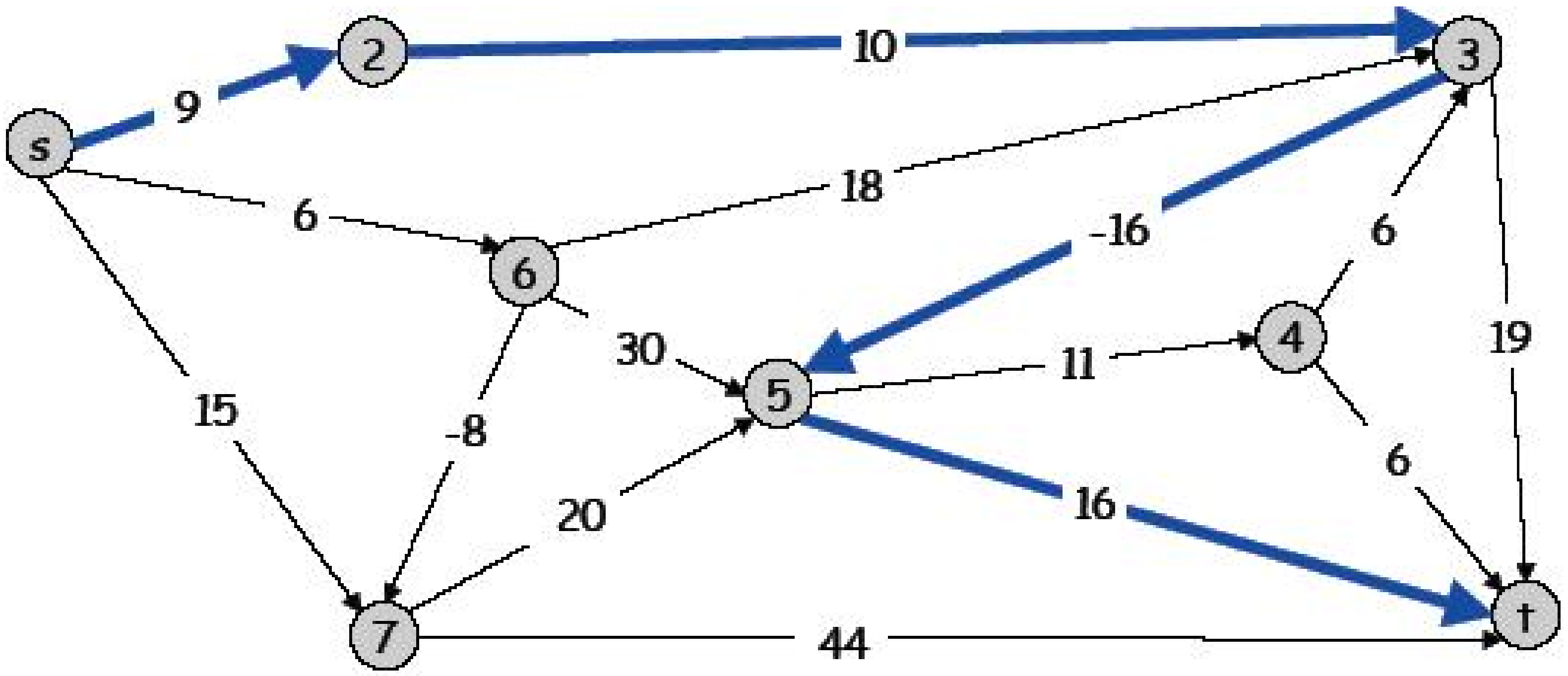
6.8 图中的最短路径

- 问题描述
- 令G = (V, E)是一个有向图，假定每条边(i,j)有一个相关的权C_{ij}, 表示从结点i直接到结点j的费用。
- 这里费用可能为负：比如，结点可能是表示在一个金融背景中的代理，在交易中我们*从代理i买入*，*然后卖给代理j*.

