- The assignment is due at Gradescope on Friday, January 20 at 5:00p.
- You can either type your homework using LaTex or scan your handwritten work. We will provide a LaTex template for each homework. If you writing by hand, please fill in the solutions in this template, inserting additional sheets as necessary. This will facilitate the grading.
- You are permitted to study with up to 2 other students in the class (any section) and discuss the problems; however, *you must write up your own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Similarly, please list any other source you have used for each problem, including other textbooks or websites. *Consulting problem solutions on the web is not allowed.*
- *Show your work.* Answers without justification will be given little credit.

PROBLEM 1 (25 POINTS) *Answer the questions below using the following preference lists:*

Group A's preference lists (from most preferred to least preferred):

a_1: b_1, b_4, b_3, b_2

a_2: b_4, b_3, b_1, b_2

a_3: b_1, b_2, b_4, b_3

a_4: b_3, b_1, b_2, b_4

Group B's preference lists (from most preferred to least preferred):

b_1: a_4, a_2, a_3, a_1

b_2: a_1, a_4, a_2, a_3

b_3: a_3, a_1, a_4, a_2

b_4: a_4, a_3, a_1, a_2

(a) Run the Gale-Shapley algorithm with group A making the proposals, to obtain a stable matching. For your answer (and for your answer to part b as well), at each step please give the proposal that is made and whether or not this proposal is accepted. You should also give the final matching you obtain.

(b) Now run the Gale-Shapley algorithm with group B making the offers to obtain another stable matching. Which people are happier in this new stable matching (compared to the stable matching found in part a)?

(c) What other stable matching(s) are there, if any? Note: For full credit you should show that you have indeed found all of the possible stable matchings. (Careful thought should allow you to avoid exhaustive search over all perfect matchings.)

Solution: (a)

By GS algorithm, order of proposing does not matter. Therefore, let the lowest unmatched member of group A propose.
Step 1: $a1$ proposes to first preference, $b1$. As $b1$ is not engaged, match($a1, b1$).
Unmatched: $a2, a3, a4$
Step 2: $a2$ proposes to first preference, $b4$. As $b4$ is not engaged, match($a2, b4$).
Unmatched: $a3, a4$
Step 3: $a3$ proposes to first preference, $b1$. As $b1$ prefers $a3$ to $a1$, unmatch($a1, b1$), then match($a3, b1$). Unmatched: $a1, a4$
Step 4: $a1$ proposes to second preference, $b4$. As $b4$ prefers $a1$ to $a2$, unmatch($a2, b4$), then match($a1, b4$). Unmatched: $a2, a4$
Step 5: $a2$ proposes to second preference, $b3$. As $b3$ is not engaged, match($a2, b3$).
Unmatched: $a4$

Step 6: $a4$ proposes to first preference, $b3$. As $b3$ prefers $a4$ to $a2$, unmatch$(a2, b3)$, then match$(a4, b3)$. Unmatched: $a2$
Step 7: $a2$ proposes to third preference, $b1$. As $b1$ prefers $a2$ to $a3$, unmatch$(a3, b1)$, then match$(a2, b1)$. Unmatched: $a3$
Step 8: $a3$ proposes to second preference, $b2$. As $b2$ is not engaged, match$(a3, b2)$. Unmatched: None. GS terminates. Stable matching is produced through GS algorithim
Final matching: $(a1, b4), (a2, b1), (a3, b2), (a4, b3)$

(b)

By GS algorithm, order of proposing does not matter. Therefore, let the lowest unmatched member of group B propose.
Step 1: $b1$ proposes to first preference, $a4$. As $a4$ is not engaged, match$(a4, b1)$. Unmatched: $b2, b3, b4$
Step 2: $b2$ proposes to first preference, $a1$. As $a1$ is not engaged, match$(a1, b2)$. Unmatched: $b3, b4$
Step 3: $b3$ proposes to first preference, $a3$. As $a3$ is not engaged, match$(a3, b3)$. Unmatched: $b4$
Step 4: $b4$ proposes to first preference, $a4$. As $a4$ prefers $b1$, $b4$ is rejected. Unmatched: $b4$
Step 5: $b4$ proposes to second preference, $a3$. As $a3$ prefers $b4$ to $b3$, unmatch$(a3, b3)$, then match$(a3, b4)$. Unmatched: $b3$
Step 6: $b3$ proposes to second preference, $a1$. As $a1$ prefers $b3$ to $b2$, unmatch$(a1, b2)$, then match$(a1, b3)$. Unmatched: $b2$
Step 7: $b2$ proposes to second preference, $a4$. As $a4$ prefers $b1$ to $b2$, $b2$ is rejected. Unmatched: $b2$
Step 8: $b2$ proposes to third preference, $a2$. As $a2$ is not engaged, match$(a2, b2)$. Unmatched: None. GS terminates with stable matching
Final matching: $(a1, b3), (a2, b2), (a3, b4), (a4, b1)$

Happier people: $b1$ ($a4 > a2$), $b2$ ($a2 > a3$), $b3$ ($a1 > a4$), $b4$ ($a3 > a1$). Therefore all of group B is happier when they engage proposals (in part b vs. part a).

