HWK9:

16.1-5 将活动按照截止时间由近到远排序为 $a_1\ a_2\ \dots\ a_n$

a_i 的开始时间为 s_i，结束时间为 f_i

开始时间为 0，结束时间为 t.

设 $A[i][j]$ 为时间 $i\sim j$ 内 $\sum v_k$ 的最大值.

$$A[i][j] = \max_{k\in J,\ s_k\geq i,\ f_k\leq j} A[i][s_k] + A[f_k][j] + v_k. \quad J=\{0,1,\dots,n\}$$

利用此式递归找出 $A[0][t]$　　时间复杂度为 $O(n^3)$

16.2-4 策略：选择尽量远但距离上一次补水地点小于 m 公里的补水点.

证明：假设补水点的最优解为 $B=[b_1,\dots,b_m]$

贪心策略生成的解为 $G=[g_1,\dots,g_k]$

(eg: b_i、g_i 可理解为对应补水站距离起点的里程数)

设 b_t, g_t 为从左往右第一对不同解

则 $b_t < g_t$

设 B 中距 g_t 最近的两个补水点为 b_j, b_{j+1}

则 $b_t \leq b_j < g_t < b_{j+1}$

则显然 $b_{j+1} - g_t < m$，因为 $b_{j+1} - b_j \leq m$

∴ b_t 可以被 g_t 替换，并且 b_t 与 g_t 中间的补水点被略去

∴ B 可以经过多次上述操作变为 G

且 $|B| \geq |G|$

∴ G 就是最优解.

16.3-7. 只需要将从优先队列每次选2个数改为选3个数

注意到如果节点总数 n 为偶数，则不能保证每次都能取出3个节点，则此时需要先补一个频率为0的节点，再进行编码.

下证：当 n 为偶数，是否加入频率为0的节点不会改变其它节点的最优编码解.

设 T 为满三叉树，度为3的结点数为 n_3，叶节点数为 n_0.

则 $3n_3+1 = n_0+n_3$

$\Rightarrow n_0 = 2n_3+1$

也就是说为 n 为偶数则对应的编码树一定不是满三叉树，则此时将0加入未满的节点下，依然为最优编码树

∴得证.

而保证编码树为满编码树之后，证明就跟二进制相同，只是将最小的2个节点与最优解最底层2个节点交换改为：

将最小的3个节点与最优解最底层3个节点交换

在证明最优子结构时将3个子节点替换为1个节点即可.

16-2. a. 先执行时间短的任务.

设由贪心策略得到的任务序列 G 为 $[a_{k_1}, a_{k_2}, \dots a_{k_n}]$

最优策略得到的任务序列 B 为 $[a_{j_1}, a_{j_2}, \dots a_{j_n}]$

设 (a_{k_m}, a_{j_m}) 是从左往右第一对不同的任务.

则 $p_{k_m} < p_{j_m}$

设 $j_q = k_m$ 且显然 $q > m$.

将 p_{j_m} 与 p_{j_q} 交换 则得到 $a_{j_1} \dots a_{j_{m-1}} a_{j_q} \dots a_{j_m} a_{j_{q+1}} \dots a_{j_n}$

则只有 $C_m \cdots C_{f-1}$ 会改变为 $C_m' \cdots C_{f-1}'$

可得 $\sum_{i=m}^{f-1} C_i = \sum_{i=m}^{f-1} C_i' + (P_{jm} - P_{km}) \cdot (f-m)$

∴ 对换后 $\sum_{i=1}^{n} C_i$ 减小.

∴ 矛盾. $B=G$ ⇒ 贪心就是最优.

仅用快排, $t = O(n \log n)$

6. 调度算法为执行当前时间点下的剩余时间最短算法.

仅用最小堆, $t = O(n \lg n)$

设一个时间为一个时间片, 所有任务执行时间总和为 N

设由贪心策略得到的时间片任务序列 G 为 $[a_{k_1}, a_{k_2}, \cdots a_{k_N}]$

最优策略得到的时间片任务序列 B 为 $[a_{j_1}, a_{j_2}, \cdots a_{j_N}]$

L_i 为任务 a_i 的剩余时间, t_i 为第 i 个时间片

设 (a_{km}, a_{jm}) 是从左往右第一对不同的任务时间片

则 $L_{km} < L_{jm}$

设在 B 中 t_m 后最近次执行任务 a_{km} 的时间为 t_f

将 $t_m \sim t_f$ 中执行任务 a_{jm} 的时间依次改为

执行 a_{km}, 直到 a_{km} 的剩余时间为 0 或全部替换.

显然只影响 C_{km} C_{jm}

那么考察任务 a_{km} 和 a_{jm} 的关系.

设某后一个 a_{km} 与 a_{jm} 对换的时间为 t_i

	$\downarrow t_m$			$\downarrow t_f$	$\downarrow t_i$
对换前	a_{jm}	a_{jm}	a_{jm} ⋯	a_{km} ⋯	a_{km}
对换后	a_{km}	a_{km}	a_{km} ⋯	a_{jm} ⋯	a_{jm}

如果 t_i 时 $l_{km} \cdot l_{jm}$ 都不为0 则 $C_{km}+C_{jm}=C'_{km}+C'_{jm}$

t_i 时 $l_{km}=0$ $l_{jm}\neq 0$ 则 $C'_{km}<C_{km}$ $C_{jm}=C'_{jm}$

t_i 时 $l_{km}\neq 0$ $l_{jm}=0$ 不可能，因为 T_m 时 $l_{km}<l_{jm}$，

如果 t_i 时 $l_{jm}=0$ 则 t_i 之前会执行 a_{jm} 可以继续对换.

t_i 时 $l_{km}=0$ $l_{jm}=0$ 则显然 $C_{km}+C_{jm}>C'_{km}+C'_{jm}$

∴ 此次对换操作不会增大策略的平均完成时间.

而B可以通过多次对换操作变为G

∴ G为最优策略.