图中的最短路径

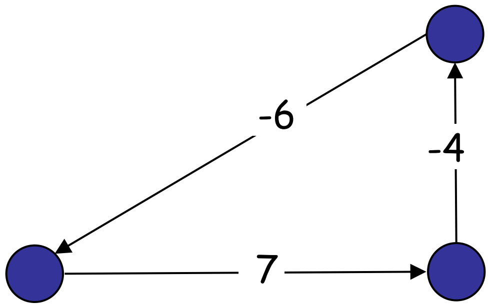


图中的最短路径

- 实际情形中，一个负圈与金融问题中一个有利可图的交易序列相对应,

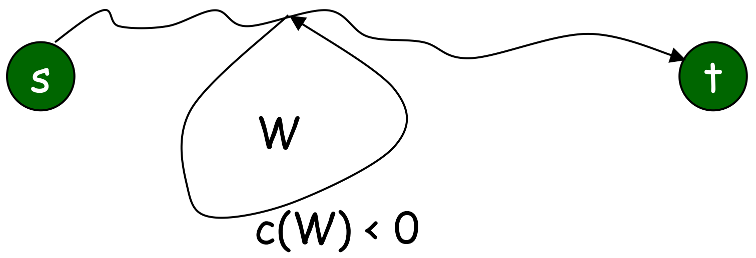
  负圈可以看成一个好的盈利机会 $c_{ij} + P_i - P_j$

- 负圈

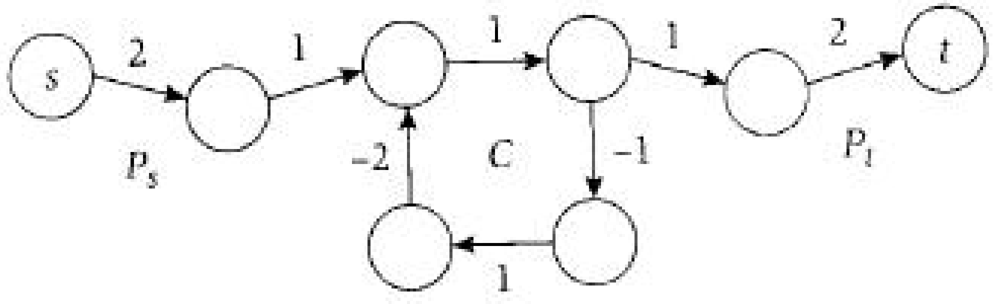


图中的最短路径

- 如果存在负圈，s-t中不存在“**最短**”路径



在这里，没有负圈的假定下考虑最小费用s-t路径才有意义



可以找到任意负费用！

图中的最短路径

我们将集中考虑下面两个相关问题

1. 给定一个带权图G,确定G是否有负圈，即一个有向圈C使得 $\sum_{ij \in C} c_{ij} < 0$

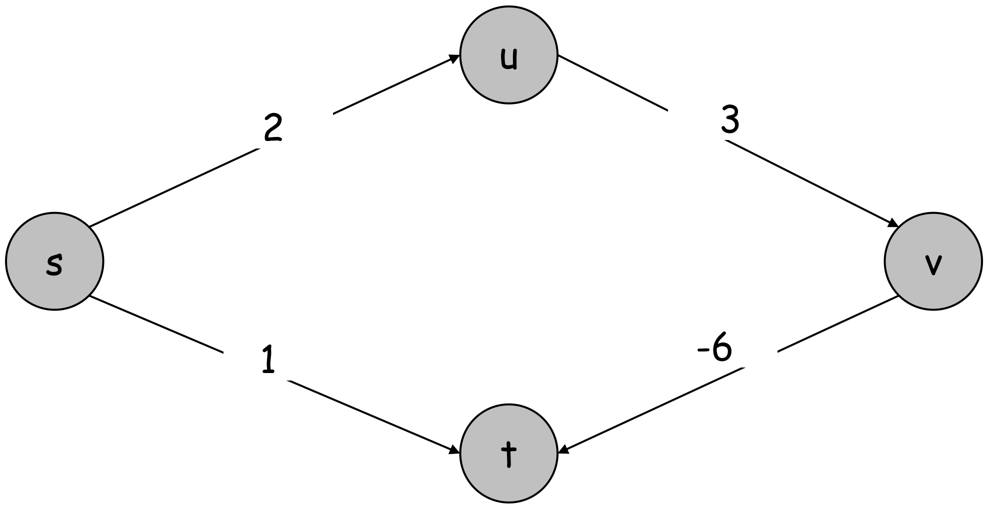
2. 如果这个图**没有负圈**，找一条从始点s到终点t的路径P使得路径具有最小总费用：

