Gale-Shapley Applied
Group Exercises
By Cole Frederick

Please form groups of five students (four is ok too), arrange yourselves in a circle across the tables in the classroom, facing each other, and introduce yourselves in counterclockwise order starting with the student who is last to touch their dominant hand’s index finger to their nose after reading this run on.

Thank you. Now turn your attention to these problems:

Love’s Labour’s Lost
As a roughly five actor theater troupe, your choice of scenes is limited. Luckily, Love’s Labour’s Lost Act IV, Scene 1 lends itself to small companies. Your names are Alice, Bob, Charlie, Dani, and Edward (in counterclockwise order starting with the student who was last to touch their nose) (if you are a group of four, someone is Alice and Edward). You have all tried out for each part, and the talent coordinator has created a ranking of actors for each part. They are:

- Princess of France - “But come, the bow: now mercy goes to kill. And shooting well is then accounted ill.”
  - Bob, Charlie, Alice, Edward, Dani
- Boyet - “I fear too much rubbing. Good night, my good owl.”
  - Charlie, Dani, Alice, Bob, Edward
- Costard - “God dig-you-den all!”
  - Charlie, Bob, Edward, Dani, Alice
- Rosaline - “Thou canst not hit it, hit it, hit it. Thou canst not hit it, my good man”
  - Edward, Dani, Charlie, Bob, Alice
- Maria - “A mark marvellous well shot, for they both did hit it.”
  - Dani, Alice, Edward, Charlie, Bob

Please each of you write down your ranking of the parts, from the one you most want to play to the one you least want to play.

Now that you have a stable matching problem, play out the Gale-Shapley algorithm. You can have the actors “propose” to play the parts, or the parts (really the talent coordinator) “propose” which actors should play them. Run the algorithm through, noting the matching and verifying it is stable. Try it again, making different choices for proposer order, to see that you get to the same outcome. Do it with the role of proposers flipped.
Some groups may be lucky enough to have created a problem where the two directions of Gale-Shapley produce the same matching. What would this mean about the number of stable matchings that exist for your problem? Think about the best/worst proofs.

Non-Smoking, Please

For those of you old enough to remember segregated restaurants, did you ever eat in the smoking section? It seemed even smokers preferred to eat their breakfast sans smoke. Imagine a stable matching problem where some number of the men and the same number of women are smokers. Let’s say that everyone, even smokers, categorically prefers non-smoking partners. So everyone’s preference list looks like [non-smoker1 non-smoker2 ... smoker1 smoker2 ...].

If we use Gale-Shapley to produce matchings for such problems, will a non-smoker ever be paired with a smoker? How would you convince a friend of this result?

Boats at dock

There are boats of all sizes approaching the dock for lunch hour half price shrimp tacos. This dock has stalls of various sizes and each stall is managed by an attendant. For various reasons, attendants have preferences for working with the various boat captains (tipping generosity, pleasantry of conversation, etc). And for other reasons, captains prefer some attendants/stalls over others (cleanliness of stall, trustworthiness, closeness to taco shack, etc). You can see where this is going.

But the wrinkle is that some stalls simply cannot accommodate certain boats, because of their width or draft (depth of water) requirements. So certain pairs should not be allowed in a matching. This might mean for a given problem that not all boats can be docked and/or not all stalls can be used. We would nevertheless like to find a maximal stable matching, which means:
- No instabilities (which also means if there is an undocked boat, it’s because every stall either can’t fit it or prefers its current boat, and vice versa for unused stalls)
- No undocked boat-unused stall pairs, unless it can’t fit. If an unpaired boat can fit in an unpaired stall, you should pair them.

Come up with a way (an algorithm) to find such a matching. (GS can be used as a tool, reshape the problem to fit it)

Summer Software Internships

There are students looking for internships and companies looking for interns, with preference lists of course. The twist here is that some companies want more than one intern; they have multiple positions to fill. Come up with a way (an algorithm) to find a stable matching in this situation. (Again, GS can’t be used directly, but you can mold the problem)