# 课程回顾

- 贪心算法
  - 贪心算法可求解问题的特性
  - 活动选择问题：动态规划算法 vs. 贪心算法
  - 贪心算法原理
    - 一般设计步骤
    - 贪心算法要素：贪心选择性质、最优子结构
  - 0-1背包 vs. 分数背包
  - 贪心算法理论：拟阵

# 贪心算法理论 (续)

## ■最大独立集

- ➢独立子集的**扩展**：在拟阵$M=(S,\mathcal{I})$中，若$A\in\mathcal{I}$且$x\in S-A$，若$A\cup\{x\}\in\mathcal{I}$，则元素$x$称为$A$的一个**扩展**，即元素$x$扩充到独立子集$A$后仍保持独立性
  - 例：在图拟阵$M_G$中，若$A$是一个独立子集，则$e$是$A$的扩展是指加入$e$后仍不产生回路
- ➢拟阵的最大独立子集
  - 若$A$是拟阵$M$的独立子集，且无法进行任何扩展，则$A$称为$M$的**最大独立子集**，即在$M$中没有更大的独立子集能包含$A$

# 贪心算法理论 (续)

■**定理16.6** 拟阵中所有最大独立子集都具有相同大小

证明（反证法）：假设拟阵$M$存在一个最大独立子集$A$和另一个更大的独立子集$B$
由交换性质：对于某个$x \in B - A$，有$A \cup \{x\} \in \mathcal{I}$，即$A$可扩展，与$A$是最大独立子集矛盾

# 贪心算法理论 (续)

- **加权拟阵**：若对$\forall x \in S$，为$x$指派一个正的权值$w(x)$，则称$M=(S,\mathcal{I})$是加权拟阵。$S$的子集（独立子集）的权重函数可定义为：

$$w(A)=\sum_{x\in A} w(x), \quad \forall A \subseteq S$$

  - 例：若令$w(e)$表示图拟阵$M_G$中边$e$的权重（例如边的长度等），那么$w(A)$为$A$中所有边的权重之和

# 贪心算法理论 (续)

- 可用贪心算法求出最优解的很多问题（但不是全部）可形式化为在一个加权拟阵中寻找最大权重的独立子集问题。即：给定加权拟阵 $M=(S,\mathcal{I})$，希望寻找独立集 $A\in\mathcal{I}$，使 $w(A)$ 最大
- 拟阵的最优子集：拟阵中权值最大的独立子集
  最优子集一定是拟阵中的一个最大独立子集，反之不然
- 加权拟阵的贪心算法
  - 适用于任何加权拟阵求最优子集 $A$
  - 贪心之处：尽可能选权值最大的元素扩充到 $A$

# 贪心算法理论 (续)

```
GREEDY (M, w)
    A ← ∅
    按权重w单调递减对M.S中元素排序
    for each x∈M.S, 按w单调递减次序取出x do
        if A ∪ {x} ∈ M.𝓘        //独立性检验
            A ← A ∪ {x}        //扩展x未破坏独立性
    return A
```

■时间复杂度：$O(n\lg n + nf(n))$

- ➢排序：$O(n\lg n)$
- ➢检查$A \cup \{x\}$独立性：$O(f(n))$

定理16.11说明了加权拟阵上贪心算法的正确性

# 贪心算法内容

- ■活动选择问题
- ■贪心算法原理
- ■分数背包问题
- ■Huffman编码
- ■拟阵
- ■其他应用

# 其他应用——最优装载

- 问题：有一批集装箱要装上一艘载重量为$c$的轮船。其中集装箱$i$的重量为$w_i$。最优装载问题要求确定在装载体积不受限制的情况下，将尽可能多的集装箱装上轮船
- 最优装载问题可用贪心算法求解。采用重量最轻者先装的贪心选择策略，可产生最优装载问题的最优解
- 时间复杂度：$O(n\lg n)$

# 其他应用——单源最短路径

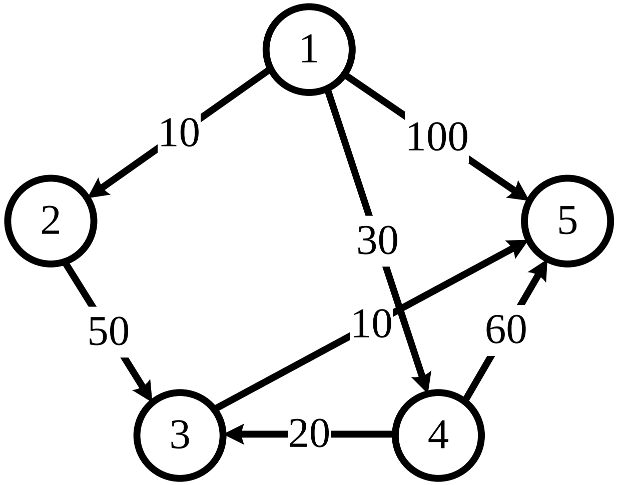
- 给定带权有向图$G = (V, E)$，其中每条边的权是非负实数
- 给定$V$中的一个顶点，称为源
- 单源最短路径问题：计算从源到所有其它各顶点的最短路径长度，这里路径长度是指路上各边权之和
- Dijkstra算法是解单源最短路径问题的贪心算法
  - 基本思想：设置顶点集合$S$并不断地作贪心选择来扩充这个集合。一个顶点属于集合$S$当且仅当从源到该顶点的最短路径长度已知

