数据结构与算法I 作业13

2019201409 于倬浩

15-11

设`f[i][j]`表示，当前考虑到了前i个月，第i个月做完后，仓库内剩余j台机器的最小代价。

考虑最优子结构：如果目前我们有一组`MinCost(p,q)`的最优解，如果在第p个月中，库存增加了k，那么前p-1个月的库存即为p-k，那么其实就对应了子问题`MinCost(p-1,q-k)`。如果之前的子问题存在更优的解法，那么由于状态维护了`p`、`q`两个参数，直接将子问题的解换成更优的可以做到无后效性，且可以改善当前问题的答案，因此该问题具有最优子结构。

每次计算`f[i][j]`时，首先当月的订单尽量使用当月刚生产的来满足，不足部分则用上个月来补足。

枚举`i`，若$d_i \le m$，则：

$$f[i][j] = f[i-1][max(j-(m-d_i),0)] + h(j)$$

若$d_i > m$，枚举`j`，则还需枚举当前月份，雇佣额外劳动力生产的机器数量k：

$$f[i][j] = min_{k=0}^{j+d_i-m}(f[i-1][j+d_i-m-k] + ck) + h(j)$$

因此，状态数为$\Theta(Dn)$，单次转移代价最坏$\Theta(D)$，因此算法的时间复杂度为$\Theta(D^2n)$。

空间复杂度可以使用滚动数组方式，从$\Theta(Dn)$优化到$\Theta(D)$。

15-5

- a. 设`f[i][j]`表示当前处理到`x[i], y[j]`时的最小编辑距离，下面来考虑如何转移：

  首先，假设我们当前正在计算`f[i][j]`，则先将其初始化为inf，然后考虑如下转移：

    - `f[i][j] = min(f[i][j], f[i - 1][j - 1] + cost(replace))`
    - `f[i][j] = min(f[i][j], f[i - 1][j] + cost(delete))`
    - `f[i][j] = min(f[i][j], f[i][j - 1] + cost(insert))`
    - `f[i][j] = min(f[i][j], f[i - 1][j - 1] + cost(copy))`
        - （当`x[i]`==`y[j]`时可使用此转移）
    - `f[i][j] = min(f[i][j], f[i - 2][j - 2] + cost(twiddle))`
        - (当`x[i]`==`y[j+1]`, `x[i + 1]`==`y[j]`时可使用此转移)

    最后，当我们计算出所有的`f[i][j]`后，只需最后利用`cost(kill)`更新`f[m + 1][j]`即可完成。

    假设我们已经维护了最优的`f[i-1][j-1]`、`f[i][j-1]`、`f[i-1][j]`、`f[i-2][j-2]`，那么当前的`f[i][j]`由定义可知，选择了最优的转移方式。因此，当子问题最优时，当前问题也满足最优性质。因此可推出具有最优子结构。

  算法的状态数为$\Theta(nm)$，单次转移复杂度$\Theta(1)$，因此时间空间复杂度显然均为$\Theta(nm)$。

- b. 首先，第一问要求的实际上是最小的编辑距离，也就是在最小化答案，但是第二问要求最大化两个序列的得分，如果想直接使用第一问的转移，那么需要先对所有权值取负。

  首先可以确定的是，twiddle和kill不会被使用，所以可以通过将两种操作的cost置为inf来达成目的。

  接下来，观察第一问的转移方程，可以规定copy对应alignment问题的+1操作，取负后cost(copy)=-1

对于delete和insert，实际上对应出现空格的-2情况，因此取负后cost(delete)和cost(insert)=2

对于replace操作，对应不等且不为空格的-1情况，取负后cost(replace)=1

最后，使用第一问的转移计算完成后，只需要对最小化的答案取负，即为我们第二问所求的最大得分。