$$\sum_{ij \in P} c_{ij} \quad \text{---}\rangle$$ **最短路径问题**

设计与分析算法

- 采用Djkstra算法类似的策略：贪心的把到源点s最短的结点加入到核心集合S中

- 存在负的边权重，是否依然有效？

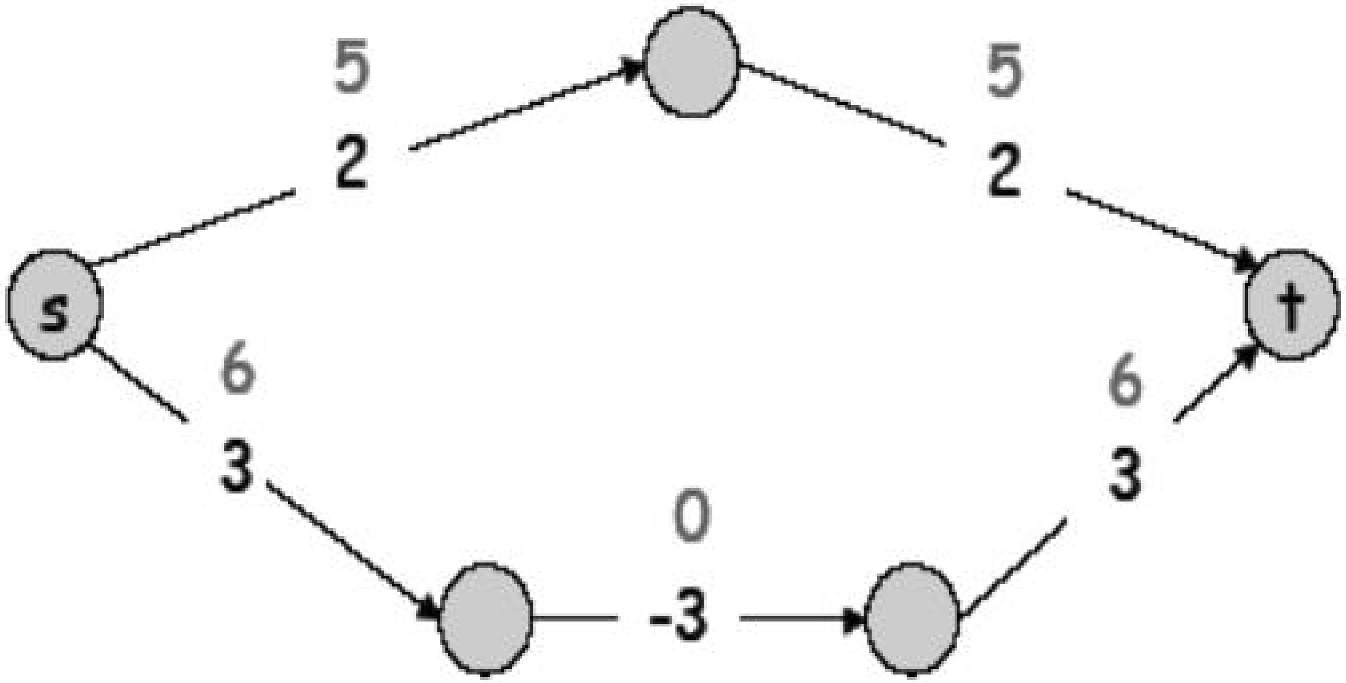


设计与分析算法

- 从上面实例可以发现： 一条路径从一条贵的边开始， 接着用负费用的边进行补偿； 比一开始从一条便宜的边出发， 路径可能要更短
- *解决办法*： 把负权重变成非负权重， 也就是对每条边(i,j), 让$c_{ij}' = c_{ij} + M$

  这样可行吗？

设计与分析算法



路径中边的数目也是一个重要因素

设计与分析算法

- 动态规划方法

- Bellman-Ford最短路径算法

- 定理6.22 如果G没有负圈，那么存在一条从s到t的简单的最短路径(没有重复结点)，因此它至多有n-1条边。

设计与分析算法

- OPT(i, v) 表示至多使用i条边的v-t最短路径的最小费用,我们目的是计算s到t的最短路径：OPT(n-1,s)
- 多重选择， OPT(i, v)

i. 如果路径P至多用i-1条边，那么OPT(i, v) =OPT(i-1,v)

ii. 如果路径P用i条边,并且第一条边是(v,w),那么余下的是w-t路径至多使用i-1条边：

$$OPT(i,v) = c_{vw} + OPT(i-1,w)$$

定理6.23

$$OPT(i,v)=\begin{cases}0 & \text{if } i=0\\ \min\left\{OPT(i-1,v),\ \min\limits_{(v,w)\in E}\{OPT(i-1,w)+c_{vw}\}\right\} & \text{otherwise}\end{cases}$$

