Analysis of Algorithms **Comprehensive Examination** **Spring 2019**

Sign this exam with your student number — not your name: ______________________

1. (10 pts) Multi-precision multiplication of integers. Suppose X and Y are two n-bit integers written in binary notation. Assume n is even and break X and Y into two (n/2)-bit integers A, B, and C, D, respectively, so

$$X = A2^{n/2} + B \qquad \text{and} \qquad Y = C2^{n/2} + D$$

The product of X and Y can be calculated as

$$XY = AC2^n + ((A - B)(D - C) + AC + BD)2^{n/2} + BD$$

which requires three multiplications of $n/2$ bit numbers: AC, BD and $(A - B)(D - C)$ plus kn operations to add the products and perform the shifts. Solve the recurrence

$$T(n) = 3T(n/2) + kn, \qquad T(1) = 1$$

2. Sorting sorting a list of values is a classic computing problem.

(a) (5 pts) Bubble-sort is well known. What is its best case, worst case, and average case time complexity?

(b) (5 pts) Quick-sort is also famous. Describe the basic way Quick-sort executes.

3. (10 pts) Dynamic Programming.

(a) How would you define edit distance between two strings?

(b) The code below computes the edit distance between two strings. What is its time complexity?

```
#include <stdio.h>
#include <string.h>

int editDistance(const char *s, const char *t) {
  int n = strlen(s), m = strlen(t);
  int d[n + 1][m + 1];

  for (int i = 0; i <= n; i++) { d[i][0] = i; }
  for (int j = 0; j <= m; j++) { d[0][j] = j; }

  for (int i = 1; i <= n; i++) {
     for (int j = 1; j <=m; j++) {
        int a = d(i-1,j) + 1;
        int b = d(i-1,j) + 1;
        int c = d(i-1, j-1) + (s[i] != t[j]);
        d[i, j] = min(a, b, c);
     }
  }
  return d(n, m);
}
```

4. Greedy Algorithms. The rational knapsack problem can be framed as a container that can hold a weight capacity C together with a list of provisions p_i, $i = 0, \dots, n$ to be placed in the container. Each provision has a corresponding weight w_i and value v_i. The provisions are to be placed into the container without exceeding the capacity C and such that the sum of values is maximized.

(a) (5 pts) Describe a greedy heuristics that could be used to fill the container.

(b) (5 pts) Assume that fractional amounts $0 \leq f \leq 1$ of each provision can be used. Describe an optimal filling strategy and its time complexity.

(c) (5 pts) Now assume an all or nothing (0 or 1) approach must be used. Describe an algorithm, and its complexity, that is guaranteed to find the maximum sum of values without exceeding the capacity C.

Brief Answer Questions

1. (5 pts) What does it mean to say $f(n) = O(g(n))$

2. (5 pts) What does it mean to say $f(n) = \Omega(g(n))$

3. (5 pts) What does it mean to say $f(n) = \Theta(g(n))$

4. (5 pts) (`True` or `False`) Explain your answer. If $f(n) = O(\sqrt{n})$ then $f(n) = O(n)$.

5. (5 pts) (`True` or `False`) Explain your answer. $\sqrt{n} = O(\lg n)$.

6. Satisfiability (SAT) is a classic NP problem: A *literal* is a Boolean variable or the negation of a variable. A *clause* is *disjunction* (logical OR) of literals. The SAT decision problem is: Given a set of clauses, Is there an assignment of Boolean values to each variable such that every clause has at at least one `True` literal?
   (a) (5 pts) If the answer to an instance of SAT is “yes”, Describe a non-deterministic polynomial–time algorithm that would prove the answer is “yes.”

   (b) (5 pts) If the answer to an instance of SAT is “no”, Describe a why any algorithm that proves the answer is “no” would take at least $O(2^n)$ time to prove the answer is “no.”

7. (5 pts) `True` or `False`: One way to prove a problem X is NP-complete is to develop a reduction (a polynomial time transformation) from a known NP-complete problem Y to X.

8. (5 pts) Suppose there exists a polynomial-time reduction of instances of problem Y to instances of problem X.
   (a) If $X \in P$ what can you conclude about the complexity of Y?

   (b) (5 pts) If Y is undecidable, what can you conclude about the complexity of X?

   (c) (5 pts) `True` or `False`: Non-deterministic algorithms can be simulated with deterministic algorithms.