

3. 贪心

每一阶段都找局部最优解

贪心选择性质

- 步骤

- 设问题有一个整体最优解，并证明可修改这个最优解，使其以贪心选择开始，修改后的解也是问题的最优解
- 运用数学归纳法证明：每步贪心选择-->问题整体最优解

活动安排

- 问题描述：n个活动集合E={1, 2, ..., n}，每个活动i占用时间 $[S_i, f_i]$
- 活动i和活动j相容： $S_i \geq f_j \text{ and } S_j \geq f_i$
- 在活动集合中选择最大的相容活动子集合
 - 最早结束的活动优先安排，n个活动按结束时间非减序排序
 - 伪代码展示

```
template <class Type>
void GreedySelector(int n, Type s[], Type f[], bool A[]){
    //n个活动, s[]存活动开始时间, f[]存活动结束时间, A[]表示活动安排与否
    A[1]=true; //排第1的活动最先结束, 直接放入A
    int j=1; //记录当前活动
    for(int i=2; i<=n; i++){ //从第2个活动开始检测
        if (s[i]>=f[j]){ //判断是否相容
            A[i]=true; //若相容则放入A
            j=i;
        }
        else A[i]=false;
    }
}
```

证明：1)证明活动安排问题有一个最优解以贪心选择开始。

设 A 是所给的活动安排问题的一个最优解，且全部活动也按 f_i 非减序排列， A 中的第一个活动是活动 k 。

若 $k=1$ ，则 A 就是一个以贪心选择开始的最优解；

否则，构造

$$B = A - \{k\} \cup \{l\}$$

(下面证明 B 也是该活动安排问题的最优解)

2) 每一步贪心选择→问题的整体最优解

若 A 是 E 的最优解，则 $A' = A - \{l\}$ 是 $E' = \{i \in E \mid s_i \geq f_1\}$ 的最优解。

如果能找到 E' 的一个解 B' ，它包含比 A' 更多的活动，则 $B' \cup \{l\}$ 优于 A ，即它包含比 A 更多的活动。

矛盾

对贪心选择次数用数学归纳法即知，贪心算法

GreedySelctor 最终产生原问题的最优解。

背包

- 问题描述：与 0-1 背包相似，但在选择物品 i 放入背包时，可以选择放入物品 i 的一部分
- 0-1 背包问题不能用贪心解决：贪心方法不能保证最终能将背包装满，部分闲置背包空间使价值降低
- 解题思路：
 - 计算每个物品的单价，按照单价从大到小排，优先装入单价高的物品

最优装载

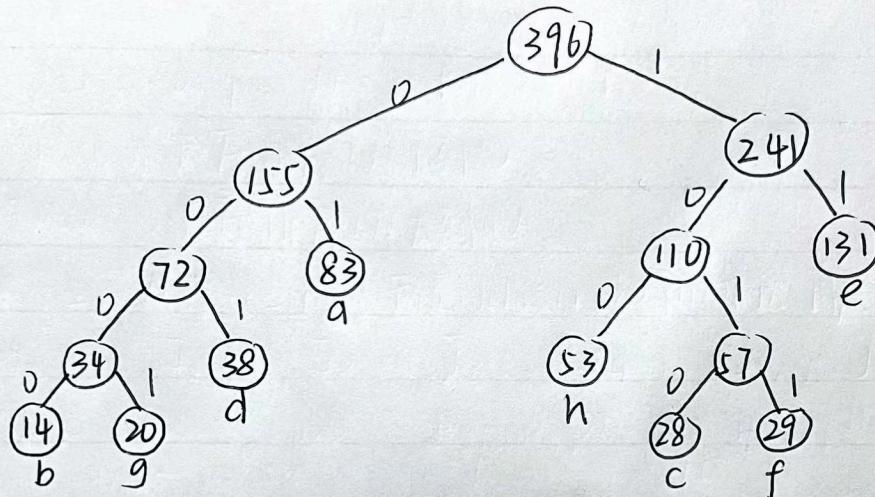
- 问题描述：有一批集装箱要装入载重量为 c 的轮船上，每个集装箱 i 的重量为 w_i ，要求在限制条件下，尽可能多地装入集装箱
- 解题思路：每次选择重量最轻者有限装载

哈夫曼编码

- 找到使平均码长最小的前缀码编码方案
- 时间复杂度: $O(n \log n)$
- 解题思路: 每次将频率最小的两个结点结合

设在1000个字母的文章中各字母出现的频率为:

$a = 83 \quad b = 14 \quad c = 28 \quad d = 38 \quad e = 131 \quad f = 29 \quad g = 20 \quad h = 13$, 求最优编码



$a = 01 \quad b = 0000 \quad c = 1010 \quad d = 001 \quad e = 11 \quad f = 1011 \quad g = 0001 \quad h = 100$

(1) 证明最优子结构

1. 假设已知大问题的一个最优解。
2. 从中取出某个部分(子问题), 假设这个部分不是最优的。
3. 那么可以用更优的子方案替换它, 从而让大问题更优 \rightarrow 矛盾。
4. 所以子问题必须最优。

在哈夫曼中:

假设 (A, B, C) 最优编码树 T , 合并 B, C 为 X 得到树 T' 。

如果 T' 对新字符集 (A, X) 不是最优的, 那么可以找到一个更优的 T'' , 再把 X 展开成 B, C , 会得到 (A, B, C) 的更优树, 与 T 最优矛盾。

(2) 证明贪心选择性质

1. 证明: 存在某个最优解包含我们的贪心选择。
2. 通常用“替换法”: 假设某个最优解没有选这个贪心选择, 我们可以修改它, 让它选这个贪心选择, 而且不会变差。

在哈夫曼中:

假设有一个最优树, 最小频率的两个字符 (B 和 C) 不是最深的兄弟。

那么我们可以交换节点, 把频率更小的字符放到更深的位置, 或者让 B 和 C 成为兄弟且最深, 这样不会增加总成本 (因为频率低的深度大是好的)。

所以存在一个最优树, 它一开始就合并了 B 和 C —— 这就是我们的贪心选择。

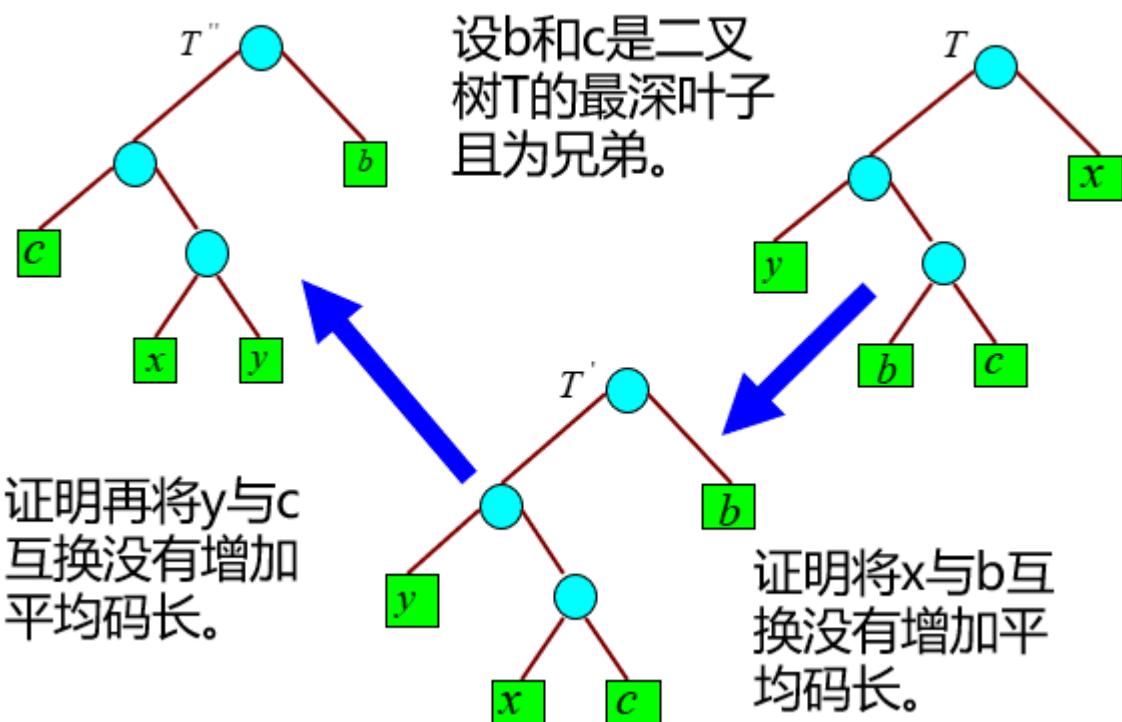
1) 贪心选择性质

设 x 和 y 是字符集 C 中具有最小频率的两个字符，证明存在 C 的最优前缀码使 x 和 y 具有最长、相同的码长且仅最后一位编码不同。

证明：

设二叉树 T 表示 C 的任意一个最优前缀码方案。只要证明可以对 T 做适当修改后，得到一棵新的二叉树 T' ，新树中， x 和 y 是最深叶子且为兄弟。
同时，新树也是 C 的最优前缀码方案。

