single-source shortest paths

2022 年 11 月 1 日

目录

1   intro and prequisition

1.1   Notation

我们研究最短路径的话, 我们必然会面对图的各种参数, 因为我们当然是在有权图上面寻找最短路径的, 如果说是那种将路径长度定义为路径所经过的节点个数的话, 正如 22 所讲的那样的话, 这种就不在我们这次的研究范围内了. 于是我们给定的是**有权, 有向图**.

定义 1. *A graph is abbreviated as* $G = (V, E)$, 这是我们已经熟知的.

定义 2. 一个 *path* 记为 p , 可以写为 $\langle v_0, v_1, \cdots, v_n \rangle$, 为了突出其终点和起点, 一个 *path* 可以记为

$$p : u \rightsquigarrow v$$

定义 3. $w : E \to \mathbb{R}, (u, v) \mapsto w(u, v)$, 将权重以函数的方式写出来当然是为了严谨. 虽然在一些人看来可能是脱裤子放屁, 但其实有很多东西的定义都是这样用函数定义的. 同时也定义了 *path* 的权重 $p \mapsto w(p) = \sum\limits_{i=1}^{\infty} w(v_i)$

定义 4. 最短路径的记号:

$$\delta(u, v) = \begin{cases} \min\{w(p) : p : u \rightsquigarrow v\}, & \textit{if there is a path from u to v} \\ \infty, & \textit{otherwise} \end{cases}$$

1.2　some variants of single-source shortest paths

我们目前的问题称为 single-source shortest paths. 对于有权有向图, 给定了一个 source, 我们要找出从 source 到其他点的最短路径的大小, 以及可以求解出这个路径. single-source shortest paths 问题有多种变体, 当然这里只是介绍一下

> **Single-destination shortest-paths problem:** Find a shortest path to a given ***destination*** vertex *t* from each vertex *v*. By reversing the direction of each edge in the graph, we can reduce this problem to a single-source problem.
>
> **Single-pair shortest-path problem:** Find a shortest path from *u* to *v* for given vertices *u* and *v*. If we solve the single-source problem with source vertex *u*, we solve this problem also. Moreover, all known algorithms for this problem have the same worst-case asymptotic running time as the best single-source algorithms.
>
> **All-pairs shortest-paths problem:** Find a shortest path from *u* to *v* for every pair of vertices *u* and *v*. Although we can solve this problem by running a single-source algorithm once from each vertex, we usually can solve it faster. Additionally, its structure is interesting in its own right. Chapter 25 addresses the all-pairs problem in detail.

图 1: the variants of single-source shortest paths problems

其中这里的 all-paired shortest paths problems 是我们在 25 中面对的问题.

1.3　optimal structure of shortest paths

rt. 最短路径具有优化子结构, 即,

定理 1. 最短路径的**子路径**也是**最短路径**

证明. trivial! □

1.4　representation of shortest paths

这涉及到前面我还没看的部分, 即无向无权图的最短路径求解. 这里我们涉及 **predecessor**, 符号 π 的出现说明其和 **predecessor** 有关. 比如说: $v.\pi$ 是 v 的一个前驱

定义 5 (predecessor subgraph)**.** $G_\pi = (V_\pi, E_\pi)$ 是 *predecessor subgraph*, 其中

$$V_\pi = \{v \in V : v.\pi \neq \varnothing\} \cup \{s\}$$

s 是 *source,* 并且

$$E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$$

虽然这里定义并不是非常清晰, 比如说 $v.\pi$ tm 的是什么我也不太清楚. 但是我们这里可以用语言将其描述清楚:

single-source shortest paths 问题其实就是求出下面这个子图
$G' = (V', E')$
V' 是所有能够达到的点的集合, 即 $\delta(s,v) \neq \infty$
并且 G' 是一个树, 并且这个树上的任意一个节点 v, 则 v 和 s 之间的距离最短.

