MATH 416, PROBLEM SET 1

Comments about homework.

- Solutions to homework should be written clearly, with justification, in complete sentences. Your solution should resemble something you'd write to teach another student in the class how to solve the problem.
- You are encouraged to work with other 416 students on the homework, but solutions must be written independently. Include a list of your collaborators at the top of your homework.
- You should submit your homework on Gradescope, indicating to Gradescope where the various pieces of your solutions are. The easiest (and recommended) way to do this is to start a new page for each problem.
- Attempting and struggling with problems is **critical** to learning mathematics. Do not search for published solutions to problems. I don't have to tell you that doing so constitutes academic dishonesty; it's also a terrible way to get better at math.

  If you get stuck, ask someone else for a hint. Better yet, go for a walk.

Warning. These are not necessarily model solutions; they are meant to help you understand the problems you didn't totally solve and maybe to give you alternative solutions. Sometimes I will give less or more detail here than I would expect from you.

Problem 1. Arrange the following list of functions in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$. Give a brief explanation for each pair of consecutive functions.

$$g_1(n) = 2^{\sqrt{\log n}} \qquad g_5(n) = n^{\log n}$$
$$g_2(n) = 2^n \qquad g_6(n) = 2^{2^n}$$
$$g_3(n) = n(\log n)^3 \qquad g_7(n) = 2^{n^2}$$
$$g_4(n) = n^{4/3}$$

Solution. (a) The correct order is

$$g_1 \ll g_3 \ll g_4 \ll g_5 \ll g_2 \ll g_7 \ll g_6.$$

Why?

- $g_1 \ll g_3$: $\log(g_1(n)) = \sqrt{\log n}$ is (for sufficiently large n) smaller than $\log n$, which in turn is smaller than $\log(g_3(n)) = \log n + 3\log(\log n)$. Exponentiating each side of this inequality gives $g_1(n) < g_3(n)$ for such n.
- $g_3 \ll g_4$: For all sufficiently large n, we have $\log n < n^{1/9}$ (a special case of a fact from class), which implies $n(\log n)^3 < n^{4/3}$.
- $g_4 \ll g_5$: The inequality $n^{4/3} < n^{\log n}$ is true for all n for which $4/3 < \log n$, i.e., $2^{4/3} < n$.
- $g_5 \ll g_2$: Notice that $g_5(n) = 2^{(\log n)^2}$.
- $g_2 \ll g_7$: $2^n \leq 2^{n^2}$ for all $n \in \mathbb{N}$.
- $g_7 \ll g_6$: Follows from the fact that $n^2 < 2^n$ for sufficiently large n.

Problem 2. Assume that $f(n)$ and $g(n)$ are functions such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether it is true or false and provide either a proof or counterexample:

(a) $\log_2 f(n)$ is $O(\log_2 g(n))$.
(b) $2^{f(n)}$ is $O(2^{g(n)})$.
(c) $f(n)^2$ is $O(g(n)^2)$.

Solution. (i) False, but for the stupid reason that $\log g(n)$ could be 0 for many n. For example, take $f(n) = 2$, $g(n) = 1$. Then f is $O(g)$ yet $1 = \log f(n) \not\leq c \log g(n) = 0$.

It's easy to correct this problem, though. For example, if we simply require that $g(n) \geq 2$ for all $n \geq n_1$, then the statement holds. Since $f(n) \leq cg(n)$ for all $n \geq n_0$, we have

$$\log f(n) \leq \log g(n) + \log c \leq (\log c)(\log g(n))$$

for $n \geq \max(n_0, n_1)$.

(ii) False. While something is true on the level of numbers ($f(n) \leq g(n)$ implies $2^{f(n)} \leq 2^{g(n)}$), the problem is the constant multiple. Take $f(n) = 2n$ and $g(n) = n$. Then f is $O(g)$ (indeed, f is $\Theta(g)$), but $2^{f(n)} = 4^n$ while $2^{g(n)} = 2^n$.

(iii) True. If n_0 and c are such that $f(n) \leq cg(n)$ for all $n \geq n_0$, then $(f(n))^2 \leq c^2(g(n))^2$ for all $n \geq n_0$. □

Problem 3. Prove that $o(f(n)) \cap \omega(f(n))$ is the empty set. (See WS3 for the definition of little o and little ω.)