1) 贪心选择性质



由于 x 和 y 是 C 中具有最小频率的两个字符，

故 $f(x) \leq f(b), f(y) \leq f(c)$ 。

$$\begin{aligned}B(T) - B(T') &= \sum_{c \in C} f(c)d_T(c) - \sum_{c \in C} f(c)d_{T'}(c) \\&= f(x)d_T(x) + f(b)d_T(b) - f(x)d_{T'}(x) - f(b)d_{T'}(b) \\&= f(x)d_T(x) + f(b)d_T(b) - f(x)d_T(b) - f(b)d_T(x) \\&= (f(b) - f(x))(d_T(b) - d_T(x)) \geq 0\end{aligned}$$

$$B(T') \leq B(T) \longrightarrow B(T') = B(T)$$

• T 是最优前缀码，所以 $B(T) \leq B(T')$

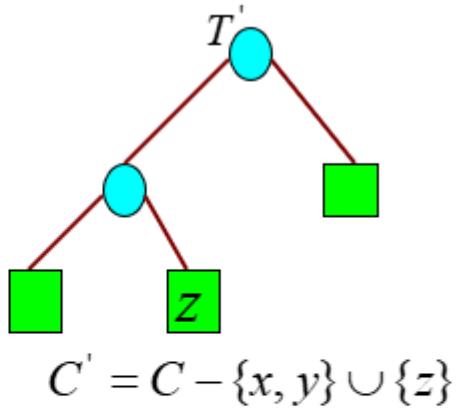
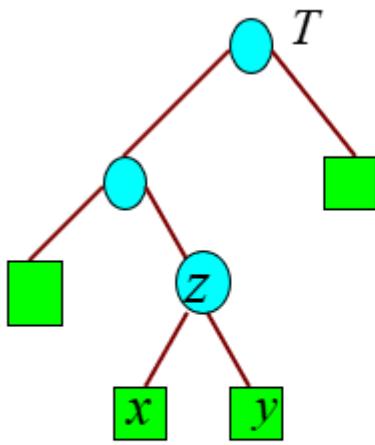
2) 最优子结构性质

设 T 表示 C 的一个最优前缀码方案。 x 和 y 是树 T 中的叶子节点且为兄弟。 z 是它们的父亲。

若将 z 看做是具有频率 $f(z) = f(x) + f(y)$ 的字符，

则证明 $\text{树 } T' = T - \{x, y\}$ 表示字符集 $C' = C - \{x, y\} \cup \{z\}$ 的一个最优前缀码即可。