# 其他应用——单源最短路径 (续)

- 初始时，*S*中仅含有源
- 设*u*是图*G*的某一个顶点，把从源到*u*且中间只经过*S*中顶点的路称为从源到*u*的特殊路径，并用数组dist记录当前每个顶点所对应的最短特殊路径长度
- Dijkstra算法每次从*V*-*S*中取出具有最短特殊路长度的顶点*u*，将*u*添加到*S*中，同时对数组dist作必要的修改
- 一旦*S*包含了所有*V*中顶点，dist就记录了从源到所有其它顶点之间的最短路径长度

# 其他应用——单源最短路径 (续)

■Dijkstra算法实例




| 迭代 | S | u | dist[2] | dist[3] | dist[4] | dist[5] |
|---|---|---|---|---|---|---|
| 初始 | {1} | - | 10 | maxint | 30 | 100 |
| 1 | {1, 2} | 2 | 10 | 60 | 30 | 100 |
| 2 | {1, 2, 4} | 4 | 10 | 50 | 30 | 90 |
| 3 | {1, 2, 4, 3} | 3 | 10 | 50 | 30 | 60 |
| 4 | {1, 2, 4, 3, 5} | 5 | 10 | 50 | 30 | 60 |

# 其他应用——单源最短路径 (续)

- 算法的正确性
  - 贪心选择性质
  - 最优子结构性质
- 算法时间复杂度$O(|V|^2)$

# 其他应用——最小生成树

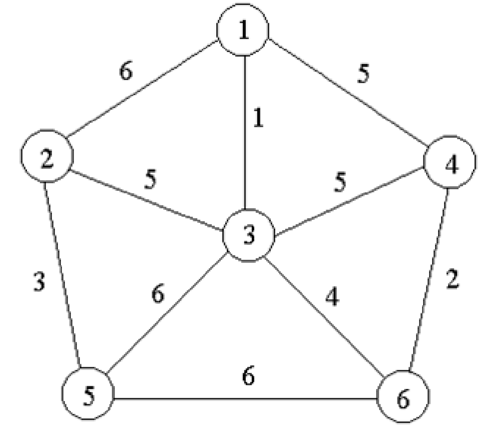
- 问题：设$G = (V, E)$是无向连通带权图，$E$中每条边$(v, w)$的权为$c(v, w)$。如果$G$的子图$G'$是一棵包含$G$的所有顶点的树，则称$G'$为$G$的生成树，生成树上各边权的总和称为该生成树的耗费。在$G$的所有生成树中，耗费最小的生成树称为$G$的最小生成树
- 最小生成树性质：设$G = (V, E)$是连通带权图，$U$是$V$的真子集。如果$(u, v) \in E$，且$u \in U$，$v \in V\text{-}U$，且在所有这样的边中，$(u, v)$的权$c(u, v)$最小，那么一定存在$G$的一棵最小生成树，它以$(u, v)$为其中一条边

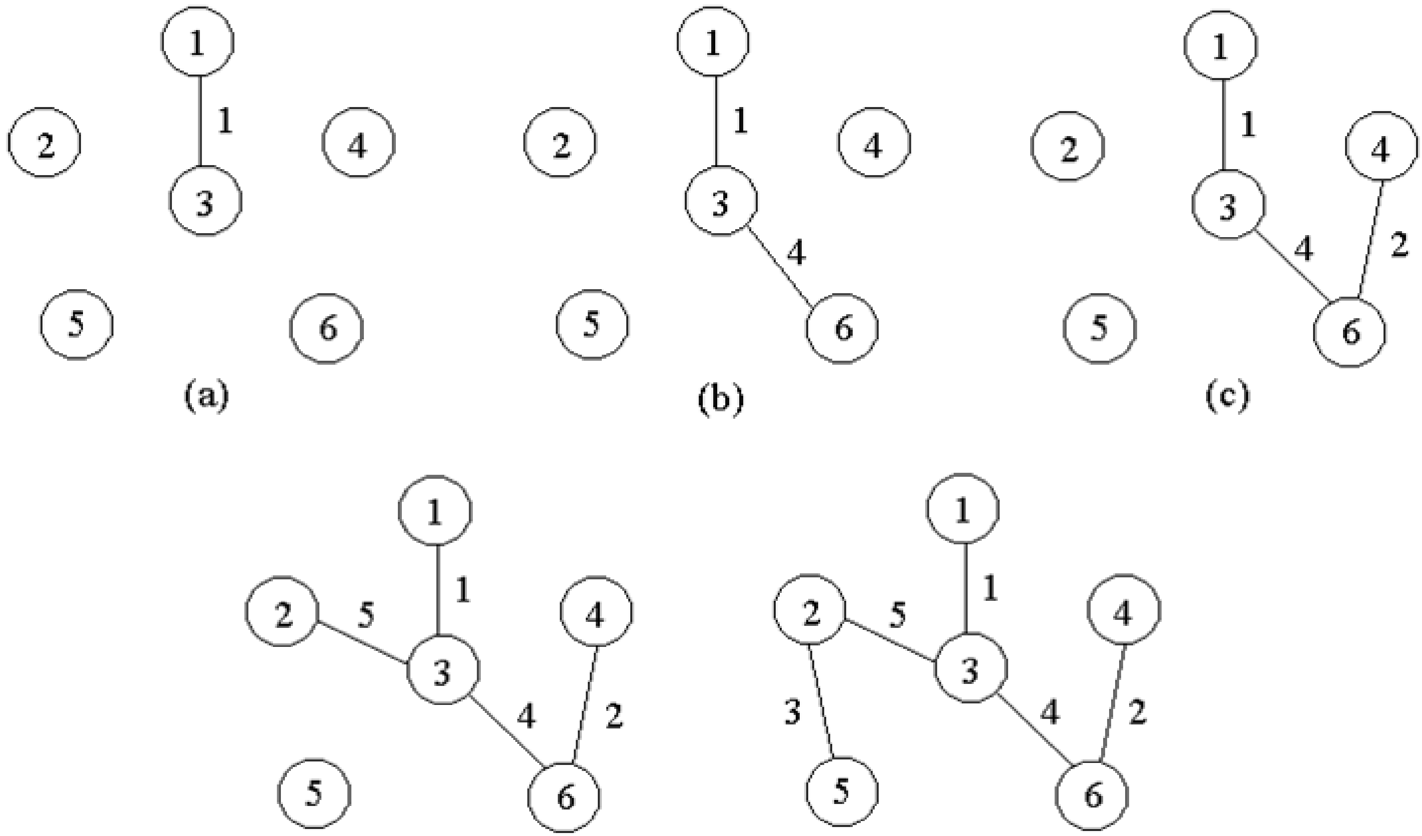
# 其他应用——最小生成树 (续)