(c)

In order to find all other stable matchings, first remove all invalid partners. Because the GS algorithm provides the *best* partner for every value in each respective group (as proved in class), then we can take the stable matchings from part (a) and (b) and remove all possible pairs ranked higher in the proposing group than the one given. This is because any matching with those pairs will not be a stable matching. For example, in (a), $a1$ is matched with $b4$, it's second preference. Therefore, it is impossible to have a stable matching where $a1$ matches with $b1$, it's first preference, because GS offers the best stable matching for every proposing person. Using this logic we remove some invalid pairs

for group A proposing and group B proposing.
We have the following potential pairs for Group A proposing, based on the preferences that are $\leq$ than the one matched with in GS.

a_1: b_4, b_3, b_2

a_2: b_1, b_2

a_3: b_2, b_4, b_3

a_4: b_3, b_1, b_2, b_4

Using the same approach, we have the following potential pairs for Group B proposing, based on the preferences that are $\leq$ than the one matched with in GS with the opposite group proposing.

b_1: a_4, a_2, a_3, a_1

b_2: a_2, a_3

b_3: a_1, a_4, a_2

b_4: a_3, a_1, a_2

Because a matching pair must be symmetrical in nature (if (a, b) is unstable than (b, a) is also unstable), we can combine the two potential pair lists and take the cases where the pairs occur in both cases (if a pair cannot produce a stable matching in either list, it cannot produce a stable matching in general) to further narrow our results by eliminating the impossible pairs. We have the following list

b_1: a_4, a_2

b_2: a_2, a_3

b_3: a_1, a_4

b_4: a_3, a_1

From these possible pairings we create the following two potential matchings. This is possible because each respective a occurs exactly twice in the set of potential pairs. To achieve this we follow the methodology: Pick a possible value for a pair, (for example for b_1, pick a_4). To obtain a stable matching, by nature, a_4 cannot be paired with any other values. Therefore we remove all other instances of a_4, and as a matching must be complete, b_3 must be matched with a_1 as the only possible values are a_1, a_4. We can apply this logic to each of the four values respectively to achieve the following matching: $(a_1, b_3), (a_2, b_2), (a_3, b_4), (a_4, b_1)$

We apply the same logic as above, though for b_1, we pick a_2 initially instead of a_4. This produces the following matching: $(a_1, b_4), (a_2, b_1), (a_3, b_2), (a_4, b_3)$
Because of the composition of these specific list preferences and the definition of a stable matching, after picking a value from a potential pair in the matching, the removal of that value from another prefrence list will lead to one of the two stable matchings above.
As these two matchings are identical to the matchings that we achieve through the GS algorithm in (a) and (b), we know that the two matchings are stable and that there are no other possible stable matchings.

PROBLEM 2 (25 POINTS) *Solve exercise 8 in Chapter 1 in the Kleinberg-Tardos textbook (on "truthfulness" in Gale-Shapley). Note: here and elsewhere, Kleinberg-Tardos use the language of men and women. You are always free to change the "storytelling" aspect if you wish, as long as the mathematical essence of the problem remains the same.*

Solution: Take the following preferences for Group A (proposed to):

a_1: b_3, b_1, b_2

a_2: b_1, b_3, b_2

a_3: b_1, b_2, b_3

Preferences for Group B (proposing):

b_1: a_1, a_2, a_3

b_2: a_1, a_3, a_2

b_3: a_2, a_1, a_3

Gale Shapely provides the following:
Step 1 Match(a_1,b_1)
Step 2 b_2 proposes to a_1 and is rejected
Step 3 Match(a_3,b_2)
Step 4 Match(a_2,b_3)
Terminate with following matching: (a_1,b_1),(a_2,b_3), (a_3,b_2)
Let a_1 (a member of the proposed group) lie about their preferences, to be

a_1: b_3, b_2, b_1

Run the Gale Shapley algorithm again with this one changed preference list to provide the following:
Step 1 Match(a_1,b_1)
Step 2 Match(a_1,b_2), unmatch(a_1,b_1)
Step 3 Match(a_2,b_1)
Step 4 b_3 proposes to a_2 and is rejected
Step 5 Match(a_1,b_3), unmatch(a_1,b_2)
Step 6 Match(a_3,b_2)
Terminate with following matching: (a_1,b_3),(a_2,b_1), (a_3,b_2)
As $b_3 > b_1$ for a_1, a member of the proposed group improved their end outcome by lying.

