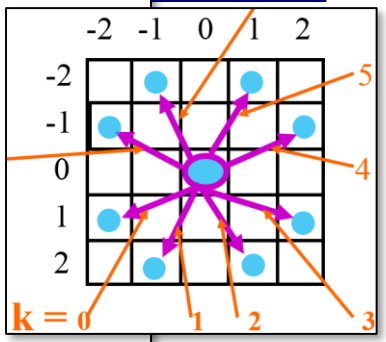
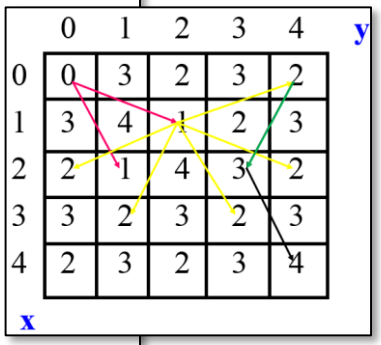
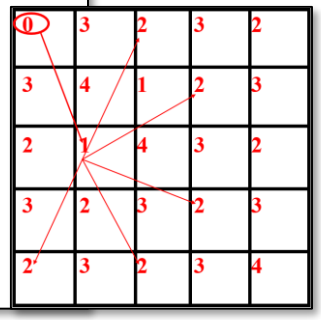
```
/*
任务：在8×8的棋盘上，输入n个骑士的出发点，假定骑士每天只能跳一步，
      计算n个人的聚会地点和走多少天。要求尽早聚会（走的天数最少）且
 n个人走的总步数最少。骑士的跳步按中国象棋的马来跳。
*/

#include <iostream>// cin, cout
#include <memory>// memset
#include <iomanip>// setw
using namespace std;

const int T = 5;  //棋盘尺寸
const int dx[8] = {1,2,2,1,-1,-2,-2,-1}; //8个跳步方向上的x增量
const int dy[8] = {-2,-1,1,2,2,1,-1,-2}; //8个跳步方向上的y增量

struct qtype { int x, y; };  // 棋盘坐标点
qtype queue[T*T]; // 搜索扩展的队列

int n;// 骑士数目n
qtype rec[T*T];  // 存储各骑士在棋盘上的起始位置

struct jtype
{
    int sum;// 记录n个骑士跳入一个棋盘格子中的总步数
    int max;// 记录n个骑士跳入一个棋盘格子中，n个中的某人的最多步数
} good[T][T];// 存储棋盘各格在骑士聚会时的跳步信息：总步数及最多步数

// 存储各骑士的跳步信息表
// 第1维是骑士号，第2和第3维是棋盘坐标x和y
int best[T*T][T][T];
```



k	0	1	2	3	4	5	6	7
dx	1	2	2	1	-1	-2	-2	-1
dy	-2	-1	1	2	2	1	-1	-2





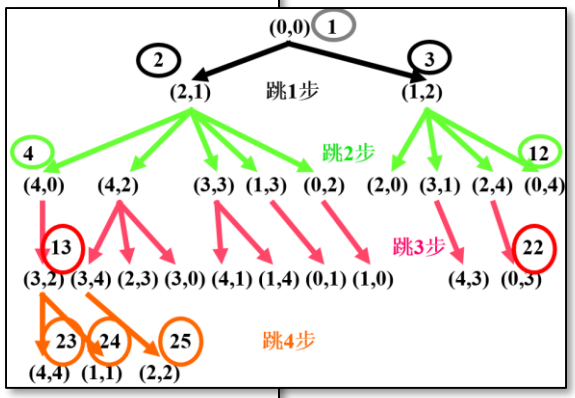
```
// 获得输入数据
void input()
{
    // 提示并输入骑士的数目n
    cout << "输入骑士数目 n = ";  cin >> n;

    // 输入n个骑士的初始位置信息
    for (int i=0; i<n; i++) {
        // 提示并输入第i号骑士的 x 坐标点
        cout << "第" << i << "号骑士行位置 x = ";   cin >> rec[i].x;

        // 提示并输入第i号骑士的 y 坐标点
        cout << "第" << i << "号骑士列位置 y = ";   cin >> rec[i].y;
    }
}

//初始化函数
void init()
{
    // 初始化best数组，使各元素值为－1
    // 表示棋盘中的每个格子都未曾填过信息
    memset(best, -1, sizeof(best));

    // 设置各骑士出发位置的跳步值
    for (int i=0; i<n; i++)
        best[i][rec[i].x][rec[i].y] = 0;
}

// 判断第i号骑士能否跳到(x,y)处
bool okjump(int i, int x, int y)
{
    //如果落点在棋盘内且该骑士从未到过该落点，则返回true，否则返回false
    return ( x>=0 && x<T && y>=0 && y<T && best[i][x][y]==-1 );
}
```

```
输入骑士数目 n = 4
第0号骑士行位置 x = 0
第0号骑士列位置 y = 0
第1号骑士行位置 x = 1
第1号骑士列位置 y = 4
第2号骑士行位置 x = 1
第2号骑士列位置 y = 2
第3号骑士行位置 x = 3
第3号骑士列位置 y = 4
```

