國立臺灣師範大學資訊工程學系

101 學年度第一學期

博士班資格考

考試科目：演算法

總分一百分

請在答案卷作答，在題目卷上作答不予計分

1. **(10 pts)** Is the average case complexity the average of the best and the worst complexity? Prove or give a counter example.

2. **(15 pts)** A sorting algorithm is *stable* if numbers with the same value appear in the output array in the same order as they do in the input array.

(a) **(8 pts)** Are the following sorting algorithms stable: insertion sort, merge sort, heapsort, and quicksort?

(b) **(7 pts)** Give a simple scheme that makes any sorting algorithm stable. How much additional time and space does your scheme entail?

3. **(25 pts)** Suppose we can buy or sell a unit of stock only once per day, and assume we know future prices, please design an algorithm that finds the best dates (a pair of dates) for trading the stock in order to maximize the profit given a sequence of stock prices. For example, given the stock prices in the table below, the best strategy is to buy the stock on 'Day 8' and sell it on 'Day 12', where the profit is 43 (186-143).

Day	1	2	3	4	5	6	7	8	9
Price	180	192	189	165	185	182	166	143	161
Day	10	11	12	13	14	15	16	17	
Price	181	174	186	181	159	173	169	175	

(a) **(5 pts)** Describe the time complexity if we use a brute force approach.

(b) **(10 pts)** Describe a divide-and-conquer approach that solves the problem. Analyze the time and space complexity.

(c) **(10 pts)** Describe a dynamic programming approach that solves the problem. Analyze the time and space complexity.

4. **(24 pts)** Stacks, queues, heaps, and sets are commonly used data structures. Breadth-first search, depth-first search, Kruskal's algorithm, and Prim's algorithm are popular graph algorithms. Please answer the following questions.

(a) **(8 pts)** List and describe the main operations supported by each of the four mentioned data structures.

(b) **(4 pts)** Each of the four mentioned structures is the core of one of the four mentioned algorithms. Match them.

(c) **(12 pts)** Describe how each of the structures helps to implement the matched algorithm.

5. **(6 pts)** Your department is asked to write a program that has two functions: the first function, ADD, is to add one user name at a time; the other, DISPLAY, is to display all user names in lexicographic order. Two programmers Gasol and Howard propose their solutions, respectively. They both use the array to store user names, but they do sorting in different places.

Gasol uses the insertion sort in ADD to put the newly added name in the correct position, and thus DISPLAY is implemented straightforward. Howard just places the newly added name at the end of the array in ADD, and the merge sort is carried out every time when his DISPLAY is called to ensure that the names can be listed in lexicographic order.

(a) **(3 pts)** According to different patterns (the number of calls, the order of calls, etc.) of calling ADD and DISPLAY, one solution is more computationally efficient than the other in some cases but less efficient in other cases. Describe how you will choose between these two solutions.

(b) **(3 pts)** The third programmer, Bryant, proposes to add a boolean flag to improve Howard's solution. The flag is set by true when ADD is called. The merge sort in DISPLAY is carried out only when the flag is true, and the flag is set by false after doing the merge sort. How do you evaluate Bryant's solution?

6. **(10 pts)** Dijkstra's algorithm solves the single-source shortest-path problem. The pseudo code is as follows.

DIJKSTRA(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)
$S \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    do $u \leftarrow$ EXTRACT-MIN(Q)
        $S \leftarrow S \cup \{u\}$
        for each vertex $v \in Adj[u]$
            do RELAX(u, v, w)

(a) **(5 pts)** The priority queue Q can be implemented by a linear array or a binary heap. Describe how you choose between them. (You can analyze the complexity and give the cases in which you will choose them.)

(b) **(5 pts)** The Bellman-Ford algorithm is another algorithm to solve the single-source shortest-paths problem. Describe how you choose between Dijkstra's algorithm and the Bellman-Ford algorithm.

7. **(10 pts)** The Floyd-Warshall algorithm solves the all-pairs shortest-paths problem. The pseudo code is as follows.

FLOYD-WARSHALL(W)

$n \leftarrow \text{rows}[W]$
$D^{(0)} \leftarrow W$
for $k \leftarrow 1$ to n
    do for $i \leftarrow 1$ to n
        do for $j \leftarrow 1$ to n
            $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $D^{(n)}$

(a) **(3 pts)** Give the time complexity of the Floyd-Warshall algorithm.

(b) **(3 pts)** The transitive closure problem is to fine whether there is a path from any vertex i to any vertex j. Describe how we can solve the transitive closure problem by assigning proper weights to the edges and then applying the Floyd-Warshall algorithm.

(c) **(4 pts)** Following the problem (b), if we want to save time and space in practice, we can modify the Floyd-Warshall algorithm by replacing some arithmetic operations in the Floyd-Warshall algorithm by logical operations. Describe how to do it.