IOI2006 中国国家集训队论文

# 信息学中的

# 参考系与坐标系

# Reference and Coordinate System

## in Olympiad in Informatics

安徽芜湖第一中学

汪晔

# 摘　要

信息学是一门新兴的学科，与数学、物理等经典学科相比显然年轻得多。因而信息学中的许多思想都来源于数学、物理。本文将通过三个方面介绍参考系与坐标系在信息学中的应用。希望能通过本文，开拓解决信息学问题的思路，使参考系与坐标系的思想成为解决信息学问题的有力武器。

# 关键字

信息学　参考系　坐标系　单位长度　离散化　数轴

# 目　　录

# 前言

我在做题中有时会灵光一现，想出一些很特殊或是未接触过的解法，这些题目和想法都会给我留下很深刻的印象。本篇论文的第三题就是其中之一，可以说这题是整篇论文的导火索。

但灵光总不会经常光顾，大多数情况下解决新到手的问题都得按部就班地分析，做过类似的题型，当然易于解决；但如果没有呢？我常常想那些“灵光一现”的题目，是否也可以按套路分析出来呢？这里的套路也就是常说的解题的思维和方法了，然而个个人都不尽相同，但其中往往有很多相同和相通的思想。

如果说解决题目是在黑暗中摸索，那么这些思想便是一个个的灯塔，正在这些灯塔的指引下，我们摸索出了道路——也就是解题方法了。那么灯塔越多，解决问题的路径便越清晰。然而很多路，我们都只凭着直觉，也就是经验走出来的，对途中未亮的灯塔视而不见。

我写本篇论文，就是想为大家点亮一个灯塔，这个灯塔就名为参考系与坐标系的思想。希望大家今后在黑暗中摸索道路时能多一片光明；更希望大家在走出一条道路后，可以将道路中的灯塔都点亮。

本篇论文主要分为两大部分，第一部分是介绍参考系与坐标系的基本概念，为后文作铺垫；第二部分是介绍参考系与坐标系思想在信息学中的应用。第二部分分为三大块：单位长度的改变，参考系的选择，坐标系的建立。每一块都由问题引入，问题描述，问题分析以及小结构成。每个问题的分析都不是单一方法，而是从简单到复杂，从常规到特殊，这也是我们面临一个未知问题时常用的思维方法。虽然这些题目的道路已经明朗了，但我仍希望本篇论文的分析能体现黑暗中的摸索，所以在分析中，会有“？”的出现，正由于这些“？”，使问题不断深入。每个问题分析后的小结正是起点亮灯塔的作用。

# 参考系与坐标系的介绍

## ♦ 参考系的概念

平时我们说树木、房屋是静止的，行驶的汽车是运动的，这是以地面作为标准来说的。坐在行驶的火车里的乘客，认为自己是静止的，路旁的树木在向后倒退，这是以车厢作为标准来说的。

在描述一个物体运动时，选来作为标准的另外物体，叫做参考系。

## ♦ 坐标系的概念

我们都有去影院看电影的经历，观众席的所有座位都按“几排几号”编号，以便确定每个座位在影院中的位置。这样，观众就能根据入场券上的“排数”和“号数”准确地“对号入座”。这正运用了坐标系的有关知识。

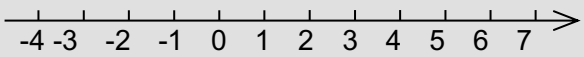
在参考物上任意选定一个参考点 $O$，并安置一个以 $O$ 为原点的坐标系，就可以把物体相对于参考系的位置定量地用坐标表示出来。

坐标系的三要素为：原点，正方向，单位长度。

## ♦ 几种坐标系

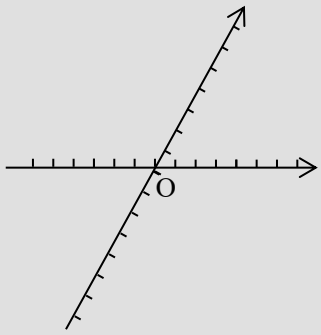
### 一维坐标系

下图是一条数轴，数轴上的点可以用一个数来表示，这个数就叫做这个点的坐标。知道一个点的坐标，这个点在数轴上的位置也就确定了。

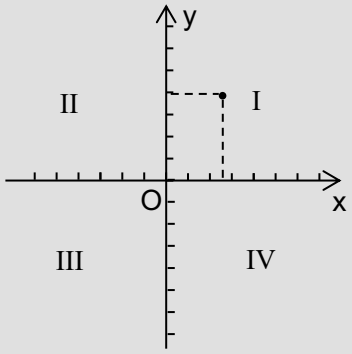



### 二维坐标系

平面上几个非平行数轴原点重合组成的坐标系就是二维坐标系。



我们接触最多的是两条互相垂直、原点重合的数轴组成的平面直角坐标系，也就是笛卡儿坐标系。水平的数轴称为 x 轴或横轴，习惯上取向右为正方向；竖直的数轴称为 y 轴或纵轴，取向上为正方向；两坐标轴的交点为平面直角坐标系的原点。

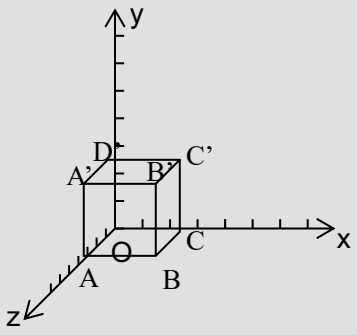



有了平面直角坐标系，平面内的点就可以用一个有序数对来表示了。一个点分别向 x 轴和 y 轴做垂线，垂足在 x 轴上的坐标就是横坐标，垂足在 y 轴上的坐标就是纵坐标。

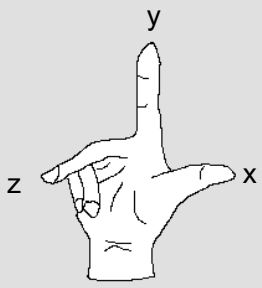
建立了平面直角坐标系以后，坐标平面就被两条坐标轴分成 I,II,III,IV 四个部分，分别叫做第一象限，第二象限，第三象限和第四象限，坐标轴上的点不属于任何象限。

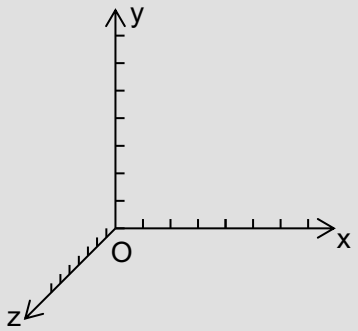
## 三维坐标系

我么常见的是三维坐标系是空间直角坐标系。如图，OABC-D'A'B'C'是单位正方体，以 O 为原点，分别以射线 OA，OC，OD'的方向为正方向，以线段 OA,OC,OD'的长为单位长，建立三条数轴：x 轴，y 轴，z 轴。这时我们说建立了一个空间直角坐标系 O-xyz。其中点 O 为坐标原点；x 轴，y 轴，z 轴叫做坐标轴，通过每两个坐标轴的平面叫做坐标平面，分别称为 xOy 平面，yOz 平面，zOx 平面。

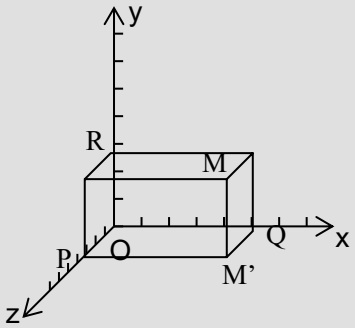



在空间直角坐标系中，让右手拇指直向 x 轴的正方向，食指指向 y 轴的正方向，如果中指指向 z 轴的正方向，则称这个坐标系为右手直角坐标系。无特别说明，三维坐标系一般都是右手直角坐标系。




设 M 为空间中一定点，过点 M 分别做垂直于 x 轴，y 轴和 z 轴的平面，依次交 x 轴，y 轴和 z 轴于点 P，Q 和 R，设点 P，Q 和 R 在 x 轴，y 轴和 z 轴上的坐标分别是 x,y,z，那么点 M 就对应唯一确定的有序实数组(x,y,z)。反过来，给定实数组(x,y,z)，我们可以在 x 轴，y 轴和 z 轴上一次取坐标为 x，y 和 z 的点 P，Q 和 R。




这样，空间中一点 M 的坐标可以由序实数组(x,y,z)来表示，有序实数组(x,y,z)叫做点 M 在此空间直角坐标系中的坐标，记作 M(x,y,z)，其中 x 叫做点 M 的横坐标，y 叫做点 M 的纵坐标，z 叫做点 M 的竖坐标。

## ♦ 参考系与坐标系的应用

### 参考系的应用

参考系的思想来源于物理。解决运动学问题的第一步就是选择合适的参考系。研究太阳系中物体的运动选择太阳为参考系比选择地球为参考系简单得多。而研究地面上物体的运动，我们不会选择太阳作为参考系。

### 坐标系的应用

坐标系的概念最初来源于数学，但在物理等学科中运用极其广泛。研究函数离不开坐标系，解析几何就是由于坐标系的发明而诞生的。实际生活中，用经纬度表示地球上一个地点的地理位置，用极坐标表示区域内地点的位置，用平面直角坐标表示区域内地点的位置等等，实际上都是利用了有序数对与点的对应关系，是坐标与点一一对应思想的表现。

