Homework 1

CS 4104 (Fall 2021)

Assigned on August 31, 2021.
Submit a PDF file containing your solutions on Canvas by 11:59pm on September 7, 2021.

Instructions:

- The Honor Code applies to this homework with the following exception:
  - You can pair up with another student to solve the homework. Please form teams yourselves. Of course, you can ask the instructor for help if you cannot find a team-mate. You may choose to work alone.
  - You are allowed to discuss possible algorithms and bounce ideas with your team-mate. **Do not discuss proofs of correctness or running time in detail with your team-mate. You must write down your solution individually and indepedently. Do not send a written solution to your team-mate for any reason whatsoever.**
  - *In your solution, write down the name of the other member in your team. If you do not have a team-mate, please say so.*
  - Apart from your team-mate, you are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine.
- Do not forget to typeset your solutions. *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as "n^2".* You can use the LaTeX version of the homework problems to start entering your solutions.
- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
- If you are proposing an algorithm as the solution to a problem, keep the following in mind (the strategies are based on mistakes made by students over the years):
  - Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*
  - Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is $m + n$, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$ time.

My team-mate is ____________________

Problem 1 (5 points) If a potential solution to the Stable Matching problem is a perfect matching but is not stable, then the number of rogue couples must be at least two. Just say "True" or "False". You do not have to provide any reasoning,

Problem 2 (10 points) In any input to the Stable Matching problem with n women and n men, what is the number of perfect matchings? Prove your answer.

Problem 3 (10 points) In class, we discussed how to implement the Gale-Shapley algorithm in $O(n^2)$ time using lists and two-dimensional arrays. However, we left out one important detail: how to maintain the set S of matched pairs of men and women. Let us remedy this gap in this problem. Describe a simple way to maintain S as the algorithm proceeds. In particular, you need to ensure that

(i) S is empty at the beginning of the algorithm,

(ii) When a free man m makes a proposal to a woman w, you can determine if w has a partner and his index in constant time, and

(iii) If w accepts the proposal from m, you can update S in constant time.

You may assume that all the men and all the women are indexed from 1 to n. *Note:* Do not use a hash map or a set data structure or anything like that. It is sufficient to use simple data structures such as lists and arrays.

Problem 4 (25 points) Consider the following algorithm to determine if a given positive interger n is prime. For each integer i between 2 and $\lfloor\sqrt{n}\rfloor$, check if i is a factor of n, i.e., if dividing n by i leaves a remainder of 0. If the answer is "yes" for any i, return that n is composite. Otherwise, return that n is prime.

What is the size of the input to this problem? Is it 1 because there is only one number in the input? Is it n because the input is n? It is not the first option, because the input may be an immensely large number which we cannot represent with a constant size. It is not the second option, since the *value* of the input is n, which is not the same as the *size* of the input. The correct way to think about the input size in this case is the *number of bits needed to represent n*.

Answer the following questions about this algorithm.

(a) (5 points) How many bits do we need to represent the number n? Just state the answer. You do not need to prove or explain your solution.

(b) (10 points) What is the running time $g(n)$ of the algorithm in terms of this input size? You may assume that we can check whether if i is a factor of n in $O(1)$, i.e., constant, time.

(c) (10 points) Let the answer to the first part be $f(n)$ (of course, you will have written something more specific). Prove that the running time $g(n) = \Omega(f(n)^d)$ for any positive value d. In other words, prove that the running time of this algorithm for testing primality is lower bounded by any arbitrary polynomial function of the input size. You can start from any statement in the textbook or the slides but be sure to explain how you derive your conclusion from the starting statement.

Note: It is likely that you will be able to prove part (c) only if you answer part (a) correctly.

Problem 5 (15 points) Solve exercise 5 in Chapter 2 (page 68) of "Algorithm Design" by Kleinberg and Tardos. In addition to what is stated in the problem, assume that $f(n)$ and $g(n)$ are increasing functions of n. If you decide that a statement is true, provide a short proof. Otherwise, provide a counterexample. *Note:* If you think one of the statements is true, you should not prove it for your own choices of $f(n)$ and $g(n)$. Your proof must hold for *any* pair of increasing functions $f(n)$ and $g(n)$ where $f(n)$ is $O(g(n))$.

Problem 6 (35 points) Solve exercise 8(a) in Chapter 2 (pages 69-70) of "Algorithm Design" by Kleinberg and Tardos. This problem involves stress-testing jars by throwing them off ladders by breaking as few bottles as possible.