Problem Set 1—Due Friday, April 13th, 3:15 PM

Homework Info Homeworks are due by 3:15 PM on the due date. They are to be turned in at the marked box in Kemper Hall, room 2131. No late homeworks will be accepted.

Much of what one learns in this course comes from trying to solve the homework problems, so work hard on them. Doing a conscientious job on the homeworks is the best preparation for the exams. We hope that you will ultimately solve the majority of the problems, but don't be surprised if some of them stump you; some of the problems may be quite challenging.

Your solutions should be terse, correct, and legible. Understandability of the solution is as necessary as correctness. Expect to lose points if you provide a "correct" solution with a not-so-good writeup. As with an English paper, you can't expect to turn in a first draft: it takes refinement to describe something well. Typeset solutions are always appreciated.

If you can't solve a problem, briefly indicate what you've tried and where the difficulty lies. Don't try to pull one over on us.

If you think a problem was missgraded, please see the grader first. (The grader will hold periodic office hours; times will be announced later.)

(25) Problem 1. Coin Changing variations

When we have quarters dimes and pennies we described an algorithm (using memeoization) that uses $O(n)$ time and space to compute the minimum number of coins needed to give change (program uscmem.c on the class web page). We now explore a faster solution.

(a) If we have quarters, dimes and pennies we showed that the greedy algorithm (use quarters till below 25 cents) does not always use the minimum number of coins. Argue that if we want to make change for an amount $n \geq 50$ it is always optimal to use quarters until the remaining amount falls below 50 cents. Hint: i) to show this consider (for contradiction) change for $n > 50$ that uses q quarters where $(n - 25q) \geq 50$; ii) also consider the remaining change, r which is an amount $(n - 25q) = r$ and has no quarters in it. Argue that we can always improve this change to use fewer coins, and thus we always use enough quarters to be sure that r, the change that has no quarters, is less than 50.

(b) For amounts between 25 and 49 cents it *may* still be correct to use another quarter, but not always. What is the largest value that uses no quarters to give change optimally? Justify your answer.

(c) Use the result from part (a) to describe a constant-time algorithm that, given an input n, returns Q, D and P the number of each coin to give change with the fewest coins. You may assume (slightly unrealistically) that you can do standard arithmetic operations on any size numbers in $O(1)$ time. Note: you may use the result of part (a) even if you didn't solve part (a).

(25) Problem 2. Problem 4.1-5 in the text (page 75). Note that you should be able to solve this using a fairly simple short piece of psuedo code (along the lines of the simple, non-recursive, algorithm we gave for coin changing). Briefly justify the correctness and run time of your solution.

(20) Problem 3.

We consider a variation of the closest point problem where distance is measured using what is called the *manhattan metric*. For two points $(x_1, y_1), (x_2, y_2)$ their distance is $|x_1 - x_2| + |y_1 - y_2|$. This is the distance if you could only move parallel to the x or y axis.

(a) Suppose that no two points are closer than d units from each other under the manhattan metric. Give a bound on the maximum number of points which could exist in a rectangle which is d units wide and $2d$ units high (and which is parallel to the x axis).

(b) Can we use the same closest point algorithm we described in class to solve the closest point problem using the manhattan metric (assuming we now compute distances for this new metric)? Justify your answer.

(20) Problem 4. We look at analyzing a variant of the coin changing algorithms discussed in class (and posted on the web page). We consider the setting with pennies, dimes, and quarters.

(a) The following recurrence describes the minimum number of coins to make change.

$$\begin{aligned} C(n) &= 1 \quad n = 1,\ n = 10,\ n = 25 \\ &= infinity \quad n < 1 \\ &= 1 + MIN(C(n-1),\ C(n-10),\ C(n-25))) \end{aligned}$$

When using the *non-memoized* recursive algorithm to make change (implied by the above recurrence relation), let $W(n)$ be the number of **calls** to the, recursive, function NCoins used to **compute** $C(n)$ (in the program usc.c on the web page). Write a recurrence relation for $W(n)$. Note: you are **not** being asked to solve $W(n)$.

(b)

Let $M(n)$ denote the number of **calls** used to compute $T(n)$ using the *memoized* version of the algorithm. Argue that $M(n) = \Theta(n)$. Hint: do **not** use a recurrence relation.