CSCE 750 Final Exam (Study Guide)
Thursday December 9, 2021

1 Exam Instructions (on the Cover Page)

- This is an in-person exam.
- Read the questions carefully and make sure to give the answers asked for. Pay particular attention to **boldface** terms. Don't give a beautiful answer to the wrong question.
- Write your answers on separate paper, except when directed by the problem to write on the handout.
- You have 150 minutes to complete this exam. Submit the exam handout along with your answers when done.
- There are 11 numbered problems, worth 20 points each. 200 points is considered full credit; any additional points are extra credit.
- This exam is open book, open notes, but no electronic devices are permitted.
- Show enough work to convince your instructor that you know what you're doing. Mark your answers clearly.
- If you continue an answer onto an additional sheet, note this clearly so the instructor can find it.
- Partial credit will be awarded for incorrect answers that demonstrate partial understanding of the relevant concepts. Therefore, it is to your advantage to explain your reasoning and show your work. However, meaningless, incomprehensible, or irrelevant writing will not earn partial credit.

2 Topics covered

Below are the topics covered by each numbered problem in the exam.

1. Solving a recurrence equation
2. Simulating and analyzing the run time of a recursive algorithm
3. Expected run time of a randomized algorithm
4. A max-treap
5. Amortized analysis
6. Dynamic programming
7. Minimum spanning trees and safe edges
8. Dijkstra's Algorithm
9. Membership in $\mathbf{P}$ and $\mathbf{NP}$
10. Understanding what it means to be in $\mathbf{P}$ or $\mathbf{NP}$
11. Understanding how a polynomial-time reduction works

3 Answers to HW8 Not-in-Textbook Questions

Not in book 1: Show, using a reduction from VERTEX COVER, that CLIQUE is NP-hard. (The CLIQUE problem is defined in Section 34.5.)

Answer: Given an instance $\langle G, k\rangle$ of VERTEX COVER, where G is a graph and k is an integer,

1. we construct a graph G' as follows: $G'.V = G.V$, and two vertices $u \neq v$ are adjacent in G' if and only if they are *not* adjacent in G. (G' is called the *complement* of graph G.);
2. we output the pair $\langle G', k'\rangle$, where $k' := |G.V| - k$.

This construction can clearly be done in polynomial time.

Let $n := |G.V|$ be the number of vertices of G (and of G'). We show that G has a vertex cover of size k if and only if G' has a clique of size $k' = n - k$. We do this by first showing that, for any set C of vertices, C is a vertex cover of G if and only if its complement $C' := G.V \setminus C$ is a clique in G': C is a vertex cover of G if and only if no pair of vertices of C' are adjacent in G, if and only if *every* pair of vertices in C' are adjacent in G', if and only if C' is a clique of G'.

Now since $|C'| = n - |C|$, we have the following: G has a vertex cover C of size k if and only if G' has a vertex cover C' of size $n-k = k'$. Thus the reduction maps 'yes'-instances of VERTEX COVER to 'yes'-instances of CLIQUE and 'no'-instances of VERTEX COVER to 'no'-instances of CLIQUE, and thus reduces VERTEX COVER to CLIQUE in polynomial time. Since VERTEX COVER is $\mathbf{NP}$-complete, it follows that CLIQUE is $\mathbf{NP}$-hard. (Since CLIQUE is also in $\mathbf{NP}$, it is $\mathbf{NP}$-complete.)

Not in book 2: Consider this problem:

> KARP-LETTERS
> *Instance:* A list of strings of varying lengths, consisting of upper- and lower-case letters.
> *Question:* Is there a way to select a letter from each string without choosing both versions of any letter?[1]

Either prove that KARP-LETTERS $\in \mathbf{P}$, or prove that KARP-LETTERS is $\mathbf{NP}$-complete. (Hint: Only one of these options can be completed correctly.[2]) For an $\mathbf{NP}$-completeness proof, you may reduce from any problem identified as $\mathbf{NP}$-hard in the lecture or in either textbook.

Answer: KARP-LETTERS is in $\mathbf{P}$. This can be seen as follows: There are 2^{26} ways of choosing either the upper- or lower-case version of each letter in the alphabet (a binary choice for each letter). Let the input be a list $\langle s_1, \ldots, s_n \rangle$ of strings, where each string s_i consists of upper and/or lower case letters. Our algorithm runs through all the 2^{26} choices described above, and for each, we determine if all of the strings $s_1, \ldots, s_n$ include some character in the choice. If we ever find this is true, we output 'yes', and otherwise output 'no'. Each iteration of the outer loop takes the same worst-case time, which is polynomial in the size of the input, and there are constantly many iterations,[3] so the whole test takes polynomial time.

[1]For example, if the input strings are 'Abc', 'BC', 'aB', and 'ac', the correct answer is 'Yes', because we can choose 'A' from the first string, 'B' from the second, 'B' from the third, and 'c' from the fourth. If the strings are 'AB', 'a', and 'b', then the correct answer is 'No'.

[2]. . . unless $\mathbf{P} = \mathbf{NP}$.

[3]The number of iterations is huge (2^{26}), but it is still constant.