参考答案

7.1

```
DPtaskSelect()
    sort by finish time
    for i←0 to n+1 do
       for j←0 to i-1 do
            c[i,j] ←0
    for i←0 to n+1 do
        for j←i+1 to n+1 do
            max1←0
            for k←i+1 to j-1 do
                if task[k].s≥task[i].f and task[k].f≤act[j].s then
                    if c[i,k]+c[k,j]+1>max1 then
                        max1←c[i,k]+c[k,j]+1
                        c[i,j] ←max1
    return c[0,n+1]
```

7.2

最优子结构性质证明同书上。

贪心选择性质证明：

子问题定义 S_{ij} 同书上。设 $S_{ij} \neq \varnothing$ 且 a_m 为 S_{ij} 中开始时间最大的任务，即

$$s_m = \min\{s_k : a_k \in S_{ij}\}$$

(1) a_m 为 S_{ij} 中开始时间最大的任务，则 a_m 一定包含在子问题中最优解中

证明：假设 A_{ij} 为子问题 S_{ij} 的最优解，且将 S_{ij} 中任务按开始时间降序排列，且设 a_k 为 S_{ij} 中第一个任务，即 $A_{ij} = \{a_k, \cdots\}$

如果 $a_k = a_m$ 则 a_m 在 S_{ij} 的最优解中

如果 $a_k \neq a_m$ 则

将 a_k 用 am 替换掉得解 $A'_{ij} = \{A_{ij} - \{a_k\}\} \cup \{a_m\}$。因为 a_m 为开始时间最大的任务，所以 $s_m \geq s_k$，所以 A'_{ij} 中各任务相互兼容而且 $|A'_{ij}| = |A_{ij}|$。这样，我们得到了一个包含任务 a_m 的最优解 A'_{ij}，故所证成立。

(2) 子问题 S_{im} 是空集，即选择 a_m 后，只剩下唯一一个可能具有非空解的子问题 S_{mj}。

证明：假设 S_{im} 不是空集，即存在一个任务 $a_k \in S_{im}$，满足

$f_i > s_i \ge f_k > s_k \ge f_m > s_m$

即$s_k > s_m$这与a_m是开始时间最大的任务相矛盾，故所证成立。

算法可以描述如下：

(1)原问题$S_{0(n+1)}$按开始时间降序排列

(2)贪心选择S_{ij}中开始时间最大的任务a_m

(3)将a_m加入到最优解中

(4)用同样的方法解决子集S_{mj}

7.3

将 n 个活动看作是直线上的 n 个半闭活动区间$[s_i, f_i)$ ，实际上就是求这 n 个半闭区间的最大重叠数，因为重叠的活动区间所对应的活动是互不相容的。若这 n 个活动区间的最大重叠数为 m，则这 m 个重叠区间所对应的活动互不相容，因此至少要安排 m 个教室来容纳这 m 个活动。

算法可以描述如下：

(1)先将活动按开始时间s_i升序排列$S = (a_1, a_2, \cdots, a_n)$

(2)维持两个教室的队列：QFree QBusy，初始时两个队列都为空。线性扫描活动序列 S，调度一个新活动时，从 QFree 中选择一个教室，若 QFree 为空则打开一个新的教室。将正在使用的教室从 QFree 中取出放入 QBusy。若一个活动结束了，则将其占用的教室从 QBusy 中取出放入 QFree 中。调度完所有活动之后，QFree 中的教室个数就为所求。

正确性证明：

假设由算法求得：调度 n 个活动需要 m 个教室，则我们可以知道 n 个活动集合 S 中一定含有这样一个活动子集

$$S_{ij} = \{a_k \in S : s_i \le s_k < f_i, f_k > f_j\} \text{且} |S_{ij}| \ge m$$

即S_{ij}表示在活动a_i开始到结束这段时间内，a_i活动之后还有 m-1 个活动开始了但还未结束，且这些活动按开始时间升序排列。

因为若不含这样的活动子集，则 S 的所有活动子集的长度都小于 m，即 S 中某时段同时处于活动状态的活动个数至多为 m-1 个，那么算法得出的至多教室数也为 m-1 而不是 m。

