第4章　贪心算法

学习要点

- 理解贪心算法的概念。
- 掌握贪心算法的基本要素

  （1）最优子结构性质

  （2）贪心选择性质
- 理解贪心算法与动态规划算法的差异
- 理解贪心算法的一般理论
- 通过应用范例学习贪心设计策略。

（1）活动安排问题；（2）最优装载问题；

（3）哈夫曼编码；（4）单源最短路径；

（5）最小生成树；（6）多机调度问题。

贪心算法

顾名思义，贪心算法总是作出在当前看来最好的选择。也就是说贪心算法并不从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择。当然，希望贪心算法得到的最终结果也是整体最优的。虽然贪心算法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解。如单源最短路经问题，最小生成树问题等。在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似。

4.2 贪心算法的基本要素

1 贪心选择性质

- 所谓贪心选择性质是指所求问题的整体最优解可以通过一系列局部最优的选择，即贪心选择来达到。这是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别。

- 动态规划算法通常以自底向上的方式解各子问题，而贪心算法则通常以自顶向下的方式进行，以迭代的方式作出相继的贪心选择，每作一次贪心选择就将所求问题简化为规模更小的子问题。

- 对于一个具体问题，要确定它是否具有贪心选择性质，必须证明每一步所作的贪心选择最终导致问题的整体最优解。

4.2 贪心算法的基本要素

2 最优子结构性质

当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。问题的最优子结构性质是该问题可用动态规划算法或贪心算法求解的关键特征。

4.6 最小生成树

设G =(V,E)是无向连通带权图，即一个网络。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。

网络的最小生成树在实际中有广泛应用。例如，在设计通信网络时，用图的顶点表示城市，用边(v,w)的权c[v][w]表示建立城市v和城市w之间的通信线路所需的费用，则最小生成树就给出了建立通信网络的最经济的方案。

4.6 最小生成树

1 最小生成树性质

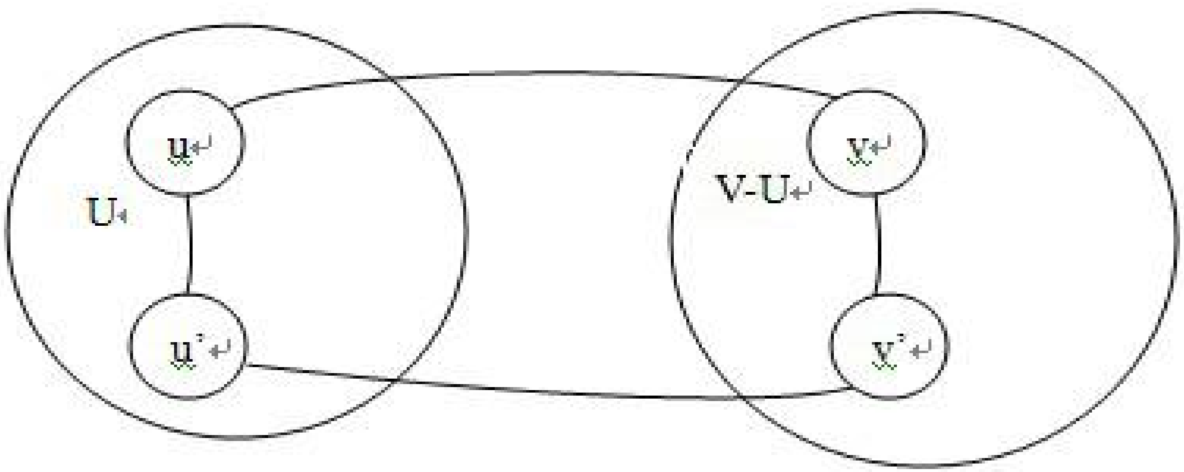
用贪心算法设计策略可以设计出构造最小生成树的有效算法。构造最小生成树的**Prim算法**和**Kruskal算法**都可以看作是应用贪心算法设计策略的例子。尽管这2个算法做贪心选择的方式不同，它们都利用了下面的**最小生成树性质**：

设G=(V,E)是连通带权图，U是V的真子集。如果(u,v)∈E，且u∈U，v∈V-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。这个性质有时也称为**MST性质**。