## ■Prim算法

- ➢设$G = (V, E)$是连通带权图，$V = \{1, 2, \ldots, n\}$
- ➢构造$G$的最小生成树的Prim算法的基本思想是：首先置$S = \{1\}$，然后，只要$S$是$V$的真子集，就作如下的贪心选择：选取满足条件$i \in S$，$j \in V\text{-}S$，且$c(i, j)$最小的边，将顶点$j$添加到$S$中。这个过程一直进行到$S = V$时为止
- ➢在这个过程中选取到的所有边恰好构成$G$的一棵最小生成树
- ➢时间复杂度：$O(|V|^2)$

# 其他应用——最小生成树 (续)





(a) (b) (c) (d) (e)

# 其他应用——最小生成树 (续)

■Kruskal算法

- 首先将$G$的$n$个顶点看成$n$个孤立的连通分支
- 将所有的边按权从小到大排序
- 按排序后顺序检查每一条边$(v, w)$，若$v$和$w$分别是当前2个不同的连通分支中的顶点时，则添加该边；直到只剩下一个连通分支时为止
- 时间复杂度：$O(|E|\lg|E|)$

# 回顾贪心算法理论

GREEDY ($M$, $w$)

$A \leftarrow \emptyset$

按权重$w$单调递减对$M.S$中元素排序

**for** each $x \in M.S$, 按$w$单调递减次序取出$x$ **do**

  **if** $A \cup \{x\} \in M.\mathcal{I}$  //独立性检验

    $A \leftarrow A \cup \{x\}$  //扩展$x$未破坏独立性

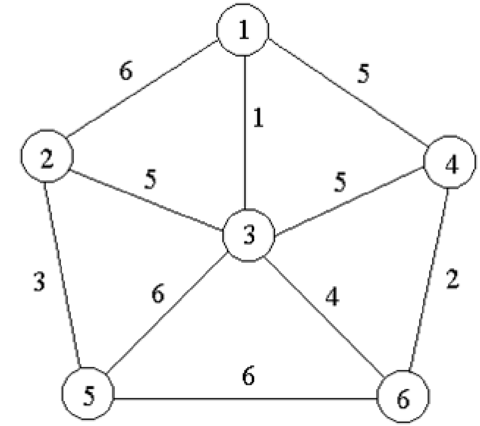
**return** $A$

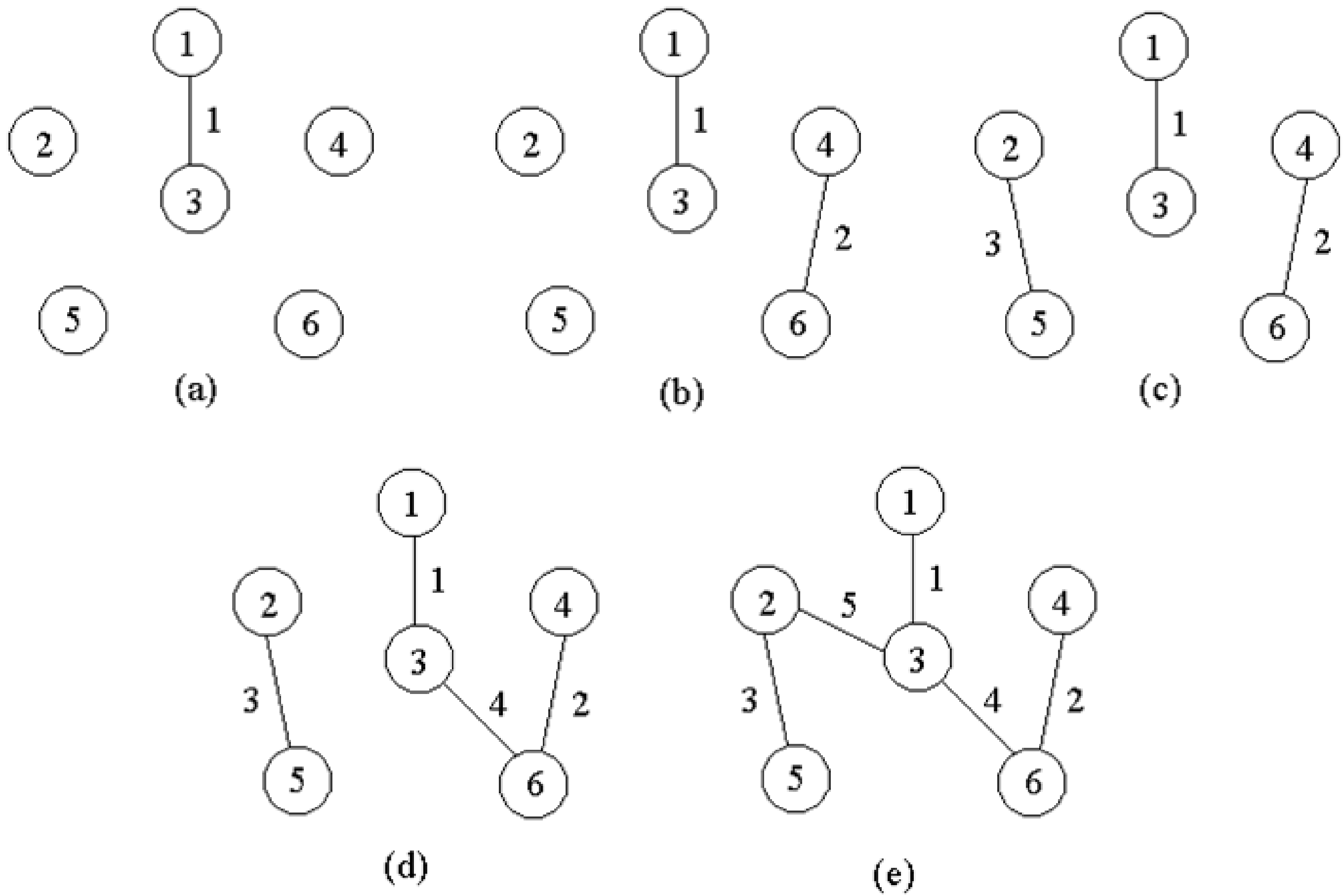
■时间复杂度：$O(n\lg n + nf(n))$

- 排序：$O(n\lg n)$
- 检查$A \cup \{x\}$独立性：$O(f(n))$

