最短通路问题

离散数学–图论初步

南京大学计算机科学与技术系

回顾

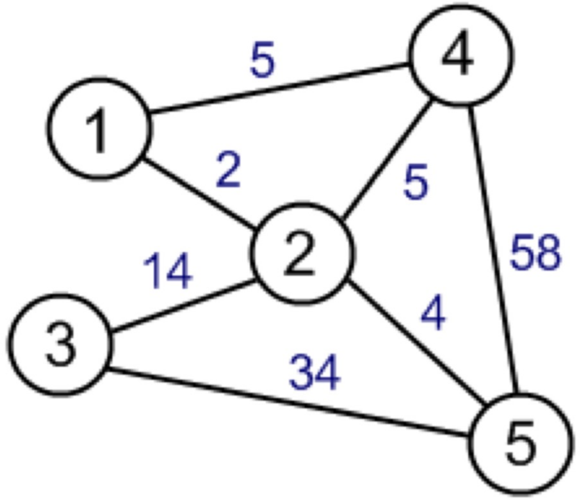
- 欧拉图
  - 充要条件
  - 构造欧拉回路的Fleury算法
- 哈密尔顿图
  - 必要条件与充分条件
  - 哈密尔顿图的应用

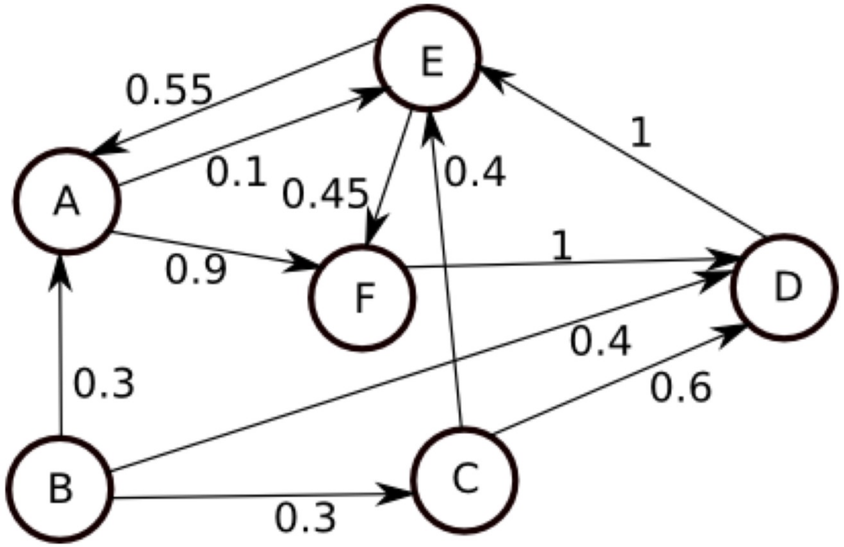
提要

- 带权图与最短通路
- 单源点最短通路问题 – Dijkstra 算法
- 所有点对间的最短通路问题 – Floyd-Washall 算法
- 旅行商问题（TSP）

带权图与最短通路问题

- 带权图：三元组 (V,E,W)，(V,E)是图，W是从E到非负实数集的一个函数。$W(e)$表示边e的权。
- 一条通路上所有边的权的和称为该通路的长度。





带权图与最短通路问题

- 带权图中两点之间长度最小的通路称为两点之间的**最短通路**。
  - 不一定是唯一的。

- **单源点最短路问题**

  给定带权图 $G(V,E,W)$并指定一个源点，确定该源点到图中其它任一顶点的最短路（长度和路径）。

Dijkstra最短路径的算法思想(1956)

- 源点s到顶点v的最短路径若为$\underline{s \dots uv}$，则$\underline{s \dots u}$是s到u的最短路径。
- $(n-1)$条最短路径按照其长度的非减次序求得，设它们的相应端点分别为$u_1, \dots, u_{n-1}$，最短路径长度记为$d(s, u_i)$，$i = 1, \dots, n-1$.
- 假设前i条最短路径已知，第$(i+1)$条最短路径长度：

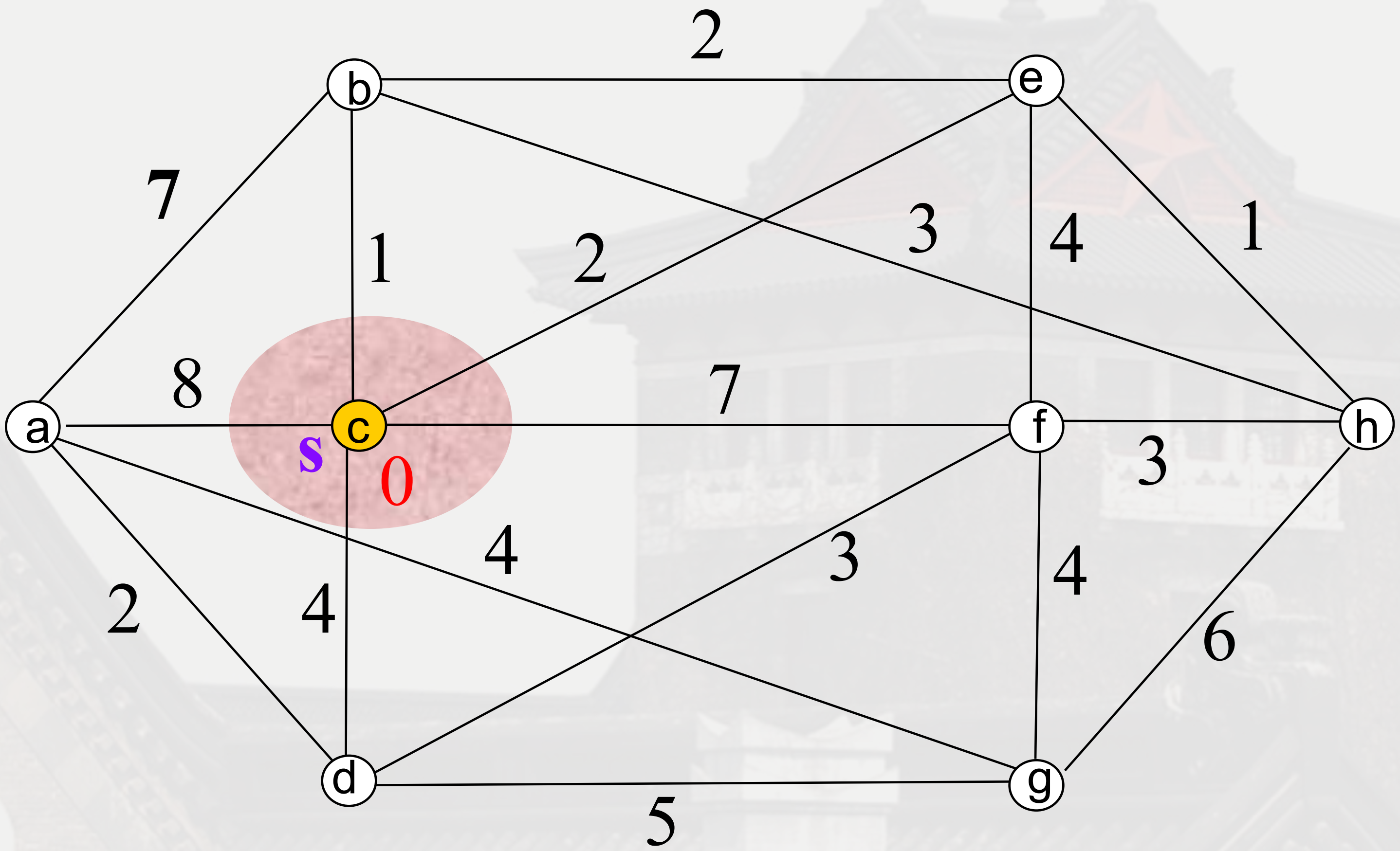
$$d(s, u_{i+1}) = \min\{d(s, u_j) + W(u_j, u_{j+1}) \mid j = 1, \dots, i\}$$

