CSC236H1 SUMMER 2018: PROBLEM SET 2

BY: ERIC KOEHLI

Date: July 4, 2018.

1. Problem 1

Let $T(i, c)$ be the maximum value obtained after considering the first i items with total capacity c.

(a) We define $T(0, c) = 0$ and $T(1, c) = \max \sum_{i=1}^{1} v_i b_i = v_1$, where $b_i \in \{0, 1\}$ and $b_1 = 1$ in this case. That is, we define the maximum value if we consider no items to be 0. If we consider only the first item, then the maximum value is simply the value of that item (i.e the value of a_1).

(b) Assume that we know $T(i, c)$ as defined above. We want to find an equation for $T(i+1, c)$ in terms of $T(i, 0), T(i, 1), \ldots, T(i, c)$. Now, we already know the maximum value of the first i items with capacity c, namely, $T(i, c)$. We now want to consider one more item (a_{i+1}).

Suppose we have the set of items $A = \{(6, 10), (2, 3), (3, 6), (1, 2)\}$ for weight, value pairs. Lets now set $i = 3$, and $c = 7$, (note $i + 1 = 4$), and try to see what happens when we consider an extra item. So we have

$T(3, 0) = 0$, $T(3, 1) = 0$, $T(3, 2) = 3$, $T(3, 3) = 6$, $T(3, 4) = 0$, $T(3, 5) = 9$, $T(3, 6) = 10$, $T(3, 7) = 0$. Then for $i + 1$, we have

$T(4, 0) = 0$, $T(4, 1) = 2$, $T(4, 2) = 3$, $T(4, 3) = 6$, $T(4, 4) = 8$, $T(4, 5) = 9$, $T(4, 6) = 11$, $T(4, 7) = 12$.

Thus we can see that when we consider an extra item $(i + 1)$ that the value of $T(i + 1, c)$ is the maximum of $T(i, c)$ and a new sum of the previous values that includes the value of a_{i+1} and retains the capacity c. Thus, as we can see in this example, $T(i + 1, c)$ is equal to the maximum of the first i items $(T(i, c))$ and the first i items minus the current capacity plus the new value. Therefore we can see the recursive structure:

$$\begin{aligned} T(i+1, c) &= \max\{T(i, c), \max\{\sum_{k=0}^{i+1} v_k b_k\}\} \\ &= \max\{T(i, c), T(i, c - c_i) + v_{i+1}\} \end{aligned}$$

again, where $b_k \in \{0, 1\}$. Therefore, we see the general recursive structure as

$$T(k, c) = \begin{cases} 0 & \text{if } k = 0 \\ v_1 & \text{if } k = 1 \\ \max\{T(k-1, c), T(k-1, c - c_k) + v_k\} & \text{otherwise} \end{cases}$$

To find the maximum value of a subset of A, we would use

$$T(i,c) = \max\{\sum_{k=0}^{i+1} v_k b_k\}$$

where $b_k \in \{0,1\}$, or the recursive formula shown above.

(c) Let C_1 and C_2 be the maximum capacities of the two bags. If we define $T(i,c)$ as the maximum total profit of both bags and $T_1(i,C_1)$, and $T_2(i,C_2)$ as the maximum total profit in bag 1 and bag 2 respectively, then $T(i,c) = T_1(i,C_1) + T_2(i,C_2)$.

2. Problem 2

Let an arbitrary $m \times n$ grid be defined as given in the question.

(a) The goal is to find a recurrence relation for the number of axis-aligned rectangles in an $m \times n$ grid. The first thing to note is that the number of axis-aligned rectangles in an $m \times n$ grid is symmetric. That is, if we define $R(m,n) := k$, where $m, n \in \mathbb{Z}^+$ and $k \in \mathbb{N}$ and k is the number of axis-aligned rectangles, then $R(2,3) = R(3,2)$, and in general, $R(p,q) = R(q,p)$. Therefore with the symmetry property in mind, we can construct our recursive formula on either m or n. We'll choose m.