定理16.11说明了加权拟阵上贪心算法的正确性

# 其他应用——最小生成树 (续)





# 其他应用——最小生成树 (续)

- 当$|E| = \Omega(|V|^2)$时，Kruskal算法比Prim算法差，但当$|E| = O(|V|^2)$时，Kruskal算法却比Prim算法好得多

# 本章小结

- 贪心算法：启发式思想，每次选择使得当前步骤收益最大化，但并不考虑全局情况，大多数情况无法得到最优解
- 贪心算法可得到最优解：活动选择问题、分数背包问题、Huffman编码、最优装载问题、单源最短路径、最小生成树等
- 寻找加权拟阵的权重最大独立子集
- 贪心算法要素：贪心选择性质、最优子结构

# 第5章 近似算法

苏州大学  计算机科学与技术学院
汪笑宇
Email: xywang21@suda.edu.cn

# 本章内容

■近似算法（教材Chapter 35）

- ➢概述
- ➢顶点覆盖问题
- ➢旅行商问题
- ➢集合覆盖问题
- ➢子集和问题

# 近似算法概述

- 许多具有实际意义的问题都是NPC问题：P≠NP时无法在多项式时间内求最优解
- 解决NPC问题方法：
  - 输入规模小：指数级运行时间算法解决
  - 多项式时间内解决特殊情况
  - 多项式时间内得到近似最优解
- 近似算法：返回近似最优解的算法

# 近似算法概述 (续)

■最优化问题$\Pi$（最小化问题/最大化问题）：

- $D$：输入实例集合
- $S(I)$：输入实例$I \in D$的可行解集合
- $f: S(I) \rightarrow \mathbb{R}$：将每个可行解进行赋值的函数，称为目标函数
- $\mathrm{OPT}(I)$：输入实例$I$对应最优解的值，即$f(s^*)$的值，其中$s^* \in S(I)$是输入$I$的最优解
- $\mathrm{SOL}_A(I)$：算法$A$得到的输入实例$I$的解的值，即$f(s)$的值，其中$s \in S(I)$是算法$A$得到的输入$I$的解