求最短路径的Dijkstra算法

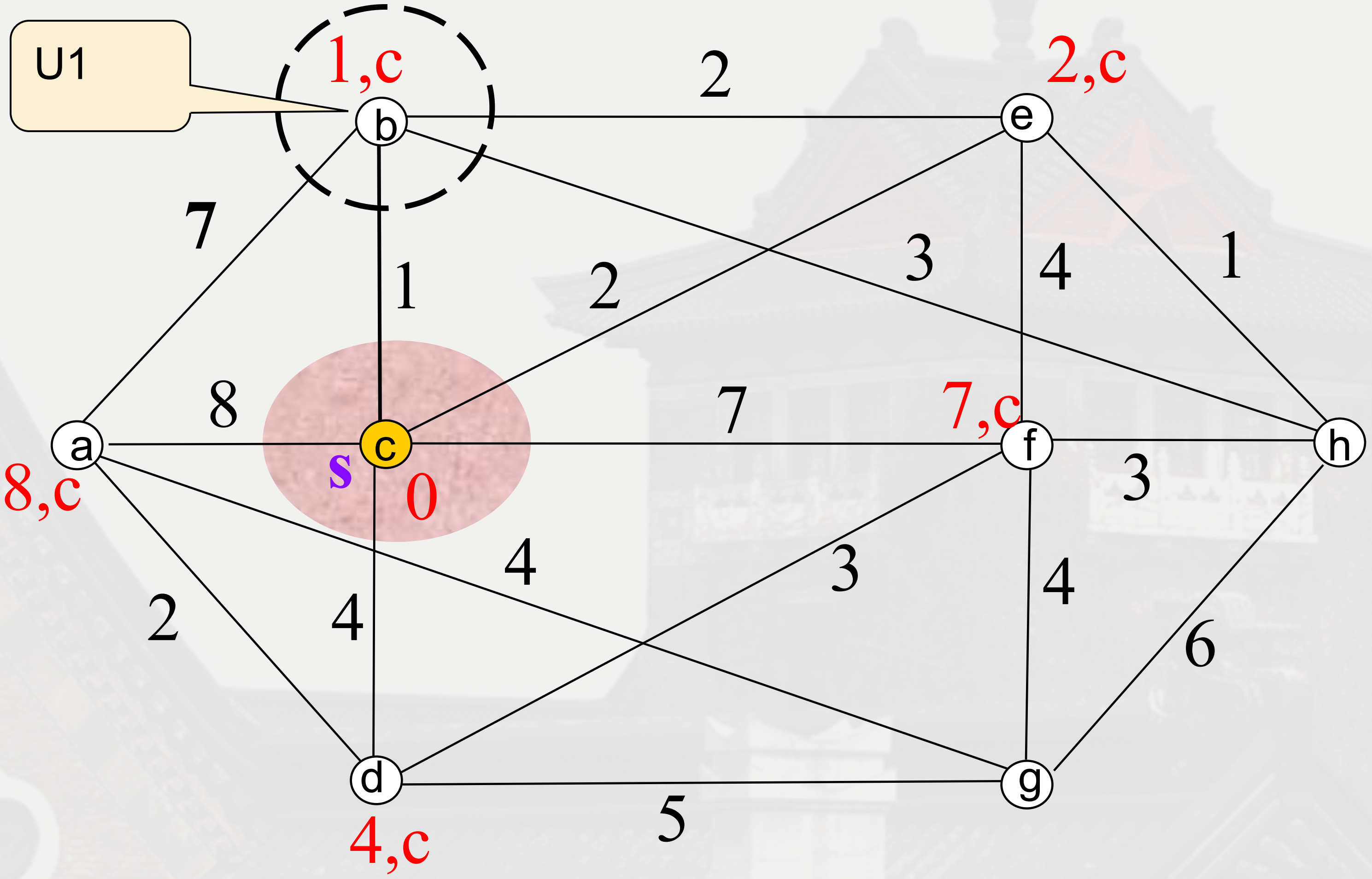
- 输入：连通带权图G，$|V_G|=n$, 指定顶点$s\in V_G$
- 输出：每个顶点v的标注$(L(v), u)$, 其中：
  - $L(v)$即从s到v的最短路径长度（目前可得的）
  - u是该路径上v前一个顶点。

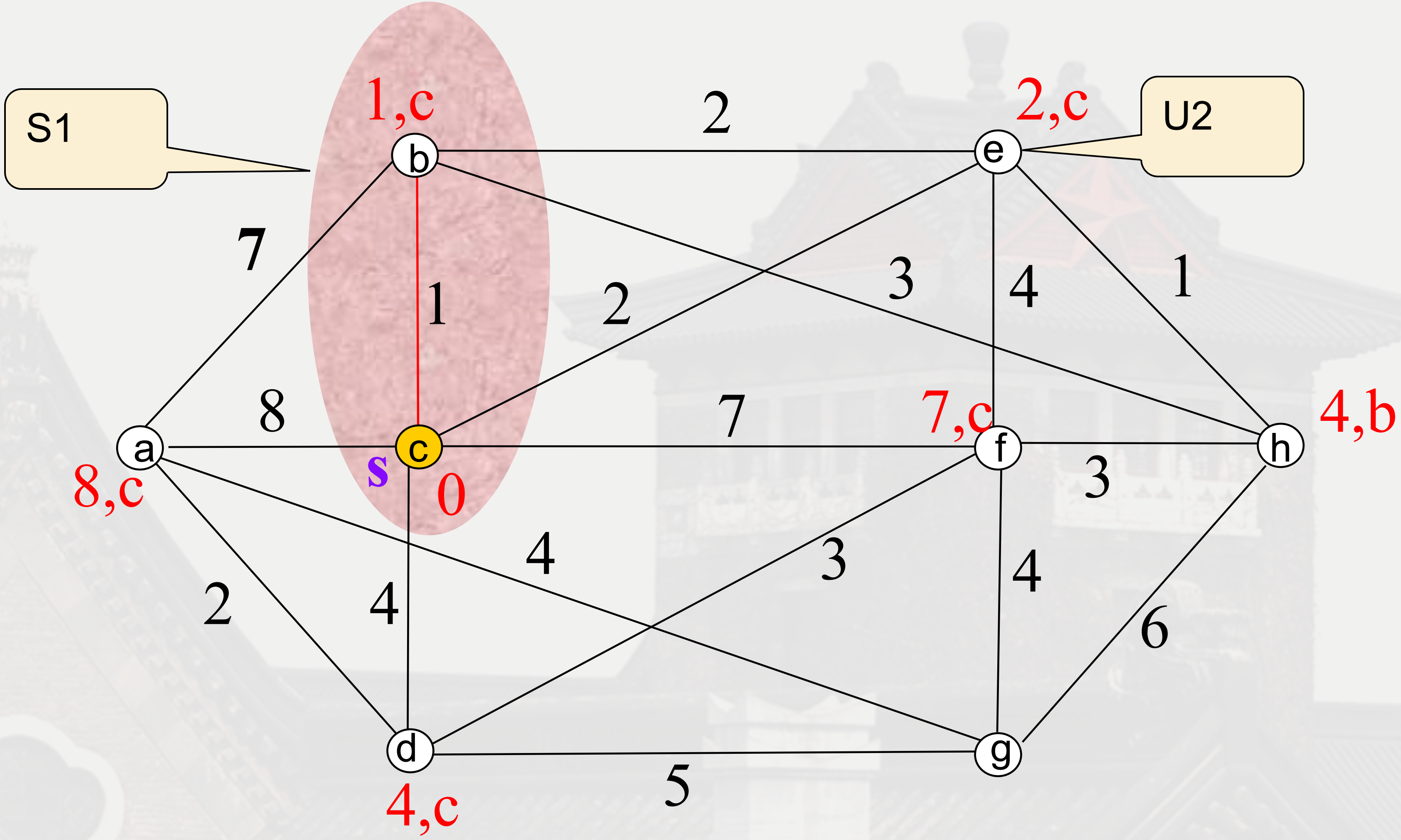
1. 初始化：$i=0$, $S_0=\{s\}$, $L(s)=0$, 对其它一切$v\in V_G$, 将$L(v)$ 置为∞。

