Dept. of Computer Science and Engineering, National Sun Yat-sen Univ.
First Semester of 2004 PhD Qualifying Exam Computer Algorithms

1. (a) What is the *2-D ranking* problem? (5%)

   (b) Design an $O(n \log n)$ algorithm to solve the 2-D ranking problem. And, analyze the time complexity. (10%)

2. (a) What is the difference between *divide-and-conquer* and *prune-and-search*? (5%)

   (b) Is the *binary search* algorithm a divide-and-conquer method or a prune-and-search method? Why? (5%)

   (c) Please give the general formula for the time complexity of an algorithm with the prune-and-search strategy. (5%)

3. (a) What is the *amortized analysis*? (4%)

   (b) What are the *move-to-front* heuristics and *transpose* heuristics in self-organizing sequential search? Give examples for illustration. (6%)

4. The *merge sort* algorithm is to repeatedly apply the *2-way merge* method, which is a linear merging scheme to merge two sorted lists into a single one. Write a *recursive* C function (program) to perform the merge sort. To implement your merge sort, you can call the following 2-way merge function as a well-known function. In other words, you need not write the 2-way merge function. If you are not familiar with C language, you can use other languages to implement the merge sort. (10%)

```
void two way(int a[ ], int b[ ], int c[ ])
/* a[ ] and b[ ] are input sorted arrays */
/* c[ ] is the output array after a[ ] and b[ ] are merged */
```

5. (a) Explain the *longest common subsequence* (LCS) problem. Give an example for illustration. (5%)

   (b) For given a sequence of distinct integers, the *longest increasing subsequence* (LIS) problem is to find the longest subsequence which is increasing. For example, the LIS of $< 10, 7, 6, 8, 9, 5, 13, 2 >$ is $< 7, 8, 9, 13 >$ or $< 6, 8, 9, 13 >$. Design an algorithm in $O(n^2)$ time, where n is the length of the input sequence, to solve the LIS problem. You can assume that the LCS algorithm is well-known. In other words, you need not write the details of the LCS algorithm when you use it. (15%)

6. A *clique* in a graph is a maximal complete subgraph. For given a graph $G = (V, E)$, a subset S of the vertex set V is called a *node cover* if all edges in E are incident to at least one vertex in S. Show that the *maximum clique decision* problem reduces to the *node cover decision* problem. (15%)

7. Suppose that you have n delayed jobs, in which each job i needs T_i, $1 \leq i \leq n$, days (T_i is an integer) to finish. You can work on only one job in each day. Since the jobs have been delayed, for each delayed day before starting to work for job i, you have to pay a fine(penalty) of S_i dollars. In other words, only the job assigned in the first day has no fine. Design an algorithm in $O(n^2)$ time to minimize the total amount of fines. Show that your algorithm is correct and analyze the time complexity. (15%)