路径规划算法

1. A\*算法

A\*算法是一种在静态路网中求解最短路径最有效的方法。给定目标区域的占据栅格地图，以及出发和终止位置，即可获取最短路径。

首先，读取地图信息，创建两个空链表，openlist和closelist，将起点A加入openlist。令F=G+H，其中G为从起点移动到指定方格的移动代价，H为从指定方格到终点B的估算成本，在本例中使用曼哈顿距离计算。接下来重复以下过程：遍历openlist，将其中F值最小的节点作为当前节点，将其移到closelist中，对于其相邻的8个方格，若它不可抵达或在closelist中，忽略它；若不在openlist中，把它加入 openlist，将当前方格设置为它的父节点；若已经在openlist中，计算G值是否比当前更小，若更小，将其父节点设置为当前节点，重新计算F值。当终点B或openlist为空，则寻路结束。最后，从终点B开始，沿着每个节点的父节点回溯至起点A，即可得到总路径。路径查找代码如图1-1所示：

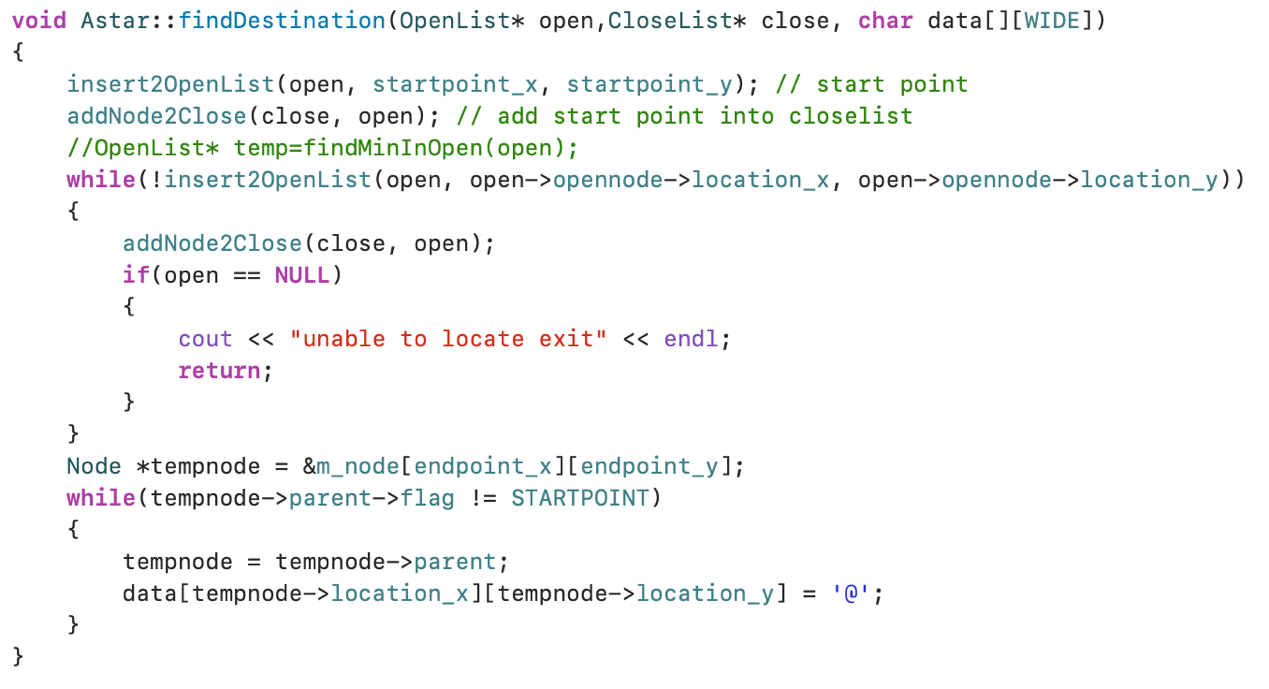


图1-1 路径查找代码

下面为A\*算法的一个实现方式，可读取存储地图信息的文本文件进行路径规划。地图如图1-2所示，其中·表示可走节点，x表示墙，s表示起点，d表示终点。程序运行结果如图1-3所示，@为使用A\*算法计算得到的最短路径。