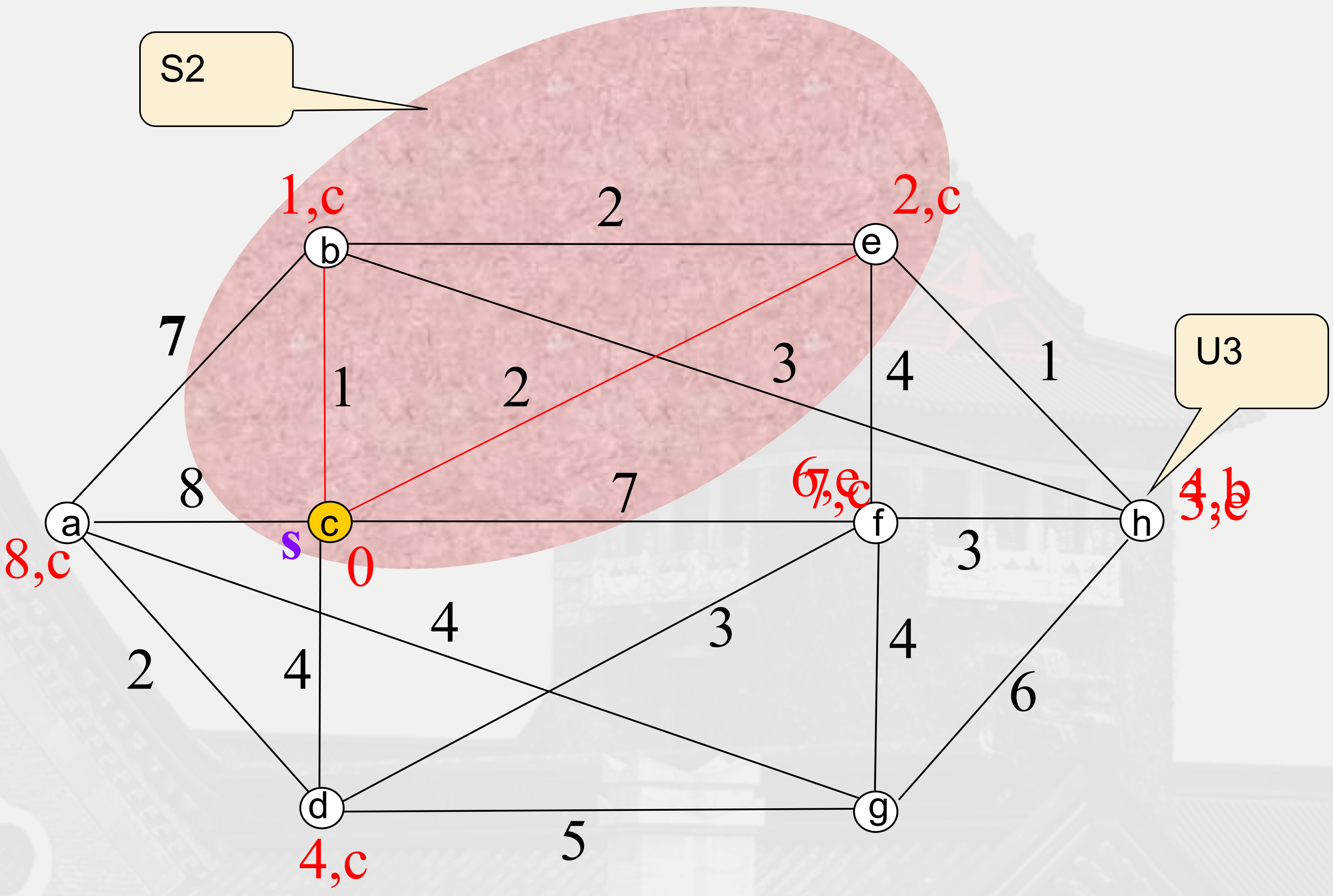
求最短路的一个例子



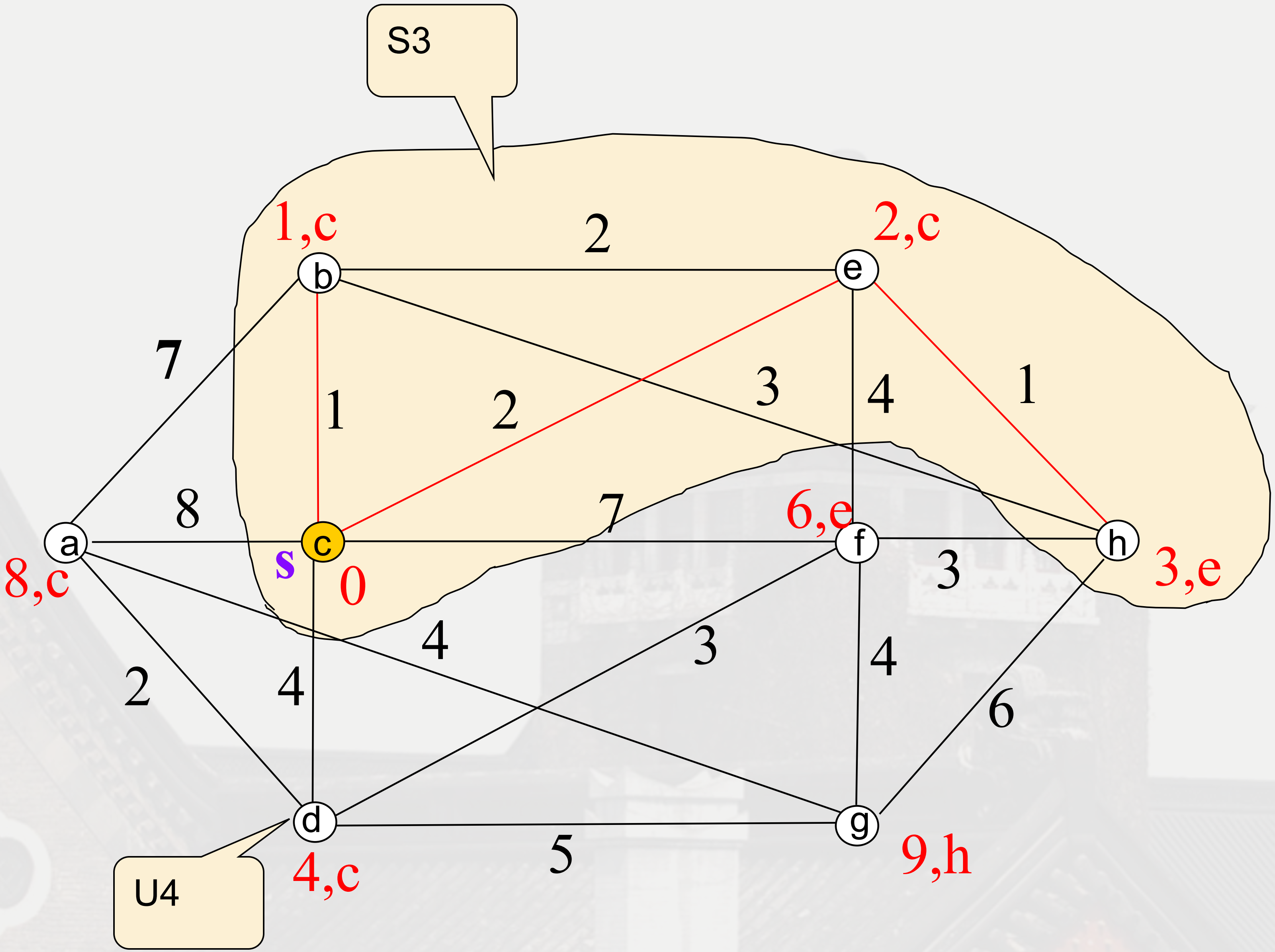
求最短路的一个例子



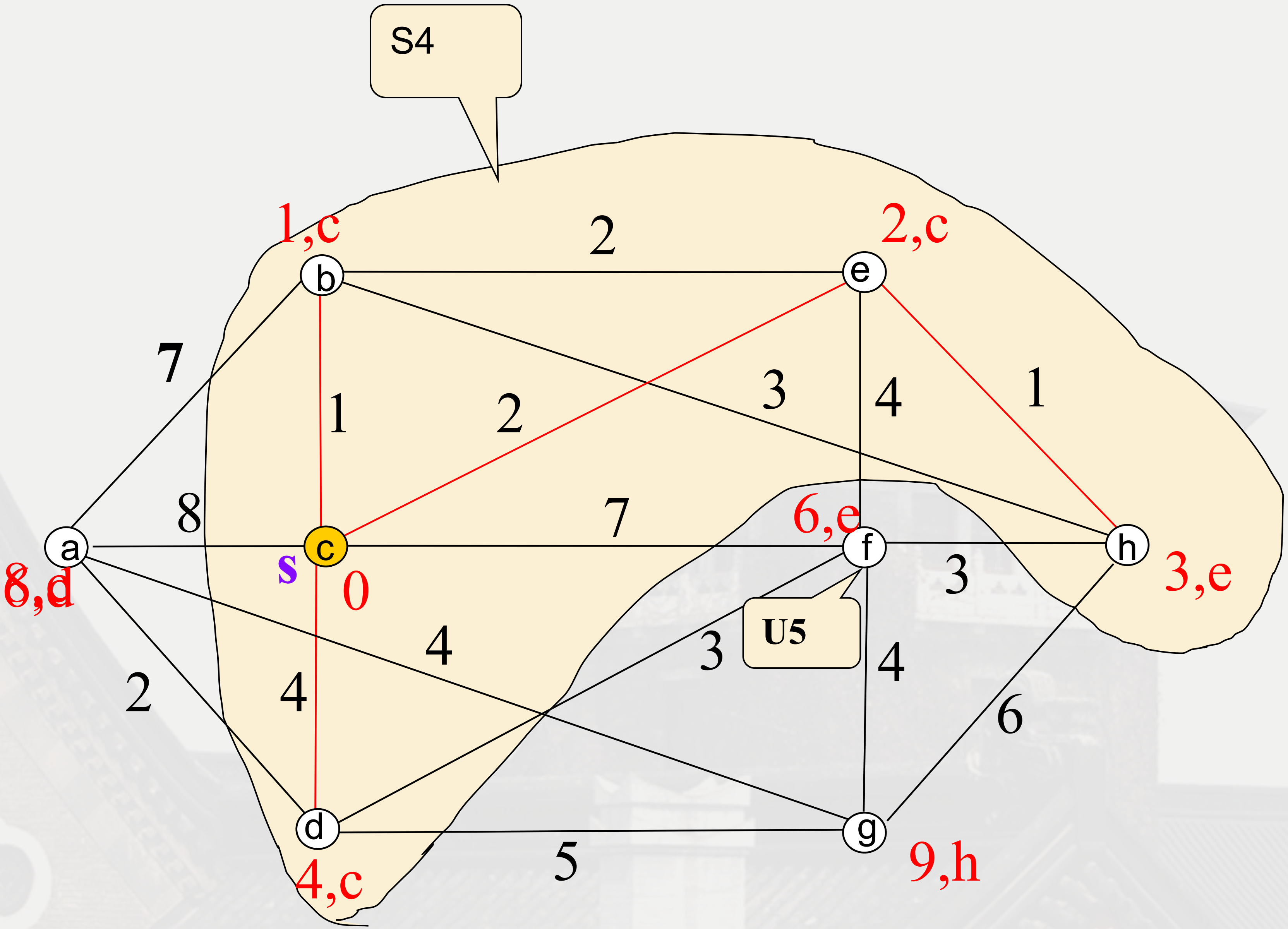

S1
1,c
2
2,c
U2
b
e
7
1
2
3
4
1
8
7
7,c
4,b
a
c
f
h
s
0
