<https://www.hackerrank.com/challenges/queens-attack-2>

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

class Solution

{

static int Norte(int rq, int cq, int ro, int co)

{

if (cq != co)

{

return int.MaxValue;

}

if (ro < rq)

{

return int.MaxValue;

}

return ro - rq - 1;

}

static int Sur(int rq, int cq, int ro, int co)

{

if (cq != co)

{

return int.MaxValue;

}

if (ro > rq)

{

return int.MaxValue;

}

return rq - ro - 1;

}

static int Este(int rq, int cq, int ro, int co)

{

if (rq != ro)

{

return int.MaxValue;

}

if (co < cq)

{

return int.MaxValue;

}

return co - cq - 1;

}

static int Oeste(int rq, int cq, int ro, int co)

{

if (rq != ro)

{

return int.MaxValue;

}

if (co > cq)

{

return int.MaxValue;

}

return cq - co - 1;

}

static int NorEste(int rq, int cq, int ro, int co)

{

//if (ro < rq || co < cq)

//{

// return 0;

//}

if (ro > rq && co > cq && (ro - rq) == (co - cq))

{

return ro - rq - 1;

}

return int.MaxValue;

}

static int SurEste(int rq, int cq, int ro, int co)

{

if (ro < rq && co > cq && (rq - ro) == (co - cq))

{

return rq - ro - 1;

}

return int.MaxValue;

}

static int NorOeste(int rq, int cq, int ro, int co)

{

if (ro > rq && co < cq && (ro - rq) == (cq - co))

{

return ro - rq - 1;

}

return int.MaxValue;

}

static int SurOeste(int rq, int cq, int ro, int co)

{

if (ro < rq && co < cq && (rq - ro) == (cq - co))

{

return rq - ro - 1;

}

return int.MaxValue;

}

static void Main(String[] args)

{

string[] tokens\_n = Console.ReadLine().Split(' ');

int n = Convert.ToInt32(tokens\_n[0]);

int k = Convert.ToInt32(tokens\_n[1]);

string[] tokens\_rQueen = Console.ReadLine().Split(' ');

int rQueen = Convert.ToInt32(tokens\_rQueen[0]);

int cQueen = Convert.ToInt32(tokens\_rQueen[1]);

int maxNorte = n - rQueen;

int maxSur = rQueen - 1;

int maxEste = n - cQueen;

int maxOeste = cQueen - 1;

int maxNorEste = n - Math.Max(rQueen, cQueen);

int maxSurEste = Math.Min(n - cQueen, rQueen - 1);

int maxNorOeste = Math.Min(n - rQueen, Math.Abs(1 - cQueen));

int maxSurOeste = Math.Min(rQueen, cQueen) - 1;

for (int a0 = 0; a0 < k; a0++)

{

string[] tokens\_rObstacle = Console.ReadLine().Split(' ');

int ro = Convert.ToInt32(tokens\_rObstacle[0]);

int co = Convert.ToInt32(tokens\_rObstacle[1]);

// your code goes here

//int ro = elem[0];

//int co = elem[1];

maxNorte = Math.Min(maxNorte, Norte(rQueen, cQueen, ro, co));

maxSur = Math.Min(maxSur, Sur(rQueen, cQueen, ro, co));

maxEste = Math.Min(maxEste, Este(rQueen, cQueen, ro, co));

maxOeste = Math.Min(maxOeste, Oeste(rQueen, cQueen, ro, co));

maxNorEste = Math.Min(maxNorEste, NorEste(rQueen, cQueen, ro, co));

maxSurEste = Math.Min(maxSurEste, SurEste(rQueen, cQueen, ro, co));

maxNorOeste = Math.Min(maxNorOeste, NorOeste(rQueen, cQueen, ro, co));

maxSurOeste = Math.Min(maxSurOeste, SurOeste(rQueen, cQueen, ro, co));

}

Console.WriteLine(maxNorte + maxSur + maxEste + maxOeste + maxNorEste +

maxSurEste + maxNorOeste + maxSurOeste);

}

}