

Ph.D. Comprehensive Exam:
Math and Algorithms

Spring 2002

The exam includes eight problems;
the length of the exam is three hours.

Problem 1

Prove or disprove the following inequality:

$$\sum_{n=1}^{\infty} \frac{1}{(2 \cdot n - 1) \cdot 2 \cdot n} < 0.8$$

Problem 2

Consider the following recurrence:

$$\begin{aligned} A_1 &= 1 \\ A_n &= (2 \cdot n - 1)^2 + A_{n-1} \quad (\text{where } n \geq 2) \end{aligned}$$

Give a formula for A_n in terms of n without the use of recurrence.

Problem 3

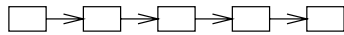
Consider the following algorithm, which inputs a natural number n and returns a natural number m . It uses a square matrix $A[1..n, 1..n]$ for intermediate results. Give a much faster algorithm that computes the same value m without using a matrix.

```
SLOW-COUNTER( $n$ )
for  $i \leftarrow 1$  to  $n$ 
    do for  $j \leftarrow 1$  to  $n$ 
        do  $max \leftarrow 0$ 
            for  $k \leftarrow 1$  to  $i - 1$ 
                do if  $max < A[k, j]$ 
                    then  $max \leftarrow A[k, j]$ 
            for  $k \leftarrow 1$  to  $j - 1$ 
                do if  $max < A[i, k]$ 
                    then  $max \leftarrow A[i, k]$ 
             $A[i, j] \leftarrow max + 1$ 
 $m \leftarrow A[n, n]$ 
return  $m$ 
```

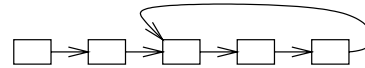
Problem 4

If we need to print all elements of a linked list (Figure 1), we can write a simple program that starts at the beginning of the list and follows pointers until reaching the end; however, novice programmers sometimes mistakenly create a list whose last element points into the middle of the list (Figure 2), and then an attempt to print all elements leads to an infinite loop.

Write an algorithm that determines whether a given linked list is “looped.” Your algorithm should run in linear time, use no extra memory, and preserve the initial contents of the list. These restrictions mean that you cannot mark the elements of the list that you have visited, because storing such marks would require a lot of additional memory.



1. Standard linked list.



2. Looped linked list.

Problem 5

We consider a set of integers and denote the number of its elements by n . Describe a data structure for representing this set that supports the following four operations.

- Add a given integer to the set; if this integer is already in the set, do nothing. The time complexity of adding an integer must be $O(\lg n)$.
- Delete a given integer from the set; if this integer is not in the set, do nothing. The complexity of this operation must be $O(\lg n)$.
- For a given natural number k , where $k \leq n$, print k smallest values in the set. For example, if $k = 2$ and the set includes integers 1, 4, 5, 7, 9, then this operation prints 1, 4. Its complexity must be $O(k)$.
- For a given natural number k , where $k \leq n$, print k largest values in the set. For example, if $k = 2$ and the set includes integers 1, 4, 5, 7, 9, then this operation prints 7, 9. Its complexity must also be $O(k)$.

Note that k may be smaller than $\lg n$, and the complexity of the last two operations must be $O(k)$. An algorithm with the complexity $O(k + \lg n)$ is not an appropriate solution.

Problem 6

Consider two problems given below. The Standard Vertex Cover problem is known to be NP-complete. Use this fact to prove that Modified Vertex Cover is also NP-complete.

Standard Vertex Cover: We consider an undirected graph, and we can put lamps in any of its vertices. An edge is lighted if at least one of its two endpoints has a lamp. The task is to input a graph and determine the minimal number of lamps required for lighting all edges.

Modified Vertex Cover: We again put lamps into vertices of an undirected graph. A vertex is lighted if we put a lamp either into the vertex itself or into one of adjacent vertices. The task is to determine the minimal number of lamps required for lighting all vertices.

Problem 7

Design a good approximation algorithm for Modified Vertex Cover, described in Problem 6. Your algorithm should find an “almost” minimal number of lamps for lighting all vertices of a given graph; its running time must be polynomial. Give an example of a graph for which your algorithm does not find the minimal number of lamps. If your algorithm is good, finding such an example should be difficult.

Problem 8

The k -Coloring Problem requires to assign colors to the vertices of a given undirected graph. Each vertex must have one of k colors, and adjacent vertices must have different colors. The Two-Coloring Problem has a polynomial-time solution, whereas Three-Coloring is known to be NP-hard. Use these facts to determine whether the 2002-Coloring Problem is NP-hard and justify your answer.