目录

[主要思路 2](#_Toc87476229)

[主要算法思想 2](#_Toc87476230)

[主要算法流程 5](#_Toc87476231)

[主要输入 6](#_Toc87476232)

[主要输出（暂定，不重要） 6](#_Toc87476233)

[详细描述 7](#_Toc87476234)

[图纸网格化处理 7](#_Toc87476235)

[转弯最少约束的处理 8](#_Toc87476236)

[房间预处理 9](#_Toc87476237)

[终点预处理 10](#_Toc87476238)

[不同坐标的处理 10](#_Toc87476239)

[风管的特殊处理 11](#_Toc87476240)

[重难点 12](#_Toc87476241)

## 主要思路

### 主要算法思想

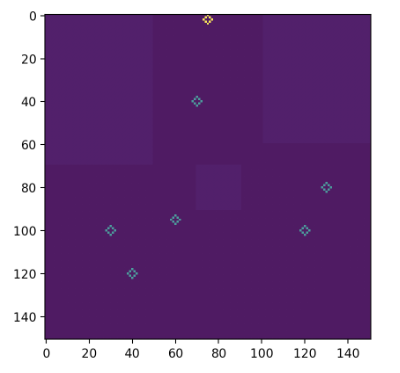
为了简化计算过程以及方便进行管道铺设，在第一版的设计中，我们暂时将管道的铺设方向默认为沿着当前坐标系的X轴与Y轴方向（横屏竖直）。

之后我们将CAD图像离散化成正方形网格，网格应当具有一种最基本的属性来表示它的内部存不存在障碍物（能通过/不能通过），每个网格与**上下左右**四个方向的网格相连接，将整个CAD图像连接构成了一张无向无权图（也可以默认权值为1）。

由于管道铺设保持横平竖直，容易得出结论：铺设路线经过的边的条数（等于经过的网格数量-1）等价于管道铺设路线长度。

也就是，求最短管道铺设长度的问题转化为：

我们想要找到一种连接方案，使得管道能够连接起点和终点的同时，经过的图的边数最少。



我们有很多算法能解决这个问题，目前我觉得比较可行的方案有如下两种：

* **对于节点个数较少的情况：**

我们可以用直接用**最小斯坦纳树算法**求出最优解。（算法细节以及伪代码见网络）

这个算法的优点是可以直接求出最优解，缺点是运算复杂度高度依赖于终点的个数。

* **对于节点个数较多的情况：**

**备注：目前打算先采取这种近优方案，之后再考虑优化。**

我们可以采取庄同学的技术文档中的**distance-map算法**（本质上是BFS算法的应用）。

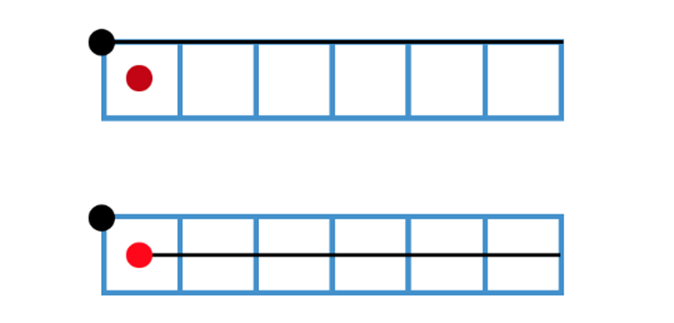
其核心思想是将互相之间距离最近的终点逐步连接成一个整体，慢慢向起点靠近，这样可以得出一个近优解。（算法细节以及伪代码见庄同学的技术文档）

