《算法设计与分析》第3次作业答案

姓名：XXX　　　　　　学号：XXXXXXXX

题目1：某公司生产长钢管，然后将钢条切割成不同的长度拿去售卖。不同长度的钢管售价不一样。钢管的长度售价表如下：

长度i(米)	1	2	3	4	5	6	7	8	9	10
售价P_i	1	5	8	9	10	17	17	20	24	30

Table 1: 钢管售价表

有一根长度为n的钢管，请设计一个动态规划算法进行切割，让公司的收益达到最大。注：若长度为n英寸的钢条的价格P_n足够大，最优解可能就是完全不需要切割。

注意：请给出算法思路、递推方程及其解释，并且分别运用自顶向下方法和自底向上的方法给出伪代码。

答：

算法思路： 令$r(n)$表示长度为n米的钢条的最大收益，则有：

$$r(n) = max_{1\leq i\leq n}\{P_i + r(n-i)\}$$

递推方程及解释： 从一端切一段长度为i的钢条下来后，可获得的最大收益即切下来的钢条价格P_i，加上剩下长度为$n-i$的钢条的最大收益

自顶向下伪代码见附录

自底向上伪代码见附录

题目2：买卖股票的最佳时机简单版：给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。如果你最多只允许完成一笔交易（即买入和卖出一支股票一次），设计一个算法来计算你所能获取的最大利润。注意：你不能在买入股票前卖出股票。示例如下：

输入: [7,1,5,3,6,4]

输出: 5

解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格。请设计一个时间复杂度为$O(n)$的算法。

注意：若使用动态规划，请给出算法思路、递推方程及其解释，并用伪代码描述算法；若不是使用动态规划，请给出算法思路、并用伪代码描述算法。

答：

算法思路： 很显然，要想设计一个时间复杂度为$O(n)$的算法，只能对数组做一次遍历。可以用动态规划的方法进行解决，假设前i天的最大收益为$P(i)$，则$P(i)$可能与前 $i-1$ 天的最大收益$P(i-1)$相等，也可能是第 i 天的价格减去前 $i-1$ 天中的最小价格,因此，我们也需要记录前 $i-1$ 天的最小股票价值min。根据以上分析，可以得出动态规划的递推方程如下所示：

$$P(i) = max\{P(i-1), Array(i) - min\}$$

伪代码见附录 只是提供一种思路，这道题比较简单，还有其他写法可以实现$O(n)$的复杂度。

题目3: n个作业$1, 2, ...n$要在又2台机器M_1和M_2组成的流水线上完成加工。每个作业加工的顺序都是先在M_1上加工，然后在M_2上加工。M_1和M_2加工作业i所需的时间分别为a_i和b_i，$1 \leq i \leq n$。流水作业调度问题要求确定这n个作业的最优加工顺序，使得从第一个作业在机器M_1上开始加工，到最后一个作业在机器M_2上加工完成所需的时间最少。

直观上，一个最优调度应使机器M_1没有空闲时间，且机器M_2的空闲时间最少。在一般情况下，机器M_2上会有机器空闲和作业积压两种情况。

设全部作业的集合$N = 1, 2, .., n, S \subseteq N$是$N$的作业子集。在一般情况下，机器$M_1$开始加工$S$中作业时，机器$M_2$还在加工其他作业，要等时间$t$后才可利用。将这种情况下完成$S$中作业所需的最短时间记为$T(S, t)$。流水作业调度问题的最优值为$T(N, 0)$.

注意： 请认真阅读自学课本3.9节流水作业调度，完成以下任务：

(1)给出课本中两种不同动态规划算法思路（递归式+基于Johnson原则）的伪代码（注意是伪代码，不是课本上的实现代码）

(2)复现这两种思路，给出程序运行截图（要有输入，输出以及运行时间）

(3)枚举n，记录运行时间并画出n与运行时间的关系图

伪代码见附录

附录：

Algorithm 1 钢管最大收益-自顶向下伪代码

Input: 价格数组 P; 长度 n; 最大收益数组 r;

Output: 长度为n的钢管进行切割的最大收益$r[n]$

```
预处理r[], 使得r[i] = -inf
function CUT(P, n, r)
    if r[n] ≥ 0 then
        return r[n]
    end if
    if n == 0 then
        temp = 0
    else
        temp = -inf
        for i = 1 → n do
            temp = max(temp, P[i] + CUT(P, n - i, r))
        end for
    end if
    r[n] = temp
    return temp
end function
```

