Algorithms, 2019/01/14

1. (10) State formal definition of $f(n) = O(g(n))$ by using: $\exists\ n_0$ and c, $(\forall\ \cdots)$.
   Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$? Justify your answers by using the definition.

2. (10) Fibonacci numbers are defined as $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$, for $n > 1$. Consider the following algorithm expressed in C-like code for computing Fibonacci number F_n.

```
integer f[N];
integer F(n) {
  if (f[n] ≥ 0) return f[n];
  f[n] = F(n − 1) + F(n − 2);
  return f[n];
}

int main() {
  read n;
  f[0] = 0; f[1] = 1;
  for (i = 2; i ≤ n; i = i + 1)
    f[i] = −1;
  print F(n);
}
```

   In the above C-like code, `integer` is a data type for integers. Assume that the input data n is a non-negative integer. How many times the statement "$f[n] = F(n-1) + F(n-2)$" will be executed?

3. (20) Let $G = (V, E, w)$ be a weighted connected undirected graph. Assume that the weight on each edge is positive and all edges weights are distinct. For a vertex $v \in V$, let $\alpha(v)$ be the edge with minimum weight among all edges incident at v. Let F be the subset of edges defined by $F = \{\alpha(v) \mid v \in V\}$.

   (a) Show that the induced graph $G[F]$ contains no cycles.

   (b) Prove or give a counter-example to show that the induced sub-graph $G[F]$ is a minimum spanning tree of G.

   (c) Based on (a) and (b), design an efficient algorithm for computing a minimum spanning tree of G.

4. (20) Let G be a weighted graph with vertex set $\{1, 2, \ldots, n\}$. Let s be a vertex of G. Suppose that you have brought a program for the single source shortest path problem. Your program has computed shortest distances from s to other vertexes, and stored the distances in an array $D[1:n]$. (That is, the shortest distance from s to vertex v is $D[v]$.)

   (a) Design a linear-time algorithm to verify that the shortest distances are all correct.

   (b) Design a linear-time algorithm to construct shortest path tree for the graph G with respect to the source vertex s.

5. (20) Let n be a large integer. Assume that n is not a prime number.

   (a) Design a non-deterministic algorithm for factoring n.

   (b) Design a probabilistic algorithm for factoring n.

   (c) Assume that n is a product of two large distinct primes p and q, and $p, q \approx \sqrt{n}$. What is the probability of your algorithm in (b) to successfully factor n?

6. (20) Let $x_1, x_2, \cdots, x_n$ be a data set of n integers. The *mode* of the n integers is the integer that appears most often in the data set. We are going to design an algorithm for finding the mode in a very large data set by using very small amount of memory. In the following C-like code, valuables a and c are used to denote a occurs c times.

```
c = 0;
for (i = 1; i ≤ n; i = i + 1) {
    read next x;        // read x_i
    if (c = 0) {a = x; c = 1;}
    elseif (x = a) {c = c + 1;}
    else {c = c − 1;}
}
print a;
```

   (a) Let the n input integers be $1, 2, 3, 3, 1, 3, 3$. Simulate the execution of the program, and show the output is correct.

   (b) Explain why the above algorithm works for the case that some data occurs at least $(n+1)/2$ times in the input.

   (c) If no data occur at least $(n+1)/2$ times, show that the algorithm will fail.