由上可知 S 中一定含有长度至少为 m 且活动同时发生活动子集，即任何求解的算法得到的教室数都等于或大于 m

因此算法得出的解 m 为最优的

7.4

同 0/1 背包问题的证明类似，略。

7.5

算法思想：按价值来贪心。

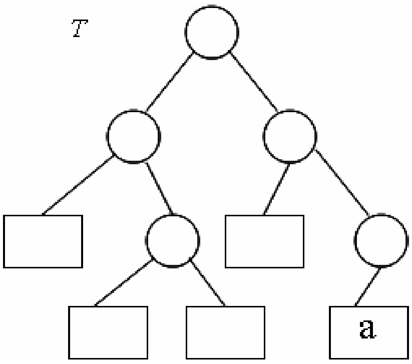
最优子结构性质类似书上证明。

贪心选择性质：

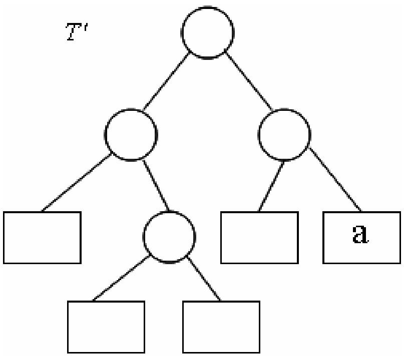
假设物品按价值大小降序排列，即$v_1 \ge v_2 \ge \cdots \ge v_n$，且$w_1 \le w_2 \le \cdots \le w_n$，令$V[i,w]$表示将物品 1 到物品 i 的物品装入载重量为 w 的背包中所能获得的最大价值，即子问题(i,w) 的最大价值。按贪心选择策略，下一步要在物品$i+1,\cdots,n$ 中选择满足条件$w+w_k \le W$ 且价值最大的物品 k。现在要证明这个物品 k 一定在某个最优解中。

假设物品 k 不在最优解中，则最优解 X ：$x_1 \ \ x_2 \cdots x_k \cdots x_j \cdots x_n$ 中，有 $x_k = 0$，且有某个 $x_j = 1 (j > k)$，其中$v_k \ge v_j$ 且$w_k \le w_j$。令$x_k = 1$，$x_j = 0$，其它不变，我们可以得到另一个解X'：$x_1 \ \ x_2 \cdots 1 \cdots 0 \cdots x_n$，这相当于从背包里取出物品 j 放入物品k，由于$v_k \ge v_j$ 且$w_k \le w_j$，显然新解X'的价值更大且没有违反约束条件，这与假设 X 是最优解矛盾。

7.6



(1)



(2)

反证法证。假设不满的二叉树T 对应一种最优前缀编码，如图(1)所示。将图(1)转化为图(2)的二叉树T'，除了字符 a 外，其他字符的深度均相同，此时有

$$\begin{aligned}
B(T') &= \sum_{c\in C} f(c) d_{T'}(c) \\
&= \sum_{c\in C} f(c) d_T(c) - f(a) d_T(a) + f(a) d_{T'}(a) \\
&= B(T) - f(a) d_T(a) + f(a)[d_T(a) - 1] \\
&= B(T) - f(a) < B(T)
\end{aligned}$$

所以T'是比T 代价还小的解，这与T 为最优解相矛盾。

7.7

反证法证。

假设我们将字符表按出现频度的单调递减顺序排序：$(c_1, c_2, ..., c_n)$，其中 $f(c_1) \geq f(c_2) ... \geq f(c_n)$，对应于此字母表不存在编码长度单调递增的最优编码。即对所有的最优编码树 T，至少存在这样一对字符 c_i, c_j，有：

$$f(c_i) \geq f(c_j),\quad d_T(c_i) \geq d_T(c_j)$$

对 T 进行改造，将叶子 c_i 与 c_j 互换，可得树 T'

$$\begin{aligned} B(T) - B(T') &= f(c_i)d_T(c_i) + f(c_j)d_T(c_j) - f(c_i)d_{T'}(c_i) - f(c_j)d_{T'}(c_j) \\ &= f(c_i)d_T(c_i) + f(c_j)d_T(c_j) - f(c_i)d_T(c_j) - f(c_j)d_T(c_i) \\ &= [f(c_i) - f(c_j)][d_T(c_i) - d_T(c_j)] \geq 0 \end{aligned}$$

即 $B(T) \geq B(T')$，这与 T 为最优解相矛盾。

7.8

7.9

贪心策略：
从大的面额开始除。

7.10

设客户 i 的等待时间为 w_i，则 $w_i = t_1 + ... + t_i$，则平均等待时间为 $\sum w_i / n$，因此贪心策略为将服务时间由小到大排序，进行处理。

7.11

假设沿路有 m 个加油站：$1,2,\cdots,m$ 且两加油站 i， j 之间的距离为 d_{ij}，如果在加油站 i 加油，则令 $x_i = 1$，否则 $x_i = 0$，则原问题可建模为

$$\min \sum x_i$$

且满足任意两加油站的之间的距离 $d_{ij} \leq n$。

使用如下贪心策略：在油量有剩余的情况下尽可能多地行使路程，在经过某个加油站时，若剩余油量不足以行使到下一站，则必须停下来加油。
正确性证明：

假设算法得出需要停靠的站点为 p 个，分别为 $O_1, O_2, \cdots, O_p$，则由贪心策略可知，

$d_{O_iO_{i+1}} \leq n$ 且 $d_{O_iO_{i+2}} > n$

若贪心算法不能得到最优解，则存在$O_1, O_2, \cdots, O_q$，其中$q < p$且满足问题的约束条件，不妨假设$q = p - 1$，即要在p个加油站的两个站点i, k之间取消一个站点j（详细地，可用归纳法证明），即