2) 最优子结构性质



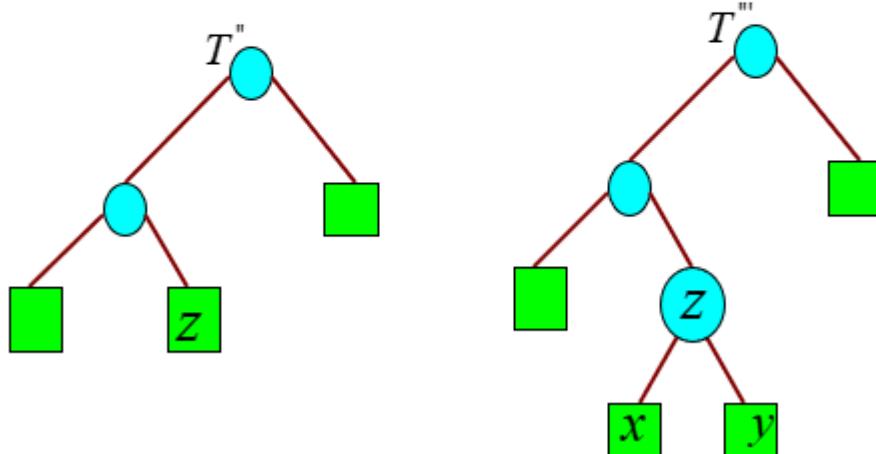
$$f(x)d_T(x) + f(y)d_T(y) =$$

$$(f(x) + f(y))(d_{T'}(z) + 1) = f(x) + f(y) + f(z)d_{T'}(z)$$

$$B(T) = B(T') + f(x) + f(y)$$

若 T' 所表示的字符集 C' 的前缀码不是最优的，

则有 T'' 表示的 C' 的前缀码使得 $B(T'') < B(T')$ 。



若将x和y加入树T'作为z的儿子，则得到表示字符集C的前缀码的二叉树T''。

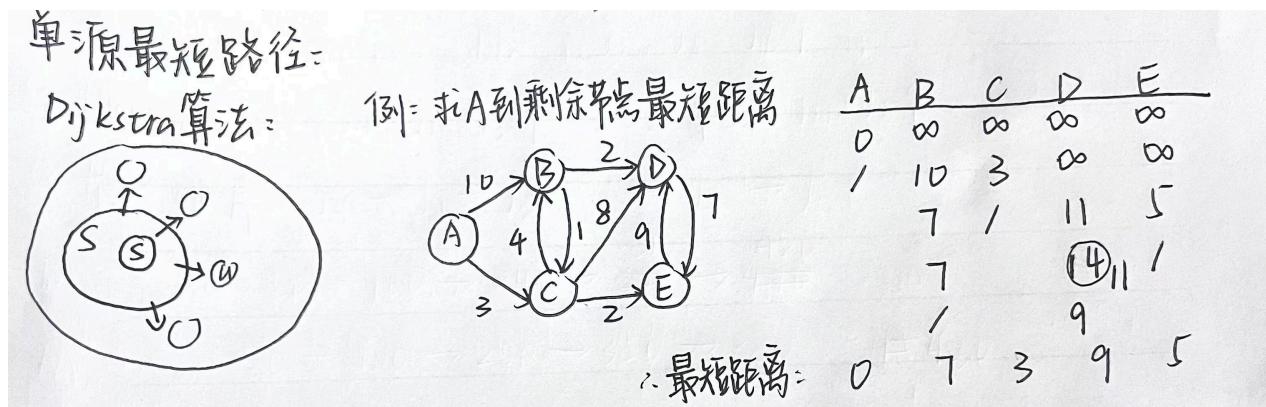
$$B(T'') = B(T') + f(x) + f(y)$$

$$< B(T') + f(x) + f(y)$$

这与
T为最优
矛盾

单源最短路径

- 找边的权值之和最小的路径
- 不存在最短路径的情况：负权回路、不可达



(1) 贪心选择性质

(2) 最优子结构性质

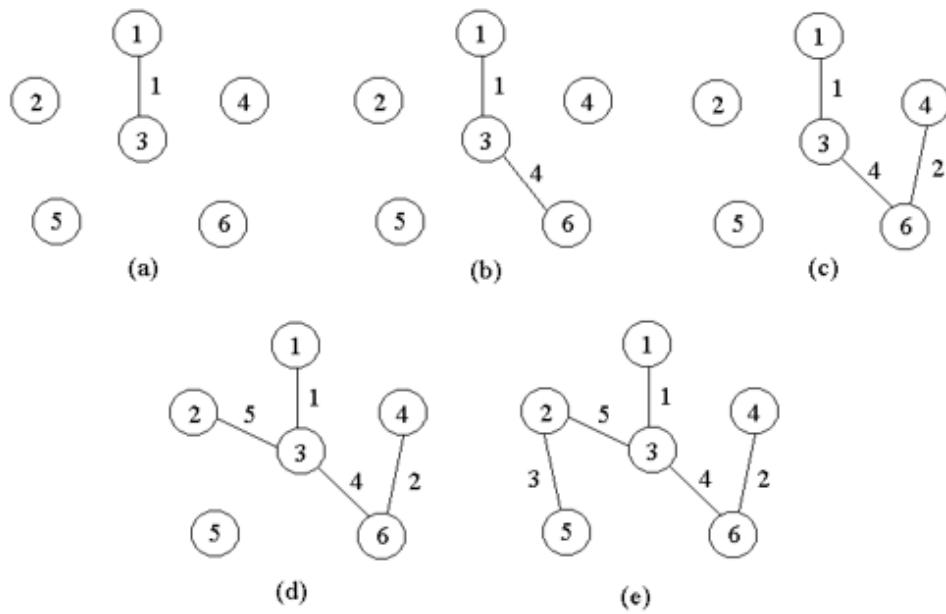
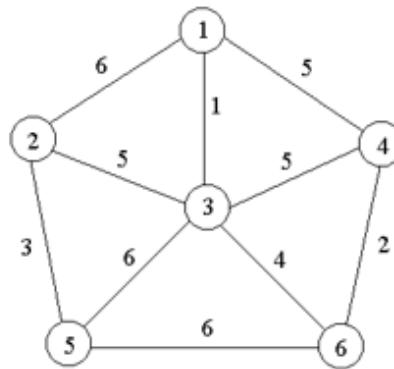
(3) 复杂性分析

