DEFERRED ACCEPTANCE

The point of this note is to take you through the calculations involved when using the deferred acceptance algorithm. Please read the papers associated with the links on the web page for motivation and applications.

We'll use the example of college placement since you might soon be involved in that, and try to work up to something realistic. To begin we'll just start with three postgraduate programs or schools, call them UBC, SFU, and UoT. When we want to refer to the set of schools we can just write *Schools*. We'll also suppose that there are exactly three students. Lets just call them student 1, 2 and 3 eventually we will have a lot of them, and it won't make sense anymore to give them names. We'll just write *Students* to refer to the set of students.

In this introductory example, lets just suppose that each school can only admit one student. Of course, each of the students can only go to one school, so we want to place the students in some reasonable way. What we mean by reasonable has to be made precise. To do this we need to know the schools and students preferences. Lets write the schools' preferences in the following way

$$SFU \to 1 \succ 2 \succ 3$$

$$UBC \to 1 \succ 3 \succ 2$$

$$UOT \to 2 \succ 3 \succ 1$$

What these bits of notation mean is that SFU wants student 1 the most. If student 1 doesn't come to SFU then they prefer to have student 2, while student 3 is their least favorite student. UBC also really wants student 1, but has a different ranking of students 2 and 3. Finally UOT really wants student 2, but if they can't get her, they prefer student 3 to student 1.

We can do the same thing for the students

$$1 \to UBC \succ SFU \succ UOT$$

$$2 \to UBC \succ UOT \succ SFU$$

$$3 \to UBC \succ SFU \succ UOT$$

(the marketing person told me I had to write the preferences that way). All three students really want to get into UBC. Otherwise, students 1 and 3 want SFU if they are unable to get into UBC. Student 2 prefers UOT if she can't get into UBC.

Now we want to 'place' the students in schools, which means that we want to define a *match* $m : Students \to Schools$. Then we would write $m(1) = UBC$ to indicate that student 1 is to be placed in UBC. So far, a match needs to be one to one and onto.

A little visual aid might help you think about the matching. Lets start by drawing a table that looks like a normal form game (but isn't).

	1	2	3
SFU	1,2	2,3	3,2
UBC	1,1	3,1	2,1
UOT	3,3	1,2	2,3

The first element of each cell is the *rank* that the university in the corresponding row assigns to the student in the corresponding column. As above, if you look across the SFU row at the top, student 1 is most preferred by UBC, so it has rank 1. Similarly, student 2 has rank 2, etc.

As you read down the column, you can read how each of the students ranks the various universities by looking at the second element in each cell. Reading down column 1, UBC has rank 1, SFU rank 2 and UOT has rank 3. This represents the preferences in a compact way.

We can represent a matching by placing three stars in the table in such a way that each row has exactly 1 start in it, and each column has exactly one star in it. For example, here is a matching that puts student 1 in SFU, student 2 in UBC and student 3 in UOT:

	1	2	3
SFU	1,2*	2,3	3,2
UBC	1,1	3,1*	2,1
UOT	3,3	1,2	2,3*

With this simple device, we can start to think about how the students and schools *should* be matched. All of the students want to be in UBC, which isn't possible since each school can only accept one student. One obvious criteria that a good match should satisfy is that it be *pareto optimal*. A match would be pareto dominated if there were some other match that made at least one school or student strictly better off without hurting anyone else.

Under this criteria, the matching $m(3) = UBC$, $m(2) = SFU$, and $m(1) = UOT$ would be something we would definitely call a bad matching. It looks like this:

	1	2	3
SFU	1,2	2,3*	3,2
UBC	1,1	3,1	2,1*
UOT	3,3*	1,2	2,3

Lets see how to improve on it by trying to make some of the schools and students better off without hurting the others. Student 3 has her favorite program, so if we make any change that moves her, she will be worse off. So lets focus on students 1 and 2 who both get their least favorite school. If we switch them around, they would both be made strictly better off. All we need to check is that the schools aren't hurt by this change. Since UOT prefers student 2 to student 1, UOT would be made better off. Similarly, since SFU prefers student 1 to student 2, SFU is also made better off.

