- 教材讨论
  - TC第24章

问题1：Bellman-Ford算法

- 我们为什么可以假设要找的最短路径中一定不含圈？
  - positive-weight cycle
  - 0-weight cycle
  - negative-weight cycle

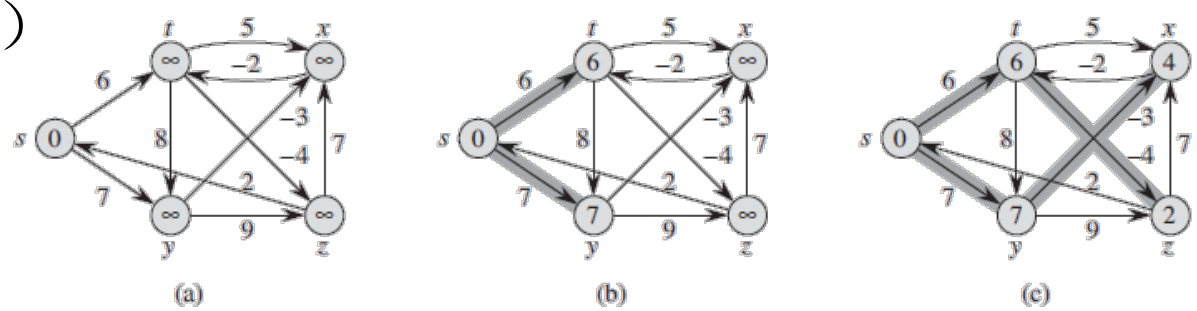
- 你理解RELAX了吗？
- 这一章中的那么多算法，它们的本质区别在哪里？

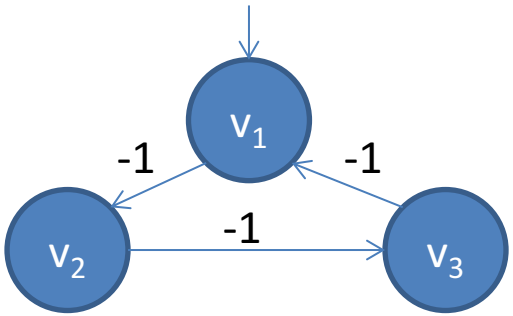
```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2      v.d = u.d + w(u, v)
3      v.π = u
```

问题1：Bellman-Ford算法 (续)

- 你能结合这个例子，解释算法的步骤吗？
- RELAX的次数和顺序？
- 循环不变量是什么？（第i次循环得到了什么重要的结果？）
- 为什么一定能发现negative-weight cycles？

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

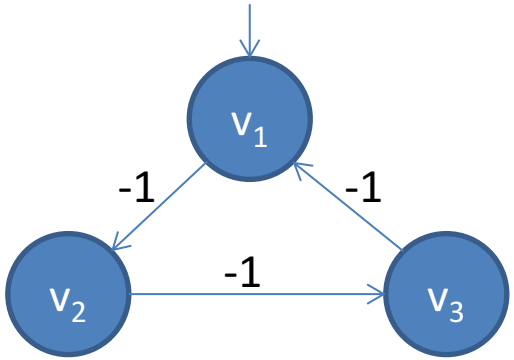


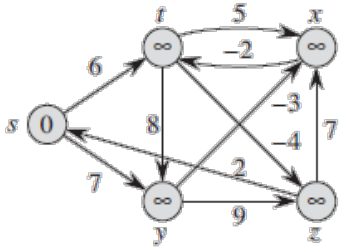


问题1：Bellman-Ford算法 (续)

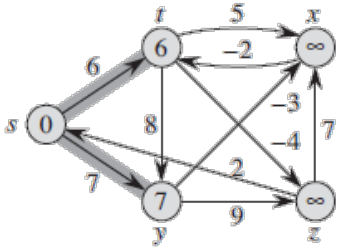
- 你能否改造BF算法，让它输出图中存在的negative-weight cycles？

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

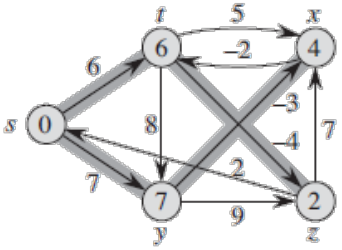




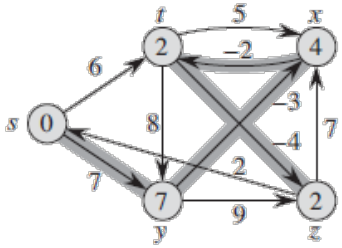
(a)



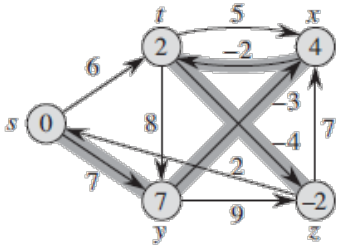
(b)



(c)



(d)