# 单位长度的改变

♦ 问题引入：

对于同一些数据，选用不同的单位长度的意义，建立的坐标系就不同，解决问题的繁简也会不同。

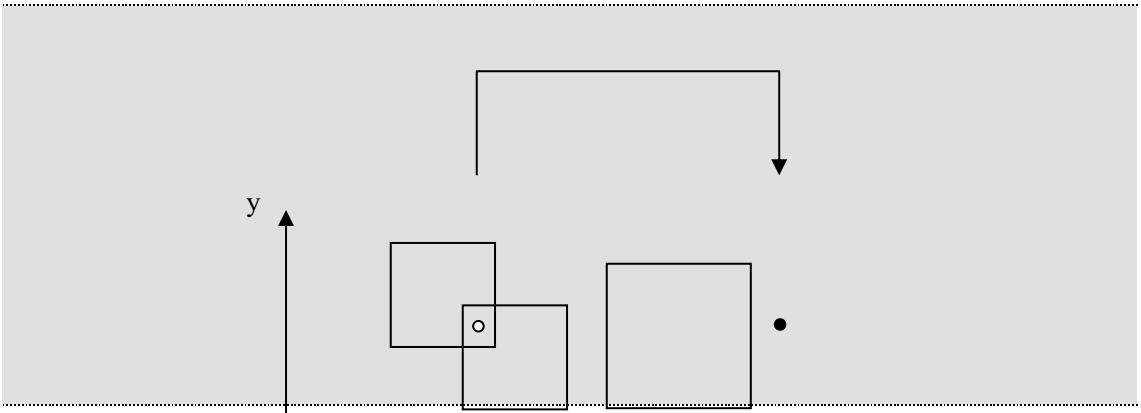
比如，我们用数轴表示分数，把一次考试中所考生的分数填入数轴中，这时单位长度都是 1 分。我们如果用数轴表示名次，单位就是 1 名。这种从分数到名次的转换，也就是我们常说的离散化。

离散化常用于处理数据规模很大，数据个数较小的问题。就刚才的例子而言，如果考试得满分为 1000 分，一共有 50 名考生，那么以名次建立的数轴范围就远小于以分数建立的数轴范围了。

♦ 问题描述：**AHOI 2005 Day1 穿越磁场（Cross）**

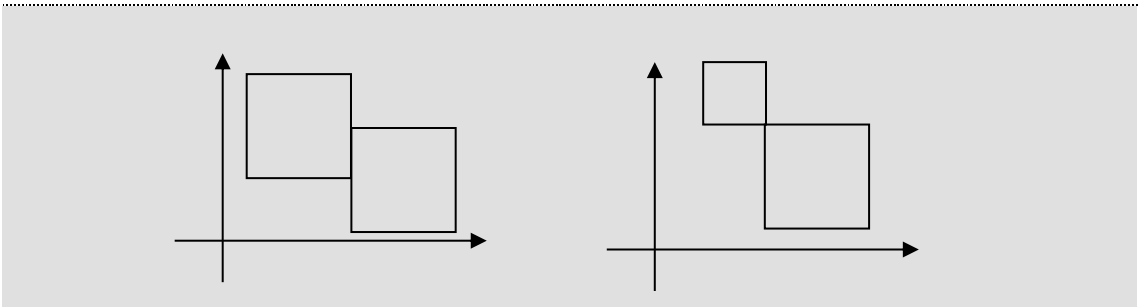
探险机器人在 ***Samuel*** 星球上寻找一块奇特的矿石，然而此时它陷入了一片神秘的磁场区域，动弹不得。

探险空间站立刻扫描了这片区域，绘制出该区域的磁场分布平面图。这片区域中分布了 N 个磁场，每个磁场呈正方形，且边与坐标轴平行。




例如下图中，存在 3 个磁场，白点表示机器人的位置，黑点表示矿石的位置：

科学家们分析平面图，进一步发现：**这些磁场为大小不一的正方形，可能相交，甚至覆盖，但是它们的边缘不会重合，顶点也不会重合。**



例如下面的两种情形是**不会出现**的：

科学家们给探险机器人启动了磁力罩，这样它就可以在磁场中自由穿越了。

**初始时，探险机器人和所有矿石都不在任何磁场的边缘。由于技术限制，在穿越过程中机器人只能够水平或垂直移动，且不能够沿着磁场的边缘行动。**

由于磁力罩的能量有限，科学家们希望探险机器人穿越尽量少的磁场边缘采集到这块矿石。例如上图中，探险机器人最少需要穿越两次磁场边缘。

现在小联请你编写程序，帮助科学家们设计探险机器人的路线，统计探险机器人最少需要穿越多少次磁场边缘。

输入：第一行有一个整数 N，表示有 N 个磁场（$1 \le N \le 100$）。随后有 N 行，每行有三个整数 X、Y、C（$0 \le X, Y, C \le 10000$），表示一个磁场左下角坐标为（X,Y），边长为 C。接下来有一行，共有四个整数 SX, SY, TX, TY，表示机器人初始坐标为（SX, SY），矿石坐标为（TX，TY）（其中，$0 \le SX, SY, TX, TY \le 10000$）。

输出：单行输出一个整数，表示机器人最少需要穿越多少次磁场边缘。

样例：

输入：

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

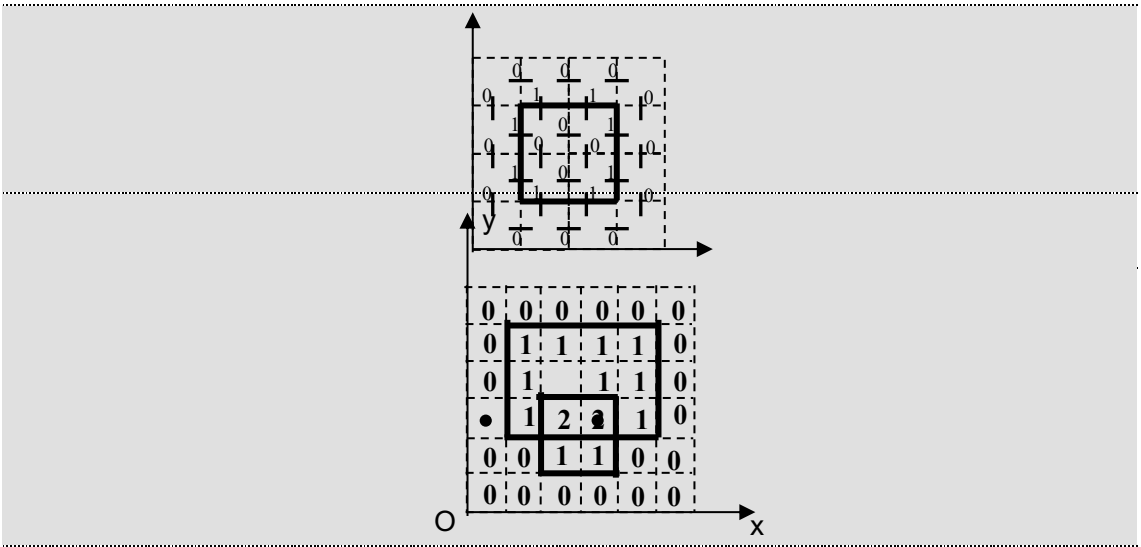
输出：

```
2
```

♦ 问题分析：

**方法 1**：

将坐标中的所有整点坐标当作顶点，在每个点与坐标系中相邻的点间加一条无向边，如果穿过磁场，边的权值为 1，否则权值为 0。



求机器人从起点到终点的最小耗费，也就是求构造的图中两点之间的最短路径，我们可以用 Dijkstra 解决。每次循环中寻找最大值的复杂度为 O(V)，改进相邻点时由于要判断是否穿过磁场边，所以复杂度为 O(N)。整个算法复杂度为就是 $O(VN+V^2)$，这里 V=整个坐标中的点的个数=10000*10000，显然超时。当然，在实现过程中我们可以有所优化，比如确定查找点的范围。

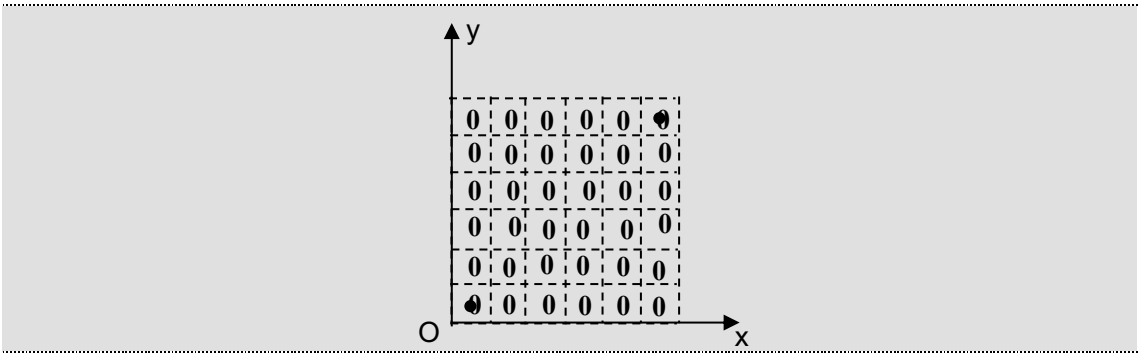
**方法 2**：

Dijkstra 分为两大部分，第一部分是取所有未标记点中的最小值，第二部分是用当前最小值改进整个图。那么建立一个上小下大的堆，堆中保存所有起始点到图中未标记的点的最短路径值，这样每次取出一个最小值的复杂度为 O(1)；由于此图中，每个点的度最多为 4，查找边的权值的复杂度为 O(N)，更新堆的复杂度为 $O(V\log_2 V)$。因而算法复杂度降为 $O(V+NV+V\log_2 V)$。但由于 V=10000*10000 仍不能在时限中出解。

