Institut for Matematik og Datalogi
Syddansk Universitet

March 27, 2020
JFB

DM553/MM850 – Complexity and Computability 2020 – Lecture 12

Lecture, March 25

I showed that more problems are NP-Complete, from chapter 34 in CLRS. We covered CLIQUE, VERTEX COVER, INDEPENDENT SET, and SUBSET-SUM. There was also a conclusion and summary of NP-Completeness.

Lecture, March 30

I will start on lower bounds from notes, which can be found under "Course Materials" in Blackboard. (Part of this is also in section 8.1 of CLRS.) I will cover up through section 2.4 and start on adversary arguments from chapter 3 in those notes.

Lecture, April 2

We will cover section 3.1, 3.2, 3.3, and 3.5 from the notes on lower bounds.

Problems to be discussed on April 16

From CLRS do:

1. Do problem 3.10 from page 141 of the notes.
2. Prove a lower bound for merging two lists of lengths n and m which meets the upper bound of $n + m - 1$ (assume $n = m$).
3. From the following pages of that book by Baase:
   (a) Consider the problem of determining if a bit string of length n contains two consecutive zeros. The basic operation is to examine a position in the string to see if it is a 0 or a 1. For each $n = 2, 3, 4, 5$ either give an adversary strategy to force any algorithm to examine every bit, or give an algorithm that solves the problem by examining fewer than n bits.

(b) a. You are given n keys and an integer k such that $1 \leq k \leq n$. Give an efficient algorithm to find *any one* of the k smallest keys. (For example, if $k = 3$, the algorithm may provide the first-, second- or third-smallest key. It need not know the exact rank of the key it outputs.) How many key comparisons does your algorithm do? (Hint: Don't look for something complicated. One insight gives a short, simple algorithm — try finding smallest, but stop when you have enough information.)
b. Give a lower bound, as a function of n and k, on the number of comparisons needed to solve this problems.

4. From Baase's textbook: Suppose $L1$ and $L2$ are arrays, each with n keys sorted in ascending order.

   a. Devise an $O((\lg n)^2)$ algorithm (or better) to find the ith smallest of the $2n$ keys. (For simplicity, you may assume the keys are distinct.)

   b. Give a lower bound for this problem.

5. Design and analyze an efficient algorithm to find the third largest item in an array.

6. Consider the problem of Sorting by Reversals. You are given a permutation of the numbers from 1 to n in an array, A. The operation you have is to choose two indices, i and j, and to reverse the elements in the subarray from $A[i]$ to $A[j]$, inclusive. The objective is to end with a sorted array. For example, given $A = [8, 6, 4, 2, 7, 5, 3, 1]$, the first operation could be $(3, 6)$, resulting in $A = [8, 6, 5, 7, 2, 4, 3, 1]$. Then doing the operations $(2, 4), (3, 4), (5, 7), (5, 6), (1, 8)$ would finish sorting the array. The question is: What is the shortest sequence of operations which will sort an arbitrary array?

   a. Give an algorithm which sorts the array in $O(n)$ operations.

   b. Prove that any algorithm needs at least $\Omega(n)$ operations in the worst case.

   c. Why doesn't the information-theoretic lower bound of $\Omega(n \log n)$ apply here?