# 近似算法概述 (续)

■最优化问题Π（最小化问题/最大化问题）：

➢算法$A$的近似比$\alpha$：

$$\forall I \in D: \begin{cases} 1 \leq \dfrac{\mathrm{SOL}_A(I)}{\mathrm{OPT}(I)} \leq \alpha, & \text{Minimization} \\ \alpha \leq \dfrac{\mathrm{SOL}_A(I)}{\mathrm{OPT}(I)} \leq 1, & \text{Maximization} \end{cases}$$

近似比限制了算法$A$所能达到的最坏情况

# 近似算法概述 (续)

■最优化问题Π（最小化问题/最大化问题）：

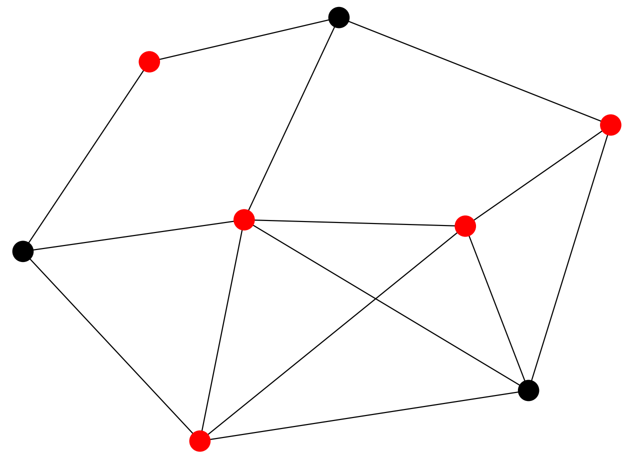
- 近似方案/近似模式（approximation scheme）：输入为$(I, \varepsilon)$，其中$I \in D$，$\varepsilon>0$为误差参数，若有$(1-\varepsilon)\mathrm{OPT}(I) \le \mathrm{SOL}_A(I, \varepsilon) \le (1+\varepsilon)\mathrm{OPT}(I)$，则称算法$A$是NP-Hard最优化问题Π的近似方案/近似模式
- PTAS (Polynomial-Time Approximation Scheme)：对于每一个确定的$\varepsilon>0$，该近似方案的运行时间以$I$的规模的多项式为上界
- FPTAS (Fully Polynomial-Time Approximation Scheme)：对于每一个确定的$\varepsilon>0$，该近似方案的运行时间以$I$的规模和$1/\varepsilon$的多项式为上界

FPTAS被认为是最值得研究的近似算法，仅有极少的NP-Hard问题存在FPTAS

# 顶点覆盖判定问题

■顶点覆盖问题VERTEX-COVER

- 给定一个无向图$G=(V, E)$和一个正整数$k$，判定是否存在$V' \subseteq V$和$|V'|=k$，使得对任意$(u,v)\in E$有$u\in V'$或$v\in V'$，如果存在，就称$V'$为图$G$的一个大小为$k$的顶点覆盖
- 即$E$中每条边至少有一个顶点在$V'$中



