Algorithms, 2016/07/11

1. (40) Efficient algorithms for computing $y = a^x \bmod n$, for very large integers a, x, and n is very useful in cryptography. We first consider the problem of computing $y = a^x$. On input two integers a and x the following two algorithms compute $y = a^x$:

   Algorithm A:
   ```
   $y = 1$;
   for ($i = 0$; $i < x$; $i++$) {
     $y = y \times a$;
   }
   print $y$;
   ```

   Let x be a k-bit integer, and $x_{k-1}x_{k-2}\ldots x_0$ be the binary representation of x.

   Algorithm B:
   ```
   $y = 1$;
   for ($i = k-1$; $i \geq 0$; $i--$) {
     $y = y^2 \bmod n$;
     if ($x_i == 1$) $y = y \times a \bmod n$;
   }
   print $y$;
   ```

   Assume that computing $a \times b$ requires $|a||b|$ bit operations, where $|a|$ ($|b|$) is the number of bits of the binary representation of a (b).

   (a) Analyze the time complexity of the above two algorithms. (10)
   (b) Which algorithm should be used for computing a^x for large x? (5)
   (c) Modify the above two algorithm for computing $a^x \bmod n$. (10)
   (d) Analyze the time complexity of the two modified algorithms. (10)
   (e) Which of the two modified algorithm should be used for computing $a^x \bmod n$ for large x and n? (5)

2. (20) The subgraph isomorphism problem is to determine whether an undirected graph contains a subgraph which is isomorphic to another undirected graph. The satisfiability problem is to determine whether a Boolean formula has a truth assignment to satisfy the formula. The k-clique problem is to determine whether a graph contains a complete subgraph of size k. The Hamiltonian problem is to determine whether an undirected graph contains a Hamiltonian cycle. Assume that the satisfiability problem, the clique problem and the Hamiltonian problem are all known to be NP-complete. Prove that the subgraph isomorphism problem is NP-complete.

3. (20) Let $a_1, a_2, \ldots, a_n$ be a sequence of n distinct integers sorted in ascending order. Binary search can be used to find an index i such that $a_i = k$ for a given input k, or report that no such integer exits in the sequence, in $O(\log n)$ time. Design an efficient algorithm for each of the following cases. The time complexity of your algorithms must be better than the time complexity of binary search. That is, your algorithms for the following cases should be $o(\log n)$.

   (a) k is relatively small, that is, k would appear near the front of the sequence, if it exists.
   (b) k is relatively large, that is, k would appear near the end of the sequence, if it exists.

4. (20) Let G be a weighted graph, s be a vertex of G. Suppose that you have brought a program for the single source shortest path problem. Suppose that your program has constructed a shortest path tree T with respect to the source vertex s. Design an efficient algorithm to verify that T is indeed a shortest path tree for the graph G with respect to the source vertex s.

5. (+20) Define the following terms.

   (a) decision problems, optimization problems;
   (b) P, NP, and NP-complete problems;
   (c) approximation algorithms, approximation ratio;
   (d) polynomial-time approximation scheme, fully-polynomial-time approximation scheme.