MA/CSSE 473 – Design and Analysis of Algorithms

Homework 7 (58 points total) Updated for Summer, 2017

When a problem is given by number, it is from the Levitin textbook. 1.1.2 means "problem 2 from section 1.1"

Problems for enlightenment/practice/review (not to turn in, but you should think about them):

How many of them you need to do serious work on depends on you and your background. I do not want to make everyone do one of them for the sake of the (possibly) few who need it. You can hopefully figure out which ones you need to do.

4.3.1 [5.4.1] (Reasonableness of generating all permutations, subsets of a 25-element set)
4.3.9 [5.4.9] (Generation of binary reflected Gray Code based on bit-flipping)
5.1.1 [4.1.1] (divide-and-conquer array max for unsorted array)
5.1.2 [4.1.2] (divide-and-conquer array max/min for unsorted array)

Problems to write up and turn in:

1. (6) 5.2.8 [4.2.8] (Negatives before positives)

   Previous questions and answers from Piazza:

   Q: can the array include zeroes? **A:** Yes

   Q: If so, is there any restriction on where they can be in the final array? **A:** No

2. (8) 5.2.9a [4.2.9] (Dutch National Flag) [do it with a one-pass algorithm for full credit]

   Previous questions and answers from Piazza:

   Q: Since the goal is let all the R's come first, the W's come next, the B's come last. Is it OK to just count how many R, W, B and then fill the array again? For example, I count that 3R, 4W, 2B. Then I just return result [R, R, R, W, W, W, W, B, B]?
   A: No. The problem explicitly says "rearrange". The goal of the problem is to better understand quicksort's partition algorithm by writing a variation of it.

3. (5) 4.1.4 [5.1.3] (generate power set)

4. (9) 4.3.2 [5.4.2] (Examples of permutation generation algorithms)

   You do not have to write any code, but you can do it that way if you wish.

5. (10) 4.3.10 [5.4.10] (Generation of all k-combinations from an n-element set)

   Previous questions and answers from Piazza:

   Q: What's the general definition of minimal-change? **A:** The change as we go from one element to the next is as simple as possible.
   Q: And in terms of generating permutation, what does minimal-change mean? My intuition says we only swap once, and we swap neighboring elements, without skipping any in-between. Is that correct? Or should it be only one swap, but we don't care how many things are in-between? **A:** Always swap two neighboring elements (constant time)

6. (10) 4.3.11 [5.4.11] (Generation of binary reflected Gray code based on Tower of Hanoi moves.

Previous questions and answers from Piazza:

Q: is it OK to just give the steps how to solve the problem? Or we also need to show why?

A: You need to describe the algorithm. For example, as part of your description you might say something like:

If the next move in the Towers of Hanoi algorithm _____ , then in our Gray Code generation, we flip bit _____ next.

7. (10) 4.3.12 [not in 2nd edition] (Fair attraction) See the "problems" document for details.

Previous questions and answers from Piazza:

Q: The last sentence in this problem description is "Design an algorithm to turn on the light bulb with the minimum number of button pushes needed in the worst case for n switches." I'm taking that to mean that even in the worst case the algorithm gets the minimum amount of button pushes, but that would be impossible as you would have to guess correctly every time. So what am I supposed to be taking this to mean?

A: Maybe "needed in" should be "needed for": "minimum number of button pushes needed for the worst case". That is, we are only considering the worst case here, The algorithm should produce the minimum number of button pushes needed to solve the problem for that case