4.6 最小生成树

MST性质证明

假设G的任何一颗最小生成树都不包含边(u,v)。将边(u,v)添加到G的一颗最小生成树T上，将产生含有边(u,v)的圈，并且在这个圈上有一条不同于(u,v)的边(u',v')，使得u' ∈ U，v ' ∈ V-U。将边(u',v')删去，得到G的另一颗生成树T'。由于c[u][v]<=c[u'][v']，所以T'的耗费<=T的耗费。于是T'是一颗含有边(u,v)的最小生成树，这与假设矛盾。



4.6 最小生成树

2 Prim算法

设G=(V,E)是连通带权图，V={1,2,…,n}。

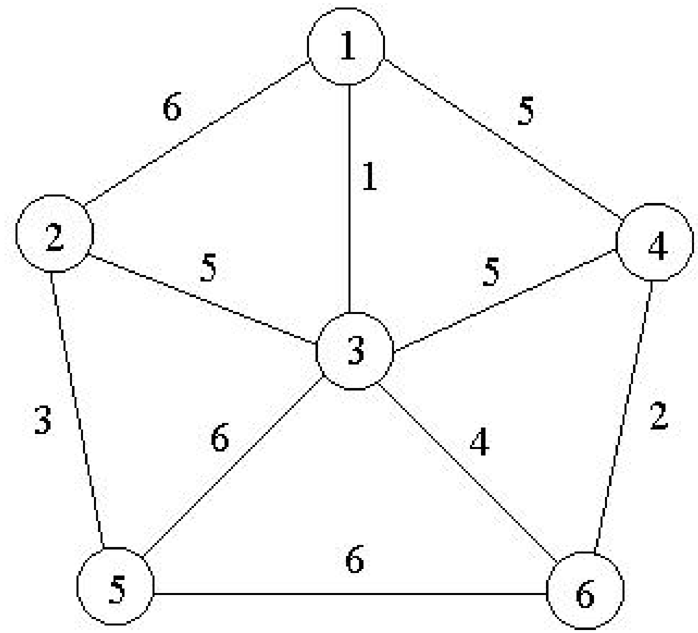
构造G的最小生成树的Prim算法的基本思想是：首先置S={1}，然后，只要S是V的真子集，就作如下的贪心选择：选取满足条件i∈S，j∈V-S，且c[i][j]最小的边，将顶点j添加到S中。这个过程一直进行到S=V时为止。

