Find a stable matching using the algorithm presented last class:

Grad students: A, B, C, D

Undergrad students: 1, 2, 3, 4

Preference lists:

A: 1 2 4 3

B: 2 1 4 3

C: 2 1 3 4

D: 1 3 2 4

1: C D B A

2: D A C B

3: B A D C

4: C B A D

If you are in CSC 425:
apply the algorithm so that the undergraduate students ask grads if they are willing to be their project partner. Grads accept if no partner, and otherwise only say yes if the undergrad student is preferred to the current partner.

If you are in CSC 520:
apply the algorithm so that the graduate students ask undergrads if they are willing to be their project partner. Undergrads accept if no partner, and otherwise only say yes if the grad students is preferred to the current partner.

Questions to ask ourselves

1. Does a stable matching always exist?
2. How many stable matchings can a graph have?
3. Does the algorithm always find a stable matching?
4. Do we get a different answer if the ringleader asks students to make their proposals in a different order?
5. Can we get a different answer when the undergrads are in charge instead of the grad students?
6. How long does the algorithm take to terminate?
7. Are there any faster algorithms for solving this problem?
8. What data structures should we use?
9. How long does it take to check is a matching is stable or not?
10. What happens if participants "cheat" and misrepresent their preferences in order to try and get a better match?
11. Is the problem harder when the underlying graphs is not bipartite (for example, roommate selection)?

There are n grads and n undergraduates. If the preferences are random and grads are proposing, what is the likely average undergraduate’s rank of their project partner, and what is the likely average grad’s rank of their partner?

[Peter Shor]:
Suppose you find the stable marriage that minimizes the average rank of the partners (averaged over both types of students). What is this average?