假定有4个骑士，初始位置分别在(0,0), (1,4), (1,2), (3,4)

	0	1	2	3	4
0	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	-1

```
// 宽度优先策略生成各骑士的“跳步信息图”
void BFS()
{
    // 对0,1,...,n-1号的每一个骑士进行跳步处理
    for(int i=0; i<n; i++)
    {
        int step = 0;    // 骑士跳步数step，初始为0
        // 定义队头head, 队尾tail, 初始值均为0
        int head = 0, tail = 0;
        // 让第i个骑士"入队": 将出发点放到队列中
        queue[tail] = rec[i];

        //从队头到队尾进行扩展, 直到队空为止
        //注意：当队头head不断增加时，队尾tail也是在不断变化的
        while (head <= tail) {
            int cur_tail = tail;    // 记住当前阶段的队尾位置

            //从 队头到当前队尾 为一个阶段进行扩展
            for( ; head <= cur_tail; head++) {
                for( int k=0; k<8; k++) { // 枚举各种跳法
                    // 从(x,y)点出发，采取第k种跳法，计算落点(x1,y1)
                    int x1 = queue[head].x + dx[k];
                    int y1 = queue[head].y + dy[k];
                    if ( okjump(i,x1,y1) ) { // i号骑士能跳至(x1,y1)吗?
                        tail++; // 队尾加1(这是新扩展出的)
                        // 记录落点位置
                        queue[tail].x = x1; queue[tail].y = y1;
                        // 记录第i号骑士跳至(x1,y1)处的步数
                        best[i][x1][y1] = step + 1;
                    }
                }
            }
            step++;       // 扩展下一段时，step步数需加1
        }
    }
}
```