对于具有n个顶点和e条边的带权有向图，如果用带权邻接矩阵表示这个图，那么Dijkstra算法的主循环体需要 $O(n)$ 时间。这个循环需要执行 $n-1$ 次，所以完成循环需要 $O(n^2)$ 时间。算法的其余部分所需要时间不超过 $O(n^2)$ 。

最小生成树

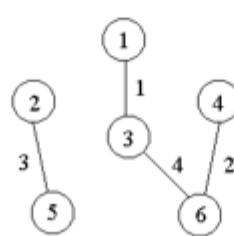
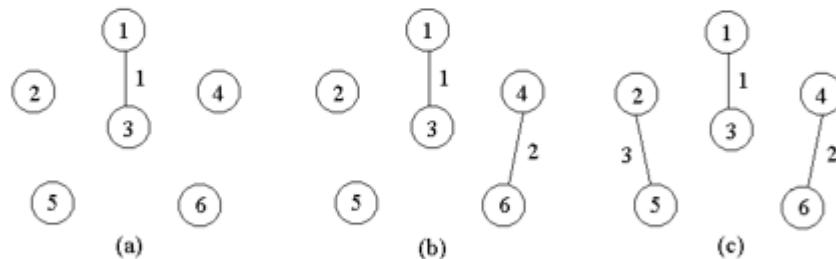
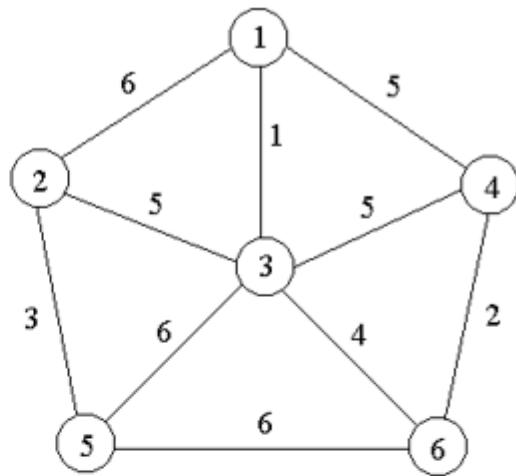
- Prim算法：集合扩散，向外选权值最小的边，直至集合为全集

Prim算法过程

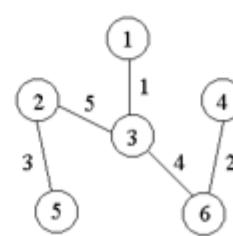


- Kruskal算法：一直选最小的边，直至连通所有孤立的点

Kruskal算法过程



•



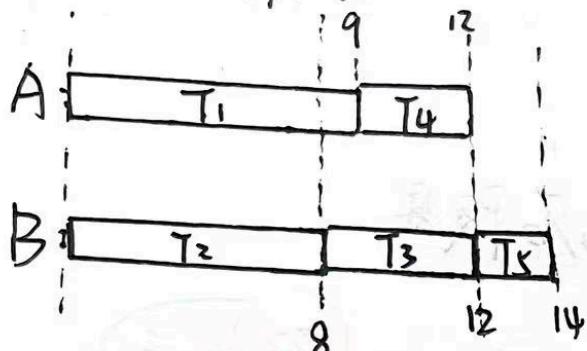
多机调度 (NP问题)

■ 多机调度问题 (NP)

任务	T ₁	T ₂	T ₃	T ₄	T ₅
耗时	9	8	4	3	2

只有两台机器可以执行任务

贪心(优先最长任务 & 分配给最早空闲)



由贪心算法，局部最优，可得最快完成时间为14。

最优分配：

$$A: T_1 + T_3 = 9 + 4 = 13$$

$$B: T_2 + T_4 + T_5 = 13$$

以上分配方法比贪心得到的完成时间更短。