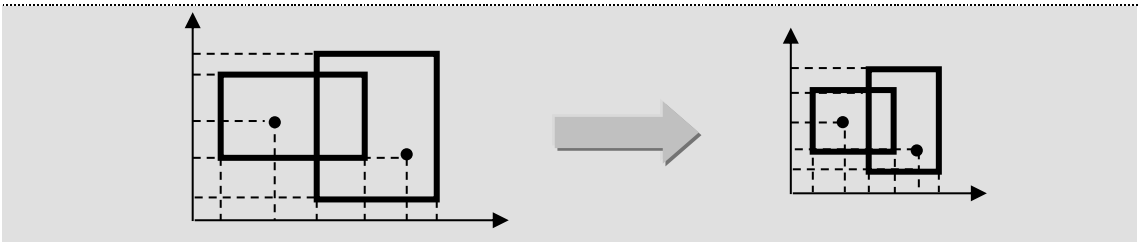
**方法 3**：

此题的数据规模有一些特性——虽然坐标系的范围巨大，但有效坐标（机器人的坐标，宝藏的坐标和磁场顶点坐标）的个数却很小。上两个方法的主要复杂度都取决于 V，也就是坐标系中的点数。如果我们可以把坐标系的范围缩小，也就相当于把 V 缩小，就可以大大降低问题的时间复杂度。

在上种方法的构图中，我们会发现很多边的权值为 0，也就是说，可能有很多连续点的最短路径值都相等。如图：



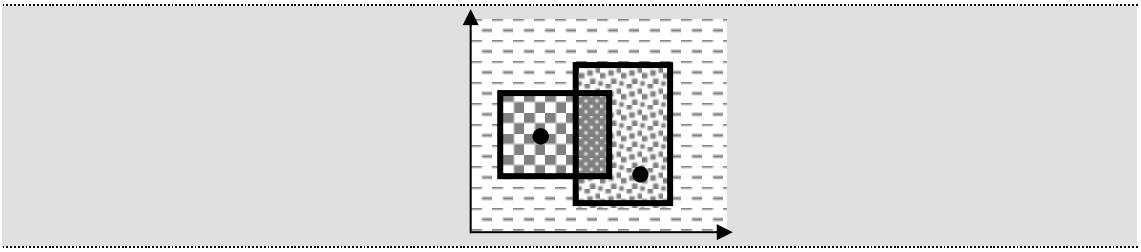

那么我们将整个图中有效坐标抽出，建立一个新的坐标系，这个坐标系中相邻两个个坐标的间距为单位“1”，但此时单位长度的意义为有效坐标的序号。如图：



这样我们依然用最短路径的方法在这个图中进行操作，算法复杂度为 $O(V+VN+V\log_2 V)$，但此时，V=204*204，问题得以解决。

**方法 4：**

离散化后对整个图中的连续无磁场部分进行染色，如下图：



问题就是求机器人所在点的颜色区域到宝藏所在点的颜色区域的最短路径。由于每相邻两个区域间的边的权值均为 1，所以算法复杂度为 O(V)。因而整个算法的复杂度为 O(NV)。这里的 N=100,V=204*204。

如果先构图，复杂度为 O(N$\sqrt{V}$ )，再染色用宽搜求最短路复杂度为 O(V)，所以总复杂度为 O(N$\sqrt{V}$ +V)。

♦ 小结：

对于同一问题，选用的单位长度的意义不同，就会有不同效果。比如 Cross 这题，开始是以题目已经建立了的坐标系的长度 1 为单位长度，到后来将有效坐标排序后，以其序号为单位长度，坐标系的范围大大缩小。这是我们常说的离散化，但也可以理解为单位长度的意义改变。我们不能局限在原有的或常用的单位长度意义，而应适宜题目做出改造或创造。

# 参考系的选择

♦ 问题引入：

选择不同的参考系来观察同一运动，观察的结果会不同。比如，马路上停着 n 辆汽车，现在有 n-1 辆汽车以相同速度向前运动，以地面为参考系，n-1 辆汽车运动，一辆汽车静止；以 n-1 辆运动的汽车为参考系，一辆汽车运动，n-1 辆汽车静止。

在不同参考系中描述物体运动，繁简是不一样的。

♦ 问题描述：**USAICO 2005 Day6　布丁(Puddin)**

FJ 建立了一个布丁工厂。在接下来的 N(1<=N<=40,000)个星期里，原料牛奶和劳动力的价格会有很大波动。FJ 希望能够在满足消费者需求的前提下尽量减小花费。

FJ 预计接下来每个星期会需要 Ci(1<=Ci<=5,000)元钱来生产一单位布丁，且消费者会需要 Pi(0<=Pi<=10,000)单位布丁。

FJ 每星期即可以生产布丁，也可以储存布丁供以后使用。它的仓库存储一星

期一单位布丁需要 S(1<=S<=100)元钱。但是由于布丁有保质期，所以最多只能保存 T(0<=T<=40,000)星期。也就是说 x 星期生产的布丁可以在(x+T)星期销售，但是不能在(x+T+1)星期销售。

请帮助 FJ 安排生产与存储的方案使得在满足消费者需求的前提下尽量减小花费。

输入文件：

第一行为 N,S 与 T.

第二行到第(N+1)行：每一行两个数，即 Ci 与 Pi。

输出文件：

仅一个数，即满足顾客需求前提下的最小花费。

样例：

输入：

```
5 10 3
12 1
21 2
27 4
45 5
52 3
```

输出：

```
488
```

♦ 问题分析：

**方法 1**：

问题是求满足每星期需求量的最小的费用 $W_{min}$，我们很容易 $W_{min}=\sum_{1}^{n} Wi$，每个星期的

最小费用是相互独立的。又因为 Wi=最小单价 Vi*Pi，Pi 已给定，那么问题就是求个个星期的最小单价 Vi。

根据 Vi=Min(Cj+S*(i-j))　(0<j<=i, j>=i-T)，我们可以求出每星期的 Vi，继而求出 Wi 和 $W_{min}$。时间复杂度为 O(NT)，由于 N,T 最大为 40000，所以不能在时限中出解。

**方法 2**：

每个星期的 Vi 都得求出，也就是 O(N)的复杂度不可少，我们试图将求解 Vi 的复杂度降低。我们想到用线段树、堆、平衡树等数据结构进行优化，就 Vi 的求解公式，我们选用堆。

建立一个上小下大的最多有 T 个节点的堆，堆中保存保质期内每个星期布丁到此星期的价值。那么从第 i 星期，到 i+1 星期对堆的操作有：

1) 删除堆中超过保质期的布丁价值
2) 将堆中所有布丁的价值加上 S
3) 插入 Ci
4) 取出堆顶作为 Vi

上面这四步操作，第 3，4 都是堆的基本操作，复杂度分别为 $O(\log_2 T)$，O(1)；第 1 步我们可以通过数组记录位置在 $O(\log_2 T)$的复杂度内解决；但第 2 步就不容易解决了。

那么是否就不可以用堆进行优化呢？

**方法 3**：

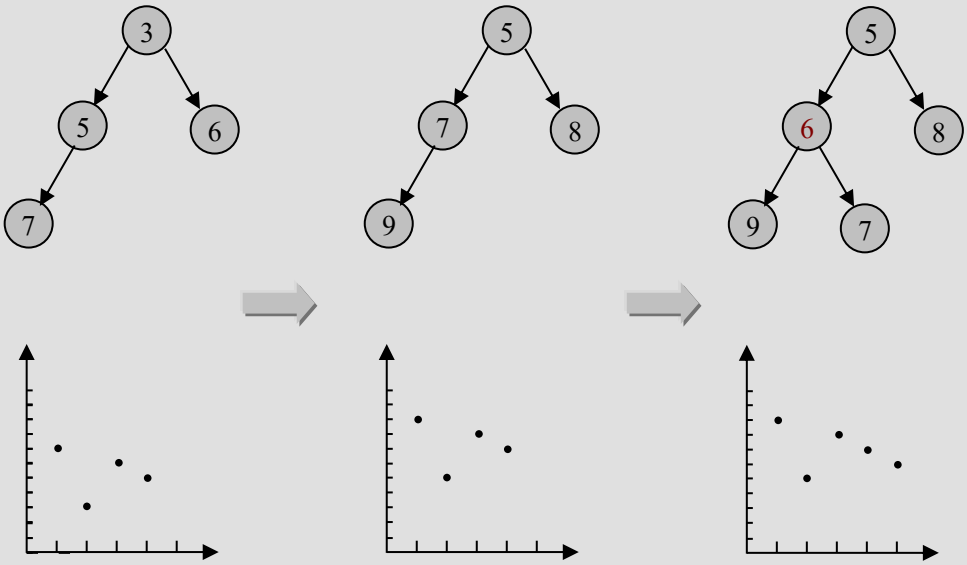
我们建立堆的目的就是求出所有保质期内的布丁在此星期时的最小单价，那么只要知道是第几星期成产的布丁此星期中价值最小便可。上种方法是完全按照题目意思进行的，那么我们不妨改变一下。我们将