图1-2 地图文件

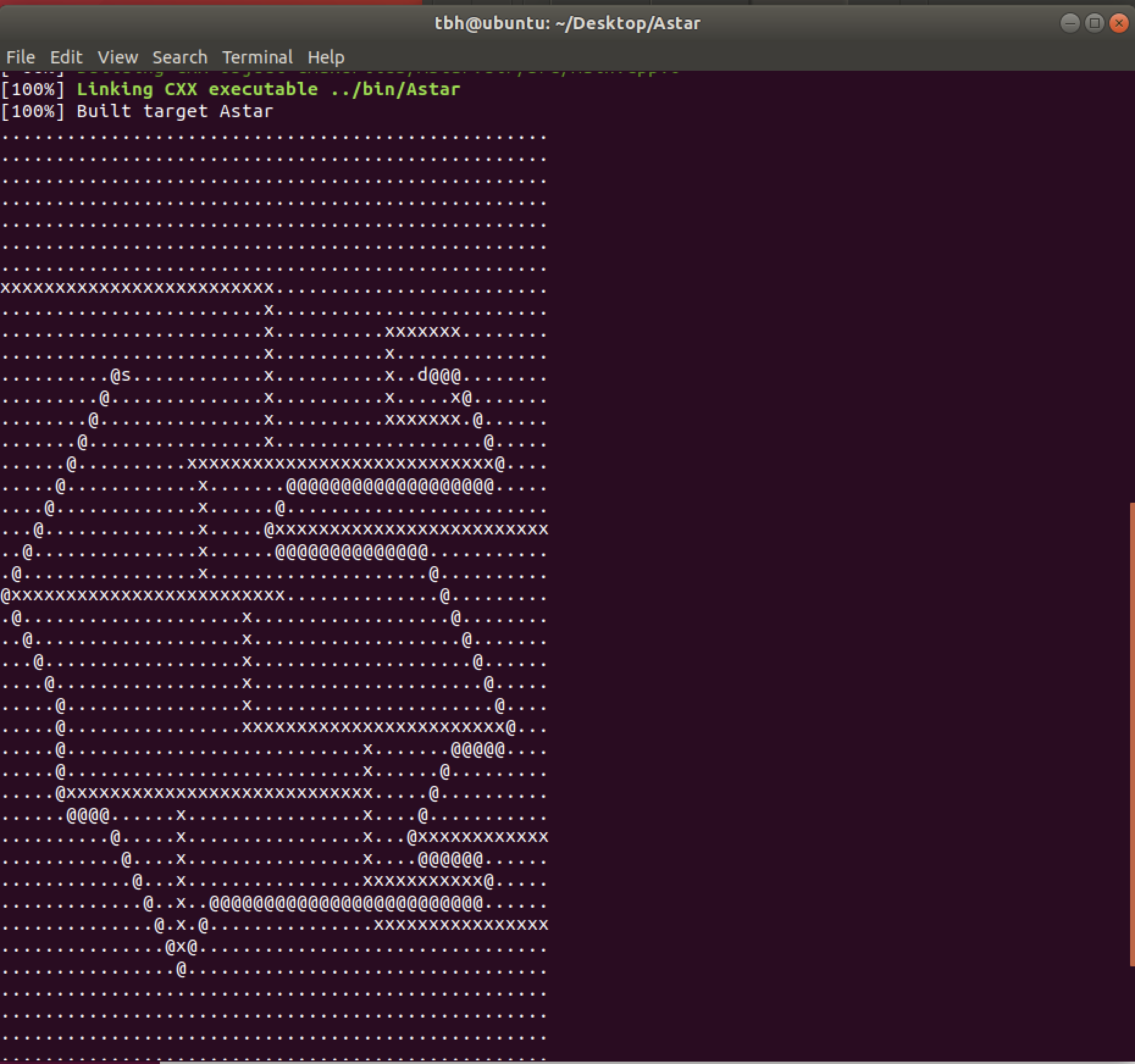


图1-3 A\*算法得到的最短路径

1. Dijkstra算法

Dijkstra算法用于解决单源最短路问题，即给定图G和起点s，通过算法得到s到达其他每个顶点的最短距离。该算法的基本思想是对图G（V，E）设置集合S，存放已被访问的顶点，然后每次从集合V-S中选择与起点s最短距离最小的顶点u，访问并加入集合S。之后，令顶点u为中介点，优化起点s与所有从u能到达的顶点v之间的最短距离。执行上述操作n次（n为顶点个数），直到集合S已经包含了所有顶点。最后递归输出从起点到终点的最短路径。