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

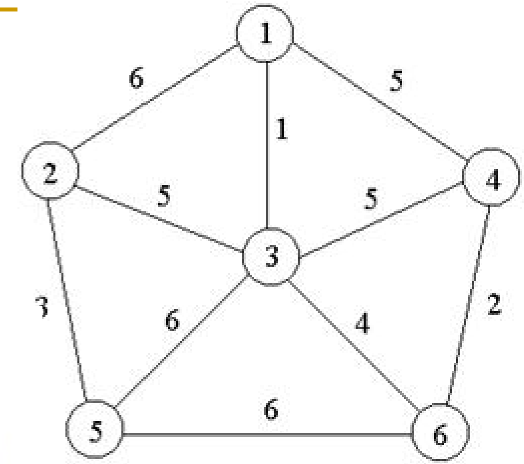
4.6 最小生成树

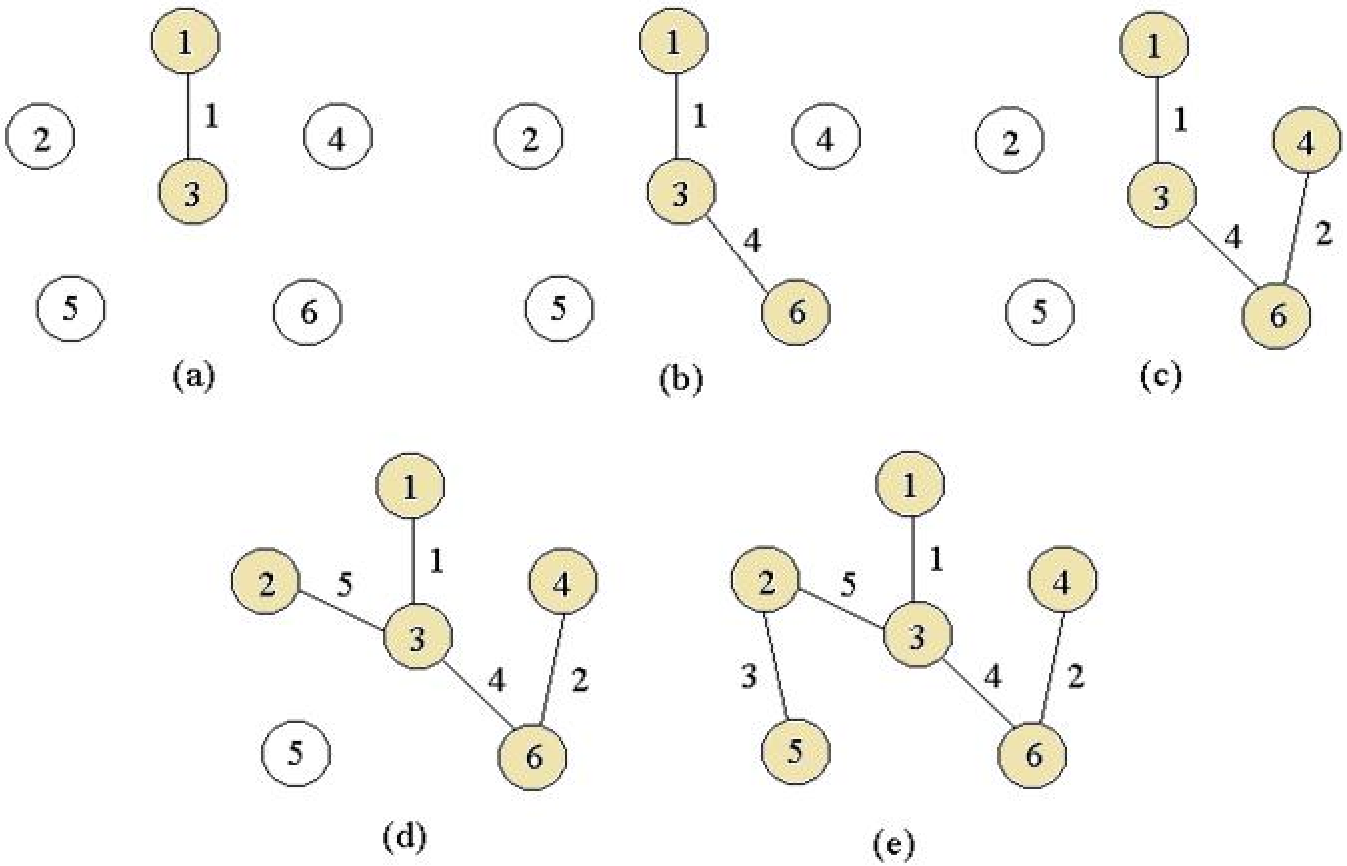
利用最小生成树性质和数学归纳法容易证明，上述算法中的边集合T始终包含G的某棵最小生成树中的边。因此，在算法结束时，T中的所有边构成G的一棵最小生成树。

例如，对于右图中的带权图，按**Prim算法**选取边的过程如下页图所示。



4.6 最小生成树





4.6 最小生成树

- 在上述Prim算法中，还应当考虑如何有效地找出满足条件$i \in S, j \in V-S$，且权c[i][j]最小的边(i,j)。实现这个目的的较简单的办法是设置2个数组closest和lowcost。

  closest[j]是j在S中的邻接顶点满足c[j][closest[j]]<=c[j][k],即为同j最近的节点； lowcost[j]=c[j][closest[j]]。

- 在Prim算法执行过程中，先找出V-S中使lowcost值最小的顶点j，然后根据数组closest选取边(j,closest[j］)，最后将j添加到S中，并对closest和lowcost作必要的修改。

