Martin Ziegler
Chansu Park, Donghyun Lim, Dongseong Seon

Issued on September 29, 2017
Due: 14:40, October 13, 2017

CS204
Fall 2017, Assignment #4

Problem 1. *5 + 2 pts*

Fill out corresponding blocks. Then give your opinion: What is more important between CPU speed improvement and algorithm time complexity improvement?

	Running time(sec)		
input size(n)	$10^{-9}2^n$	$10^{-3}n^3$	$10n\log_2 n$
2			
8			
16			
32			
64			

Problem 2. *2 + 2 + 2 pts*

In Chapter 3.1 Example 6, there is Algorithm 6: Greedy change-making algorithm. It tries to minimize the number of coins to return the change in greedy sense: given coins for each type $c_1 < \cdots < c_n$, use one coin c_i to change k satisfying $c_i \leq k < c_{i+1}$, and re-run the process with change $k - c_i$ until changes becomes 0.

a) Suppose we have infinitely many coins for each type: 100 cents, 50 cents, 10 cents, 5 cents, and 1 cent. Using that algorithm, describe how to change 486 cents.
b) Suppose that now we have infinitely many coins for each type: 120 cents, 100 cents, 20 cents, 5 cents, and 1 cent. Using that algorithm, descrive how to change 327 cents.
c) Does b) gives a minimized number of coins as a solution? If not, give a better solution.

Problem 3. *3 + 2 + 3 + 2 + 1 + 1 pts*

Fix $N = 2^n$ for some $n \in \mathbb{Z}^+$. Your task is to sort an integer array A with size N: $A[0, \ldots, N-1]$ in ascending order.

a) Construct a function that takes an integer array *arr*, integers *start*, *mid*, *finish*, and then sort $arr[start, \ldots, finish - 1]$. Assume that $arr[start, \ldots, mid - 1]$ and $arr[mid, \ldots, finish - 1]$ are already sorted in ascending order, and also assume that change of the parameter *arr* will be automatically applied to the caller.
b) Now suppose $A[0, N/4 - 1], A[N/4, N/2 - 1], A[N/2, 3N/4 - 1], A[3N/4, N - 1]$ are already sorted in ascending order. What acts you have to do to get a fully sorted A? Answer using your function in a).
c) Using the idea of b) and the function constructed in a), describe the algorithm to sort an unsorted array A, starting from sorting 2-element N/2 arrays. You would have to use the iterative loop: start from parameter = 2 and double it step by step.
d) The function in a) will compare elements in array at most *finish* − *start* times. Count the upper bound of the number of comparisons to sort A fully and answer using n.
e) Convert the answer in d) using N instead, and then give a time complexity of the algorithm using big-O notation.
f) Compare the time complexity with bubble sort.