This new matching $m(3) = UBC$, $m(1) = SFU$, and $m(2) = UOT$ is pareto optimal (if we try to move student 1 to UBC we'll hurt player 3). However, this points out that there is still a problem with this new matching because UBC would strictly prefer student 1 to student 3, and student 1 strictly prefers UBC to SFU where they are now matched. If we somehow come up with a game or a mechanism or a market that generates the matching $m(3) = UBC$, $m(1) = SFU$, and $m(2) = UOT$, then UBC will want to negotiate a side deal with student 1. We would then say that our mechanism is not *stable* so our market would *unravel*.

Formally, a match m is said to be *stable* if for any student i and university j, if student i prefers university j to university $m(i)$, then university $m(i)$ prefers

the student who it gets in m (that would be $m^{-1}(j)$) to student i. If that is true, then any attempt to unravel the matching process by some kind of offer outside the match will be rejected.

It is pretty tedious checking all this stuff as you can see. Especially if there were thousands of students and many different schools involved. You might wonder whether we could find an easier way to find a matching that is not just pareto optimal, but also stable. This is where the deferred acceptance idea comes into play.

Algorithms. An algorithm is a method of systematically checking data to find something we are looking for. You have seen one algorithm so far - we talked about checking normal form games to find all the Nash equilibria. That algorithm was really a very poor one - for example, to check for pure strategy equilibria we need to check every cell in the payoff matrix to see if there were profitable deviations.

Now instead of looking for a Nash equilibrium, we are looking for a stable matching. This is pretty complicated because we seem to have to check lots of pairings to check for stability. Fortunately there is a faster way to do this.

This is how the story goes - first we ask each of the students to *propose* to their favorite university. This doesn't seem to help much because they all like UBC. Nonetheless, lets suppose each of them makes such a proposal. UBC now responds by picking its favorite applicant - student 1 in our example. UBC *tentatively* accepts student 1's proposal. The reason for the word 'tentative' won't be clear in this example, but the idea we have in mind is that UBC might later get a better proposal and we want to allow UBC to accept this proposal if it ever comes along.

We could visualize these proposals in the following simple way:

	1	2	3
SFU	1,2	2,3	3,2
UBC	1,1*	3,1*	2,1*
UOT	3,3	1,2	2,3

This isn't a feasible matching because UBC's row has three stars in it, and it is only allowed 1. To indicate that UBC has rejected the proposals by 2 and 3 we could write

	1	2	3
SFU	1,2	2,3	3,2
UBC	1,1*	3,1*	2,1*
UOT	3,3	1,2	2,3

At this stage UBC has tentatively accepted student 1's proposal, students 2 and 3 are still left without places. We have actually learned quite a bit from this. Students 2 and 3 know that UBC has just tentatively accepted player 1's proposal. Yet they also know that if UBC ever throws that proposal out in future, it will be for a student that it strictly prefers to student 1. Then by transitivity, this student will be preferred to students 2 and 3, in other words, there is no point in students 2 and 3 bothering to propose to UBC again.

	1	2	3
SFU	1,2	2,3	3,2*
UBC	1,1*	3,1*	2,1*
UOT	3,3	1,2*	2,3

So in stage 2, we want students 2 and 3 to make proposals to schools who have not yet rejected them - 2 will propose to UOT, 3 will propose to SFU. Since the universities have only one proposal, they accept that proposal, and we are finished. This stable matching is $m(1) = UBC$, $m(2) = UOT$, $m(3) = SFU$

Unlike our previous effort, this matching won't unravel. SFU does badly in a way, since they get their least preferred applicant. Yet if SFU approaches student 1 or 2 who they prefer, they will be rejected, because each of them is already matched to a university that they prefer to SFU.

This, in a very simple form, is how the deferred acceptance *algorithm* works.

0.1. **Implementing the Deferred Acceptance Solution.** Now we are starting to get somewhere. We have something that looks like data, and a process for turning the data into an outcome that we like. The last step is to turn this process and algorithm into something that works in practice. To do this, we need to find some way to convince market participants to voluntarily give us the data we want about preferences.

One way we could do it is to have the students physically apply to each of the schools just as we suggested above. Many matching 'markets' work like this. In fact, the process of applying to firms who then choose which applicants to hire is the way job markets would be described in most economics applications.

The market with which I am most familiar is the market for junior academic economists. The applicants are economics students graduating with Ph'd degrees. The firms are Universities, Governments, and Consulting firms throughout the world. The market runs through much of the year, but the primary hiring period begins in October when Universities place job ads. In 2014 there were a little more than 4000 applicants registered at econjobmarket.org, one of the main advertising and recruiting sites. The number of academic openings is somewhere between 1200 and 1600. Junior economists apply to schools they are interested in. The schools review applications, interview applicants at a centralized meeting in January, then make offers to the applicants they like. The applicants, many of whom receive multiple offers, choose the offer that they like.