(e)

问题1：Bellman-Ford算法 (续)

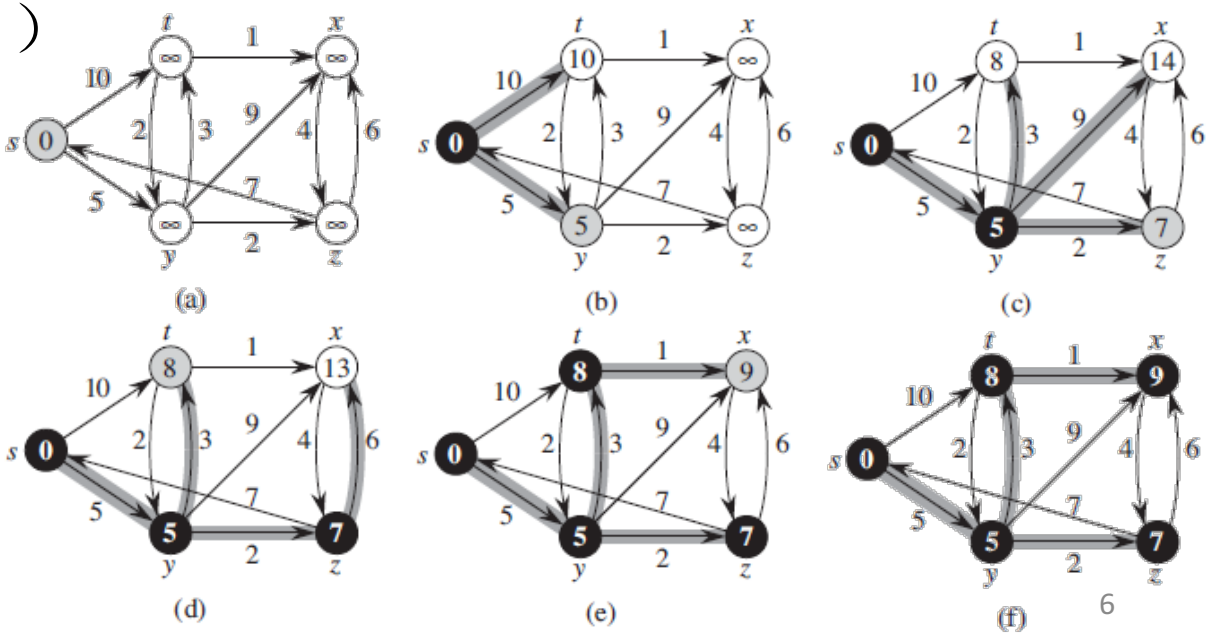
- 与BF算法相比，这个新算法的RELAX的次数和顺序是什么？
- 它的改进和局限分别是什么？
- 你能解释它取得改进的原因吗？（循环不变量是什么？）

```
DAG-SHORTEST-PATHS(G, w, s)
1  topologically sort the vertices of G
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  for each vertex u, taken in topologically sorted order
4       for each vertex v ∈ G.Adj[u]
5            RELAX(u, v, w)
```

问题2：Dijkstra算法

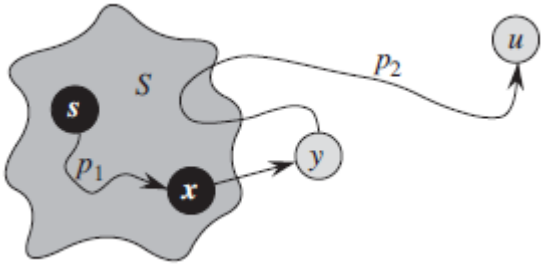
- 你能结合这个例子，解释算法的步骤吗？
- RELAX的次数和顺序？
- 循环不变量是什么？（每次循环得到了什么重要的结果？）

