A*是一种算法，也叫A Star.

这个算法可以用来寻路，但又不仅仅用来寻路。

就算法实现而言，自然是多种多样，而关键点在于：它的**启发式**（heuristic ）。

第一，数据结构

介绍其算法之前，先介绍一下两份数据结构：**Point**和**Maze**

#1. Point

`Point` 是指地图上的每一个点，主要含5个属性：

- int X
- int Y
- int G
- int H
- int F
- Point lastPoint

其中，X、Y自然不必多说，是指每一个Point的定位坐标；

需要说的是：

- **G**，指当前点到**起点**的距离；
- **H**，指当前点到**终点**的距离（之所以命名为H，便是出自单词"heuristic"）；
- **F**，即G和H的和（根据C#特性，完全可以把F定义成属性）

```
public int F {return G + H; }
```

- **lastPoint**，也是一个 `Point` 类型，即走到当前这一步的上一步位置

#2. Maze

`Maze` 是指整个地图，主要属性有：

- int[,] MazeArray
- List<Point> CloseList
- List<Point> OpenList

其中，MazeArray 是一个二维数组，是作为地图的保存方式**之一**（其中，把 `0` 视为正常路径，把 `1` 视为障碍物）。

需要说的是：

- **OpenList**，简单翻译成"**开放列表**"，存入**待处理**的 `Point`
- **CloseList**，简单翻译成"**闭合列表**"，存入**已处理**的 `Point`

所以，整个算法概括起来便是：

——在所有的`Point`中，寻找需要处理的`Point`，把它存入`OpenList`，然后处理完毕后存入`CloseList`，一直到`OpenList`中出现了终点或者`OpenList`为空。

于是，就留下两个问题：

第一：如何寻找需要处理的`Point`？

第二：如何处理`OpenList`里面的每一个`Point`

第二，算法

这个函数如下：

```
public Point FindPath(Point startPoint, Point endPoint)
{
    ...
}
```

最后返回的是一个`Point`，就是终点。（当然，也可能为`null`，即不存在可达的路径）

然后根据该函数返回的`Point`，不端地访问`Point.lastPoint`，即不断地寻找上一个位置，则可以得到整条路径。

示意代码如下：

```
public Point FindPath(Point startPoint, Point endPoint)
{
    OpenList.Add(startPoint);

    while(OpenList.Count!=0)
    {
        // q1
        OpenList排序（根据Point.F）;

        找到F值最小的Point, centerPoint = OpenList[0]; (已排序，则取[0]即可)
        OpenList.RemoveAt(0);
        CloseList.Add(centerPoint);

        // q2
        List<Point> surroundPoints = 找出centerPoint所有邻节点；（根据经验，邻节点最多9个）

        遍历所有邻节点
        foreach(Point point in surroundPoints)

        {
```

```
            if(point不在OpenList里面)
            {
                point.lastParent = center;
                计算point的G; // q3
                计算point的H; // q4
                OpenList.Add(point);
            }
            else
            {
                计算point的G；// q3
                if(刚才算出来的G < 本身的G)
                {
                    point.lastParent = center;
                    point.G = 刚才计算出来的G;
                }

            }
        }

        if(endPoint已经在OpenList里面了)
            return OpenList中的endPoint;
    }

    return OpenList中的endPoint(可能为null);

}
```

所以，根据上述示意代码，则将大问题分解为如下：

1. 如何根据 Point.F 排序OpenList？
2. 如何获取一个 Point 的所有邻节点？（因为有些邻节点为非法节点，比如，边界或障碍物等）
3. 如何计算一个 Point 的G值？
4. 如何计算一个 Point 的H值？