MA/CS 109 Third CS Homework

Discussion Exercises

You should form groups to analyze these problems during your assigned discussion periods, and hand in the solutions together with the homework problems below. You may collaborate on the discussion exercises problems with your group and hand in common solutions, but you MUST do the “homework exercises” by yourself.

Problem 1

For this problem, I would like you to write an algorithm as detailed as you can make it (within reason) which specifies how to make change for an amount of money less than 1 dollar. Specifically:

The Change Problem

**Input**: A number $N$ between 1 and 99 (inclusive), representing how many cents change is due.

**Output**: Four numbers $(Q,D,N,P)$ representing $Q =$ how many quarters, $D =$ how many dimes, $N =$ how many nickels, and $P =$ how many pennies, such that 1. The sum $D + D + N + P$ is as small as possible (give as few coins back as possible); and 2. $N = 25Q + 10D + 5N + P$ (the change correctly amounts to what is due).

The Partition Problem is as follows:

**Input**: A set $S$ of integers of size $N \geq 2$.

**Output**: A subset $S'$ of $S$ such that the sum of the numbers in $S'$ is equal to the sum of the numbers in $S – S'$; or report that “no such subset exists.”

Example 1: $S = \{1, 8, 5, 2\}$  Output: $S' = \{8\}$,  $S – S' = \{1, 5, 2\}$

Example 2: $S = \{1, 5, 7, 3, 2\}$  Output: $S' = \{1, 5, 3\}$,  $S – S' = \{7, 2\}$

Example 3: $S = \{2, 4, 3, 6\}$  Output: No such subset exists.

Here is a (very dumb) algorithm for this problem, based on a simple encoding of subsets using binary numbers: if $S = \{s_1, s_2, ..., s_n\}$ with $N$ elements, then we can use
N bits \(b_1, b_2, \ldots, b_n\) to indicate whether each individual element is in the subset \(S'\) or not: make \(b_i = 1\) if and only if \(s_i\) is in \(S'\). For example, if \(S = \{1, 2, 3\}\), then the subset \(\{2, 3\}\) would be represented by 011, and the whole set by 111. Essentially, we can think of 0 as “No” and 1 as “Yes” in answering the following question for each subset:

“Which of the integers 1, 2, 3 should I put in my subset?”

![Decision tree](image)

Algorithm for the Partition Problem

**Input:** A set of integers \(S = \{s_1, s_2, \ldots, s_n\}\) with \(N\) elements.

Step One: Let \(B = 0000\ldots01\) (an \(N\)-bit number equal to 1);

Step Two: Let \(S' = \) the subset of \(S\) indicated by \(B\), and check if \(\text{sum}(S') = \text{sum}(S - S')\); if it is, print out the set \(S'\); otherwise go to Step Three;

Step Three: Add 1 to the binary number \(B\); if the result is all 1s (the largest number representable in \(N\) bits), stop and print out “No such set exists”; otherwise, go back to Step Two with this new value of \(B\).

**Problem 2**

Now answer the following questions:

(a) Run this algorithm for \(S = \{2, 3, 4\}\) and show all work (the subsets, the sums, and the result);

(b) We did not check the empty set (\(B = \) all 0s) and the set \(S\) itself (\(B = \) all 1s) as solutions; why is this not necessary?
(c) If you have an input set with N elements, how many subsets will you have to check in the worst case?

(d) Suppose you could check 1,000,000 subsets a second on a computer and you decide to try to solve a problem with 100 numbers. How long would it take in the worst case? [Hint: express your answer very approximately using the “Short Scale” from http://en.wikipedia.org/wiki/Names_of_large_numbers.]

(d) (Optional: Just think about it) This is an exceedingly dumb algorithm: it just enumerates every possible solution, no matter how silly, and checks each one. Can you think of a better way to do it?

Homework Exercises

These should be completed by yourself.

Problem 3 (ECC - Error-Correcting Codes)

This problem is based on the lecture slides posted over the weekend on the topic of Friday’s lecture (www.cs.bu.edu/fac/snyder/cs109/).

(a) Encode the following four-bit message using the ECC: 1101.

(b) Suppose you receive the following (corrupted) message which has been encoding using the ECC, what is the corrected message? 1010101.

(c) Supposing you wanted to send a message consisting of 75 bits, using a version of the ECC, but with additional parity bits. How many parity bits would you need for 75 message bits?

Problem 4

The ECC we examined can correct any one bit error, and hence can also detect one-bit errors. But what happens with two bit errors? In this problem we will explore how the ECC deals with this.

(a) Try several examples of two bit errors using the example from lecture: 0011100 (correct), e.g., decode 1111100, 0000100, 1011110, 0101100, all with two bit errors. What happens? Is it possible to detect two bit errors or does this get confused with something else? Describe what you think is happening.

(b) Suppose we use the ECC method and then the error-detecting code, that is, we encode a 4-bit string using the ECC method (for a total of 7 bits), and then we add one more parity bit, which makes the parity for all 8 bits even. Try several 1-bit
errors, and several 2-bit errors, and describe how this solves the problem: the resulting code can correct all 1-bit errors, and detect (but not correct) all 2 bits errors. Explain what conditions would indicate a 1-bit error (and its location) and what conditions would indicate a 2-bit error.

**Problem 5**

Consider the following simple algorithm for sorting a list of N numbers into ascending order:

**Step One:** Scan from the left for the first adjacent pair of numbers which are out of order (the left one is larger than the right one); if no such pair exists, then stop, else go to step two.

**Step Two:** Swap the two numbers in the list and go back to step one.

Example (out of order pairs found are underlined, after swapping shown in red):

42715 -> 24715 -> 24175 -> 21475 -> 12475 -> 12357 done!

Suppose our measure of the complexity of this algorithm is the number of times we must swap two numbers.

(a) Sort the list 42917, show every step, and count the number of swaps.

(b) What is minimum number of swaps possible for any input list of size N? Give an example of such a “best case” list with 5 elements.

(c) What kind of input list would cause the maximum number of swaps? Play around with a few examples of 2, 3, 4, and 5-element lists, and give “worst-case” examples (they are very similar) and the number of swaps required for each of these four lengths.