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```



问题2：Dijkstra算法

- 你能简要证明循环不变量吗？




```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

$$\left.\begin{aligned} y.d &= \delta(s,y) \\ &\le \delta(s,u) \\ &\le u.d \\ u.d &\le y.d \end{aligned}\right\} \; y.d = \delta(s,y) = \delta(s,u) = u.d$$

问题3：最短路问题的应用

- 为什么我们很少讨论single-pair shortest-path problem？

问题3：最短路问题的应用 (续)

- 怎样更新设备最合算？

	2016	2017	2018	2019	2020
买入价	11	12	13	14	14

	1年后	2年后	3年后	4年后	5年后
卖出价	4	3	2	1	0

	第1年	第2年	第3年	第4年	第5年
保养费	5	6	8	11	18

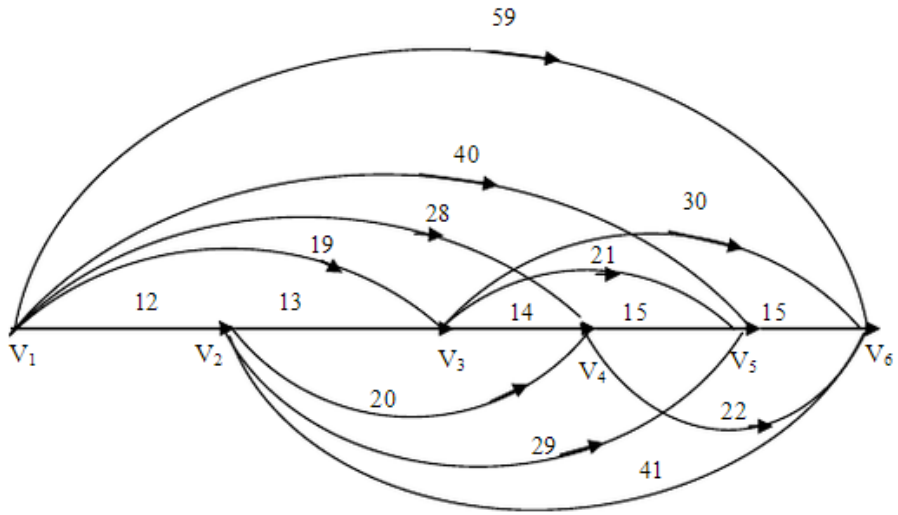
问题3：最短路问题的应用 (续)

- 怎样更新设备最合算？

	2016	2017	2018	2019	2020
买入价	11	12	13	14	14

	1年后	2年后	3年后	4年后	5年后
卖出价	4	3	2	1	0

	第1年	第2年	第3年	第4年	第5年
保养费	5	6	8	11	18



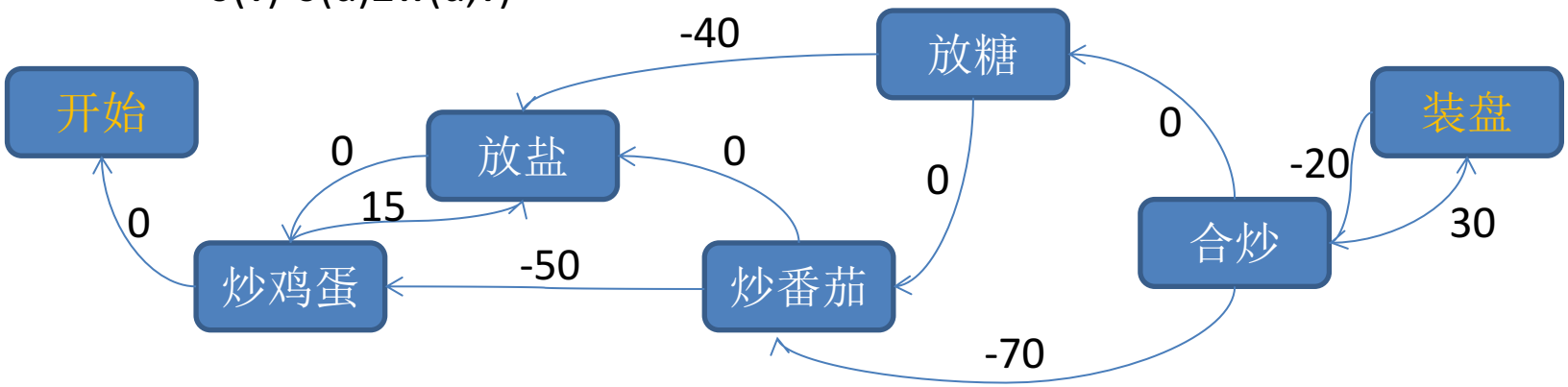
问题3：最短路问题的应用 (续)

- 你会做番茄炒蛋吗？
  - 顺序：先把鸡蛋炒熟，再炒番茄，再合起来炒
  - 要点：
    - 炒鸡蛋至少要50秒才能熟
    - 鸡蛋下锅的15秒内要放盐才能入味
    - 炒番茄至少要70秒才能出汁
    - 炒番茄时要放糖，但和之前放盐的时间要间隔至少40秒
    - 合起来炒至少要20秒才能混味，但不能超过30秒，否则就稀烂了
    - ……
- 如果一个新手要做番茄炒蛋，你能为他列出一份详细的时间表吗？（第几秒时做什么）

问题3：最短路问题的应用 (续)

- 你会做番茄炒蛋吗？
  - 顺序：先把鸡蛋炒熟，再炒番茄，再合起来炒
  - 要点：
    - 炒鸡蛋至少要50秒才能熟
    - 鸡蛋下锅的15秒内要放盐才能入味
    - 炒番茄至少要70秒才能出汁
    - 炒番茄时要放糖，但和之前放盐的时间要间隔至少40秒
    - 合起来炒至少要20秒才能混味，但不能超过30秒，否则就稀烂了

$\delta(v)-\delta(u)\le w(u,v)$



问题3：最短路问题的应用 (续)