3
8,c
4
3
2
4
4
6
d
g
5
4,c



S2
U3
1,c
2,c
0
8,c
4,c
s
a
b
c
d
e
f
g
h

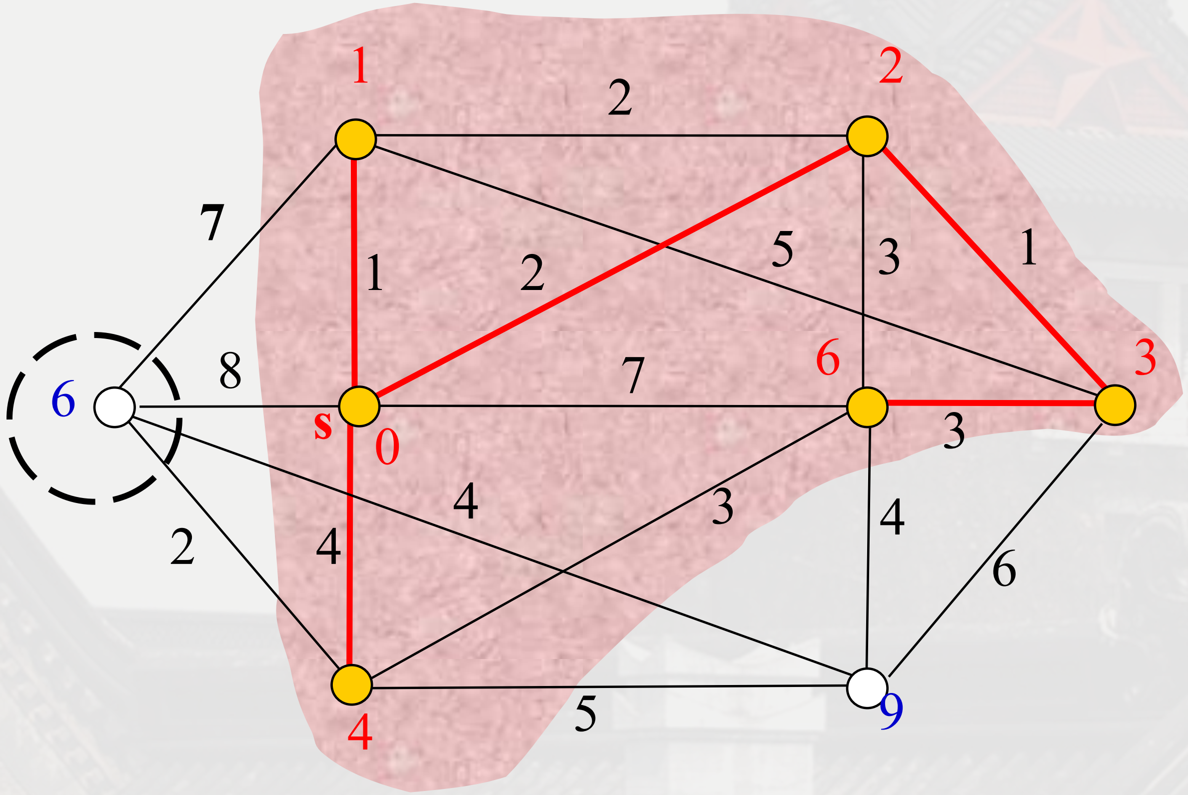


S3
1,c
2
2,c
b
e
7
1
2
3
4
1
8
7
6,e
a
c
f
h
8,c
s
0
3
3,e
4
3
2
4
4
6
d
g
5
U4
4,c
9,h