k	/	2	3	1	2	3	4	5	0	1	3	4
x	0	2	1	4	4	3	1	0	2	3	2	0
y	0	1	2	0	2	3	3	2	0	1	4	4
step	0	1	1	2	2	2	2	2	2	2	2	2
tail	1	2	3	4	5	6	7	8	9	10	11	12

```
void display()                  //显示棋局（调试用）
{
    cout << "---------DEBUG INFO BEGIN ------>>>>>>>" << endl;
    for (int i=0; i<n; i++) {               // 枚举每个骑士
        // 枚举棋盘各位置（按行列坐标），输出骑士跳至(x,y)的步数
        for (int x=0; x<T; x++) {
            for (int y=0; y<T; y++)
                cout << best[i][x][y] << " ";
            cout << endl;
        }
        cout << endl;
    }
    cout << "<<<<<<------DEBUG INFO END ------------" << endl;
}

void output_good()      // 调试用函数
{
    cout << "==============================(MAX, SUM)==============================" << endl;
    for (int i=0; i<T; i++) {
        for (int j=0; j<T; j++)
            cout << "( " << good[i][j].max << ", " << setw(2) << good[i][j].sum << ") ";
        cout << endl;
    }
    cout << "======================================================================" << endl;
}
```

```
// 搜索骑士们聚会的最佳位置
void search()
{
    // STEP 1 : 宽度优先策略生成骑士的“跳步信息表”
    BFS();
    display();// 调试: 输出各骑士的跳步信息表

    // STEP 2 : 对棋盘各格子计算n名骑士跳至该处的总步数和最大步数
    for (int x=0; x<T; x++) {
        for (int y=0; y<T; y++) {
            good[x][y].sum = 0;  // 总的步数初值设为0
            good[x][y].max = -1; // 最大步数初值设为-1

            // 枚举 n 名骑士，更新总步数和最大步数
            for (int j=0; j<n; j++) {
                good[x][y].sum += best[j][x][y];
                if (best[j][x][y] > good[x][y].max)
                    good[x][y].max = best[j][x][y];
            }
        }
    }
    output_good(); // 调试: 输出各处的聚会信息

    // STEP 3 : 查找最佳聚会位置
    int minx, miny;   // 定义最佳聚会位置
    int min = 32767;  // 定义骑士们到达聚会点的最少天数，预置大数
    int sum = 0;      // 定义最佳聚会点的众骑士的跳步总和，预置为0
```

```
// 枚举棋盘上的每个格子
    for (int x=0; x<T; x++) {
        for (int y=0; y<T; y++) {

            // 如果在(x,y)处聚会的天数小于当前最少天数
            if (good[x][y].max < min) {
                min = good[x][y].max;
                sum = good[x][y].sum;
                minx = x;                    //记录最佳聚会位置x
                miny = y;                    //记录最佳聚会位置y
            }

            //如果在(x,y)处聚会的天数等于当前最少天数
            if (good[x][y].max == min) {
                //如该处的跳步总和比前面的少,将该处作为最佳聚会位置
                if (good[x][y].sum < sum) {
                    sum = good[x][y].sum;
                    minx = x;                //记录最佳聚会位置x
                    miny = y;                //记录最佳聚会位置y
                }
            }
        }
    }

    // STEP 4: 输出最佳聚会位置、聚会时间、跳步总和
    cout << "最佳聚会位置：行x = " << minx << ", 列y = " << miny << endl;
    cout << "最佳聚会时间：" << min << endl;
    cout << "骑士跳步总和：" << sum << endl;
}
```

```
int main()
{
    input();// 输入数据
    init();// 初始化
    search();//搜索最佳聚会位置
    return 0;
}
```

```
输入骑士数目 n = 4
第0号骑士行位置 x = 0
第0号骑士列位置 y = 0
第1号骑士行位置 x = 1
第1号骑士列位置 y = 4
第2号骑士行位置 x = 1
第2号骑士列位置 y = 2
第3号骑士行位置 x = 3
第3号骑士列位置 y = 4
```

```
0 3 2 3 2
3 4 1 2 3
2 1 4 3 2
3 2 3 2 3
2 3 2 3 4
```

```
3 2 1 2 3
2 3 2 3 0
3 2 1 2 3
2 3 4 1 2
3 2 3 2 3
```

```
1 2 3 2 1
2 3 0 3 2
1 2 3 2 1
4 1 2 1 4
3 2 3 2 3
```

```
3 2 3 2 3
2 3 4 1 2
3 2 1 2 3
2 3 2 3 0
3 2 1 2 3
```

```
<<<<<<------DEBUG INFO END ------------
==================================(MAX,
SUM)==================================
( 3,  7) ( 3,  9) ( 3,  9) ( 3,  9) ( 3,  9)
( 3,  9) ( 4, 13) ( 4,  7) ( 3,  9) ( 3,  7)
( 3,  9) ( 2,  7) ( 4,  9) ( 3,  9) ( 3,  9)
( 4, 11) ( 3,  9) ( 4, 11) ( 3,  7) ( 4,  9)
( 3, 11) ( 3,  9) ( 3,  9) ( 3,  9) ( 4, 13)
==========================================================
============================
最佳聚会位置：行x = 2, 列y = 1
最佳聚会时间：2
骑士跳步总和：7
```