1. Problem 3-3 part a from the CLRS text. You need not provide justifications for your order.

If you aren’t able to find a group, you can do this individually. While I encourage you to use LaTeX, it is not strictly required for this assignment. You may hand write these solutions.

Due Friday January 6.

2. Problem 3-4. For conjectures that are untrue, explain which reasonable class of functions the statement is true for. For example, you might say ”The statement is untrue in general, and here is an example. But the statement is true if the functions are strictly increasing, and here is why.

Due Monday January 9.

3. Consider the following definitions for \( f(n,m) = O(g(n,m)) \). State which definitions are logically equivalent. Fully justify your answers. Argue about which definition you think is the best one in the context where \( f \) and \( g \) are run times of algorithms and the input size is monotonically growing in \( n \) and \( m \) (for example \( n \) might be the number of vertices in a graph and \( m \) might be the number of edges in a graph).

(a) There exists positive constants \( c > 0, n_0, m_0 \) such that for all \( (n,m) \) where \( n \geq n_0 \) and \( m \geq m_0 \) it is the case that \( f(n,m) \leq c g(n,m) \)

(b) There exists positive constants \( c > 0, n_0, m_0 \) such that for all \( (n,m) \) where \( n \geq n_0 \) or \( m \geq m_0 \) it is the case that \( f(n,m) \leq c g(n,m) \)

(c) There exists a constant \( c > 0 \) such that
\[
\limsup_{n \to \infty} \limsup_{m \to \infty} \frac{f(n,m)}{g(n,m)} < c
\]

Note that this means that you first take the limit superior with respect to \( m \). The result will be a function of just \( n \). You then take the limit superior of this function with respect to \( n \). If you don’t know what limit superior means, you can just assume that the limit exists, in which case the limit and limit superior are the same.

(d) There exists a constant \( c > 0 \) such that
\[
\limsup_{m \to \infty} \limsup_{n \to \infty} \frac{f(n,m)}{g(n,m)} < c
\]

Note that this means that you first take the limit superior with respect to \( n \). The result will be a function of just \( m \). You then take the limit superior of this function with respect to \( m \). If you don’t know what limit superior means, you can just assume that the limit exists, in which case the limit and limit superior are the same. Note that on the surface that this definition is different than the last on in that the order that you take the limits is switched.

(e) There exists a constant \( c > 0 \) such that for all but finitely many pairs \( (m,n) \) it is the case that \( f(n,m) \leq c g(n,m) \).
Due Monday January 9.

4. Problem 4-3 from the CLRS text except parts d and f. Apply the Master Theorem (Theorem 4.1) whenever applicable. Otherwise, draw the recursive call tree and sum up the costs level by level.
Due Wednesday January 11

5. Problem 4-3 parts d and f. Use induction; You will need 2 inductive proofs, one for the upper bound and one for the lower bound.
Due Friday January 13.

6. Problem 4-6 parts d and e from the CLRS text.
Due Friday January 13.

7. Problem 8.1-3 from the CLRS text. Give an adversarial strategy and prove that it is correct.
Due Friday January 13.

8. Problem 8.1-4 from the CLRS text. Give an adversarial strategy and prove that it is correct. Explain the last sentence of the problem statement from the book, that is, explain why its not rigorous/correct to simply combine the lower bounds for the individual subsequences.
Due Friday January 13.

9. Problem 8-6 from the CLRS text.
   (a) For part a and b, it is essentially asking you to consider the adversarial strategy that answers so as to maximize the number of the original ways of merging two sorted lists that are consistent with the answer. You will likely find Stirling’s approximation for $n!$ useful.
   (b) For parts c and d, come up with a different adversarial strategy.
   (c) Explain why the bound that you get using the method proposed in parts a and b isn’t as good as the bound you get using an adversarial strategy. That is, in what way are you being too generous to the algorithm in parts a and b? In this generosity in the adversarial strategy or in its analysis?
Due Monday January 23

