

Dijkstra 算法正确性的形式化验证

刘衍

2025 年 12 月 6 日

1 算法介绍

Dijkstra 通常指一种在非负权图上求单源最短路径的贪心算法。绝大部分情况下不能处理带负边的图。

在这里，我们证明对一个简单有限的无向正权图和一个给定的源点，进行 Dijkstra 算法得到的数组正确描述了从源点出发到图中所有顶点的最短路径长度。

2 算法描述

Algorithm 1 Dijkstra Algorithm

Require: Undirected, Simple, Positive-Weighted Graph $G(V, E)$, a selected vertex S in G , $dist$ is a $|V|$ array prepared for minimum distances from S

Ensure: $\forall v$, $dist[v]$ describes the length of the shortest path from S to v

```
1: for each vertex  $v$  do
2:    $dist[v] = +\infty$ 
3: end for
4:  $dist[S] = 0$ 
5:  $visitedSet = \emptyset$ 
6: while  $visitedSet \neq V$  do
7:    $u = \text{vertex in } V - visitedSet \text{ with minimum } dist[u]$ 
8:    $visitedSet += [u]$ 
9:   for each edge  $(u, v)$ ,  $v \in visitedSet$  do
10:     $dist[v] = \min(dist[v], dist[u] + weight(u, v))$ 
11: end for
12: end while
```

Dijkstra 算法首先创建一个包含所有已经访问顶点的集合 $visitedSet$ ，并记所有未访问节点的集合 $unvisitedSet$ 。初始化时，为每个顶点赋值为只经过 $visitedSet$ 从 S 到它的最短距离，即除了 S 为设为 0 外都设为 $+\infty$ 。

从 $unvisitedSet$ 中选择一个 $dist$ 值最小的顶点 u ，将其加入 $visitedSet$ ，并进行松弛操作：考虑 u 所有未访问的邻居 v ，如果从 S 到 u 再到 v 的路径更短，那么将 $dist[v]$ 的值更新为 $dist[u] + weight(u, v)$ 。

可以对已访问节点的数量归纳证明 Dijkstra 算法的正确性。即，证明循环不变量：
 $\forall u, u \in visitedSet \rightarrow dist[u]$ 是图中从源点 S 到达 u 的最短距离；
 $\forall u, u \notin visitedSet \rightarrow dist[u]$ 是从源点 S 只能经过 $visitedSet$ 中的顶点到达 u 的最短路径长度。

归纳奠基平凡。假设对于已访问 k 个顶点时成立，来证明循环不变量对 $k+1$ 个顶点时也成立，

设 u 是第 $k+1$ 个访问的顶点，首先来考虑第一个不变量，即证明 $dist[u]$ 是图中从源点 S 到 u 的最短距离。

反证法。假设存在比 $dist[u]$ 更短的路径，这条路径要么包含另一个属于 $unvisitedSet$ 的顶点，要么不包含任何 $unvisitedSet$ 的顶点。

(1) 如果这条更短的路径 p 包含另一个属于 $unvisitedSet$ 的顶点，设 w 是 p 上的第一个属于未访问节点，那么从 S 到 w 的距离至少为 $dist[w]$ (用到了第二个循环不变量，以及 w 是第一个未访问顶点的特性)，从 w 到 v 的成本是一个正数，所以这条路径的开销 $sumWeight(p) > dist[w]$ 。与假设 $sumWeight(p) < dist[u]$ 和 u 的选择 $dist[u] < dist[w]$ 相矛盾。

(2) 如果这条路径 q 不包含任何一个属于 $unvisitedSet$ 的顶点，设 w 是 q 上倒数第二个顶点。由假设有 $dist[w] + weight(w, u) < dist[u]$ 。但这与因为在访问 w 时的松弛操作矛盾，这一步应该已经将 $dist[u]$ 设置为至多 $dist[w] + weight[w, u]$ 。说明这一步比较传统的解决方法是增加第三个不变量：

$$\forall edge(u, v), u \in visitedSet \rightarrow v \in unvisitedSet \rightarrow dist[u] + weight(u, v) \leq dist[v]$$

这个不变量的证明从从松弛操作的程序片段来看是相对容易的，不过你可能还需要增加 $dist[v]$ 单调不增的不变量，等等。

然后来证明第二个不变量， $visitedSet$ 中加入了 u ，我们来考虑 $\forall v, v \notin visitedSet \rightarrow dist[u]$ 是从源点 S 只能经过 $visitedSet$ 中的顶点到达 v 的最短路径是否经过 u ：

- (1) 不经过 u ，则 $dist[v] = dist[v]$;
 - (2) 经过 u ，则 $dist[v] = dist[u] + weight(u, v)$;
- 即 $dist[v] = \min(dist[v], dist[u] + weight(u, v))$ 。

3 任务指导

我们给出了 Monad 算子定义的 Dijkstra 算法程序，见 `Dijkstra.v`，并且我们给出了关于 (最短) 路径的性质列表。

- (A) 证明性质列表中所有关于 (最短) 路径的性质。
- (B) 证明 `Dijkstra.v` 中关于 Dijkstra 算法的正确性 (*Theorem Dijkstra_correct*) 的描述。在合适的位置写出需要证明的循环不变量，并使用性质列表中的性质完成不变量的证明。

例如提到的：

$\forall u, u \in visitedSet \rightarrow dist[u]$ 是图中从源点 S 到达 u 的最短距离。

$\forall u, u \notin visitedSet \rightarrow dist[u]$ 是从源点 S 只能经过 $visitedSet$ 中的顶点到达 u 的最短路径长度。

$\forall edge(u, v), u \in visitedSet \rightarrow v \in unvisitedSet \rightarrow dist[u] + weight(u, v) \leq dist[v]$ 。

4 性质列表

性质的形式化描述详见 `dijkstra.v`。

Lemma (path_the_first_is_cut). 从 i 到 j 的路径上第一个满足 P 的顶点的前一个顶点是非 P 的。

Lemma (path_splice_trans). 从 i 到 k 的路径拼接上从 k 到 j 的路径是一条从 i 到 j 的路径。

Lemma (path_splice_length). 从 i 到 k 的路径拼接上从 k 到 j 的路径的长度等于这两条路径的长度之和。

Lemma (shortest_path_all_shortest). 从 i 到 j 的最短路径的任何两个顶点在这条路径中的路径都是一条最短路径。

Lemma (path_splice_trans). 从 i 到 k 的路径拼接上从 k 到 j 的路径是一条从 i 到 j 的路径。

Lemma (path_splice_length). 从 i 到 k 的路径拼接上从 k 到 j 的路径的长度等于这两条路径的长度之和。

Lemma (path_in_vset_mono). 从 i 到 j 限制在 $1, 2, \dots, k-1$ 上的路径也是限制在 $1, 2, \dots, k$ 上的路径。

Lemma (shortest_path_all_shortest). 从 i 到 j 的最短路径的任何两个顶点在这条路径中的路径都是一条最短路径。

Lemma (shortest_path_is_simple). 任意的最短路径都是简单路径。

Lemma (shortest_path_segment). 从 i 到 j 限制在 $1, 2, \dots, k$ 的最短路径如果经过 k , 则可以把路径切分为从 i 到 k 和从 k 到 j 限制在 $1, 2, \dots, k-1$ 上的两条最短路径。

Lemma (shortest_path_triangle). 从 i 到 j 的最短路径长度不大于从 i 到 k 的最短路径长度加上从 k 到 j 的最短路径长度。