This might give you some idea about the potential size of a market, and how complex it might be to find some kind of stable matching. It mostly illustrates that it won't be feasible to find a good solution by scribbling on pieces of paper, as we did above. The market for junior economists has some complexities that we'll discuss later.

In 2013, the Vancouver School Board implemented a matching program for children in special kindergarten programs. There were four programs, French Immersion, Mandarin Immersion, Montessori, and Fine Arts, spread over 19 schools. In the year prior to their program, about 1400 students participated. The school board discussed the possibility of matching students using deferred acceptance, but later changed their mind and used a simple computerized lottery using only parents' first choice. This system replaced one in which parents physically traveled to as many schools as they wanted and submitted an application. Schools then ran their own lotteries.

Neither of these examples actually use the deferred acceptance algorithm that we will discuss below. In the school board case, the board was concerned about the difficulty parents might have expressing their preferences over schools. In the the academic job market, universities don't have clear preferences at the start of

the process and need time to develop them. There is no hope of implementing a pareto optimal and stable matching if you don't know participants preferences. The Vancouver school board example is an interesting case in point. The way their current matching program works is that each parent lists three schools in order or their preference. The actual lottery only makes use of the first choice. Each child is entered into a draw for their parents first choice program. If they win the draw, that is great. If they don't, they are put on a waiting list for that school in case another family who did win the lottery decides to decline. Since parents who listed a school as their first choice are always put on a waiting list, the second and third choice are basically not attainable. If parents are very anxious to get their children into a special program, they need to choose the program where they think they are most likely to get in. This may or may not be the one they most prefer for their child.

This is one of the reasons that deferred acceptance is advantageous. The side who make proposals (parents in the case of the Vancouver School Board, students in the college application example) have no reason to mis-represent their preferences the way they do with the Vancouver School Board example. One of the things that is important in market design is which side should make proposals.

We could alter our procedure above and have the schools make proposals. First, UBC and SFU could both send proposals, or acceptance offers to student 1 whom they most prefer. UOT sends an acceptance letter to student 2 since they like her best. Student 1 tentatively accepts the offer by UBC and rejects SFU, while student 2 accepts the offer from UOT. SFU sends an offer to student 2, but student 2 rejects it because he likes UOT better, finally SFU makes the offer to student 3, who accepts it and we are done. Same matching as when students propose.

As long as student 1 can hold onto his offer until he learns whether or not he will get a better one, this procedure works pretty well. One of the things that can make this procedure fail is something called an 'exploding' offer. UBC might send an acceptance to student 1, but require that student 1 accept it almost immediately - in particular, before student 1 learns whether he might get offers from other schools. This doesn't make any difference here since student 1 won't get any better offers anyway. Yet this is generally a major problem for matching markets.

This brings us to modern mechanism design since computers give us a way to do all this almost instantaneously. Suppose we organize this market the following way - each student simply goes to a web page and writes down her preferences exactly as we did at the top of this note. In other words, student 1 simply looks at a list consisting of the items UBC,SFU and UOT and shifts the elements of the list up and down until her favorite school is at the top, second favorite in the middle, etc. She clicks on submit, and a computer now knows her preferences in the same way we did when we discussed this above. The other students and the universities do exactly the same thing. Now the computer simply goes through the steps we describe above, and finds the matching $m(1) = UBC$, $m(2) = UOT$, $m(3) = SFU$ using either the student proposing or school proposing deferred acceptance method.

Now we are at the point we wanted since the students and schools are basically involved in a simple game. The actions available to the students are the various preferences they could provide on the web page, i.e $UBC \succ SFU \succ UOT$, $UBC \succ UOT \succ SFU$, etc (there are 3! or 6 of them in all). Given the strategies of the other students and schools, each such preference results in a matching. For example, if

the others are expected to announce their preference as we described them above, we could calculate what student 1 could achieve for each possible preference she might express. It looks like this

$UBC \succ SFU \succ UOT$	UBC
$UBC \succ UOT \succ SFU$	UBC
$SFU \succ UBC \succ UOT$	SFU
$SFU \succ UOT \succ UBC$	SFU
$UOT \succ SFU \succ UBC$	SFU
$UOT \succ UBC \succ SFU$	SFU

Lets go through the second to last line to see how the table works. The first entry is $UOT \succ SFU \succ UBC$ which means that student 1 submits the preference $UOT \succ SFU \succ UBC$ to the computer that will then use the student proposing version of the algorithm to find a matching. If player 1 submits this preference, then the computer will submit applications to UBC for students 2 and 3 and submit an application to UOT for student 1. At that point, the computer can see that UBC prefers student 3 to student 2, while UOT has only an application from student 1, so the computer will reject student 2 from UBC and go on to the next round.