下面为使用C++实现的Dijkstra算法的例子：首先根据输入地图信息，使用邻接矩阵G储存地图信息，使用一个bool型数组vis[]储存集合S，当vis[i]为true时表示顶点Vi已被访问，使用int型数组d[]表示从起点s到达顶点Vi的最短距离，算法的核心实现代码如图2-1所示。

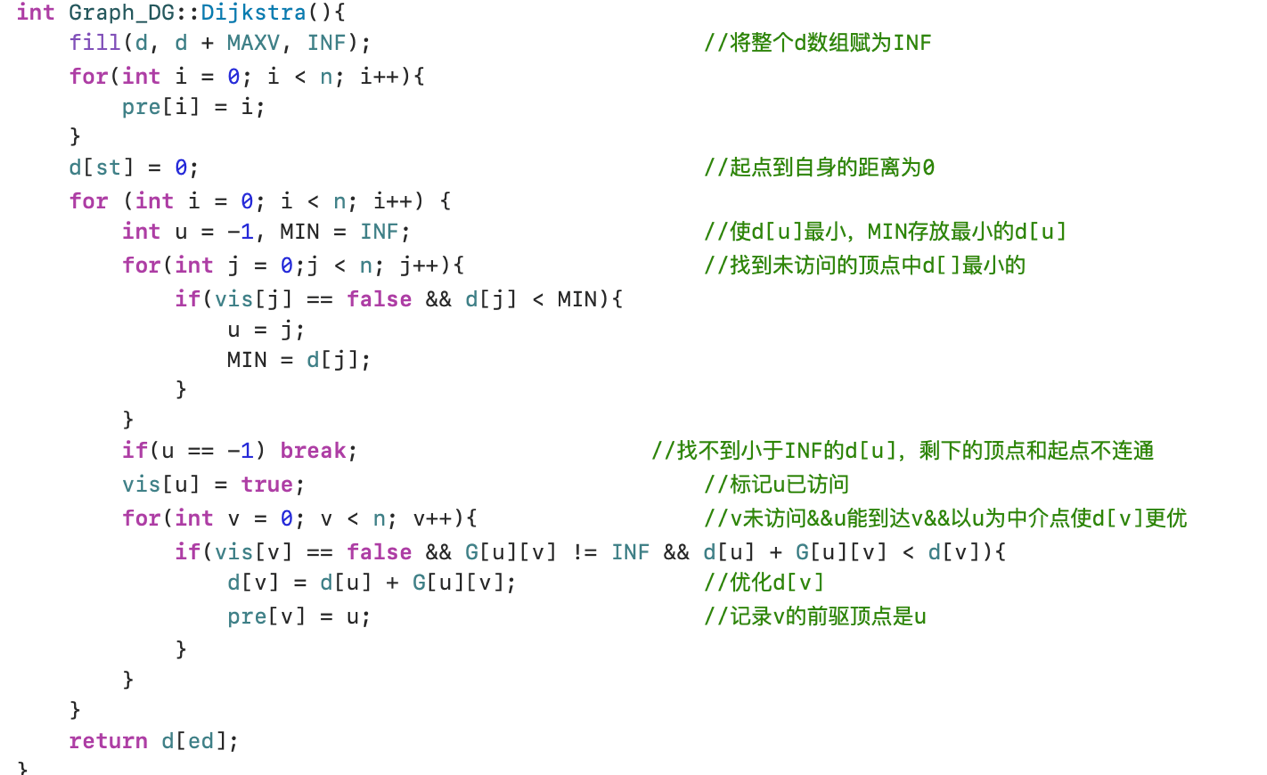


图2-1 算法核心实现代码

在给定顶点个数，边的条数，以及起点的情况下，使用如图2-2所示的函数依次输出起点至各个顶点的路径以及距离。其中DFS为递归函数，通过从终点向前遍历父节点，递归出口为起点；若当前顶点距起点的距离为INF，即无路径，则结束递归。



