Illinois Institute of Technology
Department of Computer Science

Homework Assignment 6

CS 430 Introduction to Algorithms
Spring Semester, 2016

Due: Monday, April 11

The first four problems are based on problem 35 in A. Levitin & L. Levitin, *Algorithmic Puzzles*, Oxford University Press, 2011 and chapter 5 of T. S. Michael, *How to Guard an Art Gallery*, Johns Hopkins University Press, 2009.

1. Suppose you are given an 8-pint jug full of water and two empty jugs of 3- and 5-pint capacity; you must get exactly 4 pints of water in one of the jugs by completely filling/emptying jugs into each other. This problem can be modeled by an implicit graph: The vertices are triples (i, j, k) where i is the amount of water the 3-pint jug, j is the amount of water in the 5-pint jug, and k is the amount of water in the 8-pint jug. The (directed) edges connect a triple T to triples that can be obtained from T by pouring. Thus, for example, there is an edge from $(3, 3, 2)$ to $(1, 5, 2)$ because we can pour from the 3-pint to fill the 5-pint jug, leaving 1 pint in the 3-pint jug.

   You are given jugs of capacities c_1, c_2, and c_3 and have to get a target amount t. Design an algorithm based on breadth-first search to determine how to get t, or to verify its impossibility. Remember, the graph is implicit—you do not have an explicit form of it (in a an adjacency structure, say).

2. Justify the correctness of your algorithm in problem (1). This is just shy of a formal CLRS3-style proof.

3. Analyze the time required by your algorithm in problem (1).

   (*Hint*: See `http://en.wikipedia.org/wiki/Pseudo-polynomial_time`.)

4. Implement, in any language and on any platform you choose, your algorithm in problem (1). The user must be able to enter any three positive integers c_1, c_2, c_3, and a target t. The graph might too big to be kept explicitly. It may be that that target value is impossible to reach and the program must so report. An ideal implementation would allow for more than three jugs—say, c_1, c_2, c_3, c_4, and c_5. Your algorithm must give the sequence of pourings to reach the target, when possible, and must indicate when the target cannot be achieved.

5. This is a version of Problem 22-3 on page 623 of CLRS3. Given an undirected, connected graph $G = (V, E)$, an *Eulerian cycle* is a cycle that traverses each edge of G exactly once, although it may through a vertex more than once. An *Eulerian path* is a path that traverses each edge of G exactly once, although it may through a vertex more than once.

   (a) Prove that an undirected graph has an Eulerian cycle if and only if all vertices have even degree and has an Eulerian path if at most two vertices have odd degree. What happens if only one vertex has odd degree?

   (b) Give an $O(|V| + |E|)$ algorithm to find an Eulerian cycle (or path) if one exists and to report that neither exists, if that is the case.