```cpp
 1: #include <bits/stdc++.h>
 2: #define INF 0x3f3f3f3f
 3:
 4: using namespace std;
 5:
 6: typedef long long ll;
 7:
 8: int n, k;
 9: pair<ll, ll> v[1024];
10: ll sumW[1024], sumXW[1024];
11:
12: ll dp[1024][1024];
13:
14: int hullSize, hullPtr;
15: struct line {
16:     // y = m*x + n
17:
18:     ll m, n;
19:
20:     line(){}
21:     line(ll _m, ll _n):
22:         m(_m), n(_n){}
23:
24: } hull[1024];
25:
26: ll y(int idx, ll x) {
27:     return hull[idx].m*x + hull[idx].n;
28: }
29:
30: double intersection(line t, line r) {
31:     double num = double(r.n - t.n);
32:     double den = double(t.m - r.m);
33:     return num / den;
34: }
35:
36: void insert_line(line l) {
37:     while(hullSize >= 2 && ( intersection(l, hull[hullSize-2]) <
intersection(hull[hullSize-1], hull[hullSize-2]) ) ) {
38:         if (hullPtr == hullSize - 1) hullPtr--;
39:         hullSize--;
40:     }
41:
42:     hull[ hullSize++ ] = l;
43: }
44:
45: ll minimize(ll x) {
46:     while(hullPtr+1 < hullSize && y(hullPtr, x) > y(hullPtr+1, x)) hullPtr++;
47:     return y(hullPtr, x);
48: }
49:
50: int main() {
51:     while(scanf("%d%d", &n, &k)!=EOF) {
52:         for(int i=0; i<n; i++) scanf("%lld%lld", &v[i].first, &v[i].second);
53:
54:         sumW[0] = ll(v[0].second);
55:         sumXW[0] = ll(v[0].first)*ll(v[0].second);
56:         for(int i=1; i<n; i++) {
57:             sumW[i] = sumW[i-1] + ll(v[i].second);
58:             sumXW[i] = sumXW[i-1] + ll(v[i].first)*ll(v[i].second);
59:         }
60:
61:         memset(dp, INF, sizeof(dp));
62:
63:         for(int i=0; i<n; i++)
64:             dp[1][i] = v[i].first*sumW[i] - sumXW[i];
65:
66:         for(int i=2; i<=k; i++) {
67:             hullSize = hullPtr = 0;
68:             for(int j=i-1; j<n; j++) {
69:                 insert_line( line(-sumW[j-1], dp[i-1][j-1] + sumXW[j-1]) );
```

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

```cpp
 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;
```

```cpp
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
 1: #include <stdio.h>
 2: #include <algorithm>
 3: using namespace std;
 4:
 5: /* Exemplo de calculo de distancia maxima entre pontos
 6:     considerando distancia do tipo manhatann
 7:     solucao do problema "codeforces 366E - Dima and Magic Guitar" */
 8:
 9: const int MAX_K = 10;
10: const int INF = 0x3f3f3f3f;
11:
12: int v[MAX_K][1 << 2];
13:
14: int comp(int mask) {
15:     return ((1 << 2) - 1) ^ mask;
16: }
17:
18: int solve(int x, int y) {
19:     int aux = 0;
20:     for (int mask = 0; mask < 1 << 2; mask++) {
21:         aux = max(aux, v[x][mask] + v[y][comp(mask)]);
22:     }
23:     return aux;
24: }
25:
26: int get_val(int i, int j, int mask) {
27:     if (mask & 1) {
28:         j *= -1;
29:     }
30:     if ((mask >> 1) & 1) {
31:         i *= -1;
32:     }
33:     return i + j;
34: }
35:
36: int main(void) {
37:     int n, m, k, s;
38:     int x, y;
39:
40:     for (int i = 0; i < MAX_K; i++) {
41:         for (int j = 0; j < 1 << 2; j++) {
42:             v[i][j] = -INF;
43:         }
44:     }
45:
46:     scanf(" %d %d %d %d", &n, &m, &k, &s);
47:     for (int i = 0; i < n; i++) {
48:         for (int j = 0; j < m; j++) {
49:             scanf(" %d", &x);
50:             for (int mask = 0; mask < 1 << 2; mask++) {
51:                 int aux = get_val(i, j, mask);
52:                 v[x][mask] = max(v[x][mask], aux);
53:             }
54:         }
55:     }
56:
57:     int res = 0;
58:     scanf(" %d", &x);
59:     while(s-- > 1) {
60:         scanf(" %d", &y);
61:         res = max(res, solve(x, y));
62:         x = y;
63:     }
64:     printf("%d\n", res);
65:     return 0;
66: }
```