After drawing several grids and counting the number of rectangles, it can be seen that there are patterns across each row and down each column. What I mean by that is; $R(m,n)$, $R(m,n+1)$, $R(m,n+2)$, $\ldots$, $R(m,n+k)$ and similarly $R(m,n)$, $R(m+1,n)$, $R(m+2,n)$, $\ldots$, $R(m+k,n)$ (for integer k). Then, by brute force, a pattern was revealed down each row and column by a multiple of the triangular numbers.

Thus the recursive formula is

$$R_1(m,n) = \begin{cases} 0 & \text{if } m,n \leqslant 1 \\ 1 & \text{if } m,n \leqslant= 2 \\ R(m-1,n) + (m-1) \cdot \frac{n(n-1)}{2} & \text{otherwise} \end{cases}$$

Note that because of the symmetry, m and n could be swapped in the recursive formula above and it would still work.

(b) To find a closed form, lets first suppose we have a 3×4 grid. Then lets say we choose the point $P(1,1)$. Going across row 1, we can see there are $4 - 1 = 3$ positions that we can expand into in the x direction. Similarly, there are $3 - 1 = 2$ points we can expand into in the y. Therefore, there are a total of $3 \cdot 2 = 6$ combinations. Then we can find the total number of rectangles by finding the total number of combinations at each point. Following this pattern and starting from $P(1,1)$ on an $m \times n$ grid and moving across the rows, we have

$$\begin{aligned}
&[1 \times 1 + 1 \times 2 + \cdots + 1 \times (n-1)]+\\
&[2 \times 1 + 2 \times 2 + \cdots + 2 \times (n-1)]+\\
&\vdots\\
&[(m-1) \times 1 + (m-1) \times 2 + \cdots + (m-1) \times (n-1)]\\
&= 1(1 + 2 + \cdots + n - 1)+\\
&2(1 + 2 + \cdots + n - 1)+\\
&\vdots\\
&(m-1)(1 + 2 + \cdots + n - 1)\\
&= (1 + 2 + \cdots + m - 1)(1 + 2 + \cdots + n - 1)\\
&= m\frac{m-1}{2} \cdot n\frac{n-1}{2}\\
&= \frac{mn(m-1)(n-1)}{4}
\end{aligned}$$

Therefore the closed-form formula for the number of axis-aligned rectangles in an $m \times n$ grid is

$$R_2(m,n) = \frac{mn(m-1)(n-1)}{4}$$

(c) Prove the closed form.

Proof. Let $m, n \in \mathbb{N}$ and define the predicate $P(m,n) : R_1(m,n) = R_2(m,n)$.

Base Case:

Let $m = 1$ or $n = 1$. Then $R_1(1,n) = R_1(m,1) = 0$. Also,

$$R_2(1,n) = \frac{1 \cdot n(1-1)(n-1)}{4} = 0$$

Similarly for $R_2(m,1) = 0$.

Let $m = n = 2$. Then $R_1(2,2) = 1$ and

$$R_2(2,2) = \frac{2 \cdot 2 \cdot (2-1)(2-1)}{4} = \frac{4}{4} = 1$$

Inductive Step:

Let $r, s \in \mathbb{N}$ and assume $P(r,s)$ is true. We want to show that $P(r+1,s) : R_1(r+1,s) = R_2(r+1,s)$ is also true. Then from the LHS, we have

$$\begin{aligned}
R_1(r+1,s) &= R_1(r,s) + \frac{rs(s-1)}{2} \\
\text{(By I.H.)} \quad &= \frac{rs(r-1)(s-1)}{4} + \frac{2rs(s-1)}{4} \\
&= \frac{rs(r-1)(s-1) + 2rs(s-1)}{4} \\
\text{(Factor } rs(s-1)) \quad &= \frac{rs(s-1)[r-1+2]}{4} \\
&= \frac{rs(r+1)(s-1)}{4} \\
&= R_2(r+1,s)
\end{aligned}$$

Thus $P(r+1,s)$ follows from $P(r,s)$ and this completes the induction step. Having shown steps 1 and 2, we can conclude by the principle of mathematical induction that $P(m,n)$ is true for all $m, n \in \mathbb{N}$.