Algorithm 2 钢管最大收益-自底向上伪代码

Input: 价格数组 P; 长度 n; 最大收益数组 r;

Output: 长度为n的钢管进行切割的最大收益$r[n]$

```
function CUT(p, n, r)
    r[0] = 0
    for j = 1 → n do
        temp = -inf
        for i = 1 → j do
            temp = max(temp, P[i] + r[j - i])
        end for
        r[j] = temp
    end for
end function
```

Algorithm 3 买卖股票$O(n)$复杂度算法伪代码

Input: 股票值的数组 $Array$;

Output: 买卖股票获得的最大利润;

function MAXPROFIT($Array$)
    初始化数组$dp \leftarrow 0$
    $min \leftarrow Array[0]$
    $n \leftarrow Array.length$
    for $i \leftarrow 1$ to n **do**
        if $Array[i] < min$ **then**
            $min = Array[i]$
        end if
        $dp[i] \leftarrow max\{dp[i-1], Array[i] - min\}$
    end for
    return $dp[n]$;
end function

Algorithm 4 流水作业调度-递归伪代码　参考blog

Input: 作业个数 n; 作业在M_1上的加工时间 $a[1:n]$ 作业在M_2上的加工时间 $b[1:n]$

Output: 使得加工完成时间最少额n个作业的最优调度 $c[1:n]$和最短完成时间$minTime$;

```
Class PartsSet {
    state[n];            //1:已被加工，0：不被加工
    num                  //当前状态下，被加工的零件个数
    minTime[100]         //数组下标是等待时间t
}
function findMinTime(PartsSet set, int n)
    if set.num == 0 then
        return t
    end if
    mintime ← +inf
    withouI ← set
    for i = 0 → n − 1 do
        withoutI ← set
        if set.state[i] == 1 then
            withoutI.state[i] = 0
            withoutI.num = set.num − 1
            curMinTime ← a[i] + findMinTime(withoutI, b[i] + max(t − a[i], 0)))
            if curMinTime < minTime then
                minTime = curMinTime
            end if
        end if
    end for
    return minTime
end function
function Main
    罗列所有可能的零件组合Set[i],组合总数共为NUM，并且得到Set[i].state[1:n]和Set[i].num
    for workNum = 0 → n do
        for i = 0 → Num − 1 do
            if Set[i].num == n then
                for k = 0 → 100 do
                    Set[i].minTime[k] = findMinTime(Set[i], k)
                end for
            end if
        end for
    end for
    return Set[NUM − 1].minTime[0]
end function
```

Algorithm 5 流水作业调度-基于Johnson法则伪代码

Input: 作业个数 n; 作业在M_1上的加工时间 $a[1:n]$; 作业在M_2上的加工时间 $b[1:n]$;

Output: 使得加工完成时间最少额n个作业的最优调度 $c[1:n]$和最短完成时间k;

Class JobType {
        $key, index$;   //key为机器所用时间，index为作业序号
        job //bool型，为1表示N_1,0表示N_2
        int operator≤(JobType a) const {
                return(key≤a.key)
        }
}
function FlowShop($int\ n, int\ a[], int\ b[], int\ c[]$)
    $d \leftarrow JobType[n]$
    for $i = 0 \to n-1$ **do**
        $d[i].key = a[i] > b[i]?\ b[i] : a[i]$ //按Johnson法则分别取对应的a_i中b_i的最小值作为关键字
        $d[i].job = a[i] \leq b[i]$ //给符合条件$a_i < b_i$的放入到N_1子集标记为true
        $d[i].index = i$
    end for
    $sort(d, n)$ //升序排序
    $j \leftarrow 0,\ k \leftarrow n-1$ //两个指针，对作业调度顺序进行调整
    for $x = 0 \to n-1$ **do**
        if $d[x].job$ **then**   //最小值是a_i，放在最前面执行
            $c[j++] = d[x].index$
        else   //最小值是b_i，放在最后面执行
            $c[k--] = d[x].index$
        end if
    end for
    $j \leftarrow a[c[0]]$ //下面计算执行作业所需时间
    $k \leftarrow j + b[c[0]]$
    for $z = 1 \to n$ **do**
        $j = j + a[c[z]]$
        $k = j < k?\ k + b[c[z]] : j + b[c[z]]$
    end for
    return c, k
end function