In the second round, the computer thinks that student 2 prefers UOT to SFU, so it will submit an application to UOT for student 2, and use this to reject the application by student 1. At that point students 2 and three have been matched, so in the final round, the computer just submits an application to SFU for student 1. This is what the rightmost column says - it gives the school that student 1 will be matched to if he submits that preference to the computer. Since student 1 prefers UBC to SFU, it isn't in his interest to submit such a preference. It is a best reply for student 1 to submit his true preference to the computer if he expects the others to submit preferences as we described at the beginning of this note.

It isn't always in everyone's interest to submit their preferences truthfully. For example, suppose we look at a simpler problem with only SFU and UBC and only two students, 1 and 2. We'll add the option of not going to university and suppose that preferences look like this

$$UBC \to 1 \succ 2$$

$$SFU \to 2 \succ 1$$

while the students' preferences are

$$1 \to SFU \succ UBC \succ OUT$$

and

$$2 \to UBC \succ SFU \succ OUT$$

The student proposing algorithm has 1 applying to SFU and 2 applying to UBC. That is it, there are no more proposals. So $m(1) = SFU$, $m(2) = UBC$. The school proposing algorithm has UBC proposing to 1 and SFU proposing to 2, so the matching is $m(1) = UBC$, $m(2) = SFU$. Notice that both the schools are better off in the school proposing algorithm than they are in the student proposing algorithm.

Now notice that something strange happens in this example. Suppose that in the school proposing version, that student 1 simply lies and says that if he can't get into school, he would rather not to go school at all. In other words, he or she submits the preference $1 \to SFU \succ OUT \succ UBC$. Lets run the school proposing

version of the algorithm - UBC proposes to 1 and SFU proposes to 2. At this point, our computerized algorithm decides that 1 should reject UBC's proposal because it thinks he would rather not go to school at all. Since its offer has been rejected, UBC now proposes to its second favorite choice, student 1, who drops the SFU offer and goes to UBC. SFU now has to make an offer to student 1, who accept it. By lying student 1 makes himself strictly better off, but only in the school proposing version of the algorithm.

Here are the two remarkable facts about this computerized procedure:

(1) The student proposing version of the deferred acceptance algorithm (which we just described) will always lead (in a finite number of steps) to a matching that is stable given the submitted preferences: and
(2) It is always in the interests of the students to submit their preferences truthfully no matter what preferences they think the others are going to submit.

It is hard to emphasize how useful these observations are. First, the process of sending letters and acceptances and agonizingly waiting to hear if a better offer will come is accomplished in the blink of an eye by the computer. Speed is not the only issue, all the time spent handing the communication is also eliminated.

Second, students send their preferences to the computer which can record them for future use. Students never want to lie about their preferences, this isn't a happiness survey, it is in the interests of the students to reveal they way they would make choices on their own. Now instead of relying on universities web pages, which almost always make the claim that they are one of the top universities in the world, you can just look at the preferences that students submitted and see for yourself. Recall that even if you find that one university is very popular, there is no reason for you to pretend that you like some less popular university in the hopes that you will get in there once you are refused at the popular university. The computer already takes care of that.

Just so you can see the kind or reasoning used in the literature on this topic, we can go through a proof of the first of these two facts. As always, you should be a skeptic. Just because the algorithm produces a stable matching with the examples that I used above doesn't mean it always will. Anyway, why exactly does the algorithm work in the the examples above. When you think of it, it is pretty hard to check. To do so, one really should take every pair of universities and students and ask whether the student chosen would prefer to be in this alternative university instead of being in the match produced by the algorithm, while at the same time the university would prefer to admit the student (possibly after kicking out someone else) to the group with whom they are matched during the algorithm.