存在一个规模$k$为5的顶点覆盖

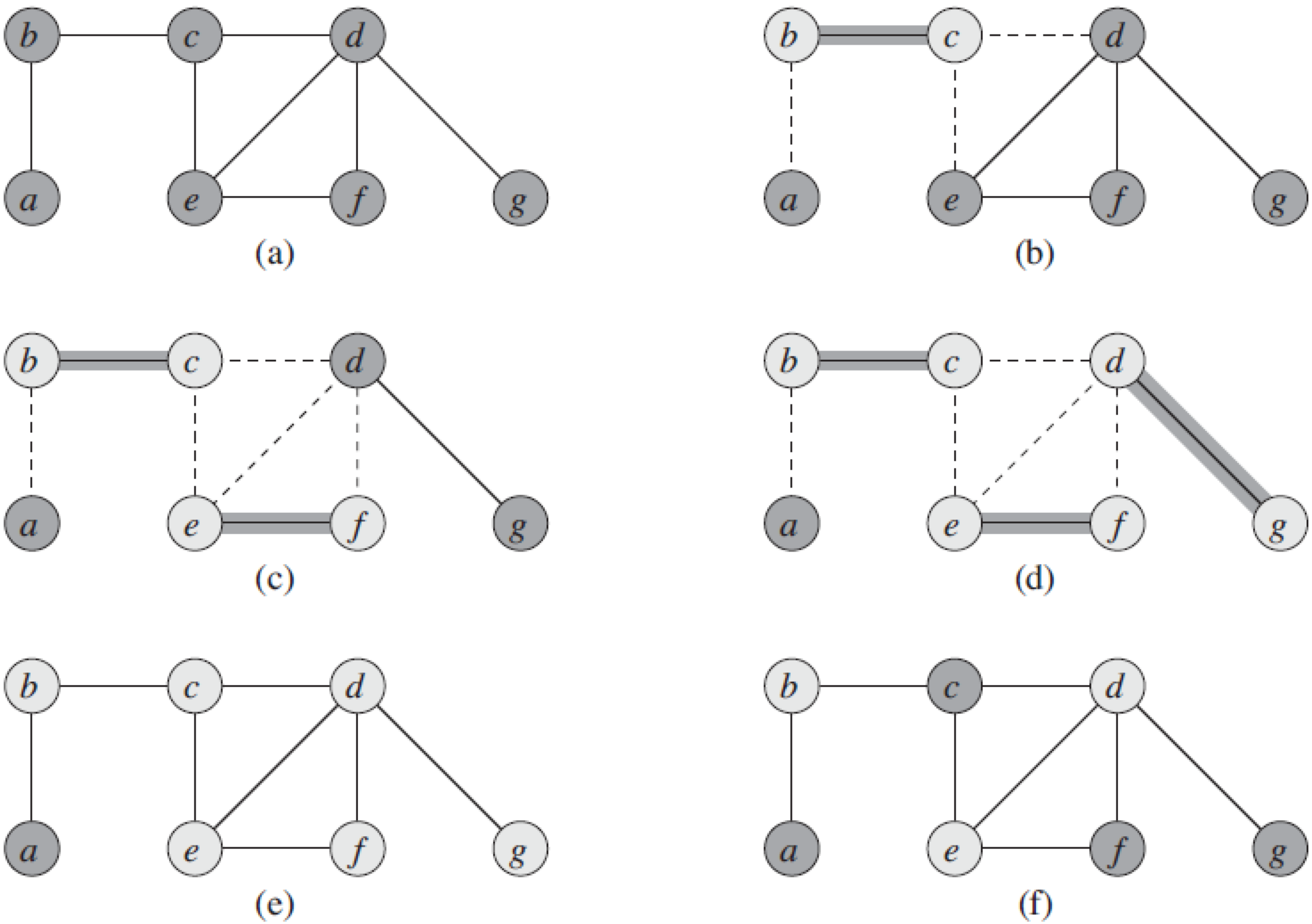
# 顶点覆盖问题

- 顶点覆盖的规模：$V'$包含的顶点数
- 顶点覆盖问题：在一个给定的无向图中找出一个具有**最小规模**的顶点覆盖（**最优顶点覆盖**）

APPROX_VERTEX_COVER($G$)
$C \leftarrow \emptyset$
$E' \leftarrow G.E$
**while** $E' \neq \emptyset$ **do**
  选择$E'$中任一条边$(u, v)$
  $C \leftarrow C \cup \{u, v\}$
  将$E'$中与$u$及$v$邻接的边全部删除
**return** $C$

$O(|V|+|E|)$

# 顶点覆盖问题 (续)



# 顶点覆盖问题 (续)

■APPROX_VERTEX_COVER算法近似比为2，即返回的规模最多是最优顶点覆盖规模的2倍

- 证明：（通过算法第4行选出的边的数量进行过渡）设算法第4行选出的边的数量为$|A|$，则$|C| = 2|A|$；

  设最优解为$C^*$，因为$A$中的边不共用顶点，则$A$中每条边至少有一个顶点要在$C^*$中，即 $|A| \le |C^*|$；

  综合得到$|C| \le 2|C^*|$，即 SOL/OPT $\le 2$

APPROX_VERTEX_COVER($G$)
1 $C \leftarrow \emptyset$
2 $E' \leftarrow G.E$
3 **while** $E' \ne \emptyset$ **do**
4 　　选择$E'$中任一条边$(u, v)$
5 　　$C \leftarrow C \cup \{u, v\}$
6 　　将$E'$中与$u$及$v$邻接的边全部删除
7 **return** $C$

# 旅行商问题TSP

- 旅行商问题TSP (Traveling Salesman Problem)
  - 给定一个无向完全图$G=(V, E)$及定义在$V\times V$上的一个费用函数$c$和一个整数$k$，判定$G$是否存在经过$V$中各顶点恰好一次的回路，使得该回路的费用不超过$k$
  - 一个售货员必须访问$n$个城市，售货员希望恰好访问每个城市一次，并最终回到出发城市。售货员从城市$i$到城市$j$的旅行费用为一个整数$c(i, j)$，旅行所需的全部费用是他旅行经过的各边费用之和，售货员希望整个旅行费用最低。判定问题为：旅行费用不超过$k$。

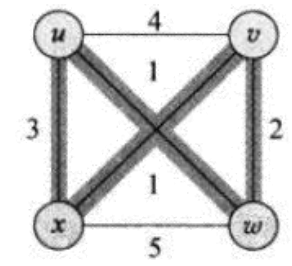



图 34-18　旅行商问题的一个实例。阴影覆盖的边表示费用最低的旅行路线，其费用为 7

# 旅行商问题TSP (续)

- ■输入：无向完全图 $G=(V, E)$，每条边$(u, v)\in E$有一个非负整数代价$c(u, v)$
- ■输出：$G$的一条具有最小代价的哈密顿回路（$G$中每个顶点只经过一次）
- ■实际情况中，代价通常满足三角不等式，即：
  $c(u, w) \le c(u, v) + c(v, w)$
  即：直达代价更小