2) 将堆中所有布丁的价值加上 S
3) 插入 Ci

这两步用图表示，如下：

**S=2**

**Ci=6**



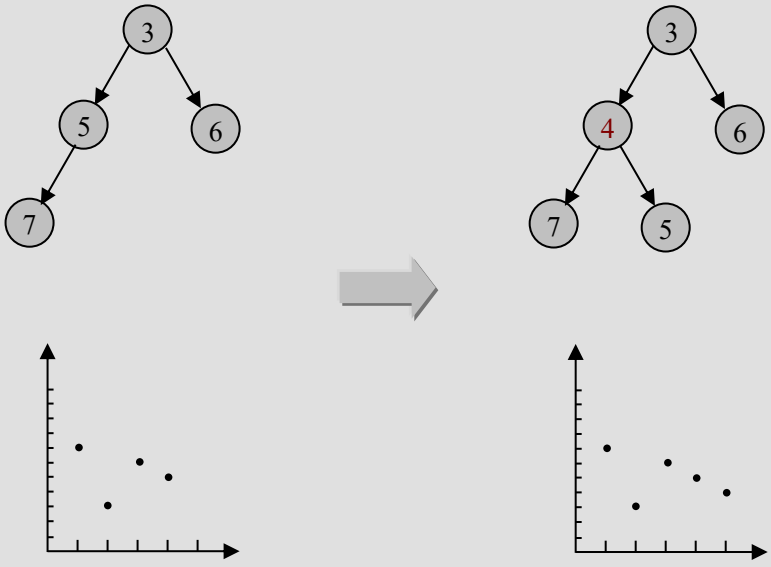

我们回想一下上面问题引入中汽车的例子，n-1 个物体运动；n-1 个物体为参照物，也就是 1 个静止的物体做相反运动了。这题也是一样，堆中所有的价值增加 S；以这些增加的值为参照物，也就相当于新插入的 Ci 减少 S。

这样一来，每个星期对布丁的操作就是：

1) 删除堆中超过保质期的布丁价值
2) 插入 Ci-S*(i-1)
3) 取出堆顶

**S=2**

**i=6**




问题复杂度降为 $O(N\log_2 T)$。

**方法 4：**

现在问题划归为每次插入 Ci-S(i-1)，求保值期 T 内的最小费用。这样，我们不妨用队列来解决。维持一个递增的长度最长为 T 的队列，步骤为：

1) 删除队列中超过保质期的布丁单价
2) 插入新的布丁单价 Ci-S(i-1)
3) 从队头删除所有大于 Ci-S(i-1)的布丁单价

由于，第 1，2 步复杂度为 O(1)，第三步平坦复杂度为 O(1)，所以整个算法的复杂度为 O(N)。

## ♦ 小结：

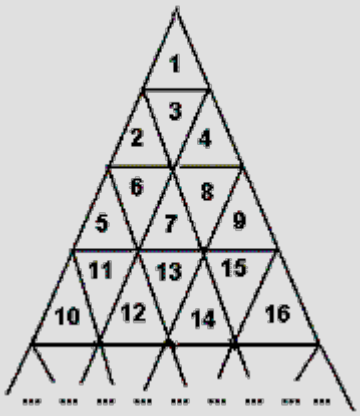
对于只要比较，不需求具体值或具体值很方便求出的问题，我们所建立的模型，可以换用非常规参考系。就像本题，原先模型中都是对真实值进行处理，不易解决；但换用已有的需改变的值为参考系后，所有的价值不是真实值，但价值的相对大小依旧，问题也就迎刃而解。所以对于这类问题，我们要开拓思维，利用参考系思想，将数值进行改造，保持其相对大小，方便解决问题。

# 坐标系的建立

♦ 问题引入：

一般来说几维坐标系就有几个非平行的原点重合的坐标轴。平面中，任意一个向量都可以用两个非平行的向量合成，所以二维坐标系只需要两个数轴，就足以表示此平面中的所有点了，如果多加一个坐标轴则是冗杂。

♦ 问题描述：**Delta-wave**




一个三角形的田野用递增的正整数按照图中方式进行编号。

一位旅行者，要从编号为 M 的格子到编号为 N 的格子(1<=N,M<=1000000000)。旅行者只能通过穿过边进入一个新格子，而不能通过点在格子中旅行。旅行者穿过的边数为旅行者的路程长度。求从 N 到 M 的最小路程长度。

输入文件：

一行包含 N,M

输出文件：

仅一个数，即最小路程长度。

样例：

输入：

```
6 12
```

输出：

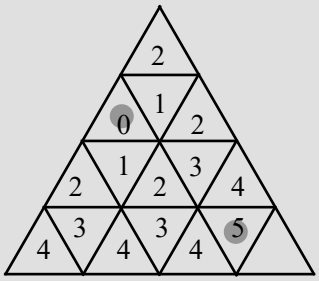
```
3
```

♦ 问题分析：

**方法 1**：

把每个小三角形视为顶点，在一个小三角形与相邻小三角形间加一条无向边，边的权值为 1。那么从起始点到目标点所要穿越的最少边数就等于构造的图中两点间的最短路径。

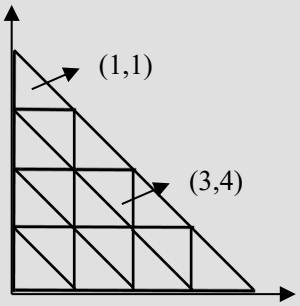
由于此图中所有边的权值均为 1，所以我们可用“灌水法”，即从起始点开始宽搜，最短路径就是目标点的层数。




算法复杂度为 O(N)，由于 N 最大时=1000000000，所以不能在规定时限中出解。

**方法 2**：

用题目中的编号来描述这个图形，显然不太方便，我们用常用的行列来表示一个小三角形的位置。这也就相当于把图形给拉直了，放入坐标系中，如下图：

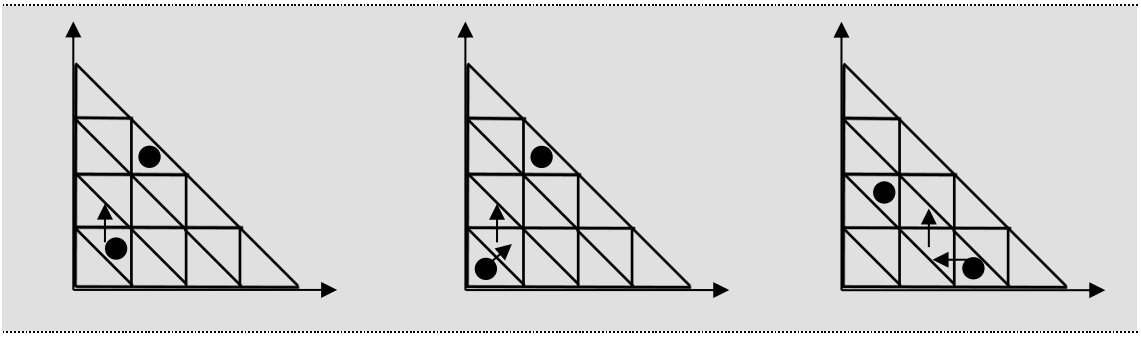



起始点的移动过程，就是不断逼近（两点之间所要穿过的边数越来越少）的过程。那么行、列坐标都要不断地逼近。但行坐标的改变不会影响列坐标；相反，列坐标的改变也不会影响行坐标。所以，我们可以先使行坐标逼近，再使列坐标逼近。不妨令目标坐标的编号小于起始坐标。

1 同行中：两个三角形所要经过的最少边数就等于列坐标的差值。

2 非同行时：

1) 行坐标偶数：直接移到临近目标三角形的一行。穿过边数为 1。
2) 行坐标奇数：必须移到相邻的偶数坐标，再移到临近目标三角形的一行。列坐标也有所改变，我们向更加逼近目标三角形列坐标的方向移动，如果列坐标相同，我们随意移到相邻的偶数坐标。穿过边数为 2。

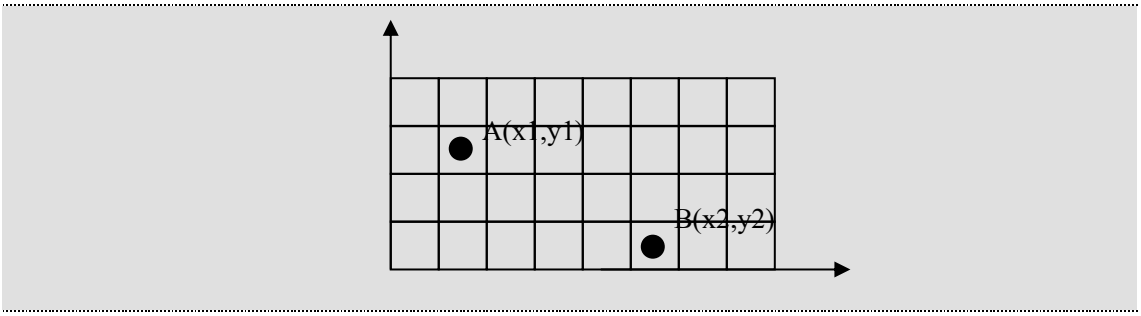


这样模拟点的运行轨迹，要经过|M-N|行，算法复杂度为 O($\sqrt{N}$)，$\sqrt{N}$ 最大为 =10000*$\sqrt{10}$，已可以在时限中出解。

那么是否有更好的方法呢？

**方法 3**：

我们先看一个与此题类似的简单题目：



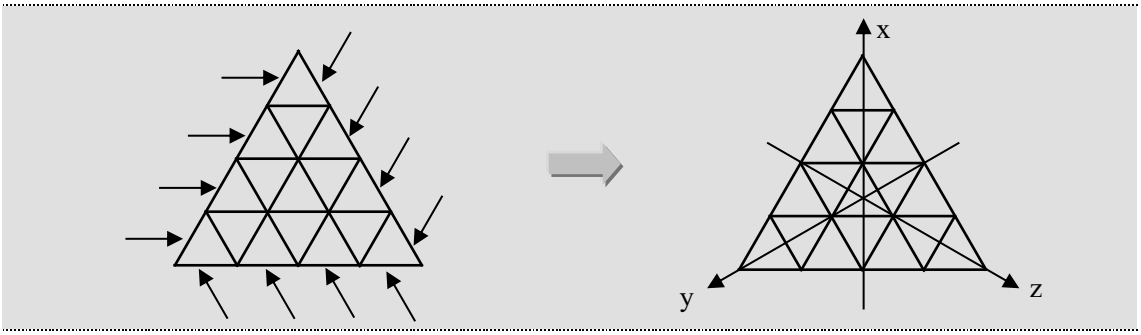

一个 m*n 的矩阵中，求 A 格到 B 格所要穿过的最少边数。

这个问题的答案似乎十分显然，我们给把矩阵放在直角坐标系中，这样每个格子都有一个坐标值(x,y)，如果 A(x1,y1)，B(x2,y2)，那么从 A 格到 B 格的最少穿过边的个数为 |x1-x2|+|y1-y2|。

