作业反馈3-1

TC第15.1节练习1、3

TC第15.2节练习2、4

TC第15.3节练习3、5、6

TC第15.4节练习3、5

TC第15.5节练习1

TC第15章问题4

15.1-3

Consider a modification of the rod-cutting problem in which, in addition to a price p_i for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem.

Original：

```
BOTTOM-UP-CUT-ROD(p, n)
let r[0..n] be a new array
r[0] = 0
for j = 1 to n
    q = −∞
    for i = 1 to j
        q = max(q, p[i] + r[j − i])
    r[j] = q
return r[n]
```

```
BOTTOM-UP-CUT-ROD(p, n)
let r[0..n] be a new array
r[0] = 0
for j = 1 to n
    q = −∞
    for i = 1 to j
        q = max(q, p[i] + r[j − i] − c);
    r[j] = max(q, p[i]);
return r[n]
```

15.3-6

Imagine that you wish to exchange one currency for another. You realize that instead of directly exchanging one currency for another, you might be better off making a series of trades through other currencies, winding up with the currency you want. Suppose that you can trade n different currencies, numbered $1, 2, \ldots, n$, where you start with currency 1 and wish to wind up with currency n. You are given, for each pair of currencies i and j, an exchange rate r_{ij}, meaning that if you start with d units of currency i, you can trade for $d r_{ij}$ units of currency j. A sequence of trades may entail a commission, which depends on the number of trades you make. Let c_k be the commission that you are charged when you make k trades. Show that, if $c_k = 0$ for all $k = 1, 2, \ldots, n$, then the problem of finding the best sequence of exchanges from currency 1 to currency n exhibits optimal substructure. Then show that if commissions c_k are arbitrary values, then the problem of finding the best sequence of exchanges from currency 1 to currency n does not necessarily exhibit optimal substructure.

If $c_k = 0$, let a array $d[i,j]$ be the maximum currencies you can exchange, to simplify the problem we just consider one unit of the original currency, so the state transformation equation is

$$d[i,j] = \max(\max_{1 \leq k \leq n}$$

实质：问题已经改变

问：

是否存在其他形式的子结构？

是否满足最后子结构特性？

If $c_i \neq 0$, we can take a expectatio

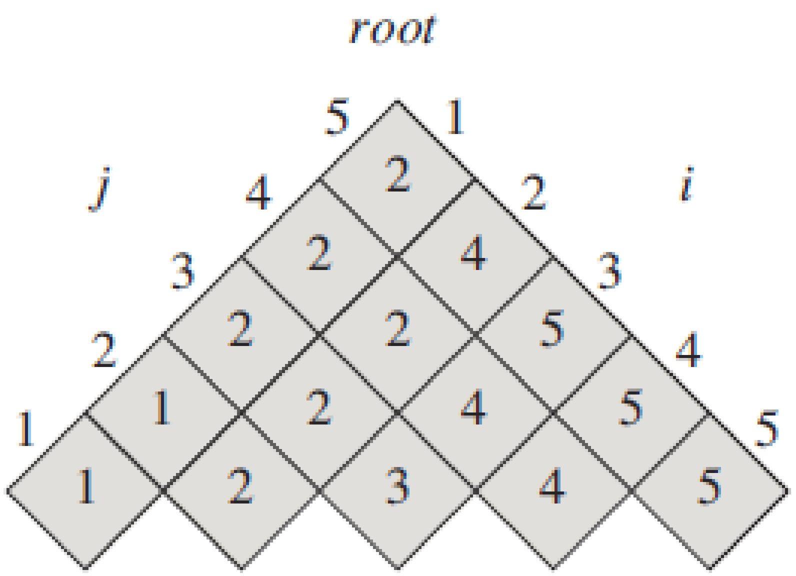
$1, c_1 = 1000$. Though $d[1,2] * d[2,3]$ is higher, we can't choose it because the fee of cut is too high.