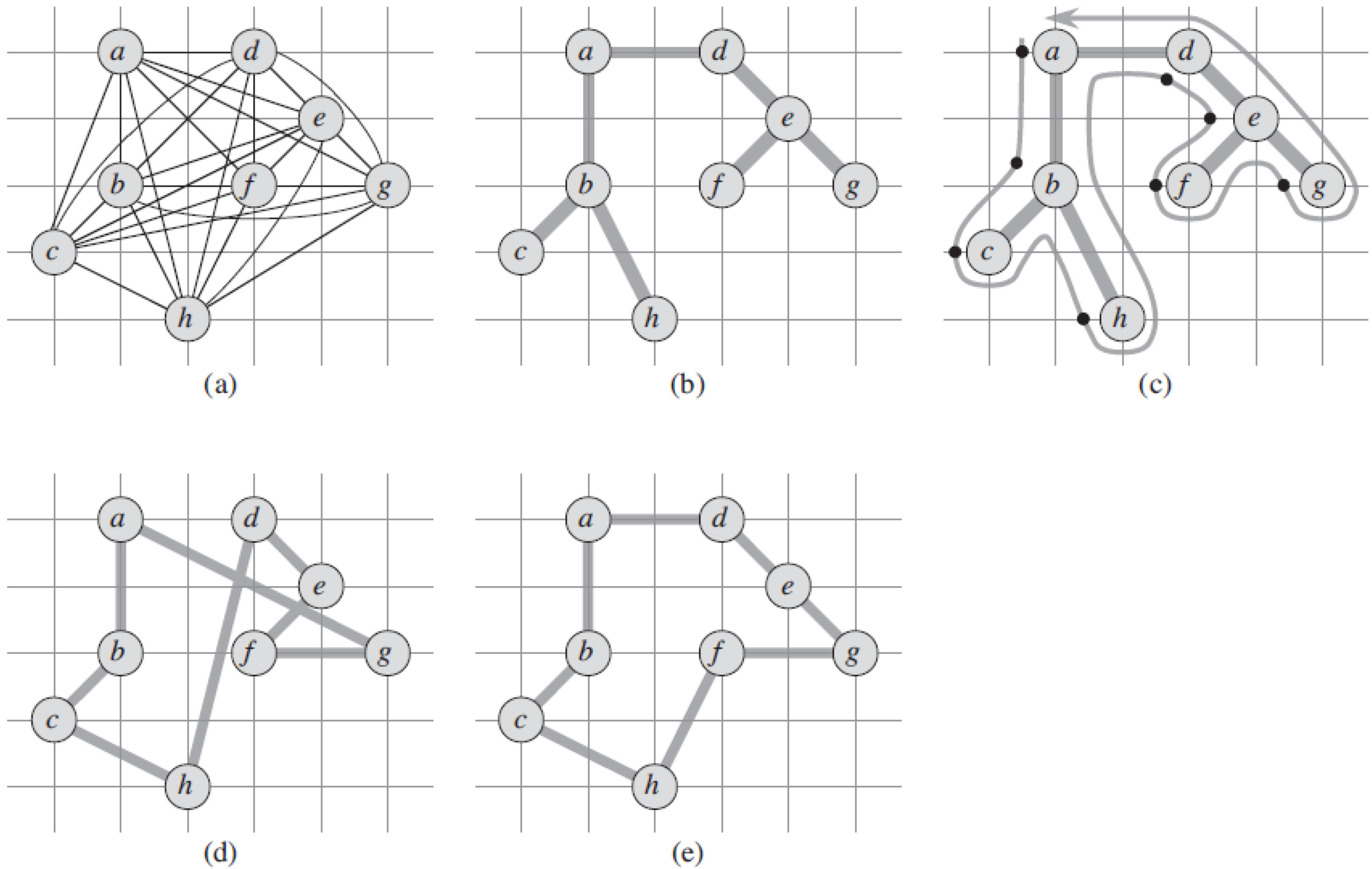
# 旅行商问题TSP (续)

## ■满足三角不等式的TSP：最小生成树法求近似解

APPROX_TSP_TOUR($G$, $c$)
1　选择一个顶点$r \in G.V$
2　使用Prim算法构造图$G$以$r$为根结点的最小生成树T
3　先序遍历T，生成顶点序列$H$
4　**return** 依次经过序列$H$中顶点、并最后回到第一个顶点的路径

- 运行时间：$\Theta(|V|^2)$
- 算法近似比：2
  - 证明：最小生成树的权值是最优旅行路线的下界：$c(\mathrm{T}) \le \mathrm{OPT}$
    设仅能在最小生成树的边上进行旅行，那么先序遍历并回到根结点的代价：$c(W)=2c(\mathrm{T})$
    由于三角不等式，算法输出的代价$\mathrm{SOL} \le c(W)$
    由此得到：$\mathrm{SOL} \le 2\mathrm{OPT}$，即$\mathrm{SOL}/\mathrm{OPT} \le 2$