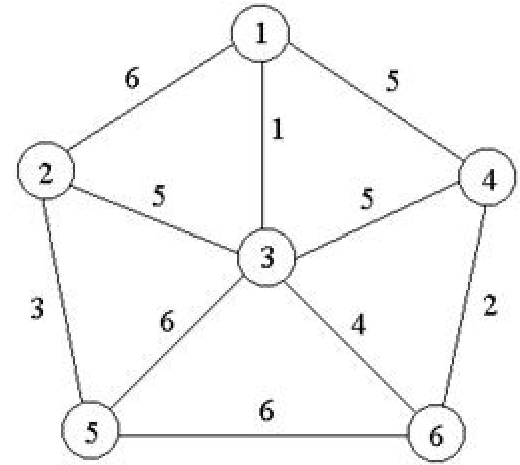
- 用这个办法实现的Prim算法所需的计算时间为 $O(n^2)$

4.6 最小生成树

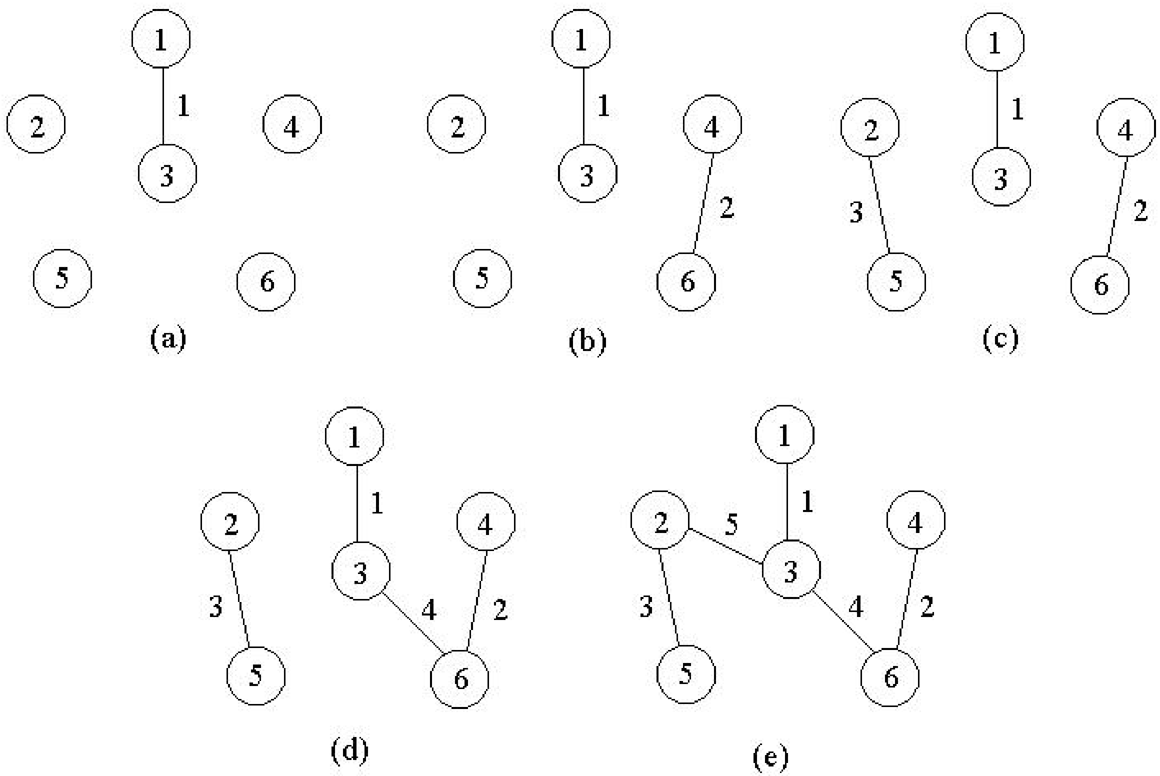
3 Kruskal算法

Kruskal算法构造G的最小生成树的基本思想是，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。然后从第一条边开始，依边权递增的顺序查看每一条边，并按下述方法连接2个不同的连通分支：当查看到第k条边(v,w)时，如果端点v和w分别是当前2个不同的连通分支T1和T2中的顶点时，就用边(v,w)将T1和T2连接成一个连通分支，然后继续查看第k+1条边；如果端点v和w在当前的同一个连通分支中，就直接再查看第k+1条边。这个过程一直进行到只剩下一个连通分支时为止。

