研究生算法课课堂笔记

上课日期：12 月 22 日　第（2）节课

组长学号及姓名：1601214489 徐力有

组员学号及姓名：1601214520 宋思捷 1601214463 李浩然

本节课主要讲了 Bellman-Ford 的优化问题，在路由协议和负环检测中的应用，同时讲了 MST 的一些拓展。

Bellman-Ford 算法的优化

空间优化：

朴素 Bellman-Ford 算法的空间复杂度为 $O(n^2)$，可以优化到 $O(n)$。具体做法为维护两个一维数组：

d(v)：目前找到的从 v->t 的最短路

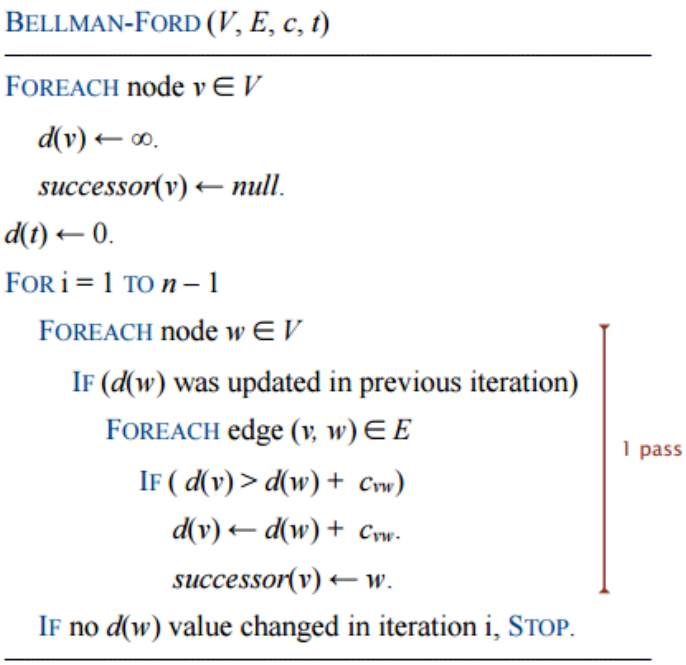
successor(v): v->t 路径上 v 的后继节点

性能优化：

若一轮循环中所有顶点的最短距离都没有发生改变，则算法收敛，可以据此提前结束 Bellman-Ford 算法。同时，如果 d(w)在第 i-1 轮循环中没有更新，则对于$(v,w)\in E$，d(v)在第 i 轮循环中也不会更新，因此在第 i 轮循环中，不必考虑 v 点。根据这一点，我们对 Bellman-Ford 算法可以做的性能优化为：这一轮只更新上一轮循环优化过的顶点。

具体实现：根据 SPFA (Shortest Path First Algorithm)的思路，维护一个队列，本次循环优化过的顶点进入队列，在下一轮循环中只松弛队列中点的上游顶点。

改进后的伪码：



其中，改变边的松弛顺序对最优结果没有影响，详见 12 月 22 日第(1)节课的笔记。

Bellman-Ford 算法在路由协议中的应用

问题建模：路由器可以视为图中的节点，路由器之间的连接为有向边，每条边上的延迟可看作边的权重。

Bellman-Ford 算法可以用于路由协议，而 Dijkstra 算法不行，原因是：

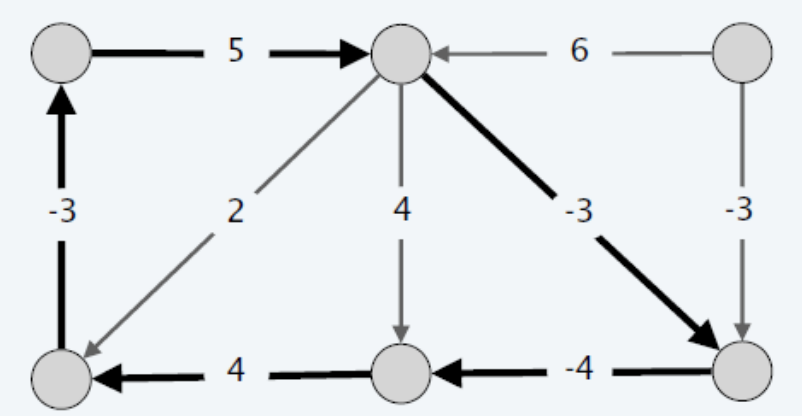
Bellman-Ford 算法只需要记住自己局部的信息，而 Dijiskra 算法则需要得到全图信息。在应用 Bellman-Ford 算法时，每个路由器只需要在自身到目的地的最短路径值改变时，通知周围自己的邻居节点重新计算到目的地的最短路径即可。放到路由算法上，每个顶点维护自己的关联节点的队列，来得知自己需不需要进行松弛操作。

如何判断当前路由网络是否达到稳定?

由于网络中路由器个数 N 未知，无法根据循环次数来判断算法是否收敛。可以在局域网络中设置假如某个节点自身到目的地的最短路径发生改变，则向上游节点发送报告，假如在一定的时间内没有收到任何报告，就说明当前这个局部网络内已经达到稳定了。

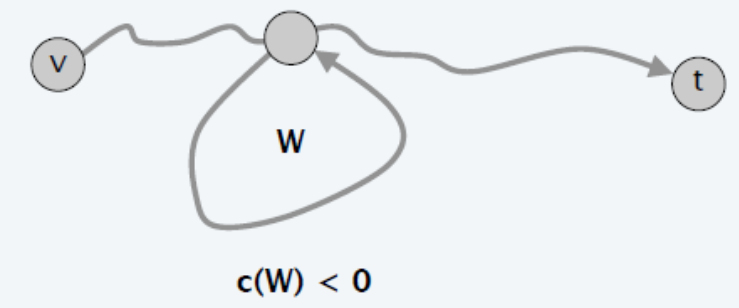
Bellman-Ford 算法与负环检测

负环检测问题：如下图，给定图 $G = (V, E)$，每条边的权重为 c_{vw}，寻找一个环（如果存在的话），其每条边的权重之和为负。



引理：如果存在节点 v，第 n 轮循环比第 n-1 轮循环更优，即 $OPT(n,v) < OPT(n\text{-}1,v)$ ，则任何从 v 到 t 的最短路径都包括环 W，且 W 是一个负环。