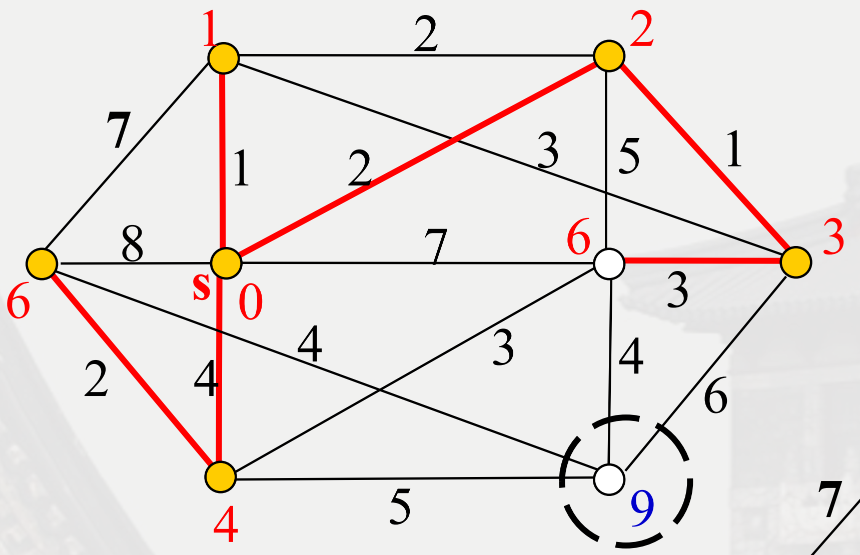


S4
U5
a
b
c
d
e
f
g
h
s
0
1,c
2,c
3,e
4,c
6,e
8,d
9,h
7
8
2
1
2
2
3
4
1
7
3
4
4
3
4
6
5

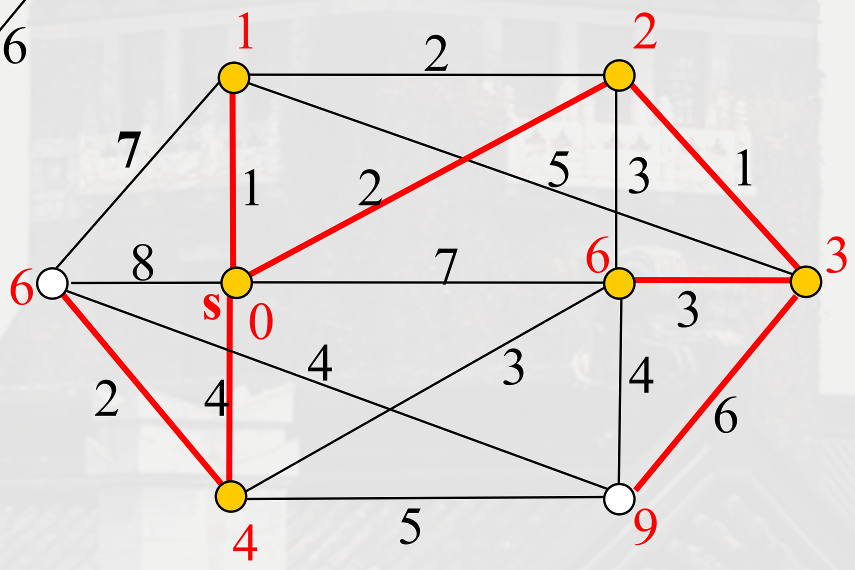
求最短路的一个例子(续)



求最短路的一个例子(续)