图2-2 输出路径及距离函数

假设输入图的顶点个数为6个，边的条数为8个，输入各边距离和运行结果如图2-3所示：

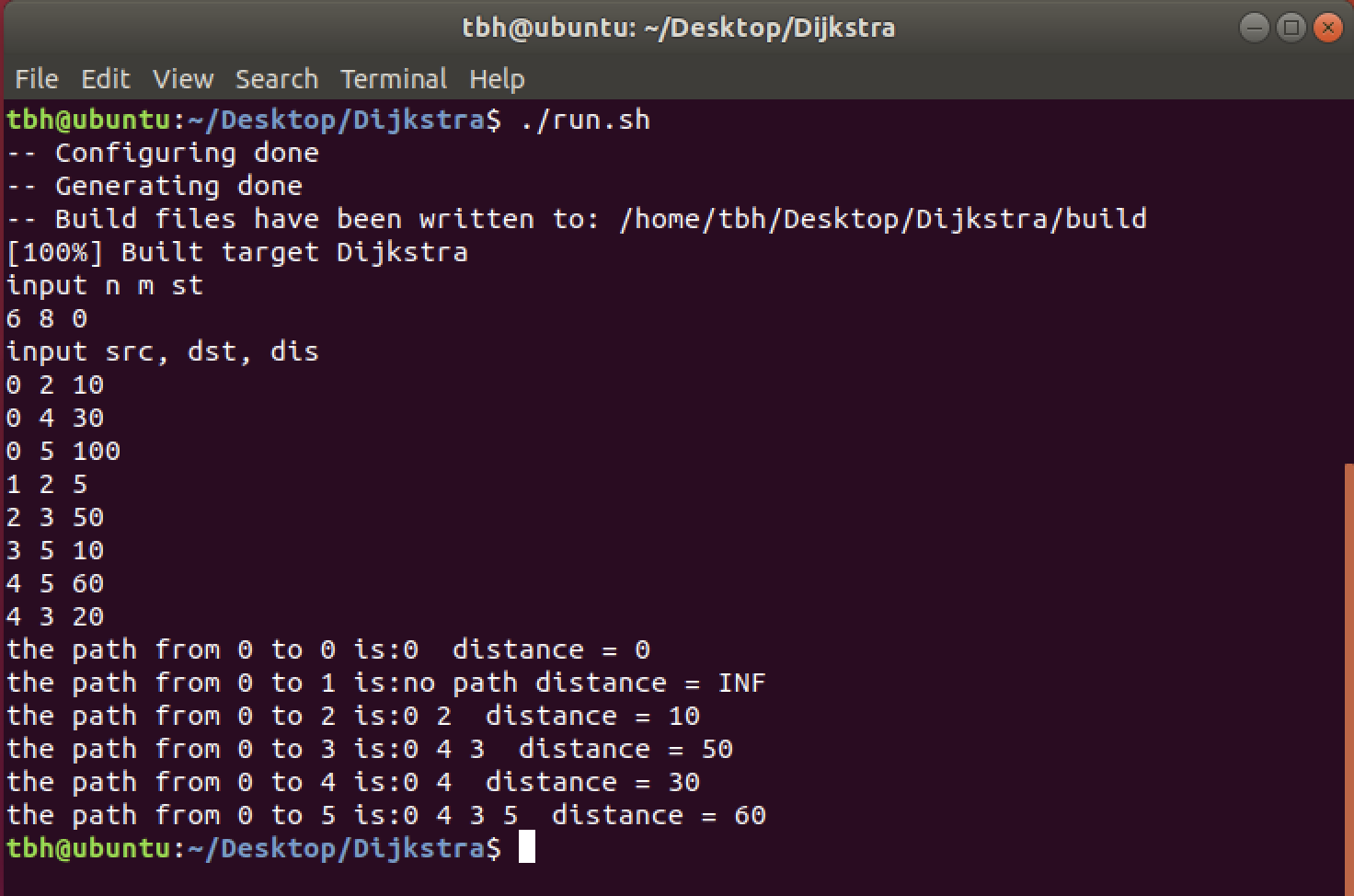


图2-3 运行结果