$O_1, \cdots, O_i, O_j, O_k \cdots, O_p$

$O_1, \cdots, O_i, O_k \cdots, O_q$

由于当取消O_j后，$d_{O_i O_k} > n$，此时即使在站点i加满油也无法行驶到站点k，所以$O_1, \cdots, O_i, O_k \cdots, O_q$不是一个可行解，这与它是最优解矛盾。

7.12

对于给定正整数x, a，函数$y = a^x, y = x^a$均为增函数，即

若$a_i > b_j$，则${a_i}^{b_i} > {a_i}^{b_j}$

若$a_i > a_j$，则${a_i}^{b_j} > {a_j}^{b_j}$

由上式可得${a_i}^{b_i} > {a_i}^{b_j} > {a_j}^{b_j}$

贪心策略：每次都选择最大的a_i, b_i，其组成的${a_i}^{b_i}$最大。

正确性证明：

设由贪心算法得到的回报为

$C = {a_1}^{b_1} {a_2}^{b_2} \cdots {a_n}^{b_n}$，其中$a_1 \ge a_2 \ge \cdots \ge a_n$，$b_1 \ge b_2 \ge \cdots \ge b_n$

假设贪心算法得到的回报不是最优的，则必定存在某个解，其回报为

$C' = {a_1'}^{b_1'} {a_2'}^{b_2'} \cdots {a_n'}^{b_n'}$

比C更优。其中$a_1', a_2', \cdots, a_n'$为$a_1, a_2, \cdots, a_n$的一个重排列，同理$b_1', b_2', \cdots b_n'$为$b_1, b_2, \cdots, b_n$的一个重排列

由于$a_1, a_2, \cdots, a_n$为按降序进行排列，则$a_1', a_2', \cdots, a_n'$中必定至少存在这样一对数(a_i', a_j')：

$$i < j \text{ 且 } a_i' \le a_j'$$

否则两个数列就相等了。

不妨设只存在一对这样的数(a_i', a_j')且先不考虑b_i'，即$b_i = b_i'$，在排列

$a_1', a_2', \cdots, a_n'$ 中，将 a_i' 和 a_j' 交换，可得排列 $a_1, a_2, \cdots, a_n$。

由 $C' = a_1'^{b_1'} a_2'^{b_2'} \cdots a_n'^{b_n'} = a_1'^{b_1'} \cdots a_i'^{b_i'} \cdots a_j'^{b_j'} \cdots a_n'^{b_n'}$

$$= a_1'^{b_1'} \cdots a_i'^{b_i'} \cdots a_j'^{b_j'} \cdots a_n'^{b_n'} (a_j'^{b_i'} a_i'^{b_j'})/(a_j'^{b_i'} a_i'^{b_j'})$$

$$= a_1'^{b_1'} \cdots a_j'^{b_i'} \cdots a_i'^{b_j'} \cdots a_n'^{b_n'} (a_i'^{b_i'} a_j'^{b_j'})/(a_j'^{b_i'} a_i'^{b_j'})$$

$$= C\ a_i'^{b_i'-b_j'} / a_j'^{b_i'-b_j'}$$

$$= C\ a_j^{b_i'-b_j'} / a_i^{b_i'-b_j'}$$

因为 $b_i \ge b_j$ 所以 $b_i - b_j \ge 0$，又因为 $a_i \ge a_j$，所以 $a_j^{b_i'-b_j'} / a_i^{b_i'-b_j'} \le 1$，

$C' \le C$，这与假设矛盾。故所证成立。

7.13

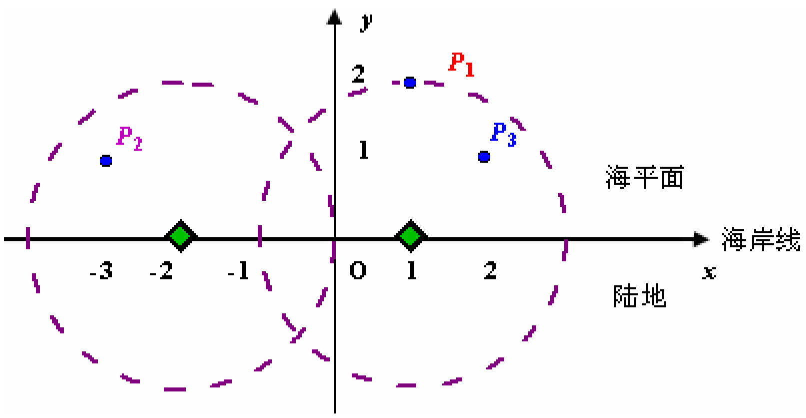


图 7.11

7.14

实验题

7.15 将背包问题分别用三种贪心算法实现，用实验分析方法分析哪个贪心算法更有效。

7.16 完成 XOJ 如下题目：1061，1062。

7.17 完成 POJ 如下题目：1017，1018，1042，1065，1083, 1089，1230，1328，1659，1716，1744，2751，3069，3687。