4.6 最小生成树



例如，对右图的连通带权图，按**Kruskal**算法顺序得到的最小生成树上的边如下图所示。



4.6 最小生成树

- 关于集合的一些基本运算可用于实现Kruskal算法。
- 按权的递增顺序查看等价于对优先队列执行removeMin运算。可以用堆实现这个优先队列。
- 对一个由连通分支组成的集合不断进行修改，需要用到抽象数据类型并查集UnionFind所支持的基本运算。
- 当图的边数为e时，Kruskal算法所需的计算时间是 $O(e\log e)$。当 $e=\Omega(n^2)$ 时，Kruskal算法比Prim算法差，但当 $e=o(n^2)$ 时，Kruskal算法却比Prim算法好得多。

4.7 多机调度问题

- 多机调度问题要求给出一种作业调度方案，使所给的n个作业在尽可能短的时间内由m台机器加工处理完成。

- 约定，每个作业均可在任何一台机器上加工处理，但未完工前不允许中断处理。作业不能拆分成更小的子作业。

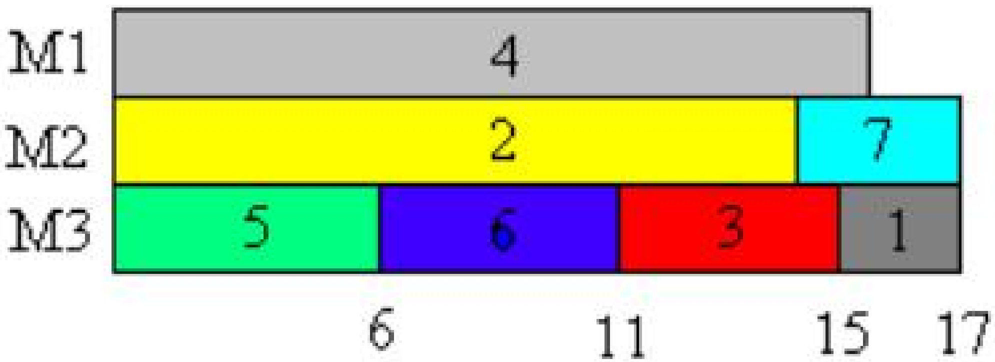
- 这个问题是NP完全问题，到目前为止还没有有效的解法。对于这一类问题,用贪心选择策略有时可以设计出较好的近似算法。

4.7 多机调度问题

- 采用最长处理时间作业优先的贪心选择策略可以设计出解多机调度问题的较好的近似算法。
- 按此策略，当 $n \le m$ 时，只要将机器i的[0, ti]时间区间分配给作业i即可，算法只需要O(1)时间。
- 当 $n > m$ 时，首先将n个作业依其所需的处理时间从大到小排序。然后依此顺序将作业分配给空闲的处理机。算法所需的计算时间为O(nlogn)。

4.7 多机调度问题

例如，设7个独立作业{1,2,3,4,5,6,7}由3台机器M1，M2和M3加工处理。各作业所需的处理时间分别为{2, 14, 4, 16, 6, 5, 3}。按算法greedy产生的作业调度如下图所示，所需的加工时间为17。



贪心法小结

(1) 适用于组合优化问题. 求解过程是多步判 断. 判断的依据是局部最优策略，使目标值达到最大(或最小)，与前面的子问题计算结果无关.

(2) 局部最优策略的选择是算法正确性的关键.

(3) 正确性证明方法：数学归纳法、交换论证. 使用数学归纳法主要通过对算法步数或者问题规模进行归纳. 如果要证明贪心策略是错误的，只需举出反例.

(4) 自顶向下求解，通过选择将问题归约为小的子问题.

(5) 如果贪心法得不到最优解，可以对问题的输入进行分析或者估计算法的近似比.

(6) 如果对原始数据排序之后，贪心法往往是一轮处理，时间复杂度和空间复杂度低.