Dept. of Computer Science and Engineering, National Sun Yat-sen Univ.
Second Semester of 2007 PhD Qualifying Exam Computer Algorithms

1. Explain the following terms. (24%)
   (a) *NP, NP-complete*
   (b) *1-center problem*
   (c) *traveling saleperson problem*
   (d) *minimum spanning tree*
   (e) *prune-and-search*
   (f) *exact set cover problem*

2. (a) Explain the *longest common subsequence* (LCS) problem. And, then give an example to illustrate your answer. Note that you should give both explanation and example. (6%)
   (b) Give the *dynamic programming* approach to solve the LCS problem. What is the time complexity of this algorithm? Why? (10%)

3. It is well-known that the *lower bound* of the *sorting* problem is $\Omega(n\log n)$ based on the comparison operations, where n is the input size. Prove that the lower bound of the *convex hull* problem is also $\Omega(n\log n)$ by problem transformation. (15%)

4. Suppose we obtain the following recurrence formula of time complexity for solving some problem:

$$T(n) = \begin{cases} b & \text{if } n \leq 2 \\ 8T(\frac{n}{2}) + cn^2 & \text{if } n > 2, \end{cases}$$

   where n is the input size of the solved problem. Please derive the time complexity and represent it with O notation. (15%)

5. Prove that the *clique* decision problem polynomially reduces to the *node cover* decision problem. (15%)

6. Given a 2-dimensional $n \times n$ array of positive and negative integers, find the sub-rectangle with the largest sum. A sub-rectangle is any contiguous sub-array of size 1×1 or greater located within the whole array. The sum of a sub-rectangle is the sum of all elements in that sub-rectangle. As an example, the sub-rectangle with the largest sum of the array:

$$\begin{array}{rrrr} 0 & -2 & -7 & 0 \\ 9 & 2 & -6 & 2 \\ -4 & 1 & -4 & 1 \\ -1 & 8 & 0 & -2 \end{array}$$

   is in the lower left corner:

$$\begin{array}{rr} 9 & 2 \\ -4 & 1 \\ -1 & 8 \end{array}$$

   and it has the sum 15. Please design an algorithm with $O(n^4)$ time to solve this problem and illustrate your algorithm with the above example. You should also give the analysis of time complexity. (15%)