Due online Monday, February 19th at 11:59 pm

Please reference the course infosheet for the complete homework/collaboration policy. Highlights below:

- You must write up your solutions by yourself, without any collaborators or external references.
- Unless otherwise stated, you may collaborate with other students and consult external references.
- You must list all collaborators and external references consulted and upload it as a separate file to Mechanical TA (do not include collaborators in your problem submission).
- Some problems will be marked as no collaboration problems. This is to make sure you have experience solving a problem start-to-finish by yourself in preparation for the midterms/final.
- Please upload your solutions as separate PDF files to Mechanical TA, and also to Dropbox (to be safe). The name of your file will be visible to graders, so if you would like to remain anonymous to graders, give your PDF a generic name.

For this problem set, you may always assume that the number of students and universities are the same, and each university has capacity one.

Problem 1: Unique Stable Matchings (5 points, no collaboration)

Prove that there is a unique stable matching when all students have the same preference ordering over universities.

Problem 2: Random Preferences (10 points)

Consider an instance where each student has uniformly random preferences over the universities (the universities have arbitrary preferences over the students), and consider an execution of the student-proposing deferred acceptance algorithm. Let $X_i$ denote the rank of student $i$'s match (that is, one plus the number of universities student $i$ prefers to their match). Prove that $E[\sum_i X_i] = O(n \log n)$.

Hint 1: Prove that deferred acceptance terminates once every university has received a proposal.

Hint 2: You may want to make use of Warm-Up Problem 3!
Problem 3: Both Sides Propose (10 points)

Consider the following algorithm, “Both-Proposing Deferred Acceptance:”

• Maintain a tentative matching $M$, initially empty.

• While there exists an unmatched student:
  
  – Pick an arbitrary unmatched student, $s$. $s$ proposes to her favorite university who hasn’t yet rejected her. If $u$ prefers $s$ to $t = M(u)$, update $M(s) = u$, $M(u) = s$, and $M(t) = \bot$.
  
  – Pick an arbitrary unmatched university (if one still exists), $u$. $u$ proposes to their favorite student who hasn’t yet rejected them. If $s$ prefers $u$ to $v = M(s)$, update $M(u) = s$, $M(s) = u$, and $M(v) = \bot$.

Either prove that Both-Proposing Deferred Acceptance always terminates in a stable matching, or provide an example of preferences and order of proposals such that BPDA does not output a stable matching.

Problem 4: Alice and Bob go to College (15 points)

Your high school guidance counselor heard you were taking COS 445 and asked you to advise the current seniors on how to decide where to apply for undergrad. You quickly realize that college admissions are a lot like university-proposing deferred acceptance (the universities “propose” to their early admits, and waitlist the rest, only proposing if their initial proposals are rejected), with one important catch: a university cannot propose to a student that didn’t apply, and students don’t apply everywhere. Fortunately, your guidance counselor is a data whiz and is able to give you the following model. Your team will be responsible for playing the role of one student deciding where to apply to college.

Setup:

• There will be one student and one university (admitting one student) per submitted bot. The number of submissions will henceforth be known as $N$.

• Every student $s$ has an aptitude $A_s$ drawn independently from $U[0, S]$. If you are student $s$, you know $S$ and $A_s$, but not $A_t$ for any other $t$.

• Every university $u$ has a quality $Q_u$ drawn independently from $U[0, T]$. Every student knows $T$, and $Q_u$ for all universities $u$.

• Every (student, university) pair has synergy $S_{s,u}$ drawn independently from $U[0, W]$. If you are student $s$, you know $W$, $S_{s,u}$ for all universities $u$, but not $S_{t,u}$ for any other student $t$.

• $S$, $T$, and $W$ are real numbers and are constant (the same) across students and universities

Admissions:

• Student $s$ forms preferences over universities in decreasing order of $Q_u + S_{s,u}$.

• Every student simultaneously selects 10 universities to apply to.

• University $u$ forms preferences over students who applied in decreasing order of $A_s + S_{s,u}$.
College-proposing deferred acceptance is performed, where universities only propose to students who applied. That is, when a university is selected to propose, they propose to their favorite student who applied and hasn’t yet rejected them. If they have already proposed to all students who applied, they are permanently unmatched.

