Analysis of Algorithms - Quiz II (Solutions)

L. Kovalchick
LCSEE,
West Virginia University,
Morgantown, WV
{lynn@csee.wvu.edu}

1 Problems

1. Sort the sequence $\mathcal{S} = \{(3,3),(1,5),(2,5),(1,2),(2,3),(1,7),(3,2),(2,2)\}$, in increasing lexicographic order, using Radix sort, showing all the intermediate steps. Note that in lexicographic ordering $(x_1, y_1) < (x_2, y_2),\ if\ x_1 < x_2,\ or\ x_1 = x_2\ and\ y_1 < y_2$

   Radix sort is a Stable Sort procedure (2 points).

   Solution: From page 242 and 243 of [GT02], we know that the Radix Sort algorithm sorts a sequence of pairs as defined by $\mathcal{S}$, by applying a stable bucket sort on the sequence twice; first using the second component and then using the first component.

   Performing a bucket sort on the second component of $\mathcal{S}$ results in the sequence $\mathcal{S}'$

   $$\mathcal{S}' = \{(1,2),(3,2),(2,2),(3,3),(2,3),(1,5),(2,5),(1,7)\}$$

   Likewise, performing a bucket sort on the first component of $\mathcal{S}'$ results in the sequence $\mathcal{S}''$

   $$S'' = \{(1,2),(1,5),(1,7),(2,2),(2,3),(2,5),(3,2),(3,3)\}$$

   It follows that $\mathcal{S}''$ is our sorted order output from Radix Sort. □

2. The coin changing problem is concerned with making change for n cents (n integral) using the fewest number of coins, where the coins are *quarters (25 cents), dimes (10 cents), nickels (5 cents)* and *pennies (1 cent)*. For instance, we can make change for 10 cents using either 10 pennies or 2 nickels or 1 dime; clearly using 1 dime is the optimal solution, since it uses the fewest number of coins. Describe a greedy strategy for the coin changing problem. Argue that your strategy always derives the optimal solution for arbitrary n, i.e., it changes n into the *fewest number of coins*? (5 points)

   Solution: Our greedy strategy consists of changing as much of n into quarters as possible, followed by changing as much of the rest into dimes as possible and so on.

   Let x_{25} indicate the number of quarters created by the greedy strategy. x_{10}, x_5 and x_1 are defined similarly. Note that the greedy strategy gives rise to Algorithm (1.1).

   Let us say that the solution returned by Algorithm (1.1), i.e., $\vec{\mathbf{x}}$ is not optimal for some n and that there exists an algorithm OPT that returns the optimal solution $\vec{\mathbf{y}} = [y_{25}\ \ y_{10}\ \ y_5\ \ y_1]^T$, for the same n.

Function COIN-CHANGER(n)

For positive numbers a and b, we define $\lfloor \frac{a}{b} \rfloor$ as the quotient when b divides a and $a \ mod \ b$ as the remainder that results when b divides a.

$x_{25} = \lfloor \frac{n}{25} \rfloor$; $r_{25} = n \ mod \ 25$

$x_{10} = \lfloor \frac{r_{25}}{10} \rfloor$; $r_{10} = r_{25} \ mod \ 10$

$x_5 = \lfloor \frac{r_{10}}{5} \rfloor$; $r_5 = r_{10} \ mod \ 5$

$x_1 = r_5$

Set $\vec{\mathbf{x}} = [x_{25} \ \ x_{10} \ \ x_5 \ \ x_1]^T$

return($\vec{\mathbf{x}}$)

Algorithm 1.1: Greedy Strategy for coin changing

We first show that if $y_{25} \neq x_{25}$, then $\vec{\mathbf{y}}$ cannot be optimal. Observe that, as per the greediness of Algorithm (1.1), $y_{25} \leq x_{25}$, since Algorithm (1.1) assigns *the maximum possible number of quarters* to x_{25}. It follows that if $y_{25} \neq x_{25}$, then we must have $y_{25} < x_{25}$. Let $c_{25} = (x_{25} - y_{25})$ and $v_{25} = 25 \cdot c_{25}$. This means that change valued at v_{25} cents has been distributed among the dimes, nickels and pennies of OPT's solution. Clearly, the optimal way to convert the amount v_{25} into change is by using precisely c_{25} quarters since converting 25 cents into change requires precisely 1 quarter but at least 2 coins, using any combination of dimes, nickels and pennies. Any other method will result in a number of coins strictly greater than c_{25}. It follows that we can take an amount v_{25} from (y_{10}, y_5, y_1) and convert it into quarters; doing so *strictly decreases* the number of coins in OPT's solution, thereby contradicting the optimality of $\vec{\mathbf{y}}$.

Thus, if $\vec{\mathbf{y}}$ is to be optimal, then we must have $y_{25} = x_{25}$. Given that $y_{25} = x_{25}$, we are now faced with the problem of showing that $\vec{\mathbf{x'}} = [x_{10} \ \ x_5 \ \ x_1]^T$ is optimal for the amount $n - (25 \cdot x_{25})$. We can repeat the above argument to show that either $y_{10} = x_{10}$ or $\vec{\mathbf{y}}$ is not optimal. If $y_{10} = x_{10}$, then the problem is reduced to showing that $\vec{\mathbf{x''}} = [x_5 \ \ x_1]^T$ is optimal for the amount $n - (25 \cdot x_{25} + 10 \cdot x_{10})$. An identical argument shows that if $y_5 \neq x_5$, then $\vec{\mathbf{y}}$ cannot be optimal. Finally, since we must have $x_{25} = y_{25}$, $x_{10} = y_{10}$ and $x_5 = y_5$, in order for $\vec{\mathbf{y}}$ to be optimal, it is clear that $y_1 = x_1 = n - (25 \cdot x_{25} + 10 \cdot x_{10} + 5 \cdot x_5)$. In other words, if $\vec{\mathbf{y}} \neq \vec{\mathbf{x}}$, then $\vec{\mathbf{y}}$ is not optimal, i.e., $\vec{\mathbf{x}}$ is the optimal solution for n.

It is very important to note that the correctness of the above proof is contingent on comparing the pairs (x_{25}, y_{25}) *first*, followed by (x_{10}, y_{10}), (x_5, y_5) and (x_1, y_1); altering this strict order destroys the correctness of the proof. □

3. Characterize the following recurrence (3 points):

$$\begin{aligned} T(n) &= 1, \ if \ n = 1 \\ &= 2 \cdot T(\frac{n}{2}) + \log n, \ n > 1 \end{aligned}$$

Hint: Use Master's Theorem

Solution: As per the specification on Page 268 of [GT02], we have: $a = 2$, $b = 2$, $d = 1$, $c = 1$ and $f(n) = \log n$. It is clear that the prerequisites of the Master's Theorem, viz., $a > 0$, $c > 0$, $b > 1$ and $f(n)$ being a positive function are satisfied. Note that $n^{\log_b a} = n$ and that $f(n) = O(n^{\log_b a - \epsilon})$, for any $0 < \epsilon < 1$. So, we can apply Case 1 of the Master's Theorem to obtain $T(n) = \Theta(n)$. (What happens if $c = 0$?) □

References

[GT02] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. John Wiley & Sons, 2002.