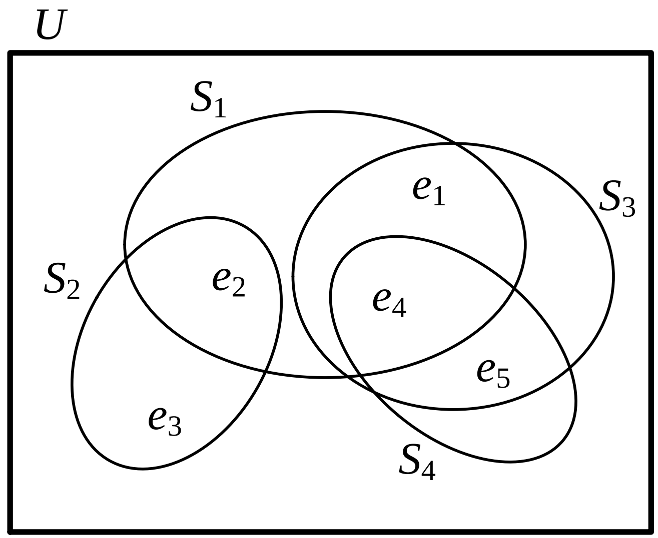
# 旅行商问题TSP (续)

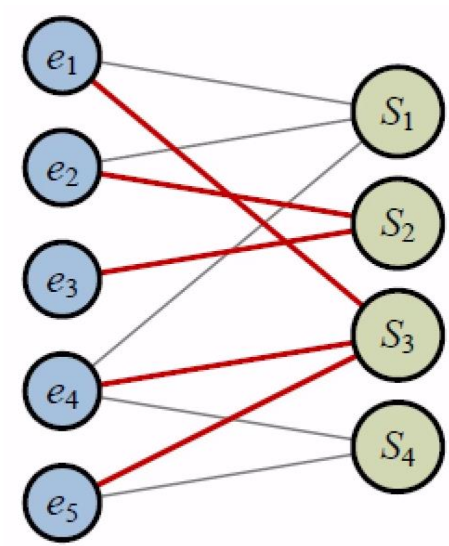


# 旅行商问题TSP (续)

- 一般TSP：P≠NP情况下，不存在多项式时间内具有常数近似比的近似算法（定理35.3）

# 集合覆盖问题




**Set Cover**

Given a universe $U$ of $n$ elements, a collection of subsets of $U$, $\mathcal{S} = \{S_1, ..., S_m\}$, and a cost function $c : \mathcal{S} \rightarrow \mathbf{Q}^+$, find a minimum cost subcollection of $\mathcal{S}$ that covers all elements of $U$.

# 集合覆盖问题 (续)

- 代价函数通常为解集合族中集合的数量，即用最少的集合覆盖所有的元素
- 贪心近似算法：每次选择覆盖元素最多的集合

GREEDY_SET_COVER($U$, $\mathcal{S}$)

$X \leftarrow U$

$\mathcal{F} \leftarrow \emptyset$

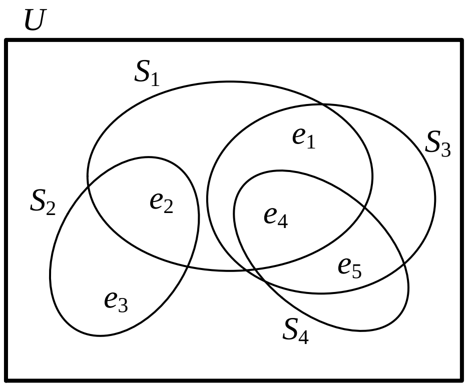
**while** $X \neq \emptyset$ **do**

  选择使得$|S_i \cap X|$最大的$S_i \in \mathcal{S}$

  $X \leftarrow X - S_i$

  $\mathcal{F} \leftarrow \mathcal{F} \cup \{S_i\}$

**return** $\mathcal{F}$




多项式时间的$(\ln|U|+1)$近似算法

# 子集和问题

与0-1背包问题联系：
物品——正整数
物品价值——正整数的值
物品重量——正整数的值
背包容量——整数和$t$

■子集和问题SUBSET-SUM

- 给定一个正整数有限集$S$和一个整数目标$t>0$，判定是否存在$S$的一个子集$S' \subseteq S$，使得$S'$中的整数的和为$t$
- 例：$S$={1, 2, 7, 14, 49, 98, 343, 686, 2409, 2793, 16808, 17206, 117705, 117993}, $t$ =138457，则子集$S'$={1, 2, 7, 98, 343, 686, 2409, 17206, 117705}是该问题的一个解
- 最优化问题：找到子集$S' \subseteq S$，使得$S'$中的整数的和尽可能大，但不能超过为$t$
- 存在$O(|S|t)$的伪多项式时间算法

# 子集和问题 (续)

■FPTAS算法：

以下设$n = |S|$，且$S=\{x_1, x_2, ..., x_n\}$

对任意$\varepsilon>0$，令 $k = \left\lfloor \frac{\varepsilon x_{\max}}{n} \right\rfloor$，其中 $x_{\max} = \max\limits_{1\le i\le n} x_i$

设置 $x_i' = \lfloor x_i/k \rfloor,\ \ i = 1, 2, ..., n$

返回以$x_i$'为物品价值、$x_i$为物品重量、$t$为背包容量的0-1背包动态规划算法的解

- 时间复杂度：$O(n^3/\varepsilon)$
- 近似比：SOL/OPT $\ge 1-\varepsilon$

# 本章小结

- 近似算法：用于解决NP-Hard问题的方案
- 近似比
- 近似算法举例：顶点覆盖问题、旅行商问题、集合覆盖问题、子集和问题