```c
// Me cung so. tim duong di cua robot
#include <stdio.h>

#define MAXN 1000
#define NO_EDGE 0
#define INFINITY 9999999

// Graph
typedef struct {
    int n;
    int L[MAXN][MAXN];
} Graph;

void init_graph(Graph* G, int n) {
    G->n = n;

    int i, j;
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j) {
            G->L[i][j] =NO_EDGE;
        }
    }
}

void add_edge(Graph* G, int x, int y, int w) {
    G->L[x][y] = w;
}

int mark[MAXN];
int pi[MAXN];
int p[MAXN];

void Dijkstra(Graph* G, int s) {
    int i, j, it;
    for (i = 1; i <= G->n; ++i) {
        pi[i] = INFINITY;
        mark[i] = 0;
    }

    pi[s] = 0;
    p[s] = -1;

    for (it = 1; it < G->n; ++it) {
```

```
        int min_pi = INFINITY;
        for (j = 1; j <= G->n; ++j) {
            if (mark[j] == 0 && pi[j] < min_pi) {
                min_pi = pi[j];
                i = j;
            }
        }

        mark[i] = 1;
        for (j = 1; j <= G->n; ++j) {
            if (G->L[i][j] != NO_EDGE && mark[j] == 0) {
                if (pi[i] + G->L[i][j] < pi[j]) {
                    pi[j] = pi[i] + G->L[i][j];
                    p[j] = i;
                }
            }
        }
    }
}

int main() {
    Graph G;
    int n, m, u, i, j, s, t, x;
    scanf("%d%d", &n, &m);
    init_graph(&G, n * m);
    int a[n + 1][m + 1];

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; ++j) {
            scanf("%d", &x);
            a[i][j] = x;
        }
    }

    int di[] = {-1, 1, 0, 0};
    int dj[] = {0, 0, -1, 1};

    for (u = 1; u <= n * m; ++u) {
        int i = (u - 1) / m;
        int j = (u - 1) % m;
        int k, ii, jj;
        for (k = 0; k < 4; ++k) {
            ii = i + di[k];
```

```
            jj = j + dj[k];
            int v;
            if ((ii >= 0) && (ii < n) && (jj >= 0) && (jj < m)) {
                v = ii * m + jj + 1;
                add_edge(&G, u, v, a[ii][jj]);
            }
        }
    }

    Dijkstra(&G, 1);

    for (int i = 1; i <= n * m; ++i) {
        for (int j = 1; j <= n * m; ++j) {
        }
    }

    printf("%d", pi[n * m] + a[0][0]);
    return 0;
}
```