INT102W7_bellmanforld 算法

Bellmanford：

定义： 求解单源最短距的一种算法，借由对边的松弛操作实现，通常在没有负权值的情况下不使用

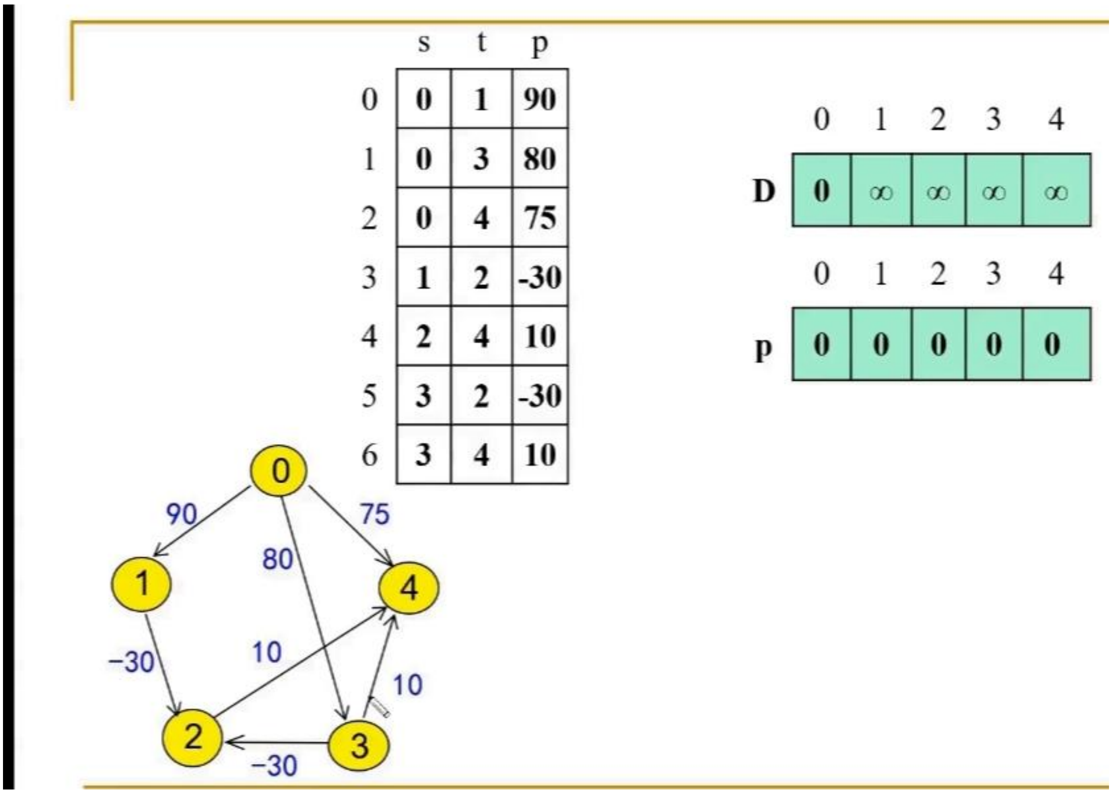
应用场景： 类似迪杰斯特拉，但是可以处理负权值

达成过程： 首先对应图将数据给上标号，确定我们的原点信息，在这之后，我们对所有的边进行松弛操作，（比如我们有一个点（b，10）和（c，15），10 和 15 代表的是目前他的最短距离，b 和 c 之间的距离为 3，那么我们就有 Vb+Vbc<Vc,此时我们用 Vb+Vbc 来替换 Vc,这就完成了这条边的松弛操作），如果说我们没有进行任何更新操作，那么算法结束，因为此时他本身的权值就是最短路径，如果说出现更新操作，那么就会重复如上步骤 n-1 次。如果最后还是有负环存在（负环是代表图中存在回路的总权重和为负数）那么代表该问题无解。因为如果存在负环，那么我们总会发现他的距离在每一圈中都会减小，无穷迭代。

存储方式： 通常采用二维数组，用第一列来代表

起 始 点 ， 第 二 列 代 表 结 束 点 ， 实 际 值 代 表 他 的 实 际 权 值 ， 与 迪 杰 斯 特 拉 相 同 他 也 会 使 用 两 个 数 组 来 分 别 存 储 它 的 当 前 最 短 距 离 和 上 一 个 节 点 的 名 称 。

例 题 ： 求 解 图 中 每 个 点 到 0 点 的 最 短 距 离



如图所示，我们可以直接把它的数据结构画一下，用 D 来存储距离，用 p 来存储上一节点，对应所有的 D 中的值我们都用 S 对应的索引在 D 上的值与 P 相加，然后与 D 中原有值进行对比，如果小于则进行更新，只要存在哪怕一次更新，那么我们就会继续下一次迭代，直到 n-1 次位置（n 为顶点个数）。

伪代码如下

```
//input: a graph G=(V,E) with a source vertex s
//output: an array d[0..|V|-1], indexed with V, d[v] is the
//length of shortest path from s to v
    d[s] ← 0
    for each v ∈ V – {s}
        do d[v] ← ∞

    for i ← 1 to |V| – 1
        do for each edge (u, v) ∈ E
            do if d[v] > d[u] + w(u, v)
                then d[v] ← d[u] + w(u, v)

    for each edge (u, v) ∈ E
        do if d[v] > d[u] + w(u, v)
            then report that a negative-weight cycle exists
```

Tir

从这里我们可以看到，他首先是规定 **d[s]**为原点，把距离设置为 **0**，然后对于所有没有经过的点的距离都初始化为无穷，在这之后，我们遍历所有的顶点，并对其进行放松，这个操作会进行 **n-1** 次，如果说新的距离更小那么就会更新最短距离。