这个算法的优点是逻辑清晰，运算速度快，缺点是得出的解不一定是最优解。

到此为止的输出是一系列的网格，只要将管道铺设在这些网格之中，就能保证管道铺设的效果接近最优。

之后需要做两件事：

* 将网格转化为具体的线条，保证线条在网格内的同时和起点终点无缝连接，（这一步应该会相对复杂，到具体实现的时候可能还得斟酌一下）。



* 对图转化成具体的管道树结构，来计算每一段管道的流量，最后将树的结构和对应的管道流量作为最终输出。（这一步实现起来应该相对简单）。

在上面的铺设过程中，有许多约束条件的实现没有细讲，对于约束条件的处理会放在“具体细节”这一部分逐一进行讨论。

### 主要算法流程

1. 收集数据，对图像进行网格切分。

对各类不同的终点（尤其是包含在房间内的终点）进行预处理操作（预处理见详细描述）。

1. 在较粗糙的网格切分之下，利用斯坦纳最小树、distance-map等算法（目前暂时打算先用后者做出一个近优版本，再试试看前者能不能有所改进）

得到粗糙网格下管道铺设的近优解（相当于框定了该近优解的范围）。

1. 将框定的最优解（网格）转化为实际的线条，输出的管道结构树。

### 主要输入

* 水管，风管的起点，终点
* 整张图的轮廓，不同障碍物的框线（房间框线要单独列出来）
* 某块区域的分界线和方向向量（如果有）

### 主要输出（暂定，不重要）

管道树结构

起点，终点，每一个道路分岔口都是节点。

每一个节点包含一组data ，一个List。

**Data：**

* 该节点下属所有管道流量
* 树的层级

**List：**

List内的每一个元素（struct）代表一条支路，每一个元素包含了：

* 该支路的行进方式 **点+点+点**，点点之间直线连接（x0，y0）-（x1-y1）……
* 该条支路的终点（下一个节点）的数据存储地址（指针？）

## 详细描述

### 图纸网格化处理

图纸网格化处理需要注意两个问题：

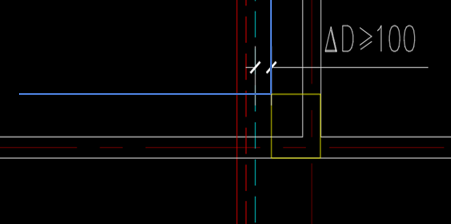
* 正方形网格的大小？

待各位工程师老师判断。

* 障碍物的边缘的网格怎么分类（归类到能通过里面还是不能通过里面）？

目前考虑的解决方案是扩大障碍物的覆盖范围，可以理解为将墙面变厚（边缘+100mm，图中的蓝色线条）。

只要网格的中心点不落在障碍物的覆盖范围内，就认为这个网格可以通过。



### 转弯最少约束的处理

两种情况：

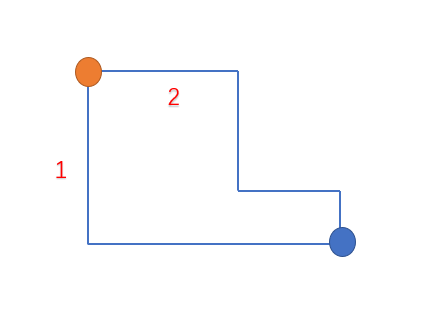
【1】少障碍物或者没障碍物的情况（大部分情境下）。

考虑到算法结构，如果在进行移动的同时（无论是BFS或是其他图相关的算法），只要移动的顺序（上下左右）考虑的优先级不变。感觉上，天然就实现了不会出现多余的转弯。

因为算法的逻辑是，如果一直往这个方向移动能得到最短路，则会一直往这个方向移动直到需要转弯为止。

（如图所示，虽然路径1和路径2都是最短路，但是只要满足在遍历每一条边时，“下上左右”的优先级保持不变，路径2永远不会是第一个出现的最短路，而算法设计中，只要记录下找到的第一个最短路的边，即可保证不会出现多余的转弯。）

不会出现多余的转弯，则代表着转弯数量最小。



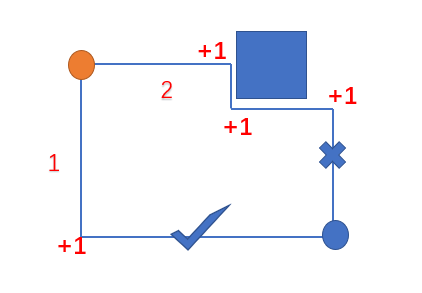
【2】有特殊障碍物的情况（小部分情景下）。

考虑在使用**distance-map** 的过程中（在BFS计算最短距离的过程中），为每一次转弯添加惩罚项。

具体做法是，假设当前出队的节点层级为X，那么新找到的节点所对应的层级理应为X+1（层级就代表着最短路的长度），且理应立刻入队。

但是我们不让他立刻入队，而是在X+1上施加惩罚，使其变为X+2，等到x层级的点全部完成出队，X+1层节点全部完成入队，X+1层节点开始出队时，将这类伪X+2其入队。