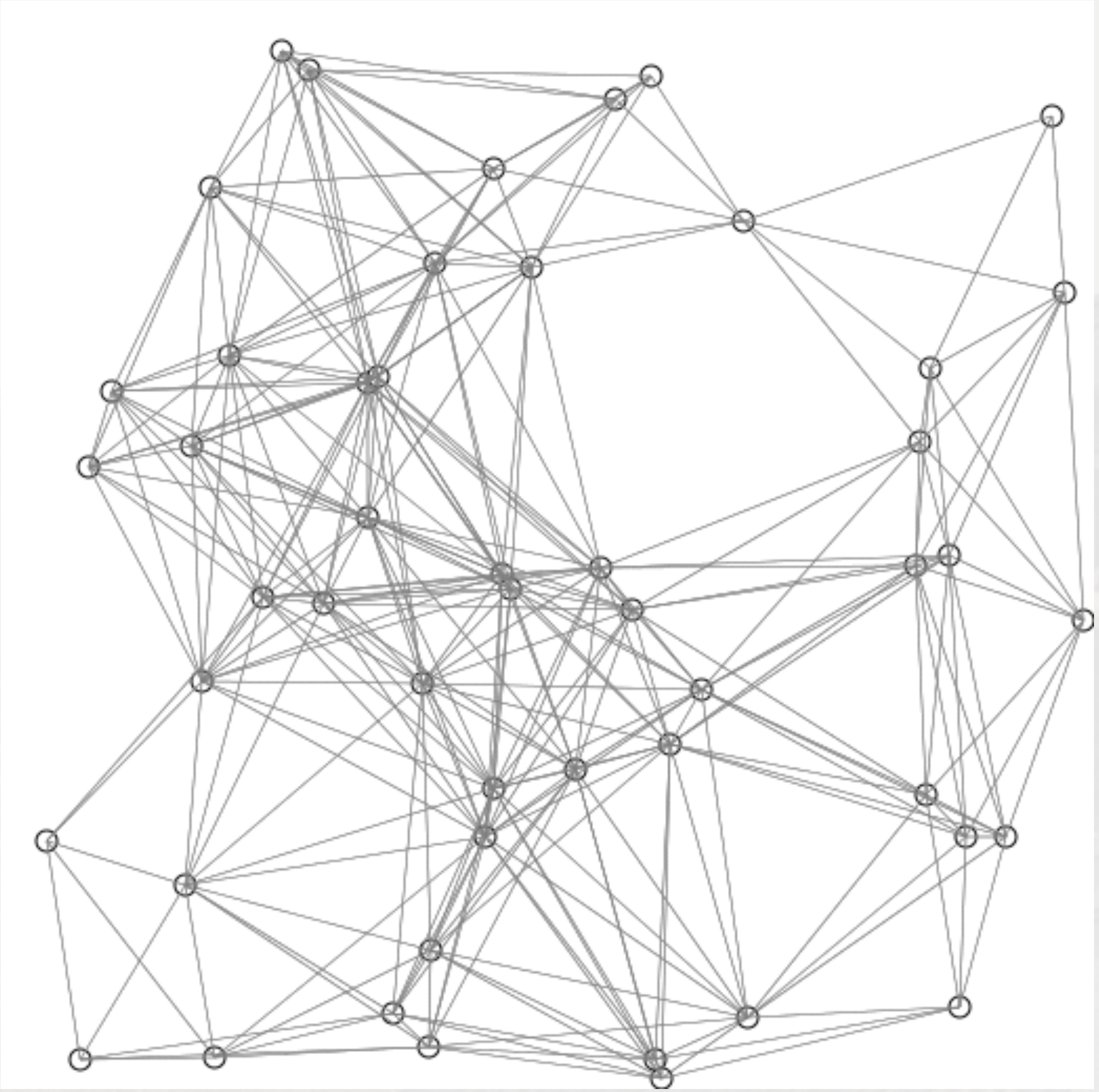


1
2
2
7
1
2
3
5
1
8
7
6
3
6
s
0
3
2
4
4
3
4
6
4
5
9

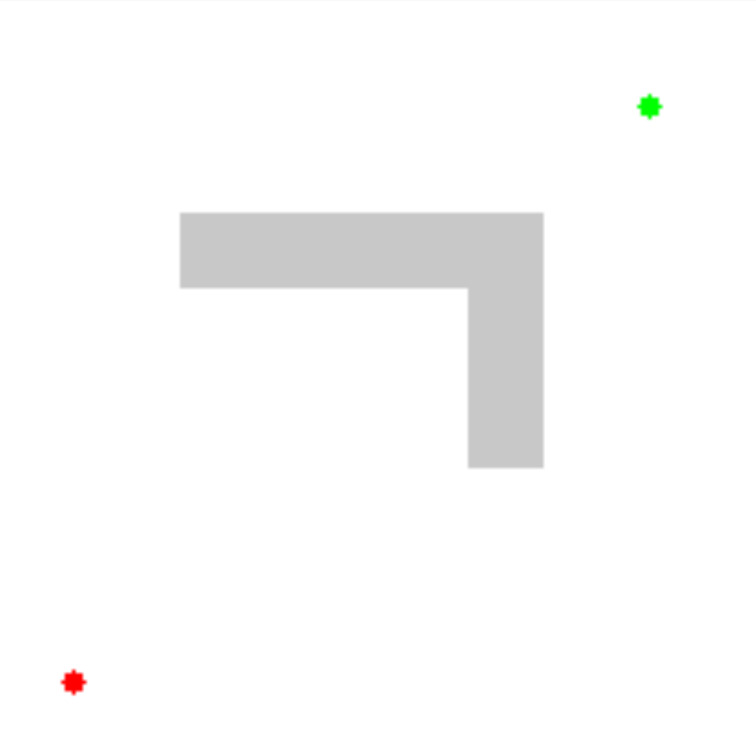


1
2
2
7
1
2
5
3
1
8
7
6
3
6
s
0
3
4
3
4
2
4
6
4
5
9

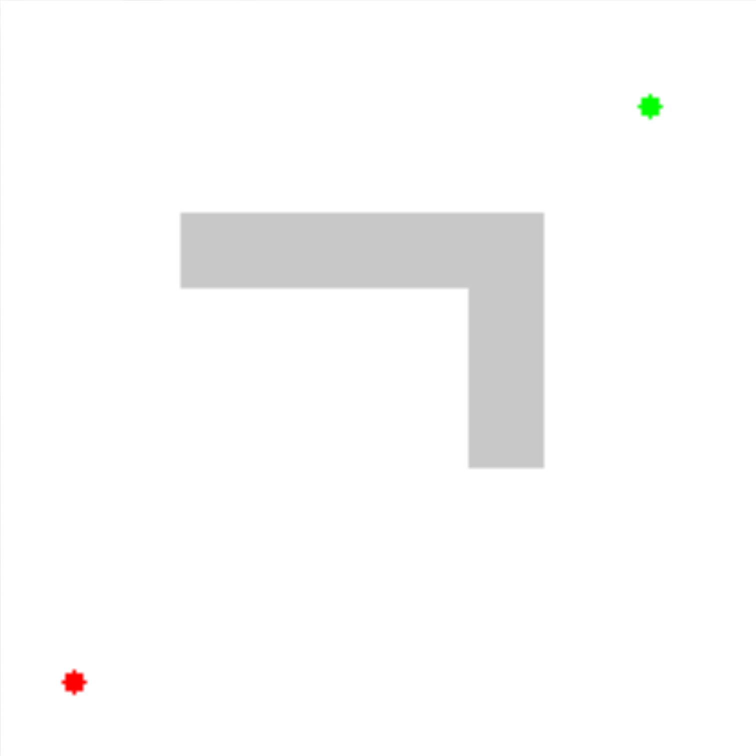
动图展示



动图展示



Dijkstra's algorithm



A* search algorithm

Dijkstra算法的描述

1. 初始化：$i=0$, $S_0=\{s\}$, $L(s)=0$, 对其它一切$v\in V_G$, 将$L(v)$ 置为∞。若$n=1$，结束。
2. $\forall v\in S_i'=V_G-S_i$, 比较$L(v)$和$L(u_i)+W(u_i, v)$的值 ($u_i\in S_i$)

   如果$L(u_i)+W(u_i, v)<L(v)$, 则将v的标注更新为$(L(u_i)+W(u_i, v), u_i)$，

   即： $L(v)=\min\{ L(v), \min_{u\in S_i}\{L(u)+W(u,v)\} \}$
3. 对所有S_i'中的顶点，找出具有最小$L(v)$的顶点v, 作为u_{i+1}
4. $S_{i+1} = S_i \cup \{u_{i+1}\}$
5. $i = i+1$; 若$i=n-1$, 终止。否则：转到第2步。

Dijkstra算法的分析

- 可终止性
  - 计数控制
- 正确性

  需证明当算法终止时

  - **L(v)=d(s, v)**对一切v成立。
  - 由标记中的诸u_i确定的路径是一条最短路径

    (这里$d(s,v)$是s到v的最短路径长度，即距离。)
- 复杂性
  - $O(|V|^2)$，对边稀疏的情况可进一步优化（在《算法设计与分析》课程中讨论）。

求所有结点间的最短距离？

- Floyd-Warshall 算法 --- $O(|V|^3)$
  - 理论上说与调用n次Dijkstra算法差不多，但实际上通常更为高效。
  - 复杂度的讨论留待《算法设计分析》课程。
- 与前面求传递闭包的算法本质上一样

求所有结点间的最短距离？

All-Pairs Shortest Paths

- 给定一个有向图 $G = (V, E)$, 边权重函数 $w: E \to R$, $|V| = n$.
  - 一个前提要求：假设不存在负权重的回路。
- 目标: 求得一个 $n \times n$ 矩阵，其元素d_{uv} 为节点 u, v 之间的最短通路长度$d(u, v)$.
- 在 $O(n^3)$ 时间复杂度内实现，无需复杂数据结构。

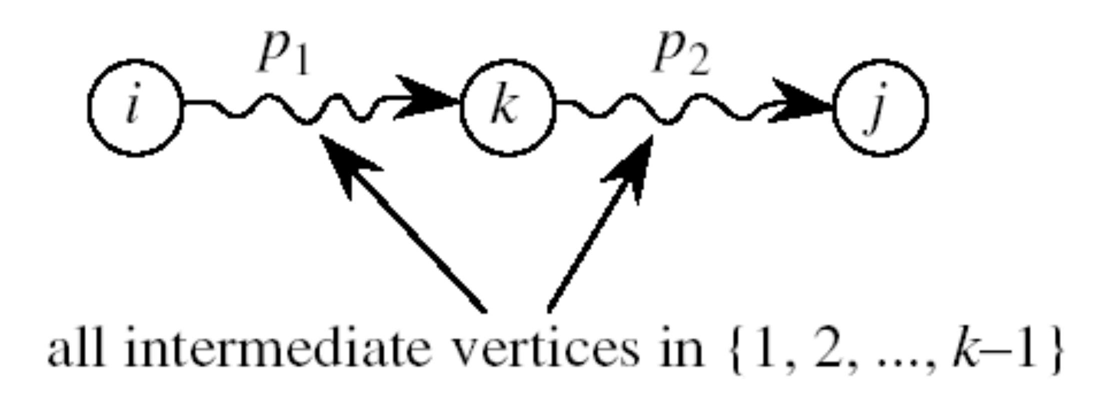
All Pairs Shortest Path – Floyd-Warshall Algorithm

- *Dynamic programming* approach. --在《算法设计与分析》课程中讨论
  - Use *optimal substructure* of shortest paths: Any subpath of a shortest path is a shortest path.

- Create a 3-dimensional table:
  - Let $d_{ij}^{(k)}$ –shortest path weight of any path from i to j where all intermediate vertices are from the set $\{1,2,\ldots,k\}$.
  - Ultimately, we would like to know the values of $d_{ij}^{(n)}$.

Computing $d_{ij}^{(k)}$

- Base condition: $d_{ij}^{(0)} = ?$
  - $d_{ij}^{(0)} = w_{ij}$.
- For k>0:
  - Let p=<v_i, . . . , v_j> be a shortest path from vertex i to vertex j with all intermediate vertices in {1,2, …, k}.
  - If k is *not* an intermediate vertex, then all intermediate vertices are in {1,2, …, k-1}.
  - If k *is* an intermediate vertex, then p is composed of 2 shortest subpaths drawn from {1,2, …, k-1}.



