数据结构与算法 作业8

姓名：邵宁录　　　学号：2018202195

15.5-4

书上源代码如下所示

```
OPTIMAL-BST(p,q,n)
1  let e[1..n+1,0..n],w[1..n+1,0..n],and root[1..n,1..n]be new tables
2  for i=1 to n+1
3      e[i,i-1]=q_{i-1}
4      w[i,i-1]=q_{i-1}
5  for l=1 to n
6      for i = 1 to n-l+1
7          j = i+l-1
8          e[i,j]=∞
9          w[i,j]=w[i,j-1]+p_j+q_j
10         for r =i to j
11             t =e[i,r-1]+e[r+1,j]+w[i,j]
12             if t<e[i,j]
13                 e[i,j]=t
14                 root[i,j]=r
15 return e and root
```

只需将第10行

```
for r to i
```

变为

```
for root[i, j - 1] to root[i + 1, j]
```

证明如下：

可知，第2行到第4行的时间复杂度为 $\mathrm{O}(2n)$ ，为线性。

因此主要分析的部分在第5行到第14行。

在改变第10行，它的复杂度如下，由于单条语句的复杂度为常数，因此只用 1 来表示：

$$
\begin{aligned}
T(n) &= \sum_{l=1}^{n}\sum_{i=1}^{n-l+1}(root[i+1,\ j]-root[i,\ j-1]+1)\\
&= \sum_{l=1}^{n}(root[1+1,\ 1+l-1]-root[1,\ 1+l-2]\\
&\quad +root[2+1,\ 2+l-1]-root[2,\ 2+l-2]+\cdots\\
&\quad +root[n-l+1+1,\ n-l+1+l-1]-root[n-l+1,\ n-l+1+l-1-1]\\
&\quad +n-l+1)\\
&= \sum_{l=1}^{n}(root[2,\ l]-root[1,\ l-1]\\
&\quad +root[3,\ l+1]-root[2,\ l]+\cdots\\
&\quad +root[n-l+2,\ n]-root[n-l+1,\ n-1]\\
&\quad +n-l+1)\\
&= \sum_{l=1}^{n}(root[n-l+2,\ n]-root[1,\ l-1]+n-l+1)
\end{aligned}
$$

由于 $root[n-l+2,\ n]-root[1,\ l-1]\le n$ ，所以
$(root[n-l+2,\ n]-root[1,\ l-1]+n-l+1)$ 该部分为 $\mathrm{O}(n)$

即上式可写为：

$$
\begin{aligned}
T(n) &= \sum_{l=1}^{n}(root[n-l+2,\ n]-root[1,\ l-1]+n-l+1)\\
&\le \sum_{l=1}^{n}\mathrm{O}(n)\\
&= \mathrm{O}(n^2)
\end{aligned}
$$

即证

15-11

1. 问题分析

首先明确本题的目标：最小化库存成本

以 $cost[i,\ j]$ 表示第 i 个月，累计生产了 j 台设备。显然该动态规划问题的答案为求解 $cost[n,\ D]$ 的值。有状态转移方程如下：

$$
cost[i,\ j]=\begin{cases}cost[i-1,\ j-k]+h(j-D_i), & k\le m\\ cost[i-1,\ j-k]+c(j-k)+h(j-D_i), & k>m\end{cases}
$$

其中，k 为这个月生产的量，D_i 表示 前 i 个月的总需求。

根据题意，要满足以下限制条件：

- 当第 i 个月时，有 $D_i\le j\le D_n$ 。
- $D_{i-1}\le j-k$ ，即 $0\le k\le j-D_{i-1}$

综上所述，有：

$$cost[i,\ j] = \begin{cases} cost[1,\ j], & i = 1 \\ \min_k(cost[i,\ j]), & i \neq 1 \end{cases}$$

其中，由于第一个月一定要出完货，所以有：

$$cost[1,\ j] = \begin{cases} h(j - D_1), & j \leq m \\ c(j - m) + h(j - D_1), & j > m \end{cases}$$

2. 伪代码

```
// d 为每个月的生产要求
// m 为固定员工的生产力
// c 额外雇佣生产每件产品的花费
F(d, m, c):
    n = d.length()
    let D[1...n] be a new array
    D[1] = d[1]
    for i = 2 to n:
        D[i] = D[i - 1] + d[i]
    let cost[1...n, 0...D[n]] and p[1...n, 0...D[n]] be new tables
    for i = 1 to n:
        for j = D[i] to D[n]:
            if i == 1:
                cost[i, j] = h(j-D[i])
                if j > m:
                    cost[i, j] += c*(j - m)
            else:
                cost[i, j] = INFINITY
                for k = 0 to (j - D[i - 1]):
                    t = cost[i - 1, j - k] + h(j - D[i])
                    if k > m:
                        t += c*(k - m)
                    if t < cost[i, j]:
                        cost[i, j] = t
                        p[i, j] = k
    return cost and p
```

3. 复杂度分析

由于一共有 nD 个子问题，且求解子问题的时候，要迭代 $\mathrm{O}(D)$ 次，因此最后的时间复杂度为 $\mathrm{O}(nD^2)$ 。