473 Levitin problems and hints HW 07

Problem 1 5.2.8 [4.2.8] (Negatives before Positives)

8. Design an algorithm to rearrange elements of a given array of n real numbers so that all its negative elements precede all its positive elements. Your algorithm should be both time- and space-efficient.

Author's hint for problem 1: Use the partition idea.

Problem 2: **(8)** 5.2.9 [4.2.9] (Dutch National Flag)

9. ▶ The *Dutch national flag problem* is to rearrange an array of characters R, W, and B (red, white, and blue are the colors of the Dutch national flag) so that all the R's come first, the W's come next, and the B's come last. Design a linear in-place algorithm for this problem.

Author's Hints:

9. a. You may want to solve first the two-color flag problem, i.e., rearrange efficiently an array of R's and B's. (A similar problem is Problem 8 in this section's exercises.)

   b. Extend the definition of a partition.

Note: THIs is two different hints for one problem, not hints for two parts of the problem (there is only one part).

Problem 3: **(5)** 4.1.4 [5.1.3] generate power set

4. Design a decrease-by-one algorithm for generating the power set of a set of n elements. (The power set of a set S is the set of all the subsets of S, including the empty set and S itself.)

Author's Hints:

4. Use the fact that all the subsets of an n-element set $S = \{a_1, ..., a_n\}$ can be divided into two groups: those that contain a_n and those that do not.

Problem 4: **(9)** 4.3.2 [5.4.2] Examples of permutation generation algorithms

2. Generate all permutations of $\{1, 2, 3, 4\}$ by

   a. the bottom-up minimal-change algorithm.

   b. the Johnson-Trotter algorithm.

   c. the lexicographic–order algorithm.

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

Author's Hints:

2. We traced the algorithms on smaller instances in the section.

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

10. ▶ Design a decrease-and-conquer algorithm for generating all combinations of k items chosen from n, i.e., all k-element subsets of a given n-element set. Is your algorithm a minimal-change algorithm?

Author's Hints:

10. There are several decrease-and-conquer algorithms for this problem. They are more subtle than one might expect. Generating combinations in a predefined order (increasing, decreasing, lexicographic) helps with both a design and a correctness proof. The following simple property is very helpful. Assuming with no loss of generality that the underlying set is $\{1, 2, ..., n\}$, there are $\binom{n-i}{k-1}$ k-subsets whose smallest element is i, $i = 1, 2, ..., n-k+1$.

Problem 6: (10) 4.3.11 [5.4.11]

11. *Gray code and the Tower of Hanoi*

    (a) ▷ Show that the disk moves made in the classic recursive algorithm for the Tower-of-Hanoi puzzle can be used for generating the binary reflected Gray code.

    (b) ▶ Show how the binary reflected Gray code can be used for solving the Tower-of-Hanoi puzzle.

Author's Hints:

11. Represent the disk movements by flipping bits in a binary n-tuple.

Problem 7: (10) 4.3.12 Not in 2nd edition

12. *Fair attraction* In olden days, one could encounter the following attraction at a fair. A light bulb was connected to several switches in such a way that it lighted up only when all the switches were closed. Each switch was controlled by a push button; pressing the button toggled the switch, but there was no way to know the state of the switch. The object was to turn the light bulb on. Design an algorithm to turn on the light bulb with the minimum number of button pushes needed in the worst case for n switches.

Author's Hints:

12. Thinking about the switches as bits of a bit string could be helpful but not necessary.