CS 330 – Spring 2016, Assignment 5

Assignments due in the drop box by 7PM, Thursday, March 31

Question 1 (10 pts)**.** *Chapter 6, Exercise 12, on p. 323.*

Question 2 (10 pts)**.** *Chapter 6, Exercise 15, on p. 325.*

Question 3 (10 pts)**.** *Exercises 4 and 5 of Chapter 7, on p. 416.*

Question 4 (20 pts)**.** *In this programming question, we will be considering the* NUMBER PARTITION *problem. As input, the number partition problem takes a sequence $A = (a_1, a_2, \ldots a_n)$ of non-negative integers, and outputs a sequence $S = (s_1, s_2, \ldots s_n)$ of signs $s_i \in \{-1, +1\}$ such that the* residue

$$u = \left| \sum_{i=1}^{n} s_i a_i \right|$$

is minimized. Another way to view the problem is to partition the sequence of numbers given by A into two disjoint subsequences A_1 and A_2 whose sums are as equal as possible. The absolute value of the difference of the sums is the residue.

(a.) Although NUMBER PARTITION *is NP-complete, it can be solved in pseudopolynomial time. In particular, if the sum of the sequence A is B, the sequence can be represented in $n \log B$ bits, and a truly polynomial time algorithm would run in time polynomial in n and $\log B$. Instead you should find and implement a dynamic programming algorithm that runs in time polynomial in n and B. Justify the running time.*

(b.) A deterministic heuristic for NUMBER PARTITION *is the Karmarkar-Karp algorithm, or the KK algorithm. The KK algorithm uses* differencing*: repeatedly take the two largest elements from A, call them a_i and a_j, and replace the larger by $|a_i - a_j|$ while replacing the smaller by zero, until there is only one non-zero element left. That final non-zero element corresponds to an achievable residue. (Each difference operation corresponds to putting a_i and a_j in different sets: if they were weight 75 and 71, then it is as if we had just one element of weight 4 around.) For example, if A is initially $(10, 8, 7, 6, 5)$, then the KK algorithm proceeds as follows:*

$$(10, 8, 7, 6, 5) \to (2, 0, 7, 6, 5) \to (2, 0, 1, 0, 5) \to (0, 0, 1, 0, 3) \to (0, 0, 0, 0, 2).$$

Argue that there is an easily computable partition that achieves the residue computed by the KK algorithm. Then implement the KK algorithm (you need not compute the partition, just the residue), and justify your algorithm's running time. You should strive to obtain an O($n \log n$) implementation.

(c.) Run several experiments for each of your implementations head-to-head on sets of 100 integers chosen uniformly starting from the range $[1, 1000]$ and working up to the range $[1, 10^{12}]$ by powers of 10. Make sure to use 64-bit integers, and make sure your random number generator is working correctly on ranges this large. Point out where your pseudopolynomial-time algorithm runs too slowly to be useful any more, and describe any differences you see in results obtained by the KK heuristic versus the exact algorithm.

You may work in teams of up to three people for this question. Teams of two are probably ideal. Each team must submit one hard copy of their code along with one joint writeup for this question. We will likely reuse some of your code in a final programming assignment.

Question 5 (Extra Credit: 10 pts for excellent writeups)**.** *Chapter 6, Exercise 19, on p. 329.*