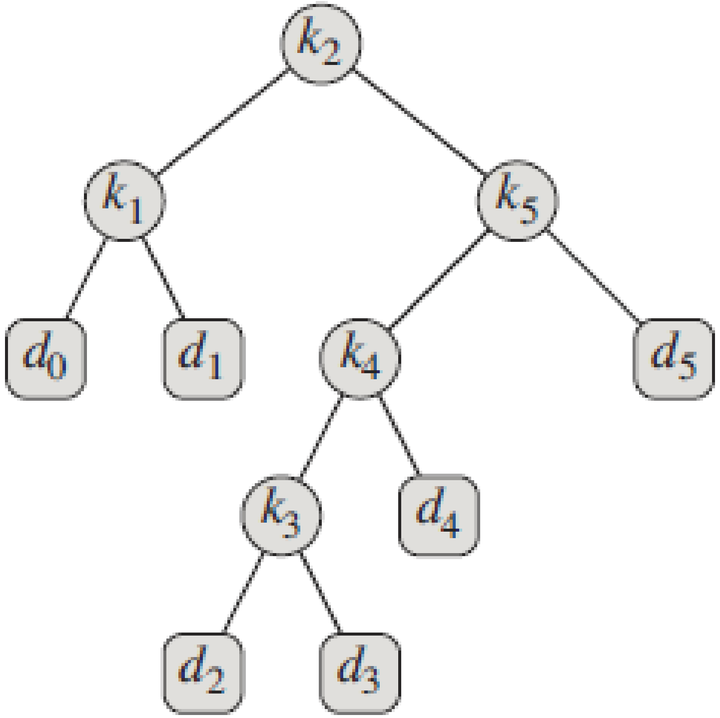
$d[i,j,x]$ = 表示货币i恰好经过x次置换，得到货币j的打额度

$$d[i,j,x] = \max_{1 \leq k \leq n} \left\{ \max_{1 \leq y \leq x} \{ (d[i,k,y] + c_y) * (d[k,j,x-y] + c_{x-y}) \}, r_{ij} \right\}$$

15.5-1

Write pseudocode for the procedure CONSTRUCT-OPTIMAL-BST(*root*) which, given the table *root*, outputs the structure of an optimal binary search tree. For the example in Figure 15.10, your procedure should print out the structure





Is this OK?

```
PRINT-OPTIMAL-BINARY-SEARCH-TREE(root, i, j)
if i == 1 ∧ j == n
    then print k_{root[i,j]} is the root.
else
        r = root[i, j]
        if r == i
            then print d_{i−1} is the left child of k_r.
        else print k_{root[i,r−1]} is the left child of k_r.
                PRINT-OPTIMAL-BINARY-SEARCH-TREE(root, i, r − 1)
        if r == j
            then print d_j is the right child of k_r.
        else print k_{root[r+1,j]} is the right child of k_r.
                PRINT-OPTIMAL-BINARY-SEARCH-TREE(root, r + 1, j)
```

例题3：最长上升子序列(LIS)

- 给定n个整数$A_1, A_2, \dots, A_n$，按照从左到右的顺序选出尽可能多的整数，组成一个上升子序列。
- 例：
  - 给定序列：1, 6, 3,5,7,2,9
  - LIS：1,3,5,7,9

最优子结构是什么？

给定一个序列A[1,…,n]的LIS为L[1,…,m]；

则L[1,…,m-1]一定为A[1,…,index(L[m-1])的LIS

状态？

d(i)表示从以A_i为结尾的LIS的长度

状态转移方程？

$$d(i) = \max\{0, d(j) | j < i, A_j < A_i\} + 1 \qquad O(n^2)$$

实际上可以转换例题2：
每个数对应一个节点，
对于A_i, A_j, 如果i<j且 $A_i < A_j$，
则可在A_i, A_j建立有向边
$A_i \to A_j$，则最终构建一个
有向无环图。找到最长路

例题3：最长上升子序列(LIS)

- 能否进一步加快？
  - 假设给定a,b满足：
    - $A_a < A_b$且$d(a) = d(b)$
  - 则对于后续所有i（$i > a, i > b$）, 选择a不会比选择b差：
    - 如果$A_b < A_i$,则$A_a < A_i$一定满足
    - 而如果$A_a < A_i$,则$A_b < A_i$未必满足
  - 所以，我们只需记录a，就不会丢失最优解
    - 对于相同的d值，我们只保留A中最小的一个
    - 用$g(i)$表示d值为i的最小A中元素的编号（如果不存在，定义为正无穷）。则:
      - $g(1) \le g(2) \le g(3) \le \cdots \le g(n)$
      - 上述g值是动态改变的。
      - 对于给定的元素A_i，我们只需考虑在A_i之前的A_j(j<i)
        - 利用二分查找找到满足$A_{g(k)} \ge A_i$的第一个k, 则：
          - $d(i) = k$
          - 更新$g(k) = i$

$O(n \lg n)$

例题4：最长公共子序列（LCS）

- 给定两个序列A，B，求长度最大的公共子序列长度。
- 例：
  - A：1 5 2 6 8 7
  - B：2 3 5 6 9 8 4
  - LCS：5 6 8 或2 6 8

最优子结构是什么？

假设C[1,k]是A[1,n]和B[1,m]的LCS，
则C[1,k-1]一定也是A[1,index(A,C[k])]和B[1,index(B,C[k])]的LCS

状态？

d(i,j)表示从以A[1,…,i]与B[1,…,j]的LCS长度

状态转移方程？

$O(nm)$

$$d(i,j)=\begin{cases} d(i-1,j-1)+1 & ,if\ A[i]=B[j] \\ \max\{d(i-1,j),d(i,j-1)\} & ,otherwise \end{cases}$$

例题4：最长公共子序列（LCS）

- 能否更快一些？
- LCS与LIS有什么联系？

例：

A：1 5 2 6 8 7

B：2 3 5 6 9 8 4

LCS：5 6 8 或2 6 8

- 能够出现在LCS的字符/数字，一定同时出现在A，B之中！
- 对A中数字，按照其出现顺序先后重新编码
  - A: 1 5 2 6 8 7
  - A’:1 2 3 4 5 6
- 按照A’，对B进行重新编码：
  - B：2 3 5 6 9 8 4
  - B’: 3 0 2 4 0 5 0
- 删除B’中的0，得到B’’：3245
- 对 B’’求LIS