Solution. Suppose $g(n) \in o(f(n))$. Suppose $C > 0$ is an arbitrary real number and let $N_1 \in \mathbb{N}$ be arbitrary. By definition of $o(f(n))$, there exists $N_0 \in \mathbb{N}$ such that

$$g(n) < Cf(n)$$

for all $n \geq N_0$. Therefore if $n \geq \max\{N_0, N_1\}$, it is **not** possible that

$$g(n) > Cf(n).$$

So $g(n)$ is **not** $\omega(f(n))$ and thus the intersection $o(f(n)) \cap \omega(f(n))$ is the emptyset.

Problem 4. Suppose that H and S together with preference arrays are given. For $h \in H$ and $s, t \in S$, define $\max_h(s, t)$ to be the higher of s and t in h's preference list. Show that if M and P are two stable matchings for this instance of the stable-matching problem, then

$$M \vee P = \{(h, \max_h(s, t)) | (h, s) \in M, (h, t) \in P\}$$

is also a stable matching. (In particular, you must show that it is a matching.)

Solution. Suppose toward a contradiction that $M \vee P$ is not a perfect matching. Then there must be $h \neq h'$ in H and $s, s', t, t' \in S$ for which

- $(h, s), (h', s') \in M$,
- $(h, t), (h', t') \in P$, and
- $\max_h(s, t) = \max_{h'}(s', t')$.

Since M and P are each matchings, $s \neq s'$ and $t \neq t'$. So without loss of generality (i.e., by renaming people if necessary), we may assume that

$$s = \max_h(s, t) = \max_{h'}(s', t') = t'.$$

Then s prefers either h or h'. If s prefers h, then (s, h) is a blocking pair in P; if s prefers h', then (s, h') is a blocking pair in M. Either way, contradiction.

Stability is easier to check: with the notation as above, if (h, s') were a blocking pair in $M \vee P$, then h would prefer s' to either s or t, which implies that (h, s') is also a blocking pair in M, contradiction. (M because that's the matching in which s' is matching with their partner in $M \vee P$.) □

Problem 5 (Examples of stable matchings).

(a) Give preference lists (for a fixed small n, say $2 \leq n \leq 10$) and a stable matching in which no hospital or student is matched with their first choice.

(b) Now for any $n \geq 2$, give preference lists and a stable matching in which every student is matched with their first choice but every hospital is matched with its last choice.

(c) Show that for every $k \geq 1$ there are preference lists for $n = 2k$ hospitals and $2k$ students that admit at least 2^k stable matchings. (So the number of stable matchings can grow exponentially in the number of parties.)

(Hint: First solve the problem for $k = 1$. Then for the general case try to paste together many 'independent' copies of your solution for $k = 1$.)

Solution. (a) Consider these preference lists:

X	A	B	C
Y	B	C	A
Z	C	A	B

A	Y	Z	X
B	Z	X	Y
C	X	Y	Z

The matching that pairs each party with their second choice (i.e., (XB, YC, ZA)) is stable.

(b) A useful observation: if all the students have different first choices, then the matching that pairs every student with their first choice is stable, no matter what the hospitals' preferences are. So make h_k the first choice of s_k for every k, and make s_k the last choice of h_k for every k. The rest of the preferences don't matter. And $\{(h_k, s_k) | k \leq n\}$ is a stable matching with the requested property.

(c) The idea is to paste together instances of the solution to the problem for $k = 1$:

X	A	B
Y	B	A

A	Y	X
B	X	Y

For these preference lists, either of the two possible perfect matchings is stable. (Why? because either both students get their first choice or both hospitals get their first choice.) Now let

$$H = \{X_1, Y_1, X_2, Y_2, \ldots, X_k, Y_k\}, \quad S = \{A_1, B_1, A_2, B_2, \ldots, A_k, B_k\}$$

and consider the preference lists

X_1	A_1	B_1	$\cdots$
Y_1	B_1	A_1	$\cdots$
X_2	A_2	B_2	$\cdots$
Y_2	B_2	A_2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	
X_k	A_k	B_k	$\cdots$
Y_k	B_k	A_k	$\cdots$