10. Consider the problem of determining whether a collection of real numbers $x_1 \ldots x_n$ is nice. A collection of numbers is nice iff the difference between consecutive numbers in the sorted order is at most 1. So 1.2, 2.7, 1.8 is nice, but 1.2, 2.9, 1.8 is not nice since the difference between 1.8 and 2.9 is more than 1. We want to show that every comparison based algorithm to determine if a collection of n numbers is nice requires $\Omega(n \log n)$ comparisons of the form $x_i - x_j \leq c$, where $c$ is some constant that the algorithm can specify. So if $c = 0$, this is a standard comparison.
Hint: This is similar to the lower bound for element uniqueness.

Another Hint: Consider the \((n - 1)!\) permutations \(\pi\) of \(\{1, \ldots, n\}\) where \(\pi(1) = 1\), and the corresponding points in \(n\) dimensional space. Note that all \((n - 1)!\) of these points are nice. Show the midpoint of any pair of these nice points is not nice. Then explain how to use this fact to give an adversarial strategy showing the \(\Omega(n \log n)\) lower bound.

Due Monday January 23.

11. Consider a setting where you have two computer networking routers \(A\) and \(B\). Each router has collected a list \(L_A\) and \(L_B\) of IP source addresses for the packets that have passed through the router that day. An IP address is \(n\) bits, and thus there are \(2^n\) possible IP addresses. Now the two routers want to communicate via a two-way channel to whether there was some source that sent a packet through one of the routers, but not the other. So more precisely, at the end of the protocol each router should commit to a bit specifying the answer to this question, and the bits for both routers should be correct. You can assume that each router can send a bit per unit time, and that a bit sent on the channel is guaranteed to arrive on the other end in one time unit. We want to consider protocols for accomplishing this goal.

(a) (warm-up) Consider the following protocol: \(A\) sends to \(B\) the list of all of the IP source addresses that it has seen; \(B\) compares \(A\)'s list to its list, and then \(B\) sends \(A\) a 0 bit if the lists are identical and a 1 bit otherwise. Show that uses protocol about uses \(n2^n + 1\) bits in the worst case.

(b) (warm-up) Give a protocol that uses \(2^n + O(1)\) bits in the worst case. Another trivial warmup problem.

(c) (warm-up) Show that there is no protocol that can solve this problem without exchanging any bits.

Hint: It's obvious that this is true. That isn't the point. The point is to understand what arguments you have to make to make this formally correct.

(d) Show that there is no protocol that can solve this problem that involves \(A\) sending one bit to \(B\). And no more bits are exchanged.

Hint: Again it's obvious that this is true. Again that isn't the point. Again the point is to understand what arguments you have to make to make this formally correct. Ask yourself how should the adversarial strategy should decide whether this first bit is a 0 or a 1?

(e) Show that there is no protocol that can solve this problem that involves \(A\) sending one bit to \(B\) and \(B\) replying with one bit to \(A\). And no more bits are exchanged.

Hint: Again its obvious that this is true. Again that isn’t the point. Again the point is to understand what arguments you have to make to make this formally correct.

(f) Prove that every protocol for this problem must send \(2^n - O(1)\) bits for its worst case instance. Of course your argument should involve an adversarial argument.
Assume that you have a computer networking router that sees a stream of $k$ IP packets, each with a source IP address consisting of $n$ bits. The router sees a packet, optionally records some information in memory, and then passes the packet on, then sees the next packet, optionally records some information in memory, and then passes that packet on, etc. The router's goal is to always know a source IP address that it has seen most frequently to date. (The most obvious way to accomplish this is to keep a count for each IP source address seen to date.) Show that if $k \leq 2^n$ then every algorithm must use $\Omega(k)$ bits of memory.

Hint: This is an "easy" consequence of the previous subproblem, provided that you think about it the right way. Assume that you had a method that solved this problem using $o(k)$ bits of memory. Explain how to use this method to get an algorithm for the previous subproblem that uses less than $2^k$ bits of communication.

Due Wednesday January 25