Discussion 2A

CS 70, Summer 2024

1 Induction (Continued)

(a) By induction on n.

Base case. For $n = 0$,

$$\sum_{i=0}^{0} F_i^2 = F_0^2 = 0 = F_0 F_1.$$

Induction case.

Induction hypothesis. For some $n \in \mathbb{N}$, assume that

$$\sum_{i=0}^{n} F_i^2 = F_n F_{n+1}.$$

Induction step. Consider $n + 1$.

$$\begin{aligned}
\sum_{i=0}^{n+1} F_i^2 &= F_{n+1}^2 + \sum_{i=0}^{n} F_i^2 \\
&= F_{n+1}^2 + F_n F_{n+1} && \text{(induction hypothesis)} \\
&= F_{n+1}(F_n + F_{n+1}) \\
&= F_{n+1} F_{n+2}. && (F_{n+1} = F_n + F_{n+1})
\end{aligned}$$

By the principle of mathematical induction, we have demonstrated the claim.

(b) By strong induction on n.

Base case. $n = 1 = F_1$.

Induction case.

Induction hypothesis. Suppose that for all positive integers $k \leq n - 1$, we can decompose k into the sum of distinct Fibonacci numbers.

Induction step. Consider n.

Let F_m be the largest Fibonacci number less than n. That is, $F_m < n$ and $F_{m+1} \geq n$. Then

$$\begin{aligned}
F_{m+1} &\geq n \\
F_m + F_{m-1} &\geq n \\
n - F_m &\leq F_{m-1}.
\end{aligned}$$

Since $n > F_m$, $n - F_m$ is a positive integer, and can be written as the sum of distinct Fibonacci numbers by the induction hypothesis:

$$n - F_m = F_{i_1} + \ldots + F_{i_\ell}.$$

Since $n - F_m \leq F_{m-1}$, none of $F_{i_1}, \ldots, F_{i_\ell}$ can be F_m. Therefore we have that

$$n = F_{i_1} + \ldots + F_{i_\ell} + F_m,$$

where $F_{i_1}, \ldots, F_{i_\ell}, F_m$ are distinct Fibonacci numbers.

By the principle of mathematical induction, we have shown that every positive integer can be writted as the sum of distinct Fibonacci numbers.

2 Stable Matching

The algorithm's run for this example is described in the table below. The bolded numbers correspond to the jobs to which the candidates said "Maybe," and the remaining numbers correspond to the jobs which candidates rejected.

Candidate	Day 1	Day 2	Day 3	Day 4	Day 5
A	**1**, 3	**1**	1, **2**	**2**	**2**
B	**2**	2, **3**	**3**	**1**, 3	**1**
C					**3**

Therefore the algorithm takes 5 days to terminate, and creates the stable matching $\{(A, 2), (B, 1), (C, 3)\}$.

3 Propose-and-Reject Proofs

(a) By the improvement lemma, since the candidate receives an offer on day m, their most preferred offer on any later day must be at least as good. This requires the candidate to receive an offer on all following days.

Another way is to prove it by induction on the day of the algorithm. Suppose that a candidate C receives on offer on day m. We prove by induction that they must receive an offer on every subsequent day $t \geq m$.

Base case. $t = m$. By assumption, C receives an offer on day m.

Induction case.

Induction hypothesis. Suppose that C receives an offer on day $t \geq m$. Let J be that job offer.

Induction step. Consider day $t + 1$. Either C prefers J to all other offers they receive on day t or C prefers some other offer J' they receive on day t to J.

(1) C prefers J to all other offers. Then on day t, C tells J to come back the next day. So C receives an offer on day $t + 1$.

(2) C prefers J' to J. Then on day t, C rejects J and tells J' to come back the next day. So C receives an offer on day $t + 1$.

By the principle of mathematical induction, we have shown that for all days $t \geq m$, the candidate receives a job offer.

Therefore, if a candidate receives an offer on day m, they receive an offer on every subsequent day.

(b) By contraposition. Suppose that a candidate has received an offer on some day j, $1 \leq j < i$. By part **(a)**, the candidate must receive a proposal on every following day, so the candidate receives an offer on day i.

(c) Let $k \in \mathbb{Z}^+$ be the number of days it takes for the algorithm to terminate. Thus every candidate receives an offer on day k.

Then there must be some candidate C on day $k - 1$ who did not receive an offer; if, by way of contradiction, every candidate received an offer on day $k - 1$, then the algorithm would have terminated on day $k - 1$.

By part **(b)**, if C did not receive an offer on day $k - 1$, then they did not receive an offer on any previous day. Furthermore, they must have received exactly one offer on day k, since the algorithm terminates that day. Thus C receives exactly one proposal throughout the entire run of the algorithm.

4 Job Optimality

(a) In the instance below, Job 1 can never be matched with candidate A in any stable matching. Thus $C^*(1) = B$ which is not the top candidate in Job 1's list.

Jobs	Candidates
1	A $\succ$ B
2	A $\succ$ B

Candidates	Jobs
A	2 $\succ$ 1
B	1 $\succ$ 2

(b) Because job J is rejected by $C^*(J)$ on day t in favour of job J^*, $C^*(J)$ must prefer J^* more.

(c) By our induction hypotheses, on days $1, 2, \ldots, t-1$, no job has been rejected by their optimal candidate. Thus J^* must have never been rejected by $C^*(J^*)$. But J^* chooses to make an offer to $C^*(J)$ on day t. Thus J^* must prefer $C^*(J)$ more than $C^*(J^*)$.

(d) By part **(b)**, $C^*(J)$ prefers J^* to J, its partner in M_J. By part **(c)**, J^* prefers $C^*(J)$ to $C^*(J^*)$, its partner in M_J. Therefore they are a rogue couple.

(e) This contradicts the fact that M_J is a stable matching. Therefore J was not rejected by its optimal partner $C^*(J)$ on day t. This finishes the induction step.

By the principle of mathematical induction, we have shown that for each $t \in \mathbb{N}$, by the end of day t of the algorithm, no job has been rejected by its optimal candidate.

Let $k \in \mathbb{Z}^+$ be the number of days it takes for the algorithm to terminate. By our induction proof, we have that by the end of day k of the algorithm, no job has been rejected by its optimal candidate. Since the algorithm terminates on day k, this means that every job is matched with their optimal candidate.