Payoffs:

- If you are unmatched, you get payoff 0. Otherwise, if there are a total of \(N\) universities, and you are matched to your \((k + 1)^{th}\) choice (that is, there exist \(k\) universities in the entire pool that you prefer to your match), then your payoff is \(N - k\).

Design a strategy that takes as input \(N, S, T, W, A_s, \langle Q_u \rangle_{u \in U}, \langle S_{s,u} \rangle_{u \in U}\), and outputs a list of ten universities to apply to. Code it up according to the specifications below, and write a brief justification. Please visit the course website for grading policies with respect to programming challenges. Recall that all of your main justification should fit in one page. If you wish to include calculations or simulation results, you may do so in-line, but your writeup should contain at most one page of English justification. This will not be strictly enforced, but graders may not read justifications that go significantly beyond the one-page guideline in full.

Your writeup should provide an overview of the main ideas in your code (remember that we also have your code — so you don’t need to provide pseudocode or a step-by-step description of your algorithm), and justify why you think it will perform well. In addition, you should address the following concrete questions:

- What does your strategy do when \(T = 0\)? Why?
- What does your strategy do when \(W = 0\)? Why?

Specifications:

You will implement the Student interface provided in Student.java, which requires the following method:

```java
public int[] getApplications(int N, double S, double T, double W,
   double aptitude, double[] schools, double[] synergies):
```
called with a profile of a student and the potential universities, and with parameters of the distributions from which the profile was created. Note \(\text{schools.length == synergies.length}\) and there are as many students as schools; and \(\text{schools}\) is sorted in descending order. Implement your strategy for deciding to which schools you shall apply. Return an \(\text{int[10]}\) containing only unique integers which are valid indices into \(\text{schools}\), which indicate your first, second, etc. . . choice schools.

We provide the following sample strategies:

- **Student_usnews**: Applies to the schools with the best overall ranking.
- **Student_synergist**: Applies to the schools with which they have the highest synergy.
- **Student_holist**: Applies to the schools which they like the most.
- **Student_random**: Applies to a uniformly random set of schools.
Your file must follow the naming convention `Student_netid.java`, where `netid` is the NetID of the submitter.

Penalties of up to 9 points may be issued if your submission does not precisely follow the API specifications. Examples of violations include: does not compile, or throws exceptions, or violates invariants documented in `Student.java`.

The provided python script (`formatBase.py`) and Makefile allow you to test your strategy against the provided strategies and any other strategies you consider. Edit `students.csv` with a list of all the strategies to run, then use `make` to rebuild the testing code with those strategies.

Extra credit may be awarded for reporting substantive bugs in our testing code.

**Extra Credit: Almost Unique Stable Matchings**

Recall that extra credit is not directly added to your PSet scores, but will contribute to your participation grade. Some extra credits are quite challenging and will contribute significantly.

Consider an instance where student preferences are uniformly random, and university preferences are arbitrary. Let each student apply only to their favorite $O(1)$ universities. Prove that the expected number of universities with the same partner in all stable matchings is $n - o(n)$ (where “unmatched” counts as a partner)[1].

**Hint 1:** You may want to prove the following fact. Let $M$ be output by student-proposing deferred acceptance, and let $M(u) = s$. Now consider modifying $u$’s preferences by “blacklisting” $s$ and all $s'$ that $u$ likes less than $s$. That is, $u$ declares that they would rather be unmatched that matched to $s$ or anyone below $s$. Let $M'$ denote the matching output by student-proposing deferred acceptance with this modified preference (and all others the same). If $u$ is unmatched, then $u$ is matched to $s$ in every stable matching.

**Hint 2:** You may use the following fact without proof: imagine throwing $kn$ balls into $n$ bins uniformly at random. Then with probability $1 - e^{-\Omega(n)}$, at least $n \cdot e^{-k}/2$ bins are empty.

---

[1] Observe that this means it barely matters which side proposes in this model because almost everyone has the same partner regardless.