这就是用另一种方法说了一边我们要求什么. 超, 我也不知道为什么要说什么 predecessor subgraph . 可能是说 G' 是 G_π 的子图. 并不是很懂

1.5 relaxation

1.5.1 initializing single source

我们使用 $v.d$ 表示目前已知的 v, s 之间的距离. 称为 **shortest path estimate**. 这时可以补充上面的 $v.\pi$ 了: $v.\pi$ 就是目前已知的" 最短路径上" v 的前驱.

```
INITIALIZE-SINGLE-SOURCE(G, s)
1   for each vertex v ∈ G.V
2       v.d = ∞
3       v.π = NIL
4   s.d = 0
```

图 2: 初始化的伪代码

ss

1.5.2 relaxation: code and definition

relaxation 是一个很简单但是很重要的操作. relaxation 这个词我们之前就已经见过了, 可能有人那时候开始就觉得: 松弛是什么叫法啊. 总之, 确实不够自然. 应该可以将其称为 renew 或者 update.

定义 6 (relaxation)**.**

```
RELAX(u, v, w)
1   if v.d > u.d + w(u, v)
2       v.d = u.d + w(u, v)
3       v.π = u
```

图 3: code

意思即为, u 到 v 的一个松弛, 如果说走到 u 然后走到 v 的长度更短, 我们就更新 $v.d$ 目前最短路径长度; $v.\pi$ v 的前驱, i.e. 更新为 u 而 `relaxation` 只需要常数时间.

> Each algorithm in this chapter calls INITIALIZE-SINGLE-SOURCE and then repeatedly relaxes edges. Moreover, relaxation is the only means by which shortest path estimates and predecessors change. The algorithms in this chapter differ in how many times they relax each edge and the order in which they relax edges. Dijkstra's algorithm and the shortest-paths algorithm for directed acyclic graphs relax each edge exactly once. The Bellman-Ford algorithm relaxes each edge $|V| - 1$ times.

1.6 some property

你可以将两点之间的最短路径视为一个度量, 即, $\delta(u,v)$, 至少在正权图中, 满足度量的三个性质:

- 1. 正定性: $\delta(v,u) \geq 0$

- 2. 忘了什么性: $\delta(u,v) = 0 \implies u = v$

- 3. 三角不等式. $\delta(u,v) + \delta(v,w) \geq \delta(u,w)$

至少这样好像挺有意思的. 然后虽然这样说并不是很好, 但是, 书本上涉及的这部分性质, 其实是非常直观并且感觉很明显的事情. 唯一说并不是很明显的就是上面这个三角不等式了.

1.7 an outline copied from textbook

Section 24.1 presents the Bellman-Ford algorithm, which solves the single-source shortest-paths problem in the general case in which edges can have negative weight. The Bellman-Ford algorithm is remarkably simple, and it has the further benefit of detecting whether a negative-weight cycle is reachable from the source. Section 24.2 gives a linear-time algorithm for computing shortest paths from a single source in a directed acyclic graph. Section 24.3 covers Dijkstra's algorithm, which has a lower running time than the Bellman-Ford algorithm but requires the edge weights to be nonnegative. Section 24.4 shows how we can use the Bellman-Ford algorithm to solve a special case of linear programming. Finally, Section 24.5 proves the properties of shortest paths and relaxation stated above. We require some conventions for doing arithmetic with infinities. We shall assume that for any real number $a \neq \infty$, we have $a + \infty = \infty + a = \infty$. Also, to make our proofs hold in the presence of negative-weight cycles, we shall assume that for any real number $a \neq 1$, we have $a + (-\infty) = (-\infty) + a = -\infty$

2 Bellman-Ford algorithm

2.1 an introd

2.2 algorithm

2.3 negative weighted cycles

2.4 the prf of theorem

3 Dijikstra algorithm

3.1 an review

3.2 algorithm

3.3 the prf of algorithm