第 *3-8* 讲*:* 单源最短路

姓名： 林凡琪　　**学号：** *211240042*

评分： _____　　**评阅：** _____

2022 年 *11* 月 *1* 日

请独立完成作业，不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

1　作业（必做部分）

题目 1 (TC 24.1-2)
Prove Corollary 24.3.

解答：
Suppose there is a path from s to v. Then there must be a shortest such path of length $\delta(s, v)$. It must have finite length since it contains at most $|V| - 1$ edges and each edge has finite length. By Lemma 24.2, $v.d = \delta(s, v) < \infty$ upon termination.

On the other hand, suppose $v.d < \infty$ when BELLMAN-FORD terminates. Recall that $v.d$ is monotonically decreasing throughout the algorithm, and RELAX will update $v.d$ only if $u.d + w(u, v) < v.d$ for some u adjacent to v. Moreover, we update $v.\pi = u$ at this point, so v has an ancestor in the predecessor subgraph. Since this is a tree rooted at s, there must be a path from s to v in this tree. Every edge in the tree is also an edge in G, so there is also a path in G from s to v.

题目 2 (TC 24.1-3)
Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let mm be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

解答：
By the upper bound theory, we know that after m iterations, no d values will ever change. Therefore, no d values will change in the $(m + 1)$-th iteration. However, we do not know the exact m value in advance, we cannot make the algorithm iterate exactly m times and then terminate. If we try to make the algorithm stop when every d values do not change anymore, then it will stop after $m + 1$ iterations.

题目 3 (TC 24.1-4)
Modify the Bellman-Ford algorithm so that it sets $v.d$ to $-\infty$ for all vertices v for which there is a negative-weight cycle on some path from the source to v.

解答:
verbatim:

```
BELLMAN-FORD'(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1
for each edge (u, v)  G.E
RELAX(u, v, w)
for each edge(u, v)  G.E
if v.d > u.d + w(u, v)
mark v
for each vertex u  marked vertices
DFS-MARK(u)

DFS-MARK(u)
if u != NIL and u.d != -ϖ
u.d = -ϖ
for each v in G.Adj[u]
DFS-MARK(v)
```

After running BELLMAN-FORD, run DFS with all vertices on negative-weight cycles as source vertices. All the vertices that can be reached from these vertices should have their d attributes set to $-\infty$.

题目 4 (TC 24.2-2)

解答:
When we reach vertex v, the last vertex in the topological sort, it must have out-degree 0. Otherwise there would be an edge pointing from a later vertex to an earlier vertex in the ordering, a contradiction. Thus, the body of the for-loop of line 4 is never entered for this final vertex, so we may as well not consider it.

题目 5 (TC 24.3-2)

解答:
Consider any graph with a negative cycle. RELAX is called a finite number of times but the distance to any vertex on the cycle is $-\infty$, so Dijkstra's algorithm cannot possibly be correct here. The proof of theorem 24.6 doesn't go through because we can no longer guarantee that

$\delta(s, y) \leq \delta(s, u)$.

题目 6 (TC 24.3-4)
Professor Gaedel has written a program that he claims implements Dijkstra's algorithm. The program produces $v.d$ and $v.\pi$ for each vertex $v \in V$. Give an $O(V + E)$-time algorithm to check the output of the professor's program. It should determine whether the d and π attributes match those of some shortest-paths tree. You may assume that all edge weights are nonnegative.

解答:

Begin by verifying that proposed shortest paths tree is indeed a tree. Next check that s.d= 0. Then, for each vertex in V
$\{s\}$, examine all edges cominginto V. Check that U. is the vertex which minimizes u.d + w(u, v) for all vertices u for which there is an edge (u, v), and that v.d = v. .d + w(v. ,v). If this is ever false, return false. Otherwise, return true. This takes O(V + E) time. Now we must check that this correctly checks whether or not the d and attributes match those of some shortest-paths tree. Suppose that this is not true; ie, that the algorithm terminates without returning false on input which doesn't correspond to correctly computed minimum distances and a minimum spanning tree. Let U be a vertex which is incorrect which minimizes $min(v.d, \delta(s, v))$.
Break ties by choosing the vertex closest to s along the proposed shortest paths tree T. (If there are still ties, choose arbitrarily from among them). First suppose $v.d \leq \delta(s, v)$. Then we must have $v.\pi.d < v.d$ since $U.\pi.d$ is closer to the root along T, so if equality held we would have selected $U.\pi$ instead of v. By construction, $v.\pi.d$ must be correct. Furthermore, if U had a better neighbor than $U.\pi$ according to the estimated distances, the algorithm would have identified it at terminated. Thus, if $U.\pi$ cannot be the parent of U on a shortest paths tree, there must exist a neighbor u of U such that u.d was computed incorrectly, but such that $\delta(s, u) + w(u, u) < v.d$. However, this implies $\delta(s, u) < v.d$, which means we should have initially chosen u instead of U.
Now suppose $\delta(s, u) < v.d$. If U has no incident edges of weight 0 then each vertex $u \in N(v) = \{u|(u, v) \in E\}$ has been computed correctly.
Since $v.d$ must equal $u.d+w(u, 0)$ for some such vertex $u \in N(v)$, the algorithm would detect if U were incorrectly computed.
Thus, V must have at least one incident edge of weight 0.
Let $S = \{v^{'}|(v^{'}, v) \in E, w(o^{'}, v) = 0\}$.
Then no $v^{'} \in S$ can be the parent of U in the proposed shortest paths tree, since it would have been chosen instead of U to begin with. Furthermore, since v.d is incorrect and each $u \in N(v) - S$ has strictly lower estimated distances, the estimated distances of each $u \in N(u) - S$ must be correct. This implies that the only possible parents of U in a true shortest paths tree must lie in S. Let z be one such vertex. Then $\delta(s, z) = \delta(s, v)$, so we may apply the exact same argument as above to z. Continuing in this fashion, follow 0-weight edges back towards the root along some true shortest paths tree. Since $0 \leq \delta(s, v) < v.d$, this process will eventually terminate because the estimated distance is nonzero, so either we'll run out of 0-weight edges or we'll hit the root and the algorithm will correctly identify an overestimated distance.