证明：若 $OPT(n,v) < OPT(n\text{-}1,v)$，则从 v 到 t 的最短路 P 恰好有 n 条边，由鸽巢原理，路径 P 中一定存在环 W。去除路径 P 中的环 W，则路径 P 中有 n-1 条边，由 $OPT(n,v) < OPT(n\text{-}1,v)$，可得 $c(W) < 0$。



利用 Bellman-Ford 算法检测负环：加入一个新顶点 t，在所有节点和 t 之间建立权重为零且指向 t 的边。对该图运行 Bellman-Ford 算法，如果 n 轮之后算法已经稳定，没有节点的值发生变化，则说明没有负环。否则说明图中存在负环。具体来讲：

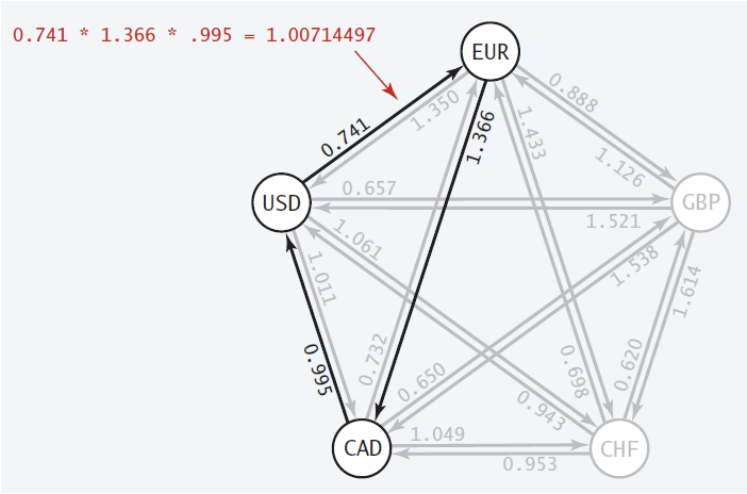
- 加入新顶点 t，利用 Bellman-Ford 算法进行 n 轮循环（注意不是 n-1 轮）
- 如果在第 n 轮循环中没有节点的值 d(v)发生变化，则没有负环
- 否则，假设 d(s)在第 n 轮循环中被更新
- 定义 pass(v)为 d(v)被最后更新的那轮循环，找到 pass(s) = n 和 pass(successor(v)) >= pass(v)-1 的点，追踪这些点的后继节点，一定可以找到负环。

时间复杂度：和基础的 Bellman-Ford 算法相同，为 O(mn)。

空间复杂度：需要额外的 pass 数组，复杂度为 O(n)，若采用优化的 Bellman-Ford 算法，总复杂度仍为 O(n)。

应用：Currency Conversion

问题描述：给定 N 个币种和币种间的汇率，是否存在套利机会？



思路分析：该问题的目标是计算上图中是否存在一个环，使该环边上的权重相乘得到的值大于 1，若存在，则有套利机会，否则没有套利机会。该问题可转化为负环检测问题，对汇率取对数，可以将乘法转化为加法，再取相反数，可以将大于零的要求转化为小于零，应用上述负环检测算法即可。

最小生成树

(1) 假如图的边权值唯一，那么最小生成树一定唯一。

证明：若有两棵最小生成树 T1 和 T2， 一定能找到一条边 e 在 T1 中不在 T2 中，将这条边加入 T2，在 T2 中会形成一个环，如果这个环中的最大边是 e，则由 Red Rule 可知，e 不可能在最小生成树 T1 中，若最大边不是 e，则去掉环中最大边可以得到一个权重和更小的生成树，即 T2 不是最小生成树，矛盾。若边权值不唯一，则可能有多个最小生成树，但是只能在权值相等的那些边之间做取舍有些变换。

(2) 引理：如果有一个环，那么环上唯一的最大边不可能出现在任何一个 MST 上。如果有多个最大边，那么不能同时出现在同一棵 MST 上。

证明：若唯一最大边在 MST 上，则添加环上的一条边，得到环，去掉该最大边，可以得到一个权重和更小的生成树，与原树是 MST 矛盾。当有多条最大边时同理可证。

(3) 若一条边权重唯一，那么它要么出现在所有 MST 上，要么不能出现在任何 MST。

证明：假设有一条唯一边 *e*，出现在图的一棵 MST1 上，但并没有出现在 MST2 上。把 *e* 放到 MST2 上，会在 MST2 上形成环。寻找该环上的最大边，两种可能：

a) *e* 是最大边，根据引理得知这条边已经出现在 MST1 上，所以 *e* 肯定不是最大边；

b) *e* 不是最大边，则这个环上可以删掉最大边来得到更小的生成树，那么 MST2 不是最小生成树。矛盾！所以假设不成立。

(4) 给定图 *G* = (*V*, *E*)，任给 G 的最小生成树 T，是否存在有效的 Kruskal 算法可以将 T 作为输出？

存在 Kruskal 算法将 T 作为输出。Kruskal 算法的思想是将所有边升序排列，只要没有环就将该边加入 MST 中。假如有若干边权值相同，要输出给定的最小生成树 T，对 T 中的边优先安排在序列的前面即可。