Algorithms Fall 2016 Homework 3 Solution

October 4, 2016

Problem 1: (40 points)

Let $X = \{x_1, x_2, ..., x_n\}$ be the set of points. We can solve this problem by using the greedy algorithm as follows:

1. Sort the numbers in increasing order.
2. Let x_{min} be the smallest number in the set.
3. Choose $[x_{min}, x_{min} + 1]$ to be a closed interval for the optimal set.
4. Remove all numbers in the range of $[x_{min}, x_{min} + 1]$ from the set.
5. Repeat steps 2 - 4 until the set is empty.

Correctness Justification
First we will prove the correctness of the greed choice. We will prove that one of the optimal solutions has the first unit-interval starting at the smallest number x_{min} in the sorted set. The smallest number x_{min} should be covered by at least one of the unit-intervals, let's say $[y, y + 1]$, starts before or at x_{min}. Since there are no numbers less than x_{min}, we can move the interval from $[y, y + 1]$ to $[x_{min}, x_{min} + 1]$ without increasing the number of intervals. Therefore our claim is correct.
Then we will prove the correctness of the optimal substructure. Suppose we have an optimal set of intervals $D = \{d_1, ..., d_m\}$ that contains m intervals. We can remove d_i from the set D and all the numbers that are covered by d_i from X. Let X' be the new set of points. Then we will prove that the set $D - \{d_i\}$ is an optimal set for the set X'. Suppose there exists another optimal solution D' that has fewer intervals but covers all the numbers in X'. Then we add d_i to D', $D' \cup \{d_i\}$ will cover all the points in X with fewer intervals than D, this contradicts with that D is an optimal set.

Problem 2: (60 points)

First, let's define the Coin Changing problem as follows: Given a set of coin denominations which is $C = \{1, 5, 10, 25\}$, find the optimal way to make change for n cents with the fewest number of coins.
The greedy algorithm is as follows:

1. Pick the largest valid denomination c_i from C.
2. Let $n = n - c_i$.
3. Repeat steps $1 - 2$ until n equals to 0.

Correctness Justification
To prove the correctness of our algorithm, we give the following lemmas:

Lemma-1 There are at most 4 pennies in any optimal solution since 5 pennies can be replaced by 1 nickel.

Lemma-2 There are at most 1 nickel in any optimal solution since 2 nickels can be replaced by 1 dime.

Lemma-3 There are at most 2 dimes in any optimal solution since 3 dimes can be replaced by 1 quarter and 1 nickel.

To prove the correctness of our greedy algorithm, we need prove that the largest valid denomination must exist in the optimal solution of making changes for given amount. Suppose we are making changes for n cents and y is the largest valid denomination.

1. If $0 < n < 5$, the only valid denomination is 1 and $y = 1$ is in any solution.

2. If $5 <= n < 10$, the valid denominations are $\{1, 5\}$ and $y = 5$. Assuming that y is not in any optimal solutions, so the optimal solution consists of only 1's and there will be n pennies in the optimal solutions, this contradict with Lemma-1 since $n > 4$.

3. If $10 <= n < 25$, the valid denominations are $\{1, 5, 10\}$ and $y = 10$. Assuming that y is not in any optimal solutions, so the optimal solution consists of only 1's and 5's. This contradicts at least one of Lemma-1 and Lemma-2.

4. If $25 <= n$, the valid denominations are $\{1, 5, 10, 25\}$ and $y = 25$. Assuming that y is not in any optimal solutions, so the optimal solution consists of only 1's, 5's and 10's. We can show the contradiction by cases:

   (a) If $n >= 30$, by lemma-2 and lemma-3, we have at most 1 nickel and 2 dimes in the optimal solution. That means we need at least $n - 25$ pennies in the optimal solution, this contradicts with Lemma-1 since $n - 25 > 4$.

   (b) If $25 <= n < 30$, we can easily prove that there must be 2 dimes and 1 nickel in the solution according to Lemma-1 and Lemma-2. This contradicts with that the solution is optimal since we can replace 2 dimes and 1 nickel by 1 quarter.

The pseudo-code is as follows:

```
COIN-CHANGING(n)
Let C be an empty set
while n >= 25
        C.add("quarter")
        n = n - 25
while n >= 10
        C.add("dime")
        n = n - 10
if n >= 5
        C.add("nickel")
        n = n - 5
while n > 0
        C.add("penny")
        n = n - 1
return C
```