PROBLEM 3 (25 POINTS) *In the Gale-Shapley algorithm with the A group proposing to the B group, does there exist an $n = |A| = |B| > 1$, a set of preference lists for any $a \in A$ and b in$B$, and an execution of the algorithm, such that every $a \in A$ ends up matched with its least-preferred b? Please, either give an example where this happens or prove it is not possible.*

Proof that it is not possible by contradiction.
Assume the Gale-Shapley algorithm has produced a stable matching such that every $a \in A$ ends up matched with its least-preferred b
If the Gale-Shapley algorithm has produced a stable matching, than the algorithm has terminated.
With size of groups $n = |A| = |B| > 1$, while the algorithm is running then there must be some member of group B who has not been proposed to yet, u. If there were no members of group B who had not been proposed to then every member of group A would be matched (as $|A| = |B|$) and the algorithm would terminate on this condition. Therefore, there must alo be some member of group A who is free and hasn't yet proposed to every member in B, v. Let (u, v) be the final match for the algorithm to terminate.
In order for every $a \in A$ to end up matched with its least-preferred b, than every $a \in A$ must have proposed to every $b \in B$ and then matched with their least-preferred b.
However, because Gale-Shapley algorithm terminates with v proposing to u and u is free before v proposes, then u has not been proposed to yet (any free member of B will accept any proposal), so it is impossible that every $a \in A$ is matched with its least-preferred b, as they all would have had to propose to u at some point, and no one proposed to u before v.
Therefore it is not possible that with the A group proposing to the B group, there exists an $n = |A| = |B| > 1$, a set of preference lists for any $a \in A$ and b in$B$, and an execution of the Gale-Shapley algorithm algorithm, such that every $a \in A$ ends up matched with its least-preferred b.

PROBLEM 4 (25 POINTS)

(a) *Identify or recall a "simple" (closed-form) formula for the value* $V_n = \sum_{i=1}^{n} i$. *Prove your formula is correct for all integer* $n \geq 1$ *by an explicit inductive proof using induction on the parameter* n.

(b) *Suppose* $f(n)$ *is a function from positive integers to positive integers. We don't know exactly which function it is, but we assume it is true that, for each* $n > 1$, *we have (using the "ceiling function")*

$$f(n) \;\leq\; 2 \cdot f(\lceil n/2 \rceil) + 10 \cdot n \; .$$

Use induction to prove that, for all integers n *of form* $n = 2^k$, *we have* $f(n) \leq C \cdot n \cdot \log_2 n$ *for some absolute constant* $C > 0$. *You are free to choose* C *but must prove that your choice works. Note that the conclusion can be shown for all* n *(not just powers of 2), but we're making the analysis a little easier.*

Solution:
(a)

Proof by Induction
Proposed formula for V_n: $\sum_{i=1}^{n} i = \frac{(n(n+1))}{2}$ for all integers $n \geq 1$

Base step: $n = 1$. $\sum_{i=1}^{1} = 1$. $\frac{(1(1+1))}{2} = 1$. Therefore for $n = 1$ $\sum_{i=1}^{1} = \frac{(1(1+1))}{2}$ and base step is true.

Hypothesis: Assume $n = k$ such that $\sum_{i=1}^{k} i = \frac{(k(k+1))}{2}$ for all integers $k \geq 1$

Induction: We want to show that $n = k + 1$ such that
$\sum_{i=1}^{k+1} i = \frac{((k+1)((k+1)+1))}{2}$ for all integers $k \geq 1$

$\frac{((k+1)((k+1)+1))}{2} = \sum_{i=1}^{k+1} i$
$\frac{k^2+3k+2}{2} = \sum_{i=1}^{k+1} i$
$\frac{k(k+1)}{2} + \frac{2k+2}{2} = \sum_{i=1}^{k+1} i$
$\sum_{i=1}^{k} i + \frac{2k+2}{2} = \sum_{i=1}^{k+1} i$ (Apply Hypothesis and substitute)
$\sum_{i=1}^{k} i + k + 1 = \sum_{i=1}^{k+1} i$
$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k+1} i$ (Combine terms)
Therefore $V_n = \frac{(n(n+1))}{2}$ for all integers $n \geq 1$