We need a bit of formality to do this. Suppose there are n students each indicated by an index j. So $j \in n$ just means that j is the name of a student. At the same time, suppose that there are m universities (probably m is much smaller than n). The notation $i \in m$ just means that i is the name of a university. This convention is adopted in what follows, i, and its variants like i' always refer to a university, j to a student. Lets refer to a matching by the Greek letter μ. Then if we put the name of a student into μ, the result will be the name of the university that the student has been placed in. For example, in the example we started with at the beginning of this note, $\mu(3) = UBC$. If we put the name of a university in the matching function, for example, $\mu(i)$, we get a list of all the students who were

placed in the university by our matching procedure. When we need it, we'll let k_i be the capacity of university i, i.e., the maximum number of students that i can admit.

Second, we need an assumption about university preferences. Let S be any collection of students who have been admitted to some university i, and that i has space for some more students. Let j and j' be two students who the university has not yet admitted.

Definition 1. University i has responsive preferences if $S \cup \{j\} \succ_i S \cup \{j'\} \implies \forall S'; S' \cup \{j\} \succ_i S' \cup \{j'\}$.

What this definition says is that whether or not a university likes one student more than another doesn't depend on the set of students the university has already admitted (or on the set it expects to admit). An example is a university that ranks students according to their GPA or SAT scores.

Theorem 2. *If universities have responsive preferences, and students and universities provide their true preferences, then the deferred acceptance algorithm will always produce a stable matching.*

Proof. The matching is supposed to be stable in this theorem, so suppose the μ' is a matching produced by the deferred acceptance algorithm which isn't stable. We'll just prove that if the matching isn't stable, then it couldn't have been produced by deferred acceptance.

First, what does it mean to say that the matching isn't stable. If that is true, then there is a student, say j and a university, say i, such that j would strictly prefer to be in university i that in university $\mu'(j)$ where they were placed by deferred acceptance, while at the same time, university i would either like to add student j to the set of students it has already accepted (if it has space), or would like to replace one of its existing students with student j (if it is full).

Now the rules of the deferred acceptance algorithm are very strict - at each step the student must send an application to the university he or she most prefers among the group of universities that have not yet rejected one of their applications. Since j must have sent an application to $\mu'(j)$, but prefers i to $\mu'(j)$, it must have been the case that j sent an application to i which was rejected. That is just the way the algorithm works, if j hadn't received a rejection from i, the program would not have sent an application to $\mu'(j)$ and would have sent it to i instead.

The next question is why i would have rejected j's application. The only time the program will allow the university to reject an application is when it has already received applications from a group of students all of whom are preferred to applicant j.[1] So the university liked its existing applicants better when j applied. After that, the algorithm would continue to process new applications for the university, some students would be admitted, and others dropped, some would be rejected. No matter how long this process continued after j's application, the university would only have admitted new students that it preferred to one of their existing acceptances. Since we now know that the existing applicant must have been preferred to j, the new applicants who are admitted must also be preferred to j (by responsive preferences), so it is impossible for the university to want to admit j after the

[1]We don't really need to worry about university i rejecting student j because their GPA isn't high enough - if that were true the university would never have wanted to admit j after the matching either.

matching is finished. By this Sherlockian deductive process we can conclude that there cannot be a matching produced by the deferred acceptance algorithm that is unstable. □

Notice a few things about this argument. First, we need the university to have responsive preferences. For example, the Stanford MBA program might prefer a student with an engineering background to a student with an English degree if it expects lots of applications from good humanities students. Yet it might find that near the end of the algorithm that it hasn't received the humanities applications it expected. Then it might decide to change its mind. This would be a violation of responsive preferences, and you can see that our argument above would fail in this case (which is why we added the assumption).

Of course, the second thing to consider here is that the algorithm uses the preferences that the university submitted at the very start. If these aren't the university's real preferences, then it might well want to admit the student once it finds out that its attempt to manipulate the outcome hasn't worked the way it wanted.

I'll stick with the student proposing version of the algorithm in what follows, and just note it isn't always in the interests of the universities to submit their preferences truthfully. Often this doesn't matter that much. Universities have moved away from screening students to graduating as many students as they can. So typically the constraints from the universities point of view are just capacity. We'll explicitly come back to deal with this since it is an important consideration in a lot of matching problems. So lets just try something slightly more complicated.