题目 7 (TC 24.3-7)

解答:

$V + \sum_{(u,v)\in E} w(u, v) - E$

题目 8 (TC 24.5-2)

Give an example of a weighted, directed graph G = (V, E) with weight function $w : E \to \mathbb{R}$ and source vertex ss such that G satisfies the following property: For every edge $(u, v) \in E$, there is a shortest-paths tree rooted at ss that contains (u, v) and another shortest-paths tree rooted at ss that does not contain (u, v).

解答:
Let G have 3 vertices s, x, and y. Let the edges be (s, x), (s, y), and (x, y) with weights 1, 1, and 0 respectively. There are 3 possible trees on these vertices rooted at s, and each is a shortest paths tree which gives $\delta(s, x) = \delta(s, y) = 1$.

题目 9 (TC 24.5-5)
Let $G = (V, E)$ be a weighted, directed graph with no negative-weight edges. Let $s \in V$ be the source vertex, and suppose that we allow $v.\pi$ to be the predecessor of v on any shortest path to v from source s if $v \in V - \{s\}$ is reachable from s, and NIL otherwise. Give an example of such a graph G and an assignment of π values that produces a cycle in G_πG.
(By Lemma 24.16, such an assignment cannot be produced by a sequence of relaxation steps.)

解答:
Suppose that we have a grap hon three vertices $\{s, u, v\}$ and containing edges $(s, u), (s, v), (u, v), (v, u)$ all with weight 0. Then, there is a shortest path from s to v of s, u, v and a shortest path from s to u of s v, u. Based off of these, we could set $v.\pi = u$and $u.\pi = v$. This then means that there is a cycle consisting of u, v in G_π.

题目 10 (TC Problem 24-3)

解答:
a. To do this we take the negative of the natural log (or any other base will also work) of all the values c_i that are on the edges between the currencies. Then, we detect the presence or absence of a negative weight cycle by applying Bellman Ford. To see that the existence of an arbitrage situation is equivalent to there being a negative weight cycle in the original graph, consider the following sequence of steps:

$$R[i_1, i_2] \cdot R[i_2, i_3] \cdot \cdots \cdot R[i_k, i_1] > 1$$
$$\ln(R[i_1, i_2]) + \ln(R[i_2, i_3]) + \cdots + \ln(R[i_k, i_1]) > 0$$
$$-\ln(R[i_1, i_2]) - \ln(R[i_2, i_3]) - \cdots - \ln(R[i_k, i_1]) < 0.$$

b. To do this, we first perform the same modification of all the edge weights as done in part (a) of this problem. Then, we wish to detect the negative weight cycle. To do this, we relax all the edges $|V|-1$ many times, as in BellmanFord algorithm. Then, we record all of the dd values of the vertices. Then, we relax all the edges $|V|$ more times. Then, we check to see which vertices had their dd value decrease since we recorded them. All of these vertices must lie on some (possibly disjoint) set of negative weight cycles. Call S this set of vertices. To find one of these cycles in particular, we can pick any vertex in S and greedily keep picking any vertex that it has an edge to that is also in S. Then, we just keep an eye out for a repeat. This finds us our cycle. We know that we will never get to a dead end in this process because the set S consists of vertices that are in some union of cycles, and so every vertex has out degree at least 1.

---

2 作业 (选做部分)

题目 1 (TC Problem 24-2)

解答:

---

3 Open Topics

Open Topics 1 (Delta stepping algorithm)
参考资料:

- https://en.wikipedia.org/wiki/Parallel_single-source_shortest_path_algorithm
- Meyer, U.; Sanders, P. (2003-10-01). "Δ-stepping: a parallelizable shortest path algorithm".

Open Topics 2 (Radius stepping algorithm)
参考资料:

- https://en.wikipedia.org/wiki/Parallel_single-source_shortest_path_algorithm
- Blelloch, Guy E.; Gu, Yan; Sun, Yihan; Tangwongsan, Kanat (2016). "Parallel Shortest Paths Using Radius Stepping". Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures - SPAA '16. New York, New York, USA: ACM Press: 443–454.

4 反馈