□

3. Problem 3

(a) Design a brute-force algorithm for minimum.

```python
def minimum(A: List[int]) -> int:
    smallest = A[0]
        for i in range(len(A)):
            if A[i] < smallest:
                smallest = A[i]
    return smallest
```

The runtime of this algorithm is $\Theta(n)$, where `n = len(A)` because the `for` loop iterates n steps. The assignment statement, `if` check , and `return` take constant time, so $RT_{minimum} = \Theta(n)$

(b) Devise a divide-and-conquer algorithm for minimum.

```python
def find_min_conquered(lst: List[int]) -> int:
    n = len(lst)
    if n < 2:
        # then there is 1 item in the list.
        return lst[0]
    elif lst[0] < lst[-1]:
        # then the list is not rotated
        return lst[0]
    elif lst[1] < lst[0]:    # n == 2
        # there are only 2 elements
        return lst[1]
    else:
        # divide the list in half. Note: n is at least 3.
        m = n // 2

        if lst[m - 1] < lst[-1]:
            # then the smallest must be before m
            return find_min_conquered(lst[:m])
        else:
            return find_min_conquered(lst[m:])
```

The runtime of find_min_conquered is $\Theta(\log n)$ as will be shown.

(c) Let `n = len(lst)` be the length of the input list and let $T(n)$ be the worst-case runtime of find_min_conquered. If $n < 3$, the non-recursive part takes constant time.

If $n \geqslant 3$, then the one of two recursive calls are made each on a list of size $\frac{n}{2}$. Therefore, we get

$$T_1(n) = \begin{cases} d & \text{if } n < 3 \\ T(\frac{n}{2}) + c & \text{otherwise} \end{cases}$$

(d) Find the closed form and prove that it is correct.

Substitute $n = 2^m$ into $T_1(n)$. Then we get

$$\begin{aligned} T(2^0) &= T(1) = d \\ T(2^1) &= T(2) = d \\ T(2^2) &= T(4) = T(2) + c = d + c \\ T(2^3) &= T(8) = T(4) + c = d + 2c \\ T(2^4) &= T(16) = T(8) + c = d + 3c \\ &\vdots \\ T(2^m) &= T(2^{m-1}) + c = d + (m-1)c \end{aligned}$$

Then $m = \log_2 n$ and substituting back for n gives

$$T_2(n) = d + c(\log_2(n) - 1)$$

Proof of closed form.

Proof. Let $n \in \mathbb{N}$ and define $P(n) : T_1(n) = T_2(n)$.

Base Case:

Let $n = 2$. Then $T_1(n) = d$. Also,

$$\begin{aligned} T_2(n) &= d + c(\log_2 2 - 1) \\ &= d + c(1 - 1) \\ &= d \end{aligned}$$

Thus $P(2)$ is true.

Inductive Step:

Let $k \in \mathbb{N}$ and assume $P(1), P(2), P(3), \ldots, P(k)$ is true. We want to show $P(k+1)$. Note that $2 \leqslant \frac{k+1}{2} \leqslant k$. Starting from $T_1(n)$, we have

$$\begin{aligned} T_1(k+1) &= T_1(\frac{k+1}{2}) + c \\ \text{(By I.H.)} \qquad &= d + c \cdot (\log_2 (\frac{k+1}{2}) - 1) + c \\ &= d + c \cdot (\log_2 (k+1) - \log_2 (2) - 1) + c \\ &= d + c \cdot (\log_2 (k+1) - 1 - 1) + c \\ &= d + c \cdot \log_2 (k+1) - 2c + c \\ &= d + c \cdot \log_2 (k+1) - c \\ &= d + c \cdot [\log_2 (k+1) - 1] \\ &= T_2(k+1) \end{aligned}$$

□

4. Problem 4

(a) We have $a = 4$, $b = 2$, $k = 2$. So we get $\log_b a = \log_2 4 = 2$. In this case $\log_b a = k$, so by the Master Theorem, we have $T(n) = O(n^2 \log n)$.

(b) We have $\log_4 2 = 1/2$ and $k = 0$. Since $1/2 > 0$, then by the Master Theorem, we get $T(n) = O(\sqrt{n})$.

(c) Here we have $\log_3 9 = 2$ and $k = 3$, so by the Master Theorem we get $T(n) = O(n^3 \log n)$.