道理和方法 2 的相类，行列都需不断逼近，且一个坐标的逼近不会影响另一个坐标，那么不妨先使一个坐标相等，再改变另一个坐标。

但方法 2 却不能直接出解，原因在于对列的奇偶坐标操作不同。这是否说明我们的坐标系建立的不完备呢？

我们观察原始三角形，会发现有三个方向，如图：




建立一个有三个轴的平面坐标系，用(x,y,z)来表示一个点的坐标，那么从 A(x1,y1,z1)到 B(x2,y2,z2)穿过的最少边数是否就等于|x1-x2|+|y1-y2|+|z1-z2|呢？

下面给出证明：

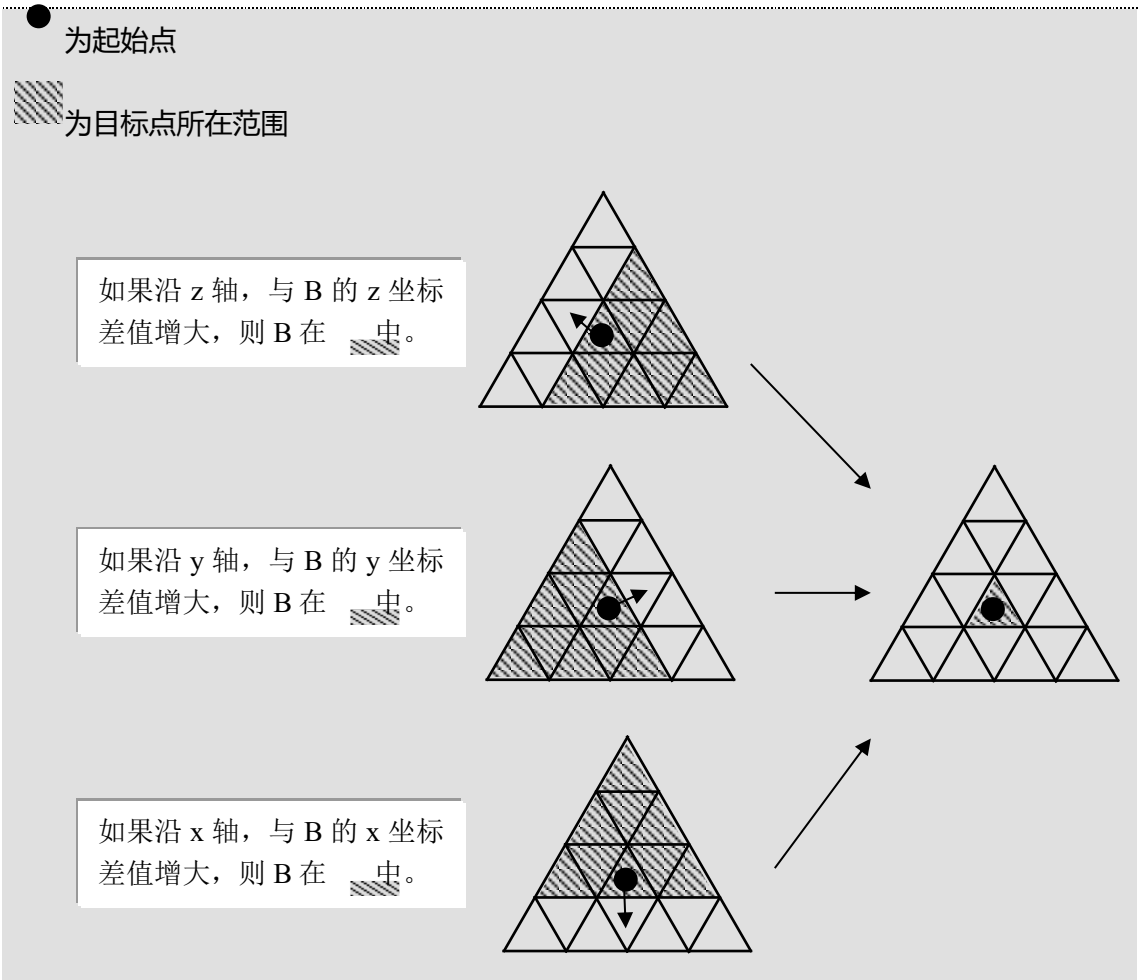
命题 1： 从 A 到 B 穿过最少边数的下限为|x1-x2|+|y1-y2|+|z1-z2|。

证明：x,y,z 坐标都是相互独立的，一个坐标的改变不会影响另一个坐标，所以命题 1 成立。

命题 2：存在从 A 到 B 穿过边数为|x1-x2|+|y1-y2|+|z1-z2|的方案。

证明：如果任意点 A(A≠B)都能向某个方向走缩小与 B 的坐标差值，就可以找到一种方案。下面用反证法证明：

假设：存在 A 点(A≠B)无论向哪个方向走，与 B 点的坐标差值都不会减少。




与假设 A≠B 矛盾！

结论：从 A(x1,y1,z1)到 B(x2,y2,z2)穿过的最少边数=|x1-x2|+|y1-y2|+|z1-z2|。

由于题目中给的是 A,B 的编号 N,M，我们只要将 N,M 转化为坐标值(x,y,z)即可。顶上的三角形坐标为(1,1,1)，编号为 N 的点的坐标值为

$x=\left\lfloor\sqrt{N-1}\right\rfloor+1$

$y=\left\lfloor\frac{N-(x-1)^2+1}{2}\right\rfloor$

$z=\left\lfloor\frac{x^2-N}{2}\right\rfloor+1$

所以算法复杂度为 O(1)。

## ♦ 小结：

本题从不用坐标系，到使用常规坐标系，再到根据题目建立特殊坐标系，问题不断明朗，复杂度不断降低。问题引入中说一般情况几维坐标系就有一个数轴，但此题的方法 3 却建立了有三个轴的二维坐标系，这种形式上的“冗杂”，换来的是思维上的简洁。所以我们要打破思维枷锁，不能只限于“常规”，而应根据“常规”继续发散。

我们分析问题时应细致认真，抓住问题的矛盾所在，继续分析。比如本题中就是由于发现方法 2 中奇偶列不能统一，才引发了方法 3。

# 总结

上面的三题看似毫无关联，因为解题思维不尽相同。但其实三题中都一个重要的灯塔，也是这三题的突破口，那就是“参考系与坐标系思想”。

虽然我们开始分析时，毫不知参考系与坐标系思想，凭着经验走出了道路。但走完这些道路后，我们再悉心回首，会发现这些道路中都有着相通点。所以在走出条道路后，就点亮道路中的灯塔，使以后的分析更加明朗。

# 参考文献与资料来源

①吴文虎 王建德，实用算法分析与程序设计，电子工业出版社，1998.1
②张维善，普通高中课程标准实验教科书 物理 必修 1，人民教育出版社，2004.5
③刘绍学，普通高中课程标准实验教科书 数学 必修 2，人民教育出版社，2004.5
④林群，义务教育课程标准实验教科书 数学 七年级下册，人民教育出版社，2004.6