根据前面的结论，如果图中没有负圈，那么OPT(n-1, v) 就是最短的v-t路径长度。

设计与分析算法

- 算法

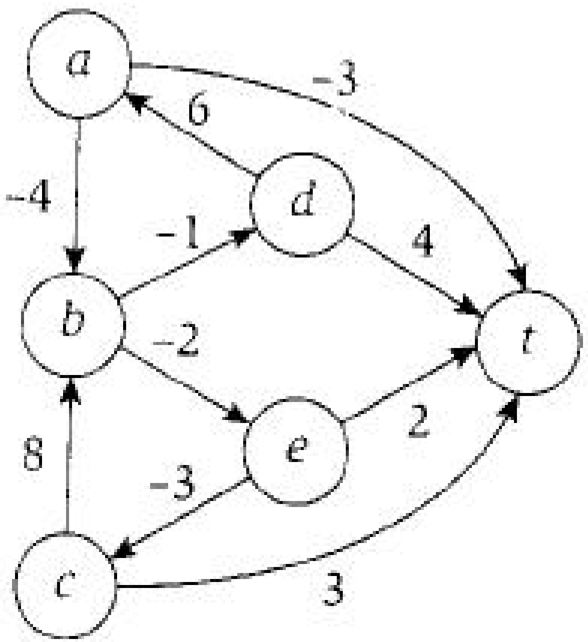
```
Shortest-Path(G, t) {
   foreach node v ∈ V
      M[0, v] ← ∞
   M[0, t] ← 0

   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] ← M[i-1, v]
      foreach edge (v, w) ∈ E
         M[i, v] ← min { M[i, v], M[i-1, w] + c_vw }
}
```

设计与分析算法

- 时间，空间复杂度？
- 时间复杂度：$O(mn)$
- 空间复杂度：$O(n^2)$
- 如何寻找最短路径？

每次记录后继元素

- 例子：Shortest-Path 算法



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	−3	−3	−4	−6	−6
b	∞	∞	0	−2	−2	−2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

运行时间估计的改进

- 定理6.25 前面的Shortest-Path方法可以改进到*O(mn)*时间。
- Pf. 运行时间的界：$O(n\sum_{v\in V} n_v)$

  其中 $\sum_{v\in V} n_v = 2m$

存储需求的改进

- 我们不对每个值i记录M[i,v],而是对每个结点v更新一个值M[v]，即我们至今已经找到的从v到t的最短路径的长度。

$$M[v]=\min(M[v],\min_{w\in V}(c_{vw}+M[w]))$$

设计与分析算法

- 命题6.26 在整个算法中，M[v]是从v到t的某条路径的长度，且在*i轮更新*后M[v]的值**不大于**从v到t*至多使用i条边*的最短路径的长度。

- 我们只需存储一个在全部结点上索引的M数组，需要O(n)的工作存储空间。

找到最短路径&存在性

- 为了帮助复原这条最短路径，记录每个结点通向终点t的路径上它后面的第一个结点；*只要距离M[v]更新，我们就更新这个值(first[v])。*

设计与分析算法

P “指针图”， 结点是v, 边(v,first[v])

- 命题6.27 如果指针图P包含一个圈C,那么这个圈一定有负费用。
- Pf. 设$v_1,v_2,\ldots,v_k$是一个圈。那么

所以，

$$M[v_i] \geq c_{v_i v_{i+1}} + M[v_{i+1}]$$

$$0 > \sum_{i=1}^{k-1} c_{v_i v_{i+1}} + c_{v_k v_1}$$

设计与分析算法

- 若G没有负圈，那么指针图没有圈，而且终点是t，算法结束的时候就得到v到t的一条最短路径。

- 命题6.28 假设G没有负圈，考虑算法终止时的指针图P. 对每个结点v,在P中从v到t的路径是G中的一条最短v-t路径。

- **算法终止信号**：某一次迭代中**所有**的M[v]值都保持不变。

设计与分析算法

```
Push-Based-Shortest-Path(G, s, t) {
   foreach node v ∈ V {
      M[v] ← ∞
      successor[v] ← φ
   }

   M[t] = 0
   for i = 1 to n-1 {
      foreach node w ∈ V {
      if (M[w] has been updated in previous iteration) {
         foreach node v such that (v, w) ∈ E {
            if (M[v] > M[w] + c_vw) {
               M[v] ← M[w] + c_vw
               successor[v] ← w
            }
         }
      }
      If no M[w] value changed in iteration i, stop.
   }
}
```

* 图中的负圈

- 如何确定图中是否包含负圈?

- 如何在包含负圈的图中找到负圈?

图中的负圈

- 命题6.31 如果G中没有负圈，那么对所有的结点v和所有的i>=n, OPT(i,v)=OPT(n-1,v).

- 命题6.33 如果G有n个结点且OPT(n,v)!=OPT(n-1,v),那么一条从v到t的费用OPT(n,v)的路径P包含一个圈C,并且C有负费用。

图中的负圈

- 定理6.34 如果G中存在这样一个负圈，存在算法找到一个负圈，并且运行在O(mn)时间。

图中的负圈

- 增加一个终点t