```cpp
#include <bits/stdc++.h>
#define INF 0x3f3f3f3f
#define MAXN 1024

using namespace std;

typedef long long ll;

int n, k;
pair<ll, ll> v[MAXN];
ll sumW[MAXN], sumXW[MAXN];

ll dp[MAXN][MAXN];

int hullSize, hullPtr;
struct line {
    // y = m*x + n

    ll m, n;

    line(){}
    line(ll _m, ll _n):
        m(_m), n(_n){}

} hull[MAXN];

ll y(int idx, ll x) {
    return hull[idx].m*x + hull[idx].n;
}

double intersection(line t, line r) {
    double num = double(r.n - t.n);
    double den = double(t.m - r.m);
    return num / den;
}

void insert_line(line l) {
    while(hullSize >= 2 && ( intersection(l, hull[hullSize-2]) <
intersection(hull[hullSize-1], hull[hullSize-2]) ) ) {
        if (hullPtr == hullSize - 1) hullPtr--;
        hullSize--;
    }

    hull[ hullSize++ ] = l;
}

ll minimize(ll x) {
    while(hullPtr+1 < hullSize && y(hullPtr, x) > y(hullPtr+1, x)) hullPtr++;
    return y(hullPtr, x);
}

int main() {
    while(scanf("%d%d", &n, &k)!=EOF) {
        for(int i=0; i<n; i++) scanf("%lld%lld", &v[i].first, &v[i].second);

        sumW[0] = ll(v[0].second);
        sumXW[0] = ll(v[0].first)*ll(v[0].second);
        for(int i=1; i<n; i++) {
            sumW[i] = sumW[i-1] + ll(v[i].second);
            sumXW[i] = sumXW[i-1] + ll(v[i].first)*ll(v[i].second);
        }

        memset(dp, INF, sizeof(dp));

        for(int i=0; i<n; i++)
            dp[1][i] = v[i].first*sumW[i] - sumXW[i];

        for(int i=2; i<=k; i++) {
            hullSize = hullPtr = 0;
            for(int j=i-1; j<n; j++) {
```

```
   70:                 insert_line( line(-sumW[j-1], dp[i-1][j-1] + sumXW[j-1]) );
   71:                 dp[i][j] = minimize( v[j].first ) + v[j].first*sumW[j] - sumXW[j];
   72:
   73:                 // the loop below is substituted by the convex hull trick
   74:                 //
   75:                 // for(int p=0; p<j; p++) {
   76:                 //     dp[i][j] = min(dp[i][j], dp[i-1][p] + v[j].first*(sumW[j]
- sumW[p]) - sumXW[j] + sumXW[p]);
   77:                 // }
   78:             }
   79:         }
   80:
   81:         printf("%lld\n", dp[k][n-1]);
   82:     }
   83:
   84:         return 0;
   85: }
```

```
    1: #include <stdio.h>
    2: #include <vector>
    3: #include <algorithm>
    4: #include <math.h>
    5: #define pb push_back
    6: using namespace std;
    7:
    8: /* Exemplo de calculo de convex hull, solucao do problema
    9:     "SPOJBR - CERCAMG - Cercadinho de Plantas" */
   10:
   11: struct pt {
   12:     int x;
   13:     int y;
   14:
   15:     pt() {}
   16:     pt(int x, int y):x(x), y(y) {}
   17: };
   18:
   19: bool operator <(const pt &a, const pt &b) {
   20:     return a.x < b.x || (a.x == b.x && a.y < b.y);
   21: }
   22:
   23: bool ccw(pt &a, pt &b, pt &c) {
   24:     return a.x * (b.y - c.y) + b.x * (c.y - a.y) + c.x * (a.y - b.y) > 0;
   25: }
   26:
   27: bool cw(pt &a, pt &b, pt &c) {
   28:     return a.x * (b.y - c.y) + b.x * (c.y - a.y) + c.x * (a.y - b.y) < 0;
   29: }
   30:
   31: double dist(pt &a, pt &b) {
   32:     int dx = a.x - b.x;
   33:     int dy = a.y - b.y;
   34:     return sqrt(dx * dx + dy * dy);
   35: }
   36:
   37: double convex_hull(vector <pt> &v) {
   38:     vector <pt> up;
   39:     vector <pt> down;
   40:
   41:     sort(v.begin(), v.end());
   42:     up.pb(v[0]);
   43:     down.pb(v[0]);
   44:     for (int i = 1; i < (int)v.size(); i++) {
   45:         while(up.size() >= 2 && !cw(up[up.size() - 2], up[up.size() - 1], v[i])) {
   46:             up.pop_back();
   47:         }
   48:         while(down.size() >= 2 && !ccw(down[down.size() - 2], down[down.size() -
1], v[i])) {
   49:             down.pop_back();
   50:         }
   51:         up.pb(v[i]);
   52:         down.pb(v[i]);
   53:     }
   54:
   55:     for (int i = down.size() - 2; i > 0; i--) {
   56:         up.pb(down[i]);
   57:     }
   58:     double res = 0;
   59:     int n = up.size();
   60:     //printf("n = %d\n", n);
   61:     for (int i = 0; i < n; i++) {
   62:         res += dist(up[i], up[(i + 1) % n]);
   63:         //printf("dist = %lf\n", dist(up[i], up[(i + 1) % n]));
   64:     }
   65:     return res;
   66: }
   67:
   68: int main(void) {
   69:     int a, d;
```

```
70:     int x, y;
71:     vector <pt> v;
72:
73:     while(scanf(" %d %d", &a, &d) != EOF) {
74:         v.clear();
75:         for (int i = 0; i < a; i++) {
76:             scanf(" %d %d", &x, &y);
77:             v.pb(pt(x, y));
78:         }
79:
80:         double res = convex_hull(v);
81:         printf("%.2lf\n", res);
82:     }
83:     return 0;
84: }
```

```cpp
#include <stdio.h>
#include <algorithm>
using namespace std;

/* Exemplo de calculo de distancia maxima entre pontos
    considerando distancia do tipo manhatann
    solucao do problema "codeforces 366E - Dima and Magic Guitar" */

const int MAX_K = 10;
const int INF = 0x3f3f3f3f;

int v[MAX_K][1 << 2];

int comp(int mask) {
    return ((1 << 2) - 1) ^ mask;
}

int solve(int x, int y) {
    int aux = 0;
    for (int mask = 0; mask < 1 << 2; mask++) {
        aux = max(aux, v[x][mask] + v[y][comp(mask)]);
    }
    return aux;
}

int get_val(int i, int j, int mask) {
    if (mask & 1) {
        j *= -1;
    }
    if ((mask >> 1) & 1) {
        i *= -1;
    }
    return i + j;
}

int main(void) {
    int n, m, k, s;
    int x, y;

    for (int i = 0; i < MAX_K; i++) {
        for (int j = 0; j < 1 << 2; j++) {
            v[i][j] = -INF;
        }
    }

    scanf(" %d %d %d %d", &n, &m, &k, &s);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf(" %d", &x);
            for (int mask = 0; mask < 1 << 2; mask++) {
                int aux = get_val(i, j, mask);
                v[x][mask] = max(v[x][mask], aux);
            }
        }
    }

    int res = 0;
    scanf(" %d", &x);
    while(s-- > 1) {
        scanf(" %d", &y);
        res = max(res, solve(x, y));
        x = y;
    }
    printf("%d\n", res);
    return 0;
}
```