最短路径问题之Dijkstra

作者 Asukaa_

发布时间 2025-12-04 08:36

分类 个人记录

最短路
引入
题目
思路
代码
总结

引入

题目

一丁想去很多地方,眼看暑假就快到了,他决定在最短的时间去一个自己想去的城市(去喜欢的D 个城市之一均可)。 因为一丁的家在一个小镇上,没有火车经过,所 以他只能去邻近的城市坐火车。 每组输入数据第一行是三个整数t,S和d,表示有 T条路,和一丁家相邻的城市s个,想去的地方D个; 接着有t行,每行有三个整数a, b, time; 接着的第t+1行有s个数,表示可以出发的城市;接着的第t+2行有D个数,表示想去的城市。

每组数据求最短用时，到达不了就输出-1 除 T 外所有数字都小于1000，所有 T 之和<=200000

思路

1. 算法核心思想 Dijkstra算法解决的是这样一个问题： 在带非负权重的有向图或无向图中，从一个给定的源点出发，到图中所有其他顶点的最短路径（及其距离）是多少?

其核心思想是 “贪心” + “广度优先搜索”：

贪心：每次从未确定最短路径的顶点中，选择一个距离源点最近的顶点，认为它的当前距离就是最终的最短距离。

广度优先：从这个确定的顶点出发，去“松弛”更新其邻居顶点的距离。

2. 重要前提 所有边的权重必须为非负数（即 w >= 0）。 如果存在负权边，Dijkstra算法可能无法得到正确结果，此时需要使用 Bellman-Ford 算法。

3. 算法步骤 我们定义：

dist[]： 从源点到每个顶点的当前已知最短距离。初始时，源点为0，其余为无穷大（∞）。

visited[]： 标记顶点是否已确定最短距离。

通常使用 优先队列（最小堆） 来高效地获取当前距离最小的未确定顶点。

步骤如下：

初始化：

设置源点 s 的 dist[s] = 0，其他所有顶点 dist[v] = ∞。

将所有顶点标记为“未确定”（visited[v] = false）。

将源点 (dist=0, 顶点=s) 加入优先队列。

循环执行，直到所有顶点都确定，或优先队列为空： a. 选取当前距离最小的未确定顶点：从优先队列中取出 dist 最小的顶点 u（此时 u 为未确定状态）。 b. 标记为确定：将 u 标记为已确定（visited[u] = true）。注意：第一次取出的就是源点。 c. 松弛操作：遍历 u 的所有未确定的邻居顶点 v。

- 计算经过 u 到 v 的候选距离：newDist = dist[u] + weight(u, v)

- 记录 v 的前驱为 u（用于最后回溯路径）
- 将 (newDist, v) 加入优先队列（如果使用简单数组遍历，则无需此步，直接更新dist即可；但堆优化需要入队新值）。

结束：dist[] 数组中存储的就是从源点到各点的最短距离。通过前驱节点可以回溯出完整路径。

代码

初始化图

```
for(int i = 1;i <= t;i++)
{
      int v,u,Dis;
      cin >> v >> u >> Dis;
      Map[v][u] = min(Map[v][u] , Dis);//更新边为最短
      Map[u][v] = Map[v][u];//无向图
      ex[v] = true;
      ex[u] = true;
}
```

一点小巧思

设置两个虚拟点，一个为终点一个为起点，使得所有的起点到虚拟起点距离为0，所有终点到虚拟终点距离为0，通过这样就能找到符合答案的最短路径。

```
int start = 1001;//设置起始1001
int tag = 1002;//设置终点1002
ex[1001] = true;
ex[1002] = true;
for(int i = 1;i <= s;i++)
{
    int st;
    cin >> st;
    Map[st][start] = 0;
    Map[start][st] = 0;
}
for(int i = 1;i <= d;i++)
{
    int targ;
    cin >> targ;
    Map[targ][tag] = 0;
    Map[tag][targ] = 0;
}
```

dijkstra算法

```
while(start != tag)
{
    int Min = INT_MAX;
    int next;
    for(int i = 0;i <= 1002;i++)
    {
        if(ex[i])
        {
            if(Map[start][i] != INT_MAX)
```

```
                Min = dis[i];
                next = i;
            }
        }
    }
    if(Min == INT_MAX) break;//如果找不到最短边了  结束循环
    start = next;//从上次最短边的末端开始寻找下一个最短边  要么为这个点向下延申  要么为一条单边
    vis[start] = true;//记录访问过的点   每个距离都为能到达这个点的最短路
}

if(start != tag) cout << -1 << endl;
else cout << dis[tag] << endl;
```

总结

核心

"局部最优 → 全局最优"：一旦某点被确定为最短路径，其距离不再改变，因为非负权重保证路径不会变得更短。

作者：Asukaa_ 创建时间：2025-12-04 08:36:30

评论区

发表评论