# 附录

```
//Cross 1
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
const char *inname="cross.in";
const char *outname="cross.out";
const int MaxN=10010;
const int Max=110;
const int Infi=100000000;

struct node
{
	int x,y,c;
};

FILE *fin,*fout;
int total,min[MaxN][MaxN],maxx,maxy,sx,sy,tx,ty,minx,miny;
bool mark[MaxN][MaxN];
node rec[Max];

void init();
void print();
void work();
void goy(int x,int y,int v);
void gox(int x,int y,int v);

void init()
{
	int i;
	fin=fopen(inname,"r");
	fscanf(fin,"%d",&total);
	for (i=0;i<total;i++)
	{
		fscanf(fin,"%d%d%d",&rec[i].x,&rec[i].y,&rec[i].c);
		if (rec[i].x-1<minx) minx=rec[i].x-1;
		if (rec[i].y-1<miny) miny=rec[i].y-1;
		if (rec[i].x+rec[i].c+1>maxx) maxx=rec[i].x+rec[i].c+1;
		if (rec[i].y+rec[i].c+1>maxy) maxy=rec[i].y+rec[i].c+1;
	}
	fscanf(fin,"%d%d%d%d",&sx,&sy,&tx,&ty);
	if (sx>maxx) maxx=sx;
```

```
        if (tx>maxx) maxx=tx;
        if (sy>maxy) maxy=sy;
        if (ty>maxy) maxy=ty;

        if (sx<minx) minx=sx;
        if (tx<minx) minx=tx;
        if (sy<miny) miny=sy;
        if (ty<miny) miny=ty;

        if (minx<0) minx=0;
        if (miny<0) miny=0;

        fclose(fin);
}

void print()
{
        fout=fopen(outname,"w");
        fprintf(fout,"%d\n",min[tx][ty]);
        fclose(fout);
}

void work()
{
        int i,j,m,n,v;
        for (i=minx;i<=maxx;i++)
                for (j=miny;j<=maxy;j++) min[i][j]=Infi;
        min[sx][sy]=0;

        for (;;)
        {
                for (v=Infi,i=minx;i<=maxx;i++)
                        for (j=miny;j<=maxy;j++)
                                if (!mark[i][j] && min[i][j]<v)
                                {
                                        v=min[i][j];
                                        m=i;
                                        n=j;
                                }
                if (m==tx && n==ty) return ;
                mark[m][n]=1;
                gox(m+1,n,min[m][n]);
                gox(m-1,n,min[m][n]);
                goy(m,n+1,min[m][n]);
```

```
            goy(m,n-1,min[m][n]);
        }
}

void gox(int x,int y,int v)
{
        int i;
        if (x>maxx || x<minx || y>maxy || y<miny || v>=min[x][y] || mark[x][y]) return ;
        for (i=0;i<total;i++)
            if ((y==rec[i].y || y==rec[i].y+rec[i].c)
                && x>=rec[i].x && x<=rec[i].x+rec[i].c) return ;
        for (i=0;i<total;i++)
            if ((rec[i].x==x || rec[i].x+rec[i].c==x)
                && y>=rec[i].y && y<=rec[i].y+rec[i].c)
                {
                    if (v+1<min[x][y]) min[x][y]=v+1;
                    return;
                }
        if (v<min[x][y]) min[x][y]=v;
}

void goy(int x,int y,int v)
{
        int i;
        if (x>maxx || x<minx || y>maxy || y<miny || v>=min[x][y] || mark[x][y]) return ;
        for (i=0;i<total;i++)
            if ((x==rec[i].x || x==rec[i].x+rec[i].c)
                && y>=rec[i].y && y<=rec[i].y+rec[i].c) return ;
        for (i=0;i<total;i++)
            if ((rec[i].y==y || rec[i].y+rec[i].c==y)
                && x>=rec[i].x && x<=rec[i].x+rec[i].c)
                {
                    if (v+1<min[x][y]) min[x][y]=v+1;
                    return;
                }
        if (v<min[x][y]) min[x][y]=v;
}

int main()
{
       init();
       work();
       print();
       return 0;
```

```
}
```

```
//Cross 2
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
const char *inname="cross.in";
const char *outname="cross.out";
const int MaxN=10010;
const int Max=110;
const int Infi=100000000;

struct node
{
	int x,y,c;
};

struct node1
{
	short int x,y;
	int v;
};

FILE *fin,*fout;
int total,maxx,maxy,sx,sy,tx,ty,minx,miny,point[MaxN][MaxN],tt;
bool mark[MaxN][MaxN];
node rec[Max];
node1 heap[MaxN*MaxN];

void init();
void print();
void work();
void goy(int x,int y,int v);
void gox(int x,int y,int v);
void swap(node1 &a,node1 &b);
void down(int v);
void up(int v);

void init()
{
	int i;
	fin=fopen(inname,"r");
	fscanf(fin,"%d",&total);
	for (i=0;i<total;i++)
```

```
        {
            fscanf(fin,"%d%d%d",&rec[i].x,&rec[i].y,&rec[i].c);
            if (rec[i].x-1<minx) minx=rec[i].x-1;
            if (rec[i].y-1<miny) miny=rec[i].y-1;
            if (rec[i].x+rec[i].c+1>maxx) maxx=rec[i].x+rec[i].c+1;
            if (rec[i].y+rec[i].c+1>maxy) maxy=rec[i].y+rec[i].c+1;
        }
        fscanf(fin,"%d%d%d%d",&sx,&sy,&tx,&ty);
        if (sx>maxx) maxx=sx;
        if (tx>maxx) maxx=tx;
        if (sy>maxy) maxy=sy;
        if (ty>maxy) maxy=ty;

        if (sx<minx) minx=sx;
        if (tx<minx) minx=tx;
        if (sy<miny) miny=sy;
        if (ty<miny) miny=ty;

        if (minx<0) minx=0;
        if (miny<0) miny=0;

        fclose(fin);
}

void print()
{
        fprintf(fout,"%d\n",heap[point[tx][ty]].v);
        fclose(fout);
}

void work()
{
        int i,j,m,n,v,cc;
        for (i=minx;i<=maxx;i++)
            for (j=miny;j<=maxy;j++)
            {
                heap[++tt].v=Infi;
                heap[tt].x=i;
                heap[tt].y=j;
                point[i][j]=tt;
            }
        heap[point[sx][sy]].v=0;
        swap(heap[point[sx][sy]],heap[1]);
```

```
        for (;;)
        {
                v=heap[1].v;
                m=heap[1].x;
                n=heap[1].y;
                if (m==tx && n==ty) return ;

                swap(heap[1],heap[tt--]);
                down(1);
                mark[m][n]=1;

                gox(m+1,n,v);
                gox(m-1,n,v);
                goy(m,n+1,v);
                goy(m,n-1,v);
        }
}

void goy(int x,int y,int v)
{
        int i,h;
        h=point[x][y];
        if (x>maxx || x<minx || y>maxy || y<miny || mark[x][y] || v>=heap[h].v) return ;

        for (i=0;i<total;i++)
                if ((x==rec[i].x || x==rec[i].x+rec[i].c) && y>=rec[i].y && y<=rec[i].y+rec[i].c)
return ;

        for (i=0;i<total;i++)
                if ((rec[i].y==y || rec[i].y+rec[i].c==y) && x>=rec[i].x && x<=rec[i].x+rec[i].c)
                {
                        if (v+1<heap[h].v)
                        {
                                heap[h].v=v+1;
                                up(h);
                        }
                        return;
                }
        heap[h].v=v;
        up(h);
}

void gox(int x,int y,int v)
{
```

```
        int i,h;
        h=point[x][y];
        if (x>maxx || x<minx || y>maxy || y<miny || mark[x][y] || v>=heap[h].v) return ;

        for (i=0;i<total;i++)
            if ((y==rec[i].y || y==rec[i].y+rec[i].c) && x>=rec[i].x && x<=rec[i].x+rec[i].c)
return ;

        for (i=0;i<total;i++)
            if ((rec[i].x==x || rec[i].x+rec[i].c==x) && y>=rec[i].y && y<=rec[i].y+rec[i].c)
            {
                if (v+1<heap[h].v)
                {
                    heap[h].v=v+1;
                    up(h);
                }
                return;
            }
        heap[h].v=v;
        up(h);
}

void down(int v)
{
        int k;
        if (v*2>tt) return;
        if (v*2+1>tt || heap[v*2].v<heap[v*2+1].v) k=v*2; else k=v*2+1;
        if (heap[v].v<heap[k].v) return ;
        swap(heap[v],heap[k]);
        down(k);
}

void up(int v)
{
        if (v==1 || heap[v].v>=heap[v/2].v) return ;
        swap(heap[v],heap[v/2]);
        up(v/2);
}

void swap(node1 &a,node1 &b)
{
        node1 temp;
        int temp1;
        temp=a;
```

```
        a=b;
        b=temp;

        temp1=point[a.x][a.y];
        point[a.x][a.y]=point[b.x][b.y];
        point[b.x][b.y]=temp1;
}

int main()
{
    init();
    fout=fopen(outname,"w");
    work();
    print();
    return 0;
}
```

```
//Cross 3
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
const char *inname="cross.in";
const char *outname="cross.out";
const int Max=210;
const int Infi=100000000;

struct node
{
    int x,y,c,xm,ym;
};

struct node1
{
    short int x,y;
    int v;
};

FILE *fin,*fout;
int total,totx,toty,sx,sy,tx,ty,point[Max][Max],tt,xaxis[Max],yaxis[Max];
bool mark[Max][Max];
node rec[Max];
node1 heap[Max*Max];

void init();
```

```
void print();
void work();
void goy(int x,int y,int v);
void gox(int x,int y,int v);
void swap(node1 &a,node1 &b);
void down(int v);
void up(int v);
int getx(int x);
int gety(int y);
void insertx(int x);
void inserty(int y);
void insert(int x,int y);

void init()
{
	int i,j;
	fin=fopen(inname,"r");
	fscanf(fin,"%d",&total);
	insert(0,0);
	insert(10001,10001);
	for (i=0;i<total;i++)
	{
		fscanf(fin,"%d%d%d",&rec[i].x,&rec[i].y,&rec[i].c);
		insert(rec[i].x,rec[i].y);
		insert(rec[i].x+rec[i].c,rec[i].y+rec[i].c);
	}
	fscanf(fin,"%d%d%d%d",&sx,&sy,&tx,&ty);
	insert(sx,sy);
	insert(tx,ty);

	for (i=0;i<total;i++)
	{
		rec[i].xm=getx(rec[i].c+rec[i].x);
		rec[i].x=getx(rec[i].x);
		rec[i].ym=gety(rec[i].c+rec[i].y);
		rec[i].y=gety(rec[i].y);
	}
	sx=getx(sx);
	sy=gety(sy);
	tx=getx(tx);
	ty=gety(ty);
	fclose(fin);
}
```

```
int getx(int x)
{
	for (int i=0;i<totx;i++) if (x==xaxis[i]) return i;
}

int gety(int y)
{
	for (int i=0;i<toty;i++) if (y==yaxis[i]) return i;
}

void insert(int x,int y)
{
	insertx(x);
	inserty(y);
}

void insertx(int x)
{
	int j;
	for (j=0;j<totx;j++) if (xaxis[j]==x) return;
	for (j=totx++;j>0;j--)
		if (xaxis[j-1]>x) xaxis[j]=xaxis[j-1]; else break;
	xaxis[j]=x;
}

void inserty(int y)
{
	int j;
	for (j=0;j<toty;j++) if (yaxis[j]==y) return ;
	for (j=toty++;j>0;j--)
		if (yaxis[j-1]>y) yaxis[j]=yaxis[j-1]; else break;
	yaxis[j]=y;
}

void print()
{
	fprintf(fout,"%d\n",heap[point[tx][ty]].v);
	fclose(fout);
}

void work()
{
	int i,j,v;
	for (i=0;i<totx;i++)
```

```
            for (j=0;j<toty;j++)
            {
                heap[++tt].v=Infi;
                heap[tt].x=i;
                heap[tt].y=j;
                point[i][j]=tt;
            }
        heap[point[sx][sy]].v=0;
        swap(heap[point[sx][sy]],heap[1]);

        for (;;)
        {
            v=heap[1].v;
            i=heap[1].x;
            j=heap[1].y;
            if (i==tx && j==ty) return ;

            swap(heap[1],heap[tt--]);
            down(1);
            mark[i][j]=1;

            gox(i+1,j,v);
            gox(i-1,j,v);
            goy(i,j+1,v);
            goy(i,j-1,v);
        }
}

void goy(int x,int y,int v)
{
        int i,h;
        if (x>=totx || x<0 || y>=toty || y<0) return ;

        h=point[x][y];
        if (mark[x][y] || v>=heap[h].v) return ;

        for (i=0;i<total;i++)
            if ((rec[i].y==y || rec[i].ym==y) && x>=rec[i].x && x<=rec[i].xm)
            {
                if (v+1<heap[h].v)
                {
                    heap[h].v=v+1;
                    up(h);
                }
```

```
                return;
            }
        heap[h].v=v;
        up(h);
}

void gox(int x,int y,int v)
{
        int i,h;
        if (x>=totx || x<0 || y>=toty || y<0) return ;

        h=point[x][y];
        if (mark[x][y] || v>=heap[h].v) return ;

        for (i=0;i<total;i++)
            if ((rec[i].x==x || rec[i].xm==x) && y>=rec[i].y && y<=rec[i].ym)
            {
                if (v+1<heap[h].v)
                {
                    heap[h].v=v+1;
                    up(h);
                }
                return;
            }
        heap[h].v=v;
        up(h);
}

void down(int v)
{
        int k;
        if (v*2>tt) return;
        if (v*2+1>tt || heap[v*2].v<heap[v*2+1].v) k=v*2; else k=v*2+1;
        if (heap[v].v<heap[k].v) return ;
        swap(heap[v],heap[k]);
        down(k);
}

void up(int v)
{
        if (v==1 || heap[v].v>=heap[v/2].v) return ;
        swap(heap[v],heap[v/2]);
        up(v/2);
}
```

```
void swap(node1 &a,node1 &b)
{
        node1 temp;
        int temp1;
        temp=a;
        a=b;
        b=temp;

        temp1=point[a.x][a.y];
        point[a.x][a.y]=point[b.x][b.y];
        point[b.x][b.y]=temp1;
}

int main()
{
      fout=fopen(outname,"w");
      init();
      work();
      print();
      return 0;
}
```

```
//Cross 4
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
const char *inname="cross.in";
const char *outname="cross.out";
const int Max=210;

struct node
{
      int x,y,c,xm,ym;
};

struct node1
{