Recursive Formulation for $d_{ij}^{(k)}$

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

Algorithm

$\text{FLOYD-WARSHALL}(W, n)$

$D^{(0)} \leftarrow W$

for $k \leftarrow 1$ **to** n

  do for $i \leftarrow 1$ **to** n

    do for $j \leftarrow 1$ **to** n

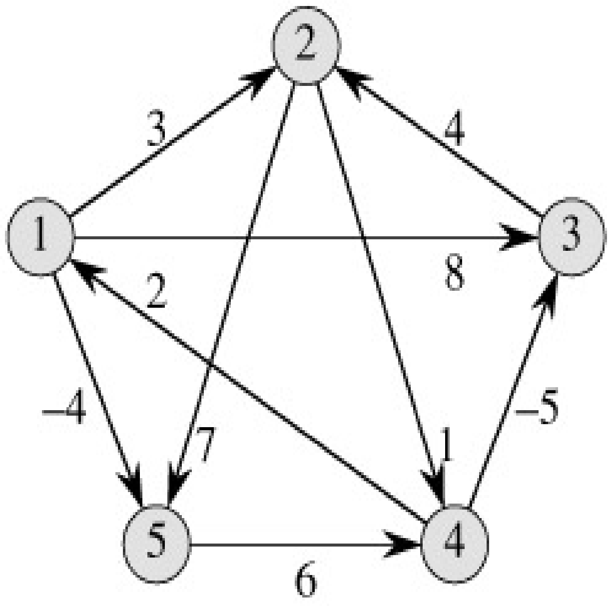
      do $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

return $D^{(n)}$

- Running time = ?
  - $O(n^3)$.
- Memory required = ?
  - $O(n^2)$ (if we drop the superscripts).

Example

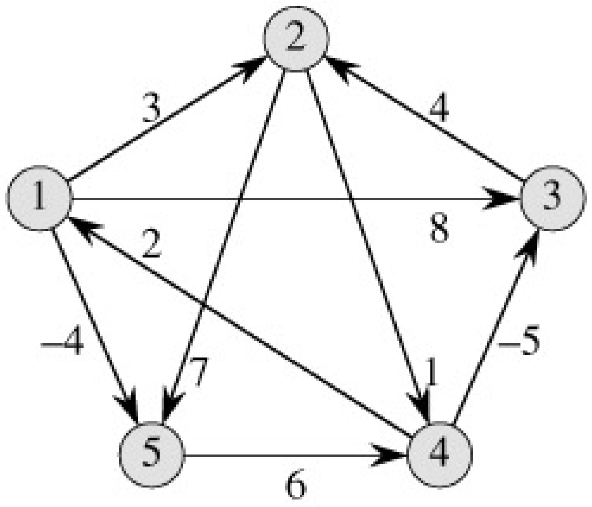
$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 . \end{cases}$$