这样就能够保证在搜索到目标点时，同时满足最短路和最少转弯的条件，不过在实现上可能会相对复杂一点，也有可能会出现一些问题。



当然，使用传统的，能够加入第二指标的最短路算法(例如堆优化的Dijkstra)，当然也能解决这个问题，不过可能由于网格太多，时间复杂度会爆炸。

### 房间预处理

【设备分组】和【直线穿过房间墙】的约束可以一起讨论。

房间的预处理过程，需要经过以下几个步骤：

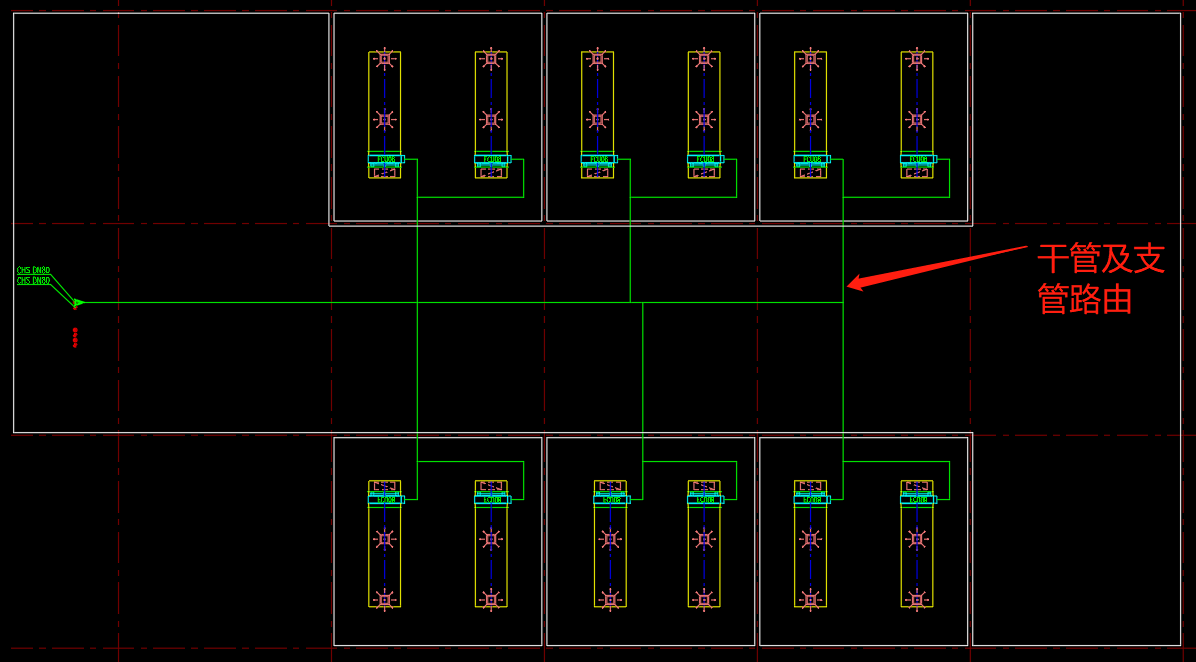
* 找出房间几面墙中，距离房间内终点的总距离最小的那一面墙（最优墙）
* 计算distance-map，得出每个房间内每个方格的distance-map数据

按照以下准则选出最优点（1-3重要性依次降低）

1. 优先 distance-map最小
2. 优先 离最优墙最近
3. 优先 离总的起点最近
4. 到此为止，评价指标依旧相同的话，任取一点作为最优点

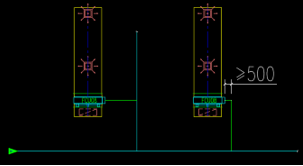
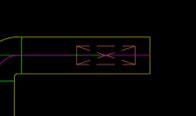
* 最优点朝着最优墙，连一条垂直穿出房间的线，穿出大概100mm后停止，作为整个房间的给水点。

到此为止解决了【设备分组】和【直线穿过房间墙】的约束。



### 终点预处理

绝大多数关于终点的约束，都可以通过对终点进行预处理，然后将预处理后的点作为算法中计算的终点得到解决。



举例：

只要将终点往外沿一段距离（500mm），将沿伸出的新点作为新的终点，这两个约束条件就消失了。

### 不同坐标的处理

如果能够获得已经划分好的坐标区域和该坐标区域的方向向量，则该问题能够按照庄同学在技术文档中写的方法得到解决。

如果不能，则需要考虑进行区域的划分和方向向量的选取。

这是所有布线过程都会遇到的问题，可能需要统一讨论得出方案。

### 风管的特殊处理

在风管铺设时，要求房间内的管道分布具有对称性，房间外的风管分布尽量具有对称性，

目前有以下解决思路：

【1】对于机器数量不多的房间，可以直接采用BFS计算到每台机器距离相同的点，并将每台机器都连到这个点上，最后连出房间。

【2】对于机器数较多的房间，每次找到距离最近的两个终点，连接，然后取中点，作为新的节点继续进行计算。

【2】存在一个问题，就是很难保证当前要连接的两个节点下游的拓扑结构相同。如果下游的拓扑结构不同，上游局部的管道长度相同则失去意义。

（如果用一颗树来存储这个节点下面的拓扑结构，每次连接时判断这两颗树是否相同，应该可行的，但是代码实现会比较复杂）。

## 重难点

目前看来的重难点有

* 不同坐标区域的划分。
* 风管的对称的实现。
* 如何由近优的正方形网格序列得到最终所需的直线才比较恰当。