      int x,y;
};

FILE *fin,*fout;
int total,totx,toty,sx,sy,tx,ty,xaxis[Max],yaxis[Max],map[Max][Max],first,last;
node1 list[Max*Max];
```

```
node rec[Max];

void init();
void print();
void work();
int goy(int x,int y);
int gox(int x,int y);
void down(int v);
void up(int v);
int getx(int x);
int gety(int y);
void insertx(int x);
void inserty(int y);
void insert(int x,int y);
void dox(int x,int y,int c);
void doy(int x,int y,int c);
void dfs(int x,int y,int c);

void init()
{
	int i,j;
	fin=fopen(inname,"r");
	fscanf(fin,"%d",&total);
	insert(0,0);
	insert(10001,10001);
	for (i=0;i<total;i++)
	{
		fscanf(fin,"%d%d%d",&rec[i].x,&rec[i].y,&rec[i].c);
		insert(rec[i].x,rec[i].y);
		insert(rec[i].x+rec[i].c,rec[i].y+rec[i].c);
	}
	fscanf(fin,"%d%d%d%d",&sx,&sy,&tx,&ty);
	insert(sx,sy);
	insert(tx,ty);

	for (i=0;i<total;i++)
	{
		rec[i].xm=getx(rec[i].c+rec[i].x);
		rec[i].x=getx(rec[i].x);
		rec[i].ym=gety(rec[i].c+rec[i].y);
		rec[i].y=gety(rec[i].y);
	}
	sx=getx(sx);
	sy=gety(sy);
```

```
        tx=getx(tx);
        ty=gety(ty);
        fclose(fin);
}

int getx(int x)
{
        for (int i=0;i<totx;i++) if (x==xaxis[i]) return i;
}

int gety(int y)
{
        for (int i=0;i<toty;i++) if (y==yaxis[i]) return i;
}

void insert(int x,int y)
{
        insertx(x);
        inserty(y);
}

void insertx(int x)
{
        int j;
        for (j=0;j<totx;j++) if (xaxis[j]==x) return;
        for (j=totx++;j>0;j--)
              if (xaxis[j-1]>x) xaxis[j]=xaxis[j-1]; else break;
        xaxis[j]=x;
}

void inserty(int y)
{
        int j;
        for (j=0;j<toty;j++) if (yaxis[j]==y) return ;
        for (j=toty++;j>0;j--)
              if (yaxis[j-1]>y) yaxis[j]=yaxis[j-1]; else break;
        yaxis[j]=y;
}

void print()
{
        fout=fopen(outname,"w");
        fprintf(fout,"%d\n",map[tx][ty]-1);
        fclose(fout);
```

```
}

void work()
{
        list[0].x=sx;
        list[0].y=sy;
        map[sx][sy]=1;
        for (first=0,last=1;first<last;first++)
             dfs(list[first].x,list[first].y,map[list[first].x][list[first].y]);
}

void dox(int x,int y,int c)
{
        int p;
        p=gox(x,y);
        if (p<0) return ;
        if (!map[x][y])
            if (p==0) dfs(x,y,c);
            else
            {
                    map[x][y]=c+1;
                    list[last].x=x;
                    list[last++].y=y;
            }
        else if (c+p<map[x][y]) map[x][y]=c+p;
}

void doy(int x,int y,int c)
{
        int p;
        p=goy(x,y);
        if (p<0) return ;
        if (!map[x][y])
            if (p==0) dfs(x,y,c);
            else
            {
                    map[x][y]=c+1;
                    list[last].x=x;
                    list[last++].y=y;
            }
        else if (c+p<map[x][y]) map[x][y]=c+p;
}

void dfs(int x,int y,int c)
```

```
{
        map[x][y]=c;
        dox(x+1,y,c);
        dox(x-1,y,c);
        doy(x,y+1,c);
        doy(x,y-1,c);
}

int goy(int x,int y)
{
        if (x>=totx || x<0 || y>=toty || y<0) return -1;
        for (int i=0;i<total;i++)
                if ((rec[i].y==y || rec[i].ym==y) && x>=rec[i].x && x<=rec[i].xm) return 1;
        return 0;
}

int gox(int x,int y)
{
        if (x>=totx || x<0 || y>=toty || y<0) return -1;
        for (int i=0;i<total;i++)
                if ((rec[i].x==x || rec[i].xm==x) && y>=rec[i].y && y<=rec[i].ym) return 1;
        return 0;
}

int main()
{
        init();
        work();
        print();
        return 0;
}
```

```
//Cross 5
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
const char *inname="cross.in";
const char *outname="cross.out";
const int Max=210;

struct node
{
        int x,y,c,xm,ym;
};
```

```
struct node1
{
	int x,y;
};

FILE *fin,*fout;
int total,totx,toty,sx,sy,tx,ty,xaxis[Max],yaxis[Max],map[Max][Max],first,last,dec[Max][Max];
node1 list[Max*Max];
node rec[Max];

void init();
void print();
void work();
int goy(int x,int y);
int gox(int x,int y);
void down(int v);
void up(int v);
int getx(int x);
int gety(int y);
void insertx(int x);
void inserty(int y);
void insert(int x,int y);
void dox(int x,int y,int c);
void doy(int x,int y,int c);
void dfs(int x,int y,int c);

void init()
{
	int i,j;
	fin=fopen(inname,"r");
	fscanf(fin,"%d",&total);
	insert(0,0);
	insert(10001,10001);
	for (i=0;i<total;i++)
	{
		fscanf(fin,"%d%d%d",&rec[i].x,&rec[i].y,&rec[i].c);
		insert(rec[i].x,rec[i].y);
		insert(rec[i].x+rec[i].c,rec[i].y+rec[i].c);
	}
	fscanf(fin,"%d%d%d%d",&sx,&sy,&tx,&ty);
	insert(sx,sy);
	insert(tx,ty);
```

```
        for (i=0;i<total;i++)
        {
            rec[i].xm=getx(rec[i].c+rec[i].x);
            rec[i].x=getx(rec[i].x);
            rec[i].ym=gety(rec[i].c+rec[i].y);
            rec[i].y=gety(rec[i].y);
        }
        sx=getx(sx);
        sy=gety(sy);
        tx=getx(tx);
        ty=gety(ty);
        fclose(fin);
}

void create()
{
        int i,k;
        for (k=0;k<total;k++)
        {
            for (i=rec[k].x;i<=rec[k].xm;i++) dec[i][rec[k].y]=dec[i][rec[k].ym]=1;
            for (i=rec[k].y;i<=rec[k].ym;i++) dec[rec[k].x][i]=dec[rec[k].xm][i]=1;
        }
}

int getx(int x)
{
        for (int i=0;i<totx;i++) if (x==xaxis[i]) return i;
}

int gety(int y)
{
        for (int i=0;i<toty;i++) if (y==yaxis[i]) return i;
}

void insert(int x,int y)
{
        insertx(x);
        inserty(y);
}

void insertx(int x)
{
        int j;
        for (j=0;j<totx;j++) if (xaxis[j]==x) return;
```

```
        for (j=totx++;j>0;j--)
            if (xaxis[j-1]>x) xaxis[j]=xaxis[j-1]; else break;
        xaxis[j]=x;
}

void inserty(int y)
{
        int j;
        for (j=0;j<toty;j++) if (yaxis[j]==y) return ;
        for (j=toty++;j>0;j--)
            if (yaxis[j-1]>y) yaxis[j]=yaxis[j-1]; else break;
        yaxis[j]=y;
}

void print()
{
        fout=fopen(outname,"w");
        fprintf(fout,"%d\n",map[tx][ty]-1);
        fclose(fout);
}

void work()
{
        list[0].x=sx;
        list[0].y=sy;
        map[sx][sy]=1;
        for (first=0,last=1;first<last;first++)
            dfs(list[first].x,list[first].y,map[list[first].x][list[first].y]);
}

void dox(int x,int y,int c)
{
        int p;
        p=gox(x,y);
        if (p<0) return ;
        if (!map[x][y])
            if (p==0) dfs(x,y,c);
            else
            {
                map[x][y]=c+1;
                list[last].x=x;
                list[last++].y=y;
            }
        else if (c+p<map[x][y]) map[x][y]=c+p;
```

```
}

void doy(int x,int y,int c)
{
        int p;
        p=goy(x,y);
        if (p<0) return ;
        if (!map[x][y])
            if (p==0) dfs(x,y,c);
            else
            {
                    map[x][y]=c+1;
                    list[last].x=x;
                    list[last++].y=y;
            }
        else if (c+p<map[x][y]) map[x][y]=c+p;
}

void dfs(int x,int y,int c)
{
        map[x][y]=c;
        dox(x+1,y,c);
        dox(x-1,y,c);
        doy(x,y+1,c);
        doy(x,y-1,c);
}

int goy(int x,int y)
{
        if (x>=totx || x<0 || y>=toty || y<0) return -1;
        return dec[x][y];
}

int gox(int x,int y)
{
        if (x>=totx || x<0 || y>=toty || y<0) return -1;
        return dec[x][y];
}

int main()
{
       init();
       create();
       work();
```

```
    print();
    return 0;
}
```

```
//Puddin 1
#include<stdio.h>
#include<fstream.h>
#include<iostream.h>
#include<string.h>
#include<stdlib.h>
const int Max=40010;
const char *inname="puddin.in";
const char *outname="puddin.out";

FILE *fin;
int n,s,t,c[Max],p[Max];
long long ans;

void init();
void print();
void work();

void init()
{
    int i;
    fin=fopen(inname,"r");
    fscanf(fin,"%d%d%d",&n,&s,&t);
    for (i=0;i<n;i++)
        fscanf(fin,"%d%d",&c[i],&p[i]);
    fclose(fin);
}

void print()
{
    ofstream fout(outname);
    fout<<ans<<endl;
}

void work()
{
    int i,j,temp,v;
    for (i=0;i<n;i++)
    {
        for (v=-1,j=i;j>=0 && j>=i-t;j--)
```

```
            {
                temp=c[j]+s*(i-j);
                if (temp<v || v<0) v=temp;
            }
            ans+=v*p[i];
        }
}

int main()
{
    init();
    work();
    print();
    return 0;
}
```

```
//Puddin 2
#include<stdio.h>
#include<fstream.h>
#include<iostream.h>
#include<string.h>
#include<stdlib.h>

const int Max=40010;
const char *inname="puddin.in";
const char *outname="puddin.out";

struct node
{
    int val,refer;
};

FILE *fin;
int n,s,t,c[Max],p[Max],total,point[Max];
node heap[Max];
long long ans;

void init();
void print();
void work();
void swap(node &a,node &b);
void up(int v);
void down(int v);
```

```
void init()
{
	int i;
	fin=fopen(inname,"r");
	fscanf(fin,"%d%d%d",&n,&s,&t);
	for (i=0;i<n;i++)
		fscanf(fin,"%d%d",&c[i],&p[i]);
	fclose(fin);
}

void print()
{
	ofstream fout(outname);
	fout<<ans<<endl;
}

void work()
{
	int i,tt;
	for (i=0;i<n;i++)
	{
	    if (i-t-1>=0)
	    {
		swap(heap[point[i-t-1]],heap[total--]);
		up(point[i-t-1]);
		down(point[i-t-1]);
	    }

	    point[i]=++total;
	    heap[total].refer=i;
	    heap[total].val=c[i]-s*i;
	    up(total);

	    tt=heap[1].refer;
	    ans+=(c[tt]+(i-tt)*s)*p[i];
	}
}

void up(int v)
{
	if (v==1 || heap[v/2].val<=heap[v].val) return ;
	swap(heap[v],heap[v/2]);
	up(v/2);
}
```

```
void down(int v)
{
        int k;
        if (v*2>total) return ;
        if (v*2+1>total || heap[v*2].val>heap[v*2+1].val) k=v*2; else k=v*2+1;
        if (heap[v].val<heap[k].val) return ;
        swap(heap[v],heap[k]);
        down(k);
}

void swap(node &a,node &b)
{
        node temp;
        int t;

        temp=a;
        a=b;
        b=temp;

        t=point[a.refer];
        point[a.refer]=point[b.refer];
        point[b.refer]=t;
};

int main()
{
      init();
      work();
      print();
      return 0;
}
```

```
//Puddin 3
#include<stdio.h>
#include<fstream.h>
#include<iostream.h>
const char *inname="puddin.in";
const char *outname="puddin.out";
const int Max=40010;

struct node
{
      int week,value;
```

```
};
FILE *fin;
ofstream fout;
int n,s,t;
int C[Max],P[Max],head,tail;
node queue[Max];
long long sum;

void init()
{
        int i;
        fin=fopen(inname,"r");
        fscanf(fin,"%d%d%d",&n,&s,&t);
        for (i=0;i<n;i++)
                fscanf(fin,"%d%d",&C[i],&P[i]);
        fclose(fin);
}

void print()
{
        fout.open(outname);
        fout<<sum<<endl;
        fout.close();
}

void work()
{
        int i,v,w;
        for (head=-1,i=0;i<n;i++)
        {
                if (i-queue[tail].week>t) tail++;
                v=C[i]-s*i;
                for (head;head>=tail;head--)
                        if (v>queue[head].value) break;
                queue[++head].value=v;
                queue[head].week=i;
                w=queue[tail].week;
                sum+=P[i]*(C[w]+s*(i-w));
        }
}

int main()
{
      init();
```

```
    work();
    print();
    return 0;
}
```

```
//Wave 1
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

const int Max=1000000;

struct node
{
    int x,y,depth;
};

int N,M,first,last;
node list[Max];
bool mark[Max];

void init();
void print(int ans);
void bfs();
inline int code(int x,int y);
void insert(int x,int y,int d);

void init()
{
    scanf("%d%d",&N,&M);
}

void print(int ans)
{
    printf("%d\n",ans);
    exit(0);
}

void bfs()
{
    int x,y,d;

    if (N==M) print(0);
```

```
        list[0].x=int (sqrt(double (N-1)))+1;
        list[0].y=N-(list[0].x-1)*(list[0].x-1);
        list[0].depth=0;

        for (first=0,last=1;first<last;first++)
        {
            x=list[first].x;
            y=list[first].y;
            d=list[first].depth+1;

            if (y-1>0) insert(x,y-1,d);
            if (y+1<x*2) insert(x,y+1,d);
            if (y%2) insert(x+1,y+1,d);
            else insert(x-1,y-1,d);
        }
}

inline int code(int x,int y)
{
        return (x-1)*(x-1)+y;
}

void insert(int x,int y,int d)
{
        int c;
        c=code(x,y);
        if (c==M) print(d);
        if (!mark[c])
        {
            mark[c]=1;
            list[last].x=x;
            list[last].y=y;
            list[last++].depth=d;
        }
}

int main()
{
      init();
      bfs();
      return 0;
}
```

