- 教材讨论
  - TC第25章

问题1：简单的动态规划法

- 你能解释这个算法是如何实现动态规划三步骤的吗？
  1. Characterize the structure of an optimal solution.
  2. Recursively define the value of an optimal solution.
  3. Compute the value of an optimal solution in a bottom-up fashion.
- 如何在计算距离的同时，记录最短路？

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \ , \\ \infty & \text{if } i \neq j \ . \end{cases}$$

$$\begin{aligned} l_{ij}^{(m)} &= \min\left(l_{ij}^{(m-1)}, \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\}\right) \\ &= \min_{1 \le k \le n} \{l_{ik}^{(m-1)} + w_{kj}\} \ . \end{aligned}$$

```
EXTEND-SHORTEST-PATHS(L, W)
n = L.rows
let L' = (l'_ij) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

问题2：Floyd-Warshall算法

- 你能解释这个算法是如何实现动态规划三步骤的吗？
  1. Characterize the structure of an optimal solution.
  2. Recursively define the value of an optimal solution.
  3. Compute the value of an optimal solution in a bottom-up fashion.
- 如何在计算距离的同时，记录最短路？
- 与上一个算法相比，为什么能够提高性能？

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0\,, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1\,. \end{cases}$$

```
FLOYD-WARSHALL(W)
n = W.rows
D^(0) = W
for k = 1 to n
    let D^(k) = (d_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

问题3：Johnson算法

- 你能解释这个算法的基本思路吗？

```
JOHNSON(G, w)
compute G′, where G′.V = G.V ∪ {s},
      G′.E = G.E ∪ {(s, v) : v ∈ G.V}, and
      w(s, v) = 0 for all v ∈ G.V
if BELLMAN-FORD(G′, w, s) == FALSE
      print "the input graph contains a negative-weight cycle"
else for each vertex v ∈ G′.V
            set h(v) to the value of δ(s, v)
                  computed by the Bellman-Ford algorithm
      for each edge (u, v) ∈ G′.E
            ŵ(u, v) = w(u, v) + h(u) − h(v)
      let D = (d_uv) be a new n × n matrix
      for each vertex u ∈ G.V
            run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
            for each vertex v ∈ G.V
                  d_uv = δ̂(u, v) + h(v) − h(u)
      return D
```

为什么要新增一个点？
一定要新增吗？

为什么reweighting要搞这么复杂？
能不能：
所有w都加上一个足够大的正数？

问题4：炼钢厂选址

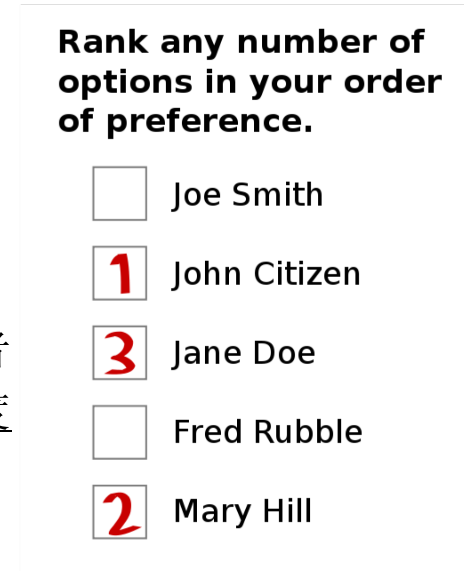
- 四川计划投资新建一个炼钢厂，集中冶炼从省内各城市开采的铁矿石。从降低生产成本的角度考虑，你认为炼钢厂应选址哪座城市？

问题5：救援机库选址

- 江苏计划采购一架救援直升机，承担省内各城市突发灾害的救援任务。从缩短救援时间的角度考虑，你认为直升机日常应停放在哪座城市？（请分别为泰州、无锡代言）

问题6：Schulze投票法

- 谁该被选为总统？
  1. 每个选民对所有候选人进行排序
  2. 将每对候选人之间的相对支持数表示成矩阵
  3. 候选人X到候选人Y的一条优势序列X…Y：
     - 相邻的每对候选人，都满足前者的相对支持数大于后者
     - 所有前者的相对支持数的最小值，称作优势序列的强度
  4. 候选人X相对于候选人Y的优势p[X, Y]：
     - 所有X-Y优势序列强度的最大值
  5. 候选人X的当选条件：
     - p[X,Y]≥p[Y,X] for every other Y
- 你能给出一种高效的算法实现吗？



Matrix of pairwise preferences

	d[*,A]	d[*,B]	d[*,C]	d[*,D]	d[*,E]
d[A,*]		20	26	30	22
d[B,*]	25		16	33	18
d[C,*]	19	29		17	24
d[D,*]	15	12	28		14
d[E,*]	23	27	21	31	

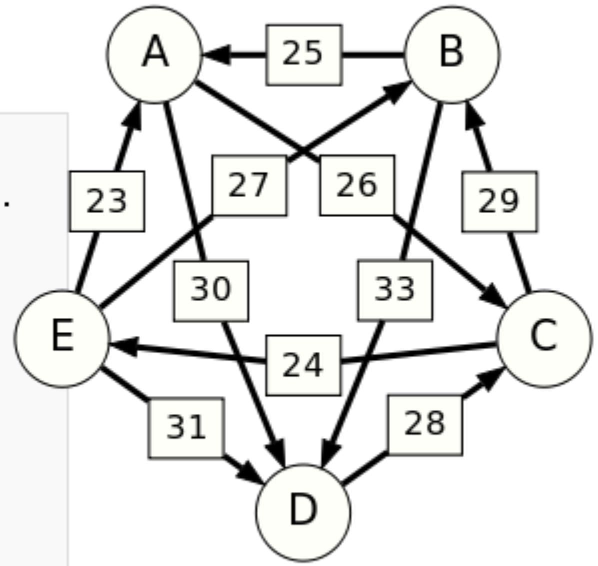
http://en.wikipedia.org/wiki/Schulze_method

问题6：Schulze投票法 (续)

```
# Input: d[i,j], the number of voters who prefer candidate i to candidate j.
# Output: p[i,j], the strength of the strongest path from candidate i to candidate j.

for i from 1 to C
   for j from 1 to C
      if (i ≠ j) then
         if (d[i,j] > d[j,i]) then
            p[i,j] := d[i,j]
         else
            p[i,j] := 0

for i from 1 to C
   for j from 1 to C
      if (i ≠ j) then
         for k from 1 to C
            if (i ≠ k and j ≠ k) then
               p[j,k] := max ( p[j,k], min ( p[j,i], p[i,k] ) )
```



FLOYD-WARSHALL(W)

$n = W.rows$

$D^{(0)} = W$

for $k = 1$ **to** n

  let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix

  for $i = 1$ **to** n

    for $j = 1$ **to** n

      $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

return $D^{(n)}$

问题7：如何实现社交网络的搭讪功能

- 功能需求：快速找到与目标账户之间长度不超过k的所有人际关系链
  - 总规模：数千万账户
  - 单次响应时间：秒级
- 请给出一种实际可行的解决方案