$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Step 1

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

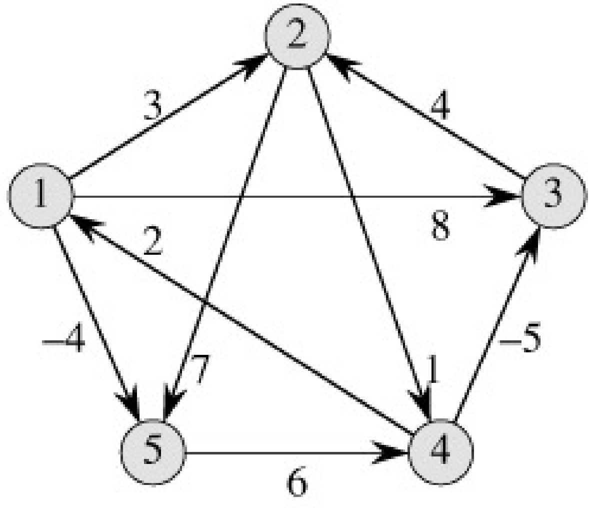


$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Step 2

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

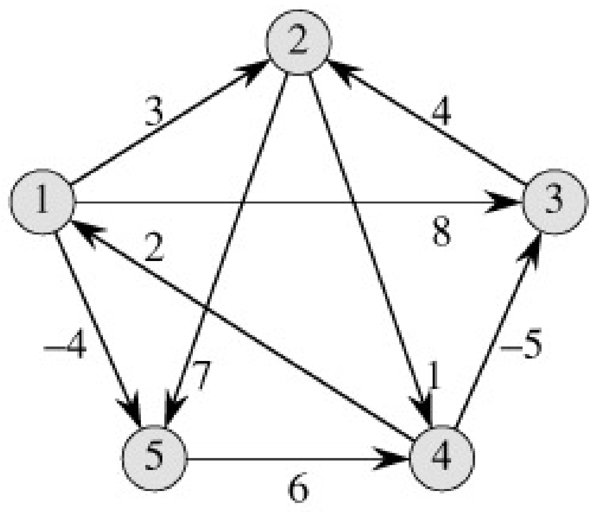


$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Step 3

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

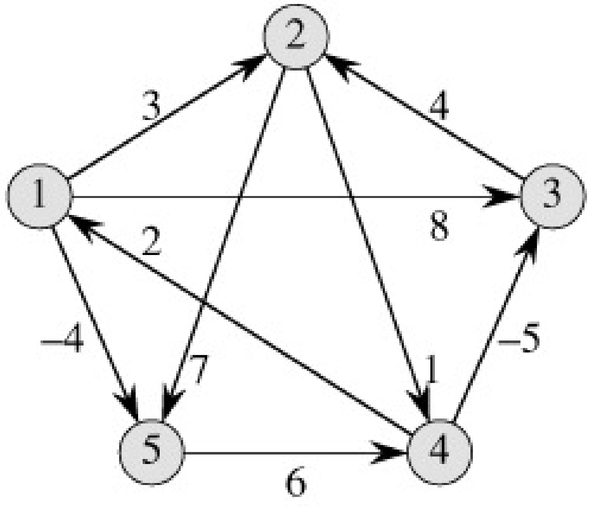


$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Step 4

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

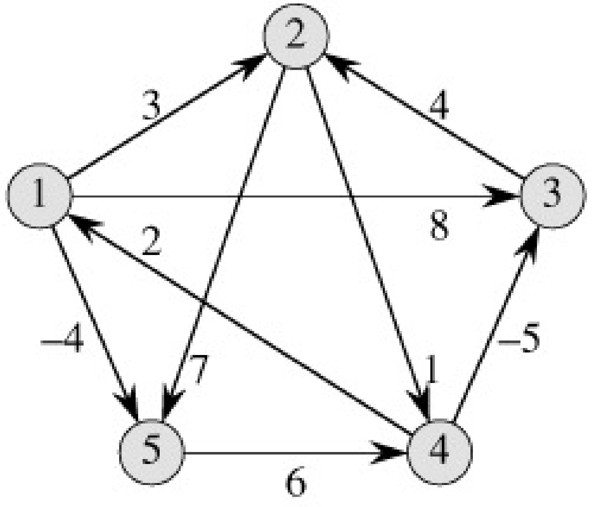


$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Step 5

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



$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

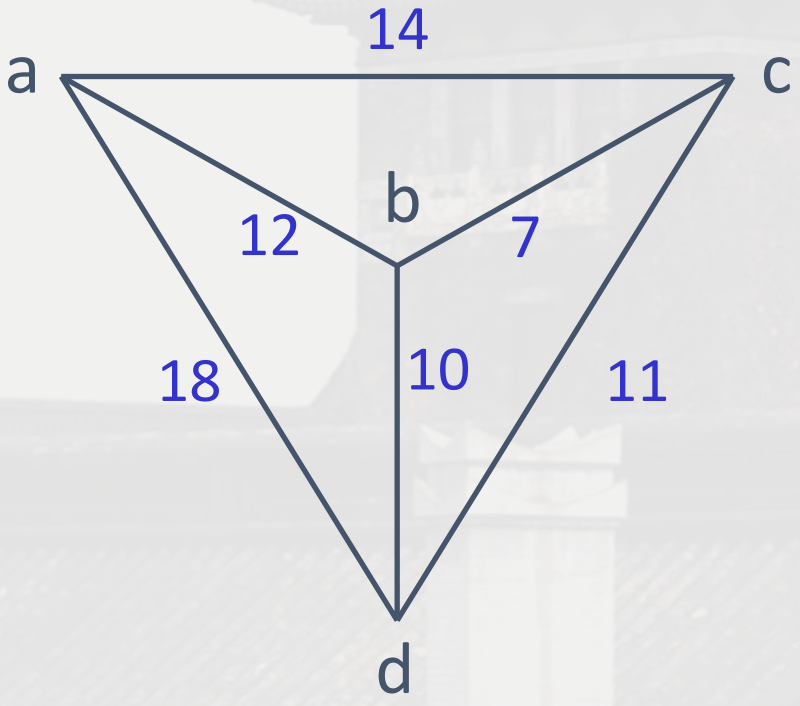
旅行商问题
(Travelling Salesman Problem, TSP)

- n个城市间均有道路，但距离不等，旅行商从某地出发，走过其它$n-1$城市各一次，最后回到原地，如何选择最短路线？
- 数学模型：
  - 无向带权图G：顶点对应于城市，边对应于城市之间的道路，道路长度用相应边的权表示。
  - 问题的解：权最小的哈密尔顿回路。
  - G是带权完全图，总共有(n-1)!/2条哈密尔顿回路。因此，问题是如何从这(n-1)!/2条中找出最短的一条。

(含25个顶点的完全图中不同的哈密尔顿回路有约3.1×10^{23}条，若机械地检查，每秒处理10^9条，需1千万年。)

旅行商问题

- 一个货郎（销售员）生活在城市a，假定访问的城市是d, b, c，然后回到a，求完成这次访问的最短路径的距离.



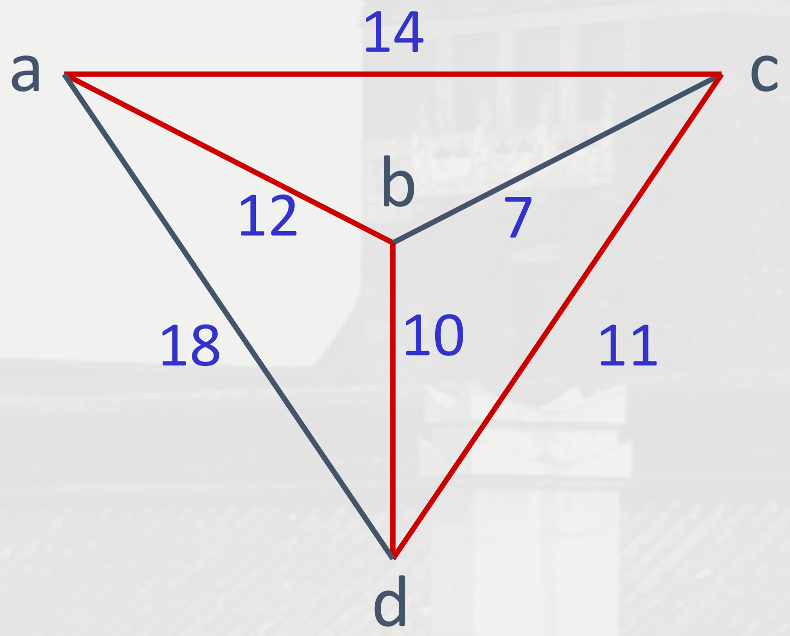
旅行商问题

- 解：列出哈密尔顿回路, 并求其距离：

（1）（abcda）＝（12+7+11+18）＝48

（2）（acbda）＝（14+7+10+18）＝49

（3）（abdca）＝（12+10+11+14）＝47



旅行商问题

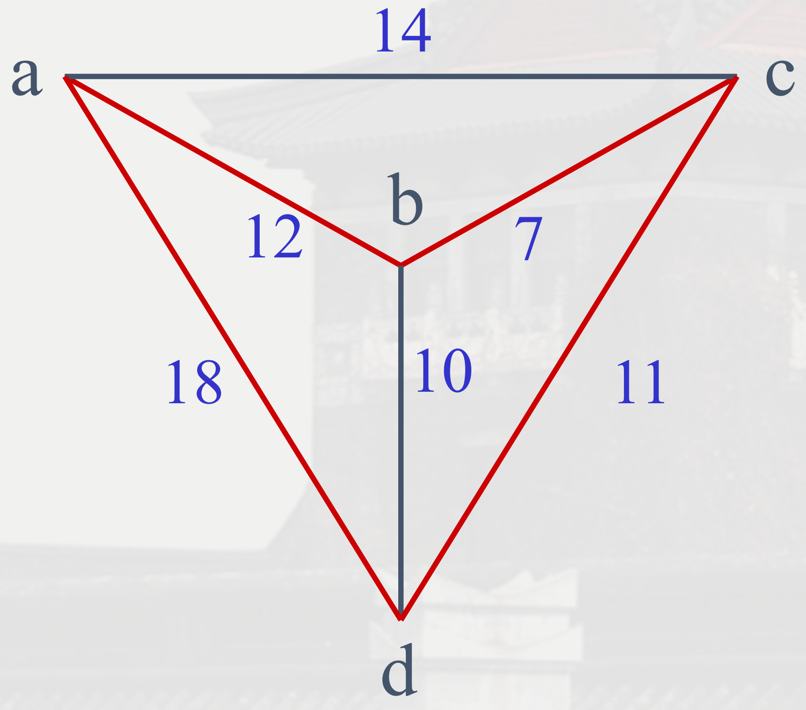
- 哈密尔顿回路（路径）的最短路径问题！
- 下面介绍一种最邻近算法：

（1）选择任一顶点作为始点，找出离始点距离最小的顶点，形成一条边的初始路径；

（2）设u是最新加到这条路径上的顶点，从不在这条路径上的所有顶点中选择一个与u距离最小的顶点,把连接u与此结点的边加入路径中；重复执行直到G中的各顶点均含在这条路径中。

旅行商问题

（3）把始点到最后加入的顶点的边放入路径中得到一条哈密尔顿回路，并为近似最短的哈密尔顿回路.



旅行商问题(TSP)的研究进展

- （在最坏情况下）时间复杂性为多项式的算法？
- （在最坏情况下）时间复杂性为多项式的近似算法
  - 保证: $W \le W' \le cW$ (c=3/2), 误差为50%
- 实际应用中，已有好的算法能够在几分钟内处理1000个节点的规模，误差在2%。

小结

- 带权图与最短通路
- 单源点最短通路问题 – Dijkstra 算法
- 所有点对间的最短通路问题 – Floyd-Washall 算法
- 旅行商问题（TSP）

编程 练习（自行练习）

- 问题描述： 给你n个点，m条无向边，每条边都有长度d>0，给你起点s终点t，要求使用Dijkstra算法输出起点到终点的最短距离及对应的一条最短路径。如果起点和终点之间没有通路，则输出距离∞ 。

编程要求

- **输入**
  - 输入n和m，顶点的编号是1~n。
  - 接着的m行，每行3个数 a,b,d，表示a和b之间有一条长度为d的边。
  - 最后一行是两个数 s,t：表示起点和终点的编号。以0 0 表示输入结束。
    (1<n<=1000, 0<m<100000, s != t)
- **输出**
  - 输出第一行最短距离，第二行最短路径的顶点序列。

样例输入：

```
3 2
1 2 5
2 3 4
1 3
0 0
```

样例输出：

```
9
1 2 3
```