0.2. **A slightly more complicated problem:** Lets make things slightly more complicated by adding a few more students, another university, and another option, The additional option will be not going to university at all. Here are the preferences. At this point you can imagine that these are preferences that were actually submitted through the web page. The problem is just to do the matching. Here are the preferences written in two different ways:

$$SFU \to 1 \succ 4 \succ 2 \succ 3$$

$$UBC \to 1 \succ 3 \succ 4 \succ 2$$

$$UOT \to 2 \succ 3 \succ 1 \succ 4$$

$$UFV \to 2 \succ 3 \succ 4 \succ 1$$

$$1 \to UBC \succ SFU \succ UOT \succ UFV \succ OUT$$

$$2 \to UBC \succ UOT \succ SFU \succ UFV \succ OUT$$

$$3 \to UBC \succ SFU \succ UOT \succ UFV \succ OUT$$

$$4 \to UBC \succ UOT \succ SFU \succ OUT \succ UFV$$

	1	2	3	4
UBC	1,1	4,1	2,1	3,1
SFU	1,2	3,3	4,2	2,3
UFV	1,4	1,4	2,4	3,5
UOT	3,3	1,2	2,3	4,2
OUT	-,5	-,5	-,5	-,4

We've added a fourth university (if you like name UFV means University of the Fraser Valley). There is also a fourth student who only wants to go to UBC, SFU or UOT, but is not interested in UFV.

Lets find a stable matching using the algorithm:

(1) All students propose to UBC, which tentatively accepts student 1 and rejects the other proposals;
(2) Students 2 and 4 propose to UOT, while student 3 proposes to SFU, SFU tentatively accepts student 3, UOT tentatively accepts student 2 who it prefers;
(3) Student 4, who has been rejected by UBC and UOT, now proposes to his remaining favorite SFU. SFU prefers 4 to their current acceptance, 3, so they throw 3 out and accept 4;
(4) Student 3 has been rejected by UBC and now SFU, so he proposed to his next favorite, UOT who has currently accepted 2 who they prefer, student 3's application is rejected;
(5) Finally, student 3 proposes to UFV who accepts him.

This gives us

$$m(1) = UBC, m(2) = UOT, m(3) = UFV, m(4) = SFU.$$

Now in pictures:

	1	2	3	4
UBC	1,1*	4,1*	2,1*	3,1*
SFU	1,2	3,3	4,2	2,3
UFV	1,4	1,4	2,4	3,5
UOT	3,3	1,2	2,3	4,2
OUT	-,5	-,5	-,5	-,4

	1	2	3	4
UBC	1,1*	4,1*	2,1*	3,1*
SFU	1,2	3,3	4,2*	2,3
UFV	1,4	1,4	2,4	3,5
UOT	3,3	1,2*	2,3	4,2*
OUT	-,5	-,5	-,5	-,4

	1	2	3	4
UBC	1,1*	4,1*	2,1*	3,1*
SFU	1,2	3,3	4,2*	2,3*
UFV	1,4	1,4	2,4	3,5
UOT	3,3	1,2*	2,3	4,2*
OUT	-,5	-,5	-,5	-,4

	1	2	3	4
UBC	1,1*	4,1*	2,1*	3,1*
SFU	1,2	3,3	4,2*	2,3*
UFV	1,4	1,4	2,4	3,5
UOT	3,3	1,2*	2,3*	4,2*
OUT	-,5	-,5	-,5	-,4

	1	2	3	4
UBC	1,1*	4,1*	2,1*	3,1*
SFU	1,2	3,3	4,2*	2,3*
UFV	1,4	1,4	2,4*	3,5
UOT	3,3	1,2*	2,3*	4,2*
OUT	-,5	-,5	-,5	-,4

Exercise:

(1) Find the stable outcome associated with the university proposing deferred acceptance algorithm for this last example. Are they the same? Can you see why this means that the stable matching is unique in this case? Can you see why neither schools or students want to misrepresent their preferences in this case?
(2) Suppose all the students agree on the ranking of the schools (pick your favorite ranking). What happens. Is there a difference between the student proposing and the school proposing outcome?
(3) As special case of the above occurs when all the schools agree on the ranking of students, and all the students agree on a ranking of schools. What happens then? This outcome is called 'assortative matching'.
(4) Go back to the problem we discussed above in which students lied about their preferences when using the school proposing algorithm. Can you describe one or more Nash equilibrium outcomes having the property that every students and schools report about their preferences is a best reply to the reports of the others? How do schools report in these equilibria?

