ECS 122A Homework 1

Hardy Jones
999397426
Professor Gysel
Fall 2014

1. (a) The most number of pennies an optimal solution can have is 9. The most number of dimes an optimal solution can have is 4. We can see this by looking at some cases:

   - $0 \leq n < 10$: We have no choice but to use pennies, as any other denomination will be more than the change required. This has a maximum of 9 coins.
   - $10 \leq n < 20$: We can use 1 dime to get down to the first case (rather than all pennies), and the rest returned in pennies. This has a maximum of 1 dime + 9 pennies = 10 coins.
   - $20 \leq n < 25$: We can use 1 dime to get down to the previous case, then continue following the previous reasoning. This is less overall coins than using pennies to get to the previous case. This has a maximum of 2 dimes + 4 pennies = 6 coins.
   - $25 \leq n < 30$: We can use 1 quarter to get to the case where $0 \leq n < 5$, then use pennies from there. This is less overall coins than using pennies and dimes to get to any previous cases. This has a maximum of 1 quarter + 4 pennies = 5 coins.
   - $30 \leq n < 35$: We can use 1 dime to get down to the case where $20 \leq n < 25$, then continue with the reasoning from there. This is less overall coins than using a quarter to get the the previous case and pennies. This has a maximum of 3 dimes + 4 pennies = 7 coins.
   - $35 \leq n < 40$: We can use 1 quarter to get to the case where $10 \leq n < 20$, then continue with the reasoning from there. This is less overall coins than using dimes and pennies alone. This has a maximum of 1 quarter + 1 dime + 9 pennies = 11 coins.
   - $40 \leq n < 45$: We can use 1 dime to get to the case where $30 \leq n < 35$, then continue with the reasoning from there. This is less overall coins than using a quarter to get to another case. This has a maximum of 4 dimes + 4 pennies = 8 coins.
   - $45 \leq n < 50$: We can use 1 quarter to get to the case where $20 \leq n < 25$, then continue with the reasoning from there. This is less overall coins than using a dime to get to another case. This has a maximum of 1 quarter + 2 dimes + 4 pennies = 7 coins.
   - $50 \leq n$: We can use 2 quarters to get down to one of the above cases, then continue with the reasoning from there. This is less overall coins than using dimes or pennies alone to get down to any of the above cases.

(b) Assume that it is not the case that using quarters initially when $n \geq 50$ provides the optimal solution.

This means we can provide the optimal solution by first using dimes and/or pennies until the change needed is less than 50¢. We can see that once the change needed is below 50¢, we can use the cases in part a to continue reasoning. So, we need 5 dimes for any $n \geq 50$ to get it down to the cases used above. However, we know that we can use just 2 quarters to get any $n \geq 50$ down to the cases above. Since 2 quarters is less overall coins than 5 dimes, we have found a solution better than what we assumed to be the optimal. Similar reasoning works with using pennies, however, we would need 50 pennies rather than 5 dimes and this is clearly non-optimal.

Thus, our assumption was wrong.

Therefore, the optimal solution uses quarters until the amount is less than 50¢.

(c) One algorithm for this is to do a direct translation of the cases above.

We first check to see if $n \geq 50$.

If so we know we need at least 2 quarters. We also subtract 50 from n to get a new amount of change needed, say n_1. If $n < 50$ we can still say the new amount of change needed is n_1.

Then, we take n_1 and find which case it fits.

Since we have already enumerated the number and denomination of quarters and dimes for each case, we subtract that amount from whichever case n_1 falls under. E.g. if $n_1 = 36$¢, when we would subtract the total of 1 quarter and 1 dime (or 35¢) from n_1. We call this new amount of change needed n_2. We also add the respective number and denomination of quarters and dimes to the coins needed.

Finally, we return n_2 pennies.

This algorithm is $O(1)$ because it only performs standard arithmetic and comparisons, each of which have $O(1)$ cost. The algorithm is not dependent on the size of n.

2. (a) An algorithm for finding the closest points on a line can be described by:

Given some array of points, first sort the array of points. Then, starting from one end of the array, we compare pairwise then distance between points. Assuming our distance function is a metric, we keep track of the minimum distance, and which points provided this distance. If a pair provides a smaller distance, we update the minimum with this distance and the pair of points that provided it. When we have traversed the entire array, we return the pair of points that provide the minimum distance.

This algorithm works because we are only measuring distance in one dimension. Since we sort the array, we know that each point is next to its closest neighbor(s). So, we don't need to check the distance between any points except the possible one before and after a point.

We can verify this by assuming that we have some points a, b, c in sorted order. Now, if $d(a, c) < d(a, b)$ (for some distance function d), then the points would not

be in sorted order, as the inequality implies that b is either greater or less than c. So, the points must not have been in sorted order.

This algorithm is $O(n\lg n)$ because the sort can be $O(n\lg n)$, and the pairwise comparison is then $O(n)$. Since $O(n) < O(n\lg n)$, the overall cost is $O(n\lg n)$.

3. (a) We want to show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$.

Using the substitution method we need to perform two steps:

i. Guess that the solution is $T(n) = O(n^2)$

ii. Show by induction that $T(n) \leq cn^2$ for some $c > 0$.

Proof. We begin with the inductive step. Assume this holds for all $m < n$ and choose $m = n - 1$.
Now we have $T(n-1) \leq c(n-1)^2$.
Substitute the recurrence:

$$\begin{aligned} T(n) &\leq c(n-1)^2 + n \\ &= c(n^2 - 2n + 1) + n \\ &= cn^2 - 2cn + c + n \\ &\leq cn^2 \end{aligned}$$

And we still need to show the base case. Let's try $n = 1$.

$$\begin{aligned} T(1) &= T(1-1) + 1 \\ &= T(0) + 1 \end{aligned}$$

We want $T(1) \leq c(1)^2 = c$ for some c we choose.
Since we have no explicit value for $T(0)$, we choose $c \geq T(0) + 1$.
Thus we have shown that the solution holds by induction. □