
    for(int i = 37; i <= 45; i ++) swap(a[i], a[i - 18]);
    for(int i = 46; i <= 54; i ++) swap(a[i], a[i - 36]);
}
void dfs(int step)
{
    if(FLAG) return;
    if(check()){FLAG = 1; return;}
    if(step == 3) return;
    for(int i = 0; i < 4; i ++){
        R(i, 0); dfs(step + 1); R(i, 1);
        R(i, 1); dfs(step + 1); R(i, 0);
    }
    go();
    for(int i = 0; i < 4; i ++){
        R(i, 0); dfs(step + 1); R(i, 1);
        R(i, 1); dfs(step + 1); R(i, 0);
    }
    go();
    for(int i = 0; i < 4; i ++){
        ML(i, 0); dfs(step + 1); ML(i, 1);
        ML(i, 1); dfs(step + 1); ML(i, 0);
    }
}

int main()
{
    scanf("%d", &casenum);
    for (casei = 1; casei <= casenum; ++casei){
        for(int i = 1; i <= 72; i ++) scanf("%d", &a[i]);
        FLAG = 0;
        dfs(0);
        if(FLAG) puts("YES");
        else puts("NO");
    }
    return 0;
}
```

三点simpson法

```
double getAppr(double le,double ri)//三点simpson法
{
    double mid=(le+ri)/2;
    return (F(le)+4.0*F(mid)+F(ri))*(ri-le)/6.0;//三点simpson公式
}

double Simpson(double le,double ri)
{
    double sum=getAppr(le,ri);
    double mid=(le+ri)/2;
```

```
    double sumLe=getAppr(le,mid);
    double sumRi=getAppr(mid,ri);
    return (fabs(sum-sumLe-sumRi)<eps)?sum:Simpson(le,mid)+Simpson(mid,ri);//eps
为精度，用于算法自适应划分区间
}

Simpson(0, INF);  // 下界，上界



double dis_point_segment(const point p, const point s, const point t)
{
    // 似乎这里去掉前两句，就变成了求p点到直线st的距离了?
    // 的确是，前两句是求到端点的距离
    if(sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
    if(sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
    return fabs(det(s - p, t - p) / dist(s, t));
}
double dis_point_ray(const point p, const point s, const point t)
{
    // 似乎这里去掉前两句，就变成了求p点到直线st的距离了?
    // 的确是，前两句是求到端点的距离
    if(sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
    //if(sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
    return fabs(det(s - p, t - p) / dist(s, t));
}
double dis_point_line(const point p, const point s, const point t)
{
    // 似乎这里去掉前两句，就变成了求p点到直线st的距离了?
    // 的确是，前两句是求到端点的距离
    //if(sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
    //if(sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
    return fabs(det(s - p, t - p) / dist(s, t));
}
bool PointOnSegment(point p, point s, point t)
{
    return sgn(det(p - s, t - s)) == 0 && sgn(dot(p - s, p - t)) <= 0;
}
bool PointOnRay(point p, point s, point t)
{
    return sgn(det(p - s, t - s)) == 0 && (sgn(dot(p - s, p - t)) <= 0 ||
dist(p, s) >= dist(p, t));
}

bool parallel(line a, line b)
{
    return !sgn(det(a.a - a.b, b.a - b.b));
}
bool line_make_point(line a, line b, point &res)
{
    if(parallel(a, b)) return false;
    double s1 = det(a.a - b.a, b.b - b.a);
    double s2 = det(a.b - b.a, b.b - b.a);
    res = (s1 * a.b - s2 * a.a) / (s1 - s2);
    return true;
}
point A, B, C, D;

// point A -> point C
double cal1()
{
    return dist(A, C);
}
// point A -> segment CD
```

```
double cal2()
{
    return dis_point_segment(A, C, D);
}
// point A -> ray CD
double cal3()
{
    return dis_point_ray(A, C, D);
}
// point A -> line CD
double cal4()
{
    return dis_point_line(A, C, D);
}

// segment AB -> point C
double cal5()
{
    return dis_point_segment(C, A, B);
}

// segment AB -> segment CD
double cal6()
{
    point res;
    if(line_make_point({A, B}, {C, D}, res) && PointOnSegment(res, A, B) &&
PointOnSegment(res, C, D)){
        return 0;
    }
    else{
        double AA = min(dis_point_segment(A, C, D), dis_point_segment(B, C, D));
        double BB = min(dis_point_segment(C, A, B), dis_point_segment(D, A, B));
        return min(AA, BB);
    }
}
// segment AB -> ray CD
double cal7(point A, point B, point C, point D)
{
    point res;
    if(line_make_point({A, B}, {C, D}, res) && PointOnSegment(res, A, B) &&
PointOnRay(res, C, D)){
        return 0;
    }
    else{
        return min(dis_point_segment(C, A, B), min(dis_point_ray(A, C, D),
dis_point_ray(B, C, D)));
    }
}
// segment AB -> line CD
double cal8(point A, point B, point C, point D)
{
    point res;
    if(line_make_point({A, B}, {C, D}, res) && PointOnSegment(res, A, B)){
        return 0;
    }
    else{
        return min(dis_point_line(A, C, D), dis_point_line(B, C, D));
    }
}
// ray AB -> point C
double cal9()
{
    return dis_point_ray(C, A, B);
}
// ray AB -> segment CD
double cal10()
```

```
{
    return cal7(C, D, A, B);
}
// ray AB -> ray CD
double cal11()
{
    point res;
    if(line_make_point({A, B}, {C, D}, res) && PointOnRay(res, A, B) &&
PointOnRay(res, C, D)){
        return 0;
    }
    else{
        return min(dis_point_ray(A, C, D), dis_point_ray(C, A, B));
    }
    //min(dis_point_ray(A, C, D), dis_point_ray(C, A, B));
}
// ray AB -> line CD
double cal12(point A, point B, point C, point D)
{
    point res;
    if(line_make_point({A, B}, {C, D}, res) && PointOnRay(res, A, B)){
        return 0;
    }
    else{
        return dis_point_line(A, C, D);
    }
}
// line AB -> point C
double cal13()
{
    return dis_point_line(C, A, B);
}
// line AB -> segment CD
double cal14()
{
    return cal8(C, D, A, B);
}
// line AB -> ray CD
double cal15()
{
    return cal12(C, D, A, B);
}
// line AB -> line CD
double cal16()
{
    point res;
    if(line_make_point({A, B}, {C, D}, res)) return 0;
    else return dis_point_line(A, C, D);
}
```