

CS 6363.004 Algorithms: Programming Project

Siddhant Sahu

April 1, 2018

1 Finding Maximum Profit

1.1 Recurrence

We define $m(i, w)$ to be the maximum profit that can be generated using the first i items and w units of gold, where i and w are both non-negative integers. Clearly, the jeweler can't make any profit with 0 items. So, $m(0, w) = 0$. When the jeweler is unable to meet the minimum quantity ($k < n_i$) according to the contract, a fine $\min(c_i, f_i \cdot (n_i - k))$ is levied. The recurrence for finding maximum profit is:

$$m(i, w) = \begin{cases} \max(m(i-1, w - k \cdot w_i) + k \cdot p_i - \min(c_i, f_i \cdot (n_i - k))) & \text{if } 0 \leq k < n_i \\ \max(m(i-1, w - k \cdot w_i) + k \cdot p_i) & \text{if } n_i \leq k \leq x_i \end{cases}$$

where k represents the quantity (i.e. number of copies) of item i .

$$0 \leq k \leq \min\left(x_i, \left\lfloor \frac{w}{w_i} \right\rfloor\right)$$

1.2 Proof of correctness

Feasibility Proof by induction on i . We claim that there exists a solution for $m(i, w)$.

Basis Step: Let $i = 0$. We can not make any profit with 0 items. Thus, $m(0, w) = 0$, which is correct. This establishes the basis step.

Inductive Step: For any (i, w) and $0 \leq k \leq \min\left(x_i, \left\lfloor \frac{w}{w_i} \right\rfloor\right)$, we have two cases as mentioned in the recursion. In both cases, we compute $m(i-1, w - k \cdot w_i)$. Since, $i-1 < i$, by induction hypothesis, $m(i-1, w - k \cdot w_i)$ is feasible. This completes the inductive step and therefore, the feasibility proof.

Optimality Let $Opt(i, w)$ be an optimal solution for the problem by using the first i items and w units of gold. We claim that $m(i, w) \geq Opt(i, w)$. Proof by induction on i .

Basis Step: Let $i = 0$. We can not make any profit with 0 items. Thus, $m(0, w) = 0 = Opt(0, w)$, which is correct.

Inductive Step: Consider $i > 0$ and $0 \leq k \leq \min\left(x_i, \left\lfloor \frac{w}{w_i} \right\rfloor\right)$. We have two cases here.

- Case 1: $0 \leq k < n_i$, i.e. the jeweler has to pay a fine. Suppose $Opt(i, w)$ selects k' quantities of item i . Thus, $Opt(i, w) = Opt(i-1, w - k' \cdot w_i) + k' \cdot p_i - \min(c_i, f_i \cdot (n_i - k'))$

$$\begin{aligned} m(i, w) &= \max_{0 \leq k \leq \min\left(x_i, \left\lfloor \frac{w}{w_i} \right\rfloor\right)} (m(i-1, w - k \cdot w_i) + k \cdot p_i - \min(c_i, f_i \cdot (n_i - k))) \\ &\geq m(i-1, w - k' \cdot w_i) + k' \cdot p_i - \min(c_i, f_i \cdot (n_i - k')) \\ &\geq Opt(i-1, w - k' \cdot w_i) + k' \cdot p_i - \min(c_i, f_i \cdot (n_i - k')) \\ &\geq Opt(i, w) \end{aligned}$$

Since $i-1 < i$, $m(i-1, w - k' \cdot w_i) \geq Opt(i-1, w - k' \cdot w_i)$ by induction hypothesis.

- Case 2: $n_i \leq k \leq x_i$. Proof is almost the same as case 1, without the fines.

This completes the proof for optimality.

1.3 Pseudocode

The input to MAX-PROFIT-CALCULATOR is the total units of gold available W and a list of items $items$ with attributes w, p, n, x, f and c . Attribute naming convention is consistent with the problem description.