A_1	Y_1	X_1	$\cdots$
B_1	X_1	Y_1	$\cdots$
A_2	Y_2	X_2	$\cdots$
B_2	X_2	Y_2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	
A_k	Y_k	X_k	$\cdots$
B_k	X_k	Y_k	$\cdots$

(The rest of the preference lists can be filled in in any way.) For any function $t\colon k \to 2$ define a matching M_t by

$$M_t := \{(X_l, A_l), (Y_l, B_l) | t(l) = 0\} \cup \{(X_l, B_l), (Y_l, A_l) | t(l) = 1\}.$$

This matching is stable, since every party is matched with their first or second choice, and a party p is matched with their second choice iff p's first choice q is paired with q's first choice. (So there are no unstable pairs.) □

Problem 6.

(a) Prove that in a stable matching produced by the GS Algorithm, at most one hospital is paired with its last choice. (Contrast with Problem ????.)

(b) For every n, give an example of preference lists such that in the stable matching produced by the GS Algorithm, there is exactly one hospital matched with its last choice and each of the other $n-1$ hospitals is matched with its second-to-last choice.

(Hint: One approach is to make one student the last choice of every hospital.)

(c) Prove that, for every $n \geq 1$, there are preference lists for $|H| = |S| = n$ for which the `while` loop in the GS algorithm must iterate $n^2 - n + 1$ times, so that the running time of the algorithm is $\Theta(n^2)$.

(Hint: This is not unrelated to the previous part.)

Solution. (a) Recall that in the GS Algorithm, once a student is matched, they remain matched (though perhaps to a different hospital) throughout the remaining execution of the algorithm. Recall also that, in the course of the execution of the GS Algorithm, the partial matching constructed so far is always a matching, meaning that no two students are matched to the same hospital and no two hospitals are matched with the same student.

If, in the execution of the algorithm, hospital h makes an offer to its last choice, then it has made offers to all $n-1$ other students, who are therefore matched. There are $n-1$ hospitals other than h, so those $n-1$ hospitals must be matched to the other $n-1$ students. So, after matching h with its last choice, the algorithm halts, because all the hospitals are matched. This means that the first time a hospital makes an offer to its last choice must also be the last, so this happens at most once.

(b) There are multiple solutions here; one approach is to make one student s_1 last on all the hospitals' preference lists, so that the n hospitals are, in effect, competing for only $n-1$ students. Then the students' preference lists can be arranged to make this competition very inefficient, meaning that it takes many proposals to settle on a stable matching.

h_1	s_2	s_3	$\cdots$	s_{n-1}	s_n	$\mathbf{s_1}$
h_2	s_3	s_4	$\cdots$	s_n	$\mathbf{s_2}$	s_1
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
h_{n-1}	s_n	s_2	$\cdots$	s_{n-2}	$\mathbf{s_{n-1}}$	s_1
h_n	s_2	s_3	$\cdots$	s_{n-1}	$\mathbf{s_n}$	s_1

s_1						
s_2	h_2	h_3	$\cdots$	h_{n-1}	h_n	h_1
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
s_{n-1}	h_{n-1}	h_n	h_1	h_2	$\cdots$	h_{n-2}
s_n	h_n	h_1	h_2	$\cdots$	h_{n-2}	h_{n-1}

The preference list of h_k for $k = 1, \ldots, n-1$ begins with a cyclic permutation of $(s_2, \ldots, s_n)$ from s_{k+1} to s_k, and lists s_1 last. The preference list for h_n is identical to the preference list for h_1.

The preference list for s_k, $k = 2, \ldots, n$, is a cyclic permutation of $(h_1, \ldots, h_n)$ from h_k to h_{k-1}. The preference list for s_1 can be anything.

With these preference lists, the offers made in the execution of the algorithm are as follows. In order, h_k makes an offer to s_{k+1}, its first choice, until h_n makes an offer to s_2, forcing h_1 to make an offer to s_3 since s_2 prefers h_n to h_1. Then all the hospitals (in order) make offers to their second choices, and then their third, and so on, until all but h_n have made offers to their second-to-last choices. Finally, h_n makes an offer to s_n, who accepts it over h_1, forcing h_1 to make an offer to s_1 and ending the algorithm.

