Class 15: Stable Matchings

Schedule

There is no class Thursday (Oct 20). Please use the class time to do something worthwhile and fulfilling.

Problem Set 6 (is posted now) is due **21 October (Friday) at 6:29pm**.

Stable Matching

Definition. $M = \{(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)\}$ is a *stable matching* between two sets $A = \{a_1, \cdots, a_n\}$ and $B = \{b_1, \cdots, b_n\}$ with respective preference orderings $\prec_A$ and $\prec_B$ if there is no pair (a_i, b_j) where $b_i \prec_{a_i} b_j$ and $a_j \prec_{b_j} a_i$.

How do we know there is always a stable matching between any two equal-size sets?

Gale-Shapley Algorithm

Download the full matching code: matching.py (assertions and comments removed here for space)

```
def gale_shapley(A, B):
    pairings = {} # dictionary b: a pairings
    unpaired = set(a for a in A.keys()) # unpaired a's
    proposals = {a: 0 for a in A.keys()} # keep track of who gets next proposal

    while unpaired:
        a = unpaired.pop()
        ap = A[a] # a's preference list (haven't been proposed to yet)
        assert proposals[a] < len(ap)
        choice = ap[proposals[a]]
        proposals[a] += 1
        if choice in pairings: # a's choice already has a match
            amatch = pairings[choice]
            bp = B[choice]
            if bp.index(a) < bp.index(amatch):
                pairings[choice] = a
                unpaired.add(amatch) # lost match
            else:
                unpaired.add(a)
        else:
            pairings[choice] = a
    return [(a, b) for (b, a) in pairings.items()]
```

```
A = {"Anna": ["Kristoff", "Olaf"], "Elsa": ["Olaf", "Kristoff"]}
B = {"Kristoff": ["Anna", "Elsa"], "Olaf": ["Elsa", "Anna"]}
gale_shapley(A, B)
```

Modeling the Gale-Shapley program as a state machine

$S =$

$G =$

$q_0 =$

Prove termination. The Gale-Shapley program, modeled by the state machine, eventually returns.

Prove correctness. The Gale-Shapley program returns a *stable matching* of the two input sets and preferences.

Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) allows two (or more) parties to jointly compute a function of their two private inputs, without leaking any information about those inputs (other than what can be inferred from the revealed result). See web versoin for links.

Garbled Logic Gate

$p_{T,F}$, $q_{T,F}$, and $x_{T,F}$ are randomly selected keys that represent the respective True and False values.

Randomly permute the rows, and only send the output column.

P	Q	$X = P \wedge Q$
p_T	q_T	$E(\_\_\_\_, x_T)$
p_T	q_F	$E(\_\_\_\_, x_F)$
p_F	q_T	$E(\_\_\_\_, x_F)$
p_F	q_F	$E(\_\_\_\_, x_F)$

XOR Secret Sharing

$A \oplus R = C$. What is $C \oplus R$?

