算法基础 HW6

PB20111686 黄瑞轩

1

此时，切割后的价值更改为 $p[i]+r[j-i]-c$

```
CUT-ROD(p, n, c)
    let r[0..n] be a new array
    r[0] = 0
    for j = 1 to n
        q = p[j]
        for i = 1 to j - 1
            q = max(q, p[i] + r[j - i] - c) // 修改书上代码这一行
        r[j] = q
    return r[n]
```

2

设问题的答案是 $f(M:\text{apple\_num},N:\text{plate\_num})$，使用带备忘录的递归。

- 如果 $M\leq N$，则 $f(M,N)=f(M,M)$
  - 如果 $M=1$，则 $f(M,M)=1$
- 如果 $M>N$:
  - 如果 $N=1$，则 $f(M,N)=1$
  - 如果 $N>1$，则 $f(M,N)=f(M-N,N)+f(M,N-1)$
    - 如果 N 个盘子里都放，则 $f(M,N)=f(M-N,N)$
    - 如果不是都放，则需要计算 $f(M,N-1)$ 直至达到边界条件

3

将 $x_1,x_2,\ldots,x_n$ 按升序排列为 $x_{i_1},x_{i_2},\ldots,x_{i_n}$，考察 x_{i_1}，覆盖它的区间若想尽可能多覆盖其他点，需要尽可能右移，所以覆盖 x_{i_1} 的区间左端点就是 x_{i_1}，假设剔除掉这个区间覆盖的那些点之后，最小的是 x_{i_k}，让新的区间左端点为 x_{i_k}。重复，直到所有点都被覆盖为止。

因为每一次的选择都是全局最优的，所以最后的结果也是最优的。

4

假设所有任务耗时都不相同。

- 当 $n\leq m$ 时，每个机器分配至多一个任务并且全部分配即可
- 当 $n>m$ 时，把任务按耗时从大到小排序，每当有机器空闲，就把当前未做的任务中耗时最长的那个任务分配给它

如果每次都把当前未做的任务中耗时最短的那个任务分配给空闲机器，可能长任务完成时间延后，由于任务不能分割，则平均起来耗时就长。

6

每次都用当前能用的面值最大的硬币找钱。

如果在某个选择中不用当前能用的面值最大的硬币，就需要用更多的低面额硬币来填补其地位，会使硬币数量增多（注意到这里 25、10、5 美分的硬币都可以用多个低面额硬币恰好填补其地位）。

7

```
int KNAP_SACK(int W, int n, vector<int>& weight, vector<int>& val) {
    vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= W; i++) {
            if (w - weight[i - 1] < 0) {
                dp[i][j] = dp[i - 1][j]; // 容量不够，不装入
            }
            else {
                dp[i][j] = max(dp[i - 1][j - weight[i - 1]] + val[i - 1], dp[i -
1][j]); // 装不装，择优
            }
        }
    }
    return dp[n][W];
}
```