(It is possible, though more difficult, to prove without discussing the execution of the algorithm that s_k is h_k's highest-ranked valid partner for $k = 2, \ldots, n-1$ and conclude that the matching given is the hospital-optimal one.)

(c) The number of iterations of the `while` loop is the number of offers, which in the previous example must be $(n-1)^2 + n = n^2 - n + 1$. (Hospitals $h_2, \ldots, h_n$ each make $n-1$ offers, while h_1 makes n offers.) This proves that the worst-case running time of the algorithm is $\Omega(n^2 - n + 1) = \Omega(n^2)$, so it is $\Theta(n^2)$. □

Problem 7. Recall that the Gale–Shapley Algorithm produces the unique hospital-optimal, student-pessimal stable matching. By falsifying their preference list, can a student end up (after running the algorithm) with a better position than they would with their true preference list? Give a proof or an example.

Solution. Yes! Consider the following preference lists:

X	B	A	C
Y	A	B	C
Z	A	B	C

A	X	Y	Z
B	Y	X	Z
C	Y	X	Z
A^*	X	Z	Y

where A^* indicates the false preferences that A will report. With the true preference lists, the GS Algorithm runs as follows:

X	B		
Y		A	
Z			ABC,

in the end producing the matching (XB, YA, ZC), in which A gets their second choice. The idea is that A should falsely claim to prefer Z to Y in order to invalidate Y as a possible partner, so that they end up with X, their first choice. With the false preferences, the algorithm runs as follows:

X	~~B~~			A	
Y		A			B
Z			~~A~~		C

and A ends up with their top choice X instead of their second choice Y. □

Problem 8. Suppose that we are given preference lists for $|H| = |S| = n$. Prove that there is no perfect matching, stable or otherwise (!), in which every hospital is matched with a student it strictly prefers to the student matched with it by the Gale–Shapley Algorithm.

(Hint: Consider the final step of the algorithm before it stops.)

Solution. Let M^* be the (hospital-optimal) stable matching produced by the GS Algorithm. Suppose toward a contradiction that M is a perfect matching in which every hospital is paired with a student that it prefers to its match in M^*. Let $(h, s) \in M^*$ be the last pair produced in the execution of the GS Algorithm. Then s does not reject any offers during the execution of the algorithm, since the algorithm halts after the final student receives their first offer. Let k be the partner of s in M, and let t be the partner of k in M^*. So (by choice of M) k prefers s to t, which means that k must have made an offer to s and been rejected by s, a contradiction. □

Problem 9. A past MATH 416 student proposed the following algorithm for the stable-matching problem.

> Start with an arbitrary perfect matching M of hospitals and students. (For instance, $\{(h_1, s_1), (h_2, s_2), \ldots, (h_n, s_n)\}$.)
>
> Search for pairs of pairs $(h, s), (h', s') \in M$ for which (h, s') is a blocking pair; if you find one, then rematch these four parties: i.e., replace the pairs $(h, s), (h', s')$ with $(h, s'), (h', s)$.
>
> Repeat until there are no more blocking pairs.

This algorithm will certainly succeed if it halts. Prove that, unfortunately, it need not halt. (Hint: You should be able to produce an example with $|H| = |S| = 3$.)

Solution. The preference lists are as follows.

X	B	A	C
Y			
Z	A	B	C

A	X	Z	Y
B	Z	X	Y
C			

(The preferences of Y and of C don't matter.) Here is a sequence of unstable matchings, with all instabilities listed to the right:

S_1 :	$\mathbf{X}A$	$Y\mathbf{B}$	ZC	$\mathbf{XB}$, ZB
S_2 :	$X\mathbf{B}$	YA	$\mathbf{Z}C$	ZA, $\mathbf{ZB}$
S_3 :	XC	$Y\mathbf{A}$	$\mathbf{Z}B$	XA, $\mathbf{ZA}$
S_4 :	$\mathbf{X}C$	YB	$Z\mathbf{A}$	$\mathbf{XA}$, XB
$S_5 = S_1$:	XA	YB	ZC	✓

This means that if the described algorithm chose the blocking pairs in the order indicated, it would never terminate. □