CS 180, Fall 2019
Problem Set 1
Due October 9, 2019

Tian Ye
UID: 704931660

October 7, 2019

Exercise 3 Page 22

There does not always exist a stable pair of schedules. We can see that from the following set of TV shows and associated ratings:

- $\mathcal{A}$ and $\mathcal{B}$ have length < 1
- Let $\mathcal{A} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ and $\mathcal{B} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$; this is valid as no two numbers are the same
- Let us match $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$. $\mathcal{B}$ is now stable but $\mathcal{A}$ can swap its slots to gain more viewers. If we swap $\mathcal{A}$'s slots such that the schedules now look like $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$, $\mathcal{A}$ is now maximized but $\mathcal{B}$ longer is. If $\mathcal{B}$ swaps its slots to remedy this issue, we return back to the first state.
- Hence, this set of schedules do not have a stable state and therefore there does not exist an algorithm to solve them.

Exercise 4 Page 23

Using the Stable Matching Algorithm:

- Any student that is not on a list and has yet to ask all the hospitals on their preference list will ask the remaining hospitals on their list.
- If the hospital is not full, it will accept the student. If the hospital is full, it can either accept or reject that student. If the student s is ranked higher on the hospital's preference list than the lowest ranked student s' currently assigned to the hospital, s' will be removed from the hospital and put on the list of students not currently assigned to a hospital while s will be assigned to the hospital.
- Repeat the previous steps until all the students that are not assigned to a hospital have gone through their entire preference list. Once this condition is met, the remaining students that are not assigned to a hospital are those that were rejected.

Proof by Contradiction for Stability Condition 1:

- Given that student s is assigned to hospital h, and student s' is assigned to no hospital, and h prefers s' to s.
- To reach this state, s' must have asked all hospitals on their preference list, including h.
- Had s' been already assigned to h before s was assigned, s should not have replaced s' as s' is preferred by h.
- Had s been already assigned to h before s' was assigned, s' should have replaced s as h prefers s' to s.

Proof by Contradiction for Stability Condition 2:

- Given that student s is assigned to hospital h, and student s' is assigned to hospital h', and the h prefers s' to s, and s' prefers h to h'.
- Since s' would have asked to join h before h' as h is higher up their preference list, in order to reach this state, h must have replaced s' with s, or h must have elected to not replace s with s', despite the fact that s' is higher up h's preference list.
- Neither of these situations would occur, as h would not have replaced s' with s, and h would replace s with s'.

Exercise 6 Page 25

Using a variant of the Stable Matching Algorithm:

- Ships will have their “preference” list be comprised of their schedule.
- Pick a ship that is not currently docked with the lowest timestamp (closest to the beginning on the month).
- Go to the next port in the ship’s schedule and dock at that port, removing any ship that is currently docked at that port.
- Repeat until all ships are matched. Note, ships will ask ports in order of preference, at maximum once per port.

Proof by Contradiction:

- Given that no two ships will dock at the same port on the same day, the only two instability case are that a ship will arrive at a port that is occupied and therefore “locked” and that a ship and a port will remain unmatched after the completion of the algorithm.
- The first is necessarily impossible as this is a step in the algorithm used to match the ships to the ports: the arriving ship will replace the previously docked ship, and the algorithm will work to match the previously docked ship to a new port.
- The second is impossible as once a single ship has occupied a port at any time, that port will always have a ship thereafter. Furthermore, each ship has all the ports on their schedule. Hence, there will be no situation where a ship and a port will remain unmatched.

Exercise 4 Page 67

$$g_1(n) = 2^{\sqrt{\log n}} \tag{1}$$
$$g_2(n) = n(\log n)^3 \tag{2}$$
$$g_4(n) = n^{\frac{4}{3}} \tag{3}$$
$$g_5(n) = n^{\log n} \tag{4}$$
$$g_2(n) = 2^n \tag{5}$$
$$g_7(n) = 2^{n^2} \tag{6}$$
$$g_6(n) = 2^{2^n} \tag{7}$$

Exercise 5

(a) We will prove the following given equation by induction:
$1+2+3+4+...+n = \frac{n(n+1)}{2}$

Proof. Base Case:

$$\begin{aligned} 1 &= \frac{1(1+1)}{2} \\ &= \frac{2}{2} \\ &= 1 \end{aligned}$$

□

Proof. N + 1 Case:

$$\begin{aligned} \frac{(n+1)(n+2)}{2} &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{n(n+1)}{2} + \frac{2(n+1)}{2} \\ &= \frac{n^2+n}{2} + \frac{2n+2}{2} \\ &= \frac{n^2+3n+2}{2} \\ &= \frac{(n+1)(n+2)}{2} \end{aligned}$$

□

(b) We will prove the following equation by induction:
$1^3+2^3+3^3+...+n^3 = \left(\frac{n(n+1)}{2}\right)^2$

Proof. Base Case:

$$\begin{aligned} 1 &= \left(\frac{1(1+1)}{2}\right)^2 \\ &= \left(\frac{2}{2}\right)^2 \\ &= 1 \end{aligned}$$

□

Proof. N + 1 Case:

$$\begin{aligned}\left(\frac{(n+1)(n+2)}{2}\right)^2 &= \left(\frac{n(n+1)}{2}\right)^2 + (n+1)^3 \\ &= \frac{n^2(n+1)^2}{4} + \frac{4(n+1)^3}{4} \\ &= (n+1)^2 * \frac{n^2+4n+4}{4} \\ &= \frac{(n+1)^2(n+2)^2}{4} \\ &= \left(\frac{(n+1)(n+2)}{2}\right)^2\end{aligned}$$

□

Egg Drop

200 Step Case:

- The worst case scenario for the egg drop requires 27 steps.
- Since we are moving up the steps in increments alternating between 14 and 15, starting with 15.
- At worst, we will find that the egg only breaks on step 200 but not the previous increment (186).
- We then start with the second egg at step 187, incrementing until it eventually reaches step 199.
- We can then conclude the minimum distance for breaking is step 200, with a total of 27 steps.

N Step Case:

- Drop the egg in step size increments.
- In a worst case scenario, this results in $\frac{n}{increment} + increment + 1$ tries.
- Taking the derivative of the previous expression, we find that the optimal increment size is $\sqrt{n}$ steps.
- Thus, the maximum number of drops required for n steps is approximately $2\sqrt{n}$ drops.