Indifference. Lets go back to our original problem in which 3 students rated 3 schools according to

$$1 \to UBC \succ UOT \succ SFU$$
$$2 \to UBC \succ SFU \succ UOT$$
$$3 \to UOT \succ UBC \succ SFU$$

Now when the students submitted these preferences, lets suppose that 1 and 2 are exactly as stated, but that 2 really doesn't care whether he goes to UBC or SFU, but strictly prefers both of them to UOT. He had to fill out something as his favorite choice, so he flipped a coin and it came out UBC. Furthermore, lets change the preferences of the schools so that they don't really care who they admit, they just want to fill their slots. What should you do then?

Recall that the student proposing deferred acceptance algorithm supports a stable outcome and has the property that students report their preferences truthfully. To make it work, we needed the schools to reject applications. We can make them do that by having them run a lottery which ranks the students. So the administrators in the different schools create a spreadsheet with the three students listed in one column and then sort the column randomly. Once they do this they submit these randomized preferences. They could look like anything, but suppose they were

$$SFU \to 3 \succ 1 \succ 2$$
$$UBC \to 2 \succ 3 \succ 1$$
$$UOT \to 1 \succ 2 \succ 3.$$

Now the program runs, and 2 gets into UBC, 1 gets into UOT and 3 gets into SFU. This is stable given the stated preferences, but these preferences are masking a lot of indifference. In fact, if we switched 1 to UBC, 2 to SFU and 3 to UOT, 1 and 3 would be strictly better off and 2 would be no worse than he was initially. Since the schools don't really care, this new matching would constitute a pareto improvement - not good.

In this simple case, there is an easy way to resolve this. It provides an example of something called the 'top cycle' algorithm. In the matching above $m(2) = UBC$, $m(1) = UT$, $m(3) = SFU$. The students learn this information. Suppose we now give them the option to propose one or more schools that they would be willing to trade for their existing allocation - encouraging them to include another school if they don't really care whether they are matched with their existing school or their alternative proposal. The proviso is that they must be willing to give up their existing school for the alternative they specify. The system will switch them to the new school if they can.

Hopefully, student 2 will propose that he is willing to trade UBC for SFU. Student 1 should propose to trade UBC for UT. Student 3 should propose trading either UT or UBC for his current allocation at SFU. The top cycle algorithm takes these proposed trades and goes through the following routine:

(1) Choose any student, say student 1 and check whether there are any schools he would like to trade for his current match. Student 1 has proposed UBC as an alternative to UT. So we start looking for a cycle using student 1.
(2) UBC is owned by student 2, so check whether there is any school that student 2 is willing to trade for UBC. We find he has proposed SFU as an alternative he would be willing to trade. So lets add student 2 to our cycle, getting $1 \to 2$.
(3) SFU is owned by student 3, so check whether there is any school that 3 is willing to trade. One of the schools he has proposed is UBC, so lets add student 3 to the cycle $1 \to 2 \to 3$.
(4) UBC is owned by student 2 who is willing to trade for UBC, so add him $1 \to 2 \to 3 \to 2$.

In this last step, we added a student who is already in the list. This gives us a 'top cycle'. If we continued to apply the steps in the algorithm above we would cycle back and forth between student 2 and 3. The reason for the cycle is that 2 prefers the school that 3 has and 3 prefers the school that 2 has. So switching them creates a pareto improvement. We might as well switch them and change the matching to $m(2) = SFU$, $m(1) = UT$, $m(3) = UBC$.

Notice that we made some students better off, and didn't hurt anyone. You only propose trades you prefer - you might get them or you might get nothing. No one is being made any worse off that they were initially. We might as well ask the question again, are there any schools you would be willing to trade for your new allocation. All three of them have schools they prefer - 1 prefers UBC to his new allocation at UT, 2 also prefers UBC while 3 prefers UT. So lets do it again:

(1) 1 prefers UBC to UT so lets add him to the cycle $1 \to$;
(2) UBC is now owned by student 3 who prefers UT, so lets add her to the cycle $1 \to 3$;
(3) UT is owned by 1 who prefers UT so we have $1 \to 3 \to 1$ - a top cycle.

So student 1 and 3 should switch schools giving $m(2) = SFU$, $m(1) = UBC$, $m(3) = UT$.

If we try this again we will find the only student we can add to a cycle is student 2 who prefers UBC, but it is owned by student 1 who is no longer willing to move - our algorithm comes to an end with an allocation that is pareto optimal and stable as long as the students haven't lied about their preferences. In fact, if you recall, we assumed that student 2 didn't really care whether he got UBC or SFU, so each of the students actually gets his favorite choice.