算法课第 3 次作业

	题目 1	题目 2	题目 3	总分
分数				
阅卷人				

1 高压工作

1.1 (a)

例子

	周一	周二	周三
l	5	5	30
h	10	20	500

显然应该是周一和周三选高压, 周二休息所获得的收益最高, 为 510. 二按照算法思想, $20 > (5+5)$, 所以周一休息, 周二高压, 进而周三只能选低压, 收益为 50. 错误!

1.2 动态规划求解答案

假设 $H(i)$ 表示第 i 周选择高压活动时前 i 周一共获得的最大收益, $L(i)$ 表示第 i 周选择低压活动时前 i 周一共获得的最大收益. 显然前 i 周的最大收益为 $Max\{H(i), L(i)\}$(**假设收益总是非负的, 第 i 周至少可以做低压活, 则第 i 周什么都不做显然不如做了低压活收益大**). 初始时

$$H(1) = h_1, L(1) = l_1$$

递推式为

$$H(i) = h_i + \max\{H(i-2), L(i-2)\}$$

$$L(i) = L_i + \max\{H(i-1), L(i-1)\}$$

当 $i <= 0$,$H(i) = L(i) = 0$,
用大小为 2n 的数组 BEF 记录取得收益最大值时的前一个状态. 为此, 我们将所以状态对应到数字上:2n-2 表示第 i 天选择高压状态的情况; 2n-1 表示第 i 天选择低压状态的情况. 如果 $H(i) = h_i + H(i-2)$, 则 $BEF(2n-2) = 2(n-2) - 2$. 另外用 H 和 L 数组记录每个状态收益最大值
等到递推结束后, 取 H(n) 和 L(n) 的最大值就是第 n 天的状态, 不妨设为 i 然后查询 BEF(i), 得到前一个状态; 如果跨越一天, 则被跨越的一天为空闲. 比如, H(n)>L(n), BEF(2n-2)=2(n-2)-2, 则第 i-1 天空闲, i-2 天选择高压工作;

复杂度分析 需要储存每天执行高压动作和低压动作时各自当天的最大收益, 还要记录转入状态, 空间复杂度为 $O(2n \times 2) = O(n)$; 循环 n 次计算, 每个循环内计算复杂度为 $O(1)$, 更新数组复杂度为 $O(1)$, 最后反向查找时的复杂度为 $O(n)$, 所以总的时间复杂度为 $O(n)$.

2 高性能计算

2.1 实例

	day1	day2	day3	day4	day5
x	51	51	51	51	51
s	50	4	3	2	1

最优解 第 1/3/5 天工作, 第 2/4 天休息, 最大处理 150TB 数据

2.2 算法

两个数组 A 和 C：A[i] 表示前 i 天最大总收益，C[i] 表示前 i 天取得最大收益时第 i 天的连续工作天数. 第一天的最大收益为 $max\{x[i], s[1]\}$，前 k 天取得最大收益时必定只有 2 种情况：(1) 继续前一天工作 (2) 前一天休息，今天最大努力工作。**因为 $x[k], s[i] \geq 0$，最后一天工作肯定不会比什么都不做收益低**，情况 (1) 的收益等于前 k-1 天的最大收益加上第 k 天的最大收益，C[k]=C[k-1]+1，情况 (2) 的最大收益为前 k-2 天的最大收益加上第 k 天的收益，C[k]=1。递推式为

$$A[n] = \begin{cases} \max\{A[n-1] + \min\{s[C[n-1]+1], x[n]\}, A[n-2] + \min\{s[1], x[n]\}\} & \text{if n>1} \\ \max\{A[n-1] + \min\{s[C[n-1]+1], x[n]\}, \min\{s[1], x[n]\}\} & \text{if n=2} \\ \min\{s[1], x[1]\} & \text{if n=1} \end{cases}$$

$$C[n] = \begin{cases} 1 & \text{if } n = 1 \text{ or (n=2 and } A[n-1] + \min\{s[C[n-1]+1], x[n]\} > \min\{s[1], x[n]\}) \\ & \text{or } A[n-1] + \min\{s[C[n-1]+1], x[n]\} < A[n-2] + \min\{s[1], x[n]\} \\ C[n-1]+1 & \text{otherwise} \end{cases}$$

最后只要查看 A[n] 的值就是最大收益。最后的 C[n] 天（包括第 n 天）是连续工作，第 n-C[n] 天休息；从第 n-C[n]-1 天开始向前的 C[n-C[n]-1] 天连续工作 (包含第 n-C[n]-1 天)... 以此类推，得到全部 n 天的排班情况。

3 期末得分

3.1 算法思路

最大的平均值等价于最大的得分和。依次求前 i 门课程在 h 小时内可以抢到的最大得分，第 i+1 门课的得分为给第 i 门课分配某一时间后的得分加上前 i 门课在剩余小时内得到的最大得分（递归子问题）。假设 A[i,h] 表示前 i 门课在总时间 h 小时下得到的最大分数和，则

$$A[i,h] = \begin{cases} 0 & \text{if i=0 and h>=0} \\ A[i-1,0] + f_{i+1}(0) & \text{if } h = 0 \\ \max\limits_{0 \leq k \leq h} \{A[i-1, h-k] + f_{i+1}(k)\} & \text{otherwise} \end{cases}$$

3.2 复杂度分析

用二维数组 A 记录最大和，空间代价为 O(nH)。按照课程数递增的顺序计算则在计算 A[i,h] 时，A[i-1,h-k] 已经算过，计算复杂度为 O(1)；计算每个 A[i,h] 的复杂度为 O(H)；对每个 i，要计算 h 在 [0,H] 的所有情况，所以复杂度为 $O(H^2)$，一共计算 n 个课程，所以总的时间复杂度为 $O(nH^2)$