(b)

Proof by Induction
We want to show that $f(n) \leq C \cdot n \cdot \log_2 n$ for any 2^k. Therefore we want to

show $f(2^n) \leq C \cdot 2^n \cdot n$ for any $n \geq 1$. This is because $f(n)$ is a positive to positive function and there is a definition conflict for $n < 1$.
Proving $f(2^n) \leq 2^n \cdot f(1) + (10 \cdot 2^n \cdot n)$ for any $n \geq 1$ to help solve previous statement.
Base case: $f(2^1) \leq 2^1 \cdot f(1) + 10 \cdot 2^1 \cdot 1$. This is given by applying ceiling function for $n = 1$.
Inductive Hypothesis: Assume $f(2^n) \leq 2^n \cdot f(1) + (10 \cdot 2^n \cdot n)$ for arbitrary n.
Induction step: Proving for $n + 1$.

$$\begin{aligned}
\text{By ceiling function: } f(2^{n+1}) &\leq 2f(\frac{2^{n+1}}{2}) + 10 \cdot 2^{n+1} \\
f(2^{n+1}) &\leq 2f(2^n) + 10 \cdot 2^{n+1} \\
\text{Inductive hypothesis: } f(2^{n+1}) &\leq 2(2^n f(1) + 10 \cdot 2^n \cdot n) + 10 \cdot 2^{n+1} \\
f(2^{n+1}) &\leq 2^{n+1} f(1) + 10 \cdot 2^{n+1} \cdot n + 10 \cdot 2^{n+1} \\
f(2^{n+1}) &\leq 2^{n+1} f(1) + 10 \cdot 2^{n+1}(n+1)
\end{aligned}$$

Therefore we have $f(2^{n+1}) \leq 2^{n+1} f(1) + 10 \cdot 2^{n+1}(n+1)$ for any $n \geq 1$, proving $f(2^n) \leq 2^n \cdot f(1) + (10 \cdot 2^n \cdot n)$ for any $n \geq 1$.
Now to prove $f(2^n) \leq C \cdot 2^n \cdot n$ for any $n \geq 1$, let $C = 10 + f(1)$. $0 < C$ as $f(1)$ is positive. Then for all $n \geq 1$,

$$\begin{aligned}
\text{As proved previously: } f(2^n) &\leq 2^n f(1) + 10 \cdot 2^n \cdot n \\
\text{As } n \geq 1\text{: } f(2^n) &\leq 2^n f(1) \cdot n + 10 \cdot 2^n \cdot n \\
f(2^n) &\leq 2^n \cdot n \cdot (f(1) + 10) \\
\text{Substitute } C\text{: } f(2^n) &\leq 2^n \cdot n \cdot C
\end{aligned}$$

Therefore we have $f(2^n) \leq 2^n \cdot n \cdot C$ which proves $f(n) \leq C \cdot n \cdot \log_2 n$ for any 2^k.

PROBLEM 5 (0 POINTS, NOT GRADED) *Do, but* ***do not turn in,*** *the following review exercises about big-O notation.*

(a) *Describe each of the following functions* $T : \mathbb{N} \to$ *mathbbR*$^+$ *using big O notation. For full credit, the big-O expression should be as simple as possible (to a reasonable observer). Note: For this problem it is sufficient to describe the upper bound for each of these functions. For example, if we had* $T(n) = 10000$*, we could say that* $T(n)$ *is both* $\Omega(1)$ *and* $O(1)$ *(a condition expressed as* $T(n) = \Theta(n)$*) but it is sufficient to say that* $T(n)$ *is* $O(1)$*.*

1. $T(n) = 5n\log_2(n) + 2n + 3$
2. $T(n) = 20n + n^2$
3. $T(n) = 100 + 2\sqrt{n} + 7\log_2(n)$
5. $T(n) = 10n^2 \cdot 4^n$

(b) *Prove that for functions* $f, g : \mathbb{N} \to \mathbb{R}^+$*,* $f(n)$ *is* $O(g(n))$ *if and only if*

$$\exists n_0, C : \forall n \geq n_0, \log(f(n)) \leq \log(g(n)) + C$$

(c) *Order the following bounds from smallest to largest:* $O(n^2)$*,* $O(2^n)$*,* $O(1)$*,* $O(\log_2(n))$*,* $O(n\log_2(n))$*,* $O(\sqrt{n})$*,* $O(n^{\log_2(n)})$*,* $O(n)$*,* $O(n^{\sqrt{n}})$*,* $O((\log_2(n))^{10})$*.* *[***Hint:** *it may help to use part (b).]*

Extra Space for your solution