MAX-PROFIT-CALCULATOR($W, items$)

```

1   $N = items.length$ 
2  let  $m[0..N, 0..W]$  be a new table
3  for  $w = 0$  to  $W$ 
4       $m[0, w] = 0$ 
5  for  $i = 1$  to  $N$ 
6      for  $w = 1$  to  $W$ 
7           $m[i, w] = -\infty$ 
8           $quantity = \min(items[i].x, \lfloor w/items[i].w \rfloor)$ 
9          for  $k = 0$  to  $quantity$ 
10              $q = m[i-1, w - k \cdot items[i].w] + k \cdot items[i].p$ 
11             if  $k < items[i].n$ 
12                  $q = q - \min(items[i].c, items[i].f \cdot (items[i].n - k))$ 
13             if  $q > m[i, w]$ 
14                  $m[i, w] = q$ 
15 return  $m$ 
```

Maximum profit is $m[N, W]$. A quick inspection tells us the running time is $O(kNW)$, where k is the maximum value in $x[1..n]$. If $k = N$, running time is $O(N^2W)$.

2 Counting Number of Solutions

2.1 Recurrence

We define $c(i, w)$ to be the number of solutions that generate the maximum profit $m(i, w)$. With 0 items, the solution set is empty. With one item, there is exactly one solution. Thus, the recurrence is:

$$c(i, w) = \begin{cases} \sum_{k \in \arg\max m(i, w)} c(i-1, w - k \cdot w_i) & \text{if } i \geq 2 \\ 1 & \text{if } i = 1 \\ 0 & \text{if } i = 0 \end{cases}$$

2.2 Proof of Correctness

Feasibility Proof by induction on i . We claim that there exists a solution for $c(i, w)$.

Basis Step: For $i = 0$, there are no solutions. Thus, $c(i, w) = 0$ and for $i = 1$, there is exactly one solution: $c(i, w) = 1$. The base case is established.

Inductive Step: For $i \geq 2$, $c(i, w) = \sum_{k \in \arg\max m(i, w)} c(i-1, w - k \cdot w_i)$. The right-hand side is a smaller subproblem as $i-1 < i$. Therefore, by inductive hypothesis, $c(i, w)$ is feasible.

Optimality This proof assumes that $m(i, w)$ is optimal, which has been proven above. Let $Opt(i, w)$ count the number of solutions to $m(i, w)$. We claim that $c(i, w) = Opt(i, w)$. We will prove this by induction on i .

Basis Step: For $i = 0$, there are no solutions. Thus $c(0, w) = Opt(0, w) = 0$. And, for $i = 1$, there is exactly one solution. $c(1, w) = Opt(1, w) = 1$. This establishes the basis step.

Inductive Step: For $i \geq 2$, suppose there are l distinct values of k , i.e., k_1, \dots, k_l which yield $m(i, w)$. Thus, $Opt(i, w) = \sum_{k=k_1}^{k_l} Opt(i-1, w - k \cdot w_i)$. By induction hypothesis, it follows:

$$c(i, w) = \sum_{k=k_1}^{k_l} c(i-1, w - k \cdot w_i) = \sum_{k=k_1}^{k_l} Opt(i-1, w - k \cdot w_i) = Opt(i, w)$$

Hence, $c(i, w)$ is optimal.

2.3 Pseudocode

The input to COUNT-SOLUTIONS is m , the table returned from MAX-PROFIT-CALCULATOR, W and a list of $items$.

COUNT-SOLUTIONS($m, W, items$)

```
1   $N = items.length$ 
2  let  $c[0..N, 0..W]$  be a new table
3  for  $w = 0$  to  $W$ 
4       $c[0, w] = 0$ 
5  for  $w = 0$  to  $W$ 
6       $c[1, w] = 1$ 
7  for  $i = 2$  to  $N$ 
8      for  $w = 1$  to  $W$ 
9           $quantity = \min(items[i].x, \lfloor w/items[i].w \rfloor)$ 
10         for  $k = 0$  to  $quantity$ 
11              $q = m[i - 1, w - k \cdot items[i].w] + k \cdot items[i].p$ 
12             if  $k < items[i].n$ 
13                  $q = q - \min(items[i].c, items[i].f \cdot (items[i].n - k))$ 
14             if  $q == m[i, w]$ 
15                  $c[i, w] = c[i, w] + c[i - 1, w - k \cdot items[i].w]$ 
16 return  $c$ 
```

Total number of solutions to the problem is $c[N, W]$. The running time is same as MAX-PROFIT-CALCULATOR, i.e. $O(kNW)$ or $O(N^2W)$ if $k = N$.