Report on "Introduction to Algorithms and Data Structures"

January 7–January 10, 2020

Ryuhei Uehara (uehara@jaist.ac.jp)

Do not forget to write your name, student ID, problems, and answers on your report. Choose one of Problems 1 and 2, and one of Problems 3 and 4, and answer them in English (or Japanese). (If you choose more, I'll take from better scores.) Submit the final report to Dr. Wint Thida Zaw (wintthidazaw@uit.edu.mm) up to January 31 (Friday).

Problem 1 (10 pts): In the first report, we learnt a function that swaps two variables by using Exclusive OR. In fact, we can perform it by using usual + and − instead of Exclusive OR. Replace the following "[??]" by + or − to complete the swap function.

```
Swap(int x, int y){
  x = x [??] y;
  y = x [??] y;
  x = x [??] y;
}
```

Problem 2 (10 pts): Suppose that we perform BFS and DFS on a complete binary tree T that consists of n nodes. (Since it is a *complete* binary tree, we have $n = 2^k - 1$ for some integer k.) Then estimate the space complexity (that is, the quantity of memory) of BFS and DFS on T.

Problem 3 (20 pts): You want to shuffle the data which is in $a[0], a[1], \ldots, a[n-1]$ by randomization. You can use a random generator function `random`(k) that returns an integer i with $0 \leq i < k$ uniformly at random. Then, show a *shuffle* algorithm for $a[]$. That is, the algorithm outputs each possible permutations of $a[]$ with the same probability. Evaluate the running time of your algorithm. (Hint: there are several ways, but there exists a simple $O(n)$ time algorithm.)

Problem 4 (20 pts): Let $a[0], a[1], \ldots, a[n-1]$ be the array of data. Then the *partition problem* is defined as follows: For the given input array $a[]$, determine if you can partition data into two subsets that make the same value. That is, when $S = \sum_{i=0}^{n-1} a[i]$, determine if there is a subset I of $\{0, 1, \ldots, n-1\}$ such that $\sum_{i \in I} a[i] = S/2$ (and $\sum_{i \notin I} a[i] = S/2$). It is intractable in general. Now, suppose that S is not so big. Then there is an algorithm that solves the partition problem in polynomial time of n and S. Show the algorithm and its running time. (Hint: construct a table $T[i, j]$ that indicates if the summation j can be made by a subset of $\{0, 1, \ldots, i\}$.)