**//Wave 2**

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int N,M,ans;

void init();
void cal();
void print();
void swap(int &a,int &b);

void init()
{
	scanf("%d%d",&N,&M);
}

void print()
{
	printf("%d\n",ans);
}

void cal()
{
	int x1,y1,x2,y2;
	if (N<M) swap(N,M);
	x1=int (sqrt(double (N-1)))+1;
	y1=N-(x1-1)*(x1-1);
	x2=int (sqrt(double (M-1)))+1;
	y2=M-(x2-1)*(x2-1);
	for (;x1>x2;x1--,ans++)
		if (y1%2==0) y1--;
		else
		{
			ans++;
			if (y2<y1) y1-=2;
		}
	ans+=abs(y1-y2);
}

void swap(int &a,int &b)
{
	int temp;
	temp=a;
	a=b;
```

```
        b=temp;
}
int main()
{
        init();
        cal();
        print();
        return 0;
}
```

```
//Wave 3
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int N,M,ans;

void init();
void cal();
void work();

void init()
{
        scanf("%d%d",&N,&M);
}

void print()
{
        printf("%d\n",ans);
}

void cal()
{
        int x1,y1,z1,x2,y2,z2;
        x1=int (sqrt(double (N-1))+1);
        y1=(N-(x1-1)*(x1-1)+1)/2;
        z1=(x1*x1-N)/2+1;
        x2=int (sqrt(double (M-1)))+1;
        y2=(M-(x2-1)*(x2-1)+1)/2;
        z2=(x2*x2-M)/2+1;
        ans=abs(x1-x2)+abs(y1-y2)+abs(z1-z2);
}

int main()
```

```
{
    init();
    cal();
    print();
    return 0;
}
```

# 致谢

感谢江涛老师在论文准备期间给予的帮助。
感谢朱晨光、周冬同学提出的宝贵意见与建议。