CS 180 Homework 1

Prithvi Kannan

405110096

1. Exercise 3, Page 22

There are examples of TV shows and ratings that do not have a stable pairing.

Consider a simple case where $n = 2$ and Network A has shows of rating $\{1,3\}$ and Network B has shows of rating $\{2,4\}$. If the initial schedule for combinations in the form of $\{A's\ show, B's\ show\}$ where Network B is winning both weeks, then Network A can switch their shows around and win one of the weeks.

We consider all 4 situations (2 possibilities for Network A's shows * 2 possibilities for Network B's shows). In each situation, the matching is unstable.

- $\{\{1,2\},\{3,4\}\} \rightarrow \{\{3,2\},\{1,4\}\}$ Network A interchanges its shows and gains a win (0 -> 1)
- $\{\{3,2\},\{1,4\}\} \rightarrow \{\{3,4\},\{1,2\}\}$ Network B interchanges its shows and gains a win (1 -> 2)
- $\{\{3,4\},\{1,2\}\} \rightarrow \{\{1,4\},\{3,2\}\}$ Network A interchanges its shows and gains a win (0 -> 1)
- $\{\{1,4\},\{3,2\}\} \rightarrow \{\{1,2\},\{3,4\}\}$ Network B interchanges its shows and gains a win (1 -> 2)

Therefore, A: $\{1,3\}$ and B: $\{2,4\}$ will never have a stable pair of schedules since one network can unilaterally change its own schedule and win more time slots.

2. Exercise 4, on Page 22

The algorithm goes as follows

```
while a hospital h has an open spot
    extend an offer to the next highest student in priority list
        if the student doesn't have an offer
            student accepts offer from h
            decrement available spots in h
        if the student already has an offer from hospital h'
            if h is ranked higher than h' on the student's priority list
                student accepts offer from h
                decrement available spots in h
                students declines offer from h'
                increment available spots in h'
            else
```

```
                    do nothing
```

We will prove the stable matching exists indirectly.

The case of instability looks like s assigned to h, s’ assigned to no hospital, but h prefers s’ to s. The algorithm would go down h’s list of students in descending preference, meaning that h will extend an offer to s’ before s. If s’ has no higher offers or no offer, s’ will accept the offer from h. Only if s’ has a better offer than h will s’ decline h’s offer. Once s’ has an offer, s’ will never not have an offer; in other words s’ will only improve their offer. Therefore, we will never see a situation where a higher priority student of h was not matched.

Another case of instability looks like student s has accepted an offer from hospital h, but would rather have go to h’. Similarly hospital h’ has given an offer to s’, but prefers s. The algorithm would give h’s offers to s and then s’ since that is the given priority. If s already has an offer, then s will only accept a new offer if it is higher in its list, meaning if s gets an offer from h’, it will never give that up for something lower. For this reason, we will never see s end up matched with h.

3. Exercise 6 on page 25

The conditions that must be met state that “no two ships can be in the same port on the same day.” We will indirectly prove a set of truncations can always be found. If a set of truncations cannot be found, that means that two ships must be in the same port on the same day. We know that each ship is guaranteed to visit each port for exactly one day during the month, so there must be a combination that results in a perfect match.

Now we must prove this is a stable matching algorithm. The case of instability goes as follows: ship s’ visits port p after ship s is already stopped for maintenance. Since s’ is higher priority in p’s list, p would have offered a spot to s’ before s, so by contradiction the case of an instable match is impossible.

This is a stable matching problem between n ships and n ports. Each ship needs to be matched to exactly one port along its schedule for maintenance. The ship’s preferences vector is its schedule, and the port’s preferences vector is the reverse chronological order in which ships visit it.

We then run the G-S algorithm on the sets.

```
While a port p is unmatched
    extend a spot to the next highest ship in priority list
        if the ship doesn't have an spot
            ship accepts offer from p
        if the ship already has a spot from port p'
            if p is ranked higher than p' on the ship's priority list
                ship accepts spot from p
                ship declines spot from p'
            else
                ship keeps spot from p'
```

4. Exercise 4 on page 67

$$2^{\sqrt{logx}} < x^{4/3} < x(logx)^3 < x^{logx} < 2^x < 2^{x^2} < 2^{2^x}$$

$2^{\sqrt{logx}}$ grows the slowest since $\sqrt{logx}$ for large x is close to 1, meaning the term $2^{\sqrt{logx}}$ will be almost constant. $x^{4/3}$ is an polynomial function, which will grow the faster than an almost-constant function. $x(logx)^3$ will grow even faster since the $(logx)^3$ term will outgain $x^{1/3}$. x^{logx} grows faster than polynomial time since the $logx$ exponent will grow faster than a constant such as 2. Next we have the remaining exponential functions. Since they all have base 2, we can compare their exponents. $x < x^2 < 2^x$.

5. (a) Prove (by induction) that sum of the first n integers (1+2+::::+mn) is n(n + 1)=2

Base case: $n = 1$ gives $1 = \frac{(1)(1+1)}{2} = \frac{(1)(2)}{2} = 1$

Inductive assumption: $1 + 2 + 3 + \cdots + k = \frac{k(k+1)}{2}$

Now we show that given the statement holds for the case of $n = k$, it also holds for $n = k + 1$.

$$1 + 2 + 3 + \cdots + k + (k + 1) = \frac{k(k + 1)}{2} + (k + 1)$$

$$1 + 2 + 3 + \cdots + k + (k + 1) = \frac{k(k + 1)}{2} + \frac{2(k + 1)}{2}$$

$$1 + 2 + 3 + \cdots + k + (k + 1) = \frac{(k + 1)(k + 2)}{2}$$

Therefore, we have shown that $1 + 2 + 3 + \cdots + k + 1 = \frac{(k+1)(k+2)}{2}$ so we can confirm our inductive assumption holds for all $n \geq 1$.

(b) What is 13 +23 +33 +:::+n3 = ?? Prove your answer by induction.

Guess the following:

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \left[\frac{n(n + 1)}{2}\right]^2$$

Prove the base case, $n = 1$,

$$1^3 = \left[\frac{1(1 + 1)}{2}\right]^2$$

$$1^3 = \left[\frac{1(2)}{2}\right]^2$$

$$1^3 = [1]^2$$

$$1 = 1$$

Assume the inductive hypothesis for $n = k$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 = \left[\frac{k(k+1)}{2}\right]^2$$

Now we must prove for $n = k + 1$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \left[\frac{k(k+1)}{2}\right]^2 + (k+1)^3$$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \frac{k^2(k+1)^2}{4} + \frac{4(k+1)^3}{4}$$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \frac{k^2(k+1)^2 + 4(k+1)^3}{4}$$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \frac{(k+1)^2 + (k^2 + 4(k+1))}{4}$$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \frac{(k+1)^2 + (k^2 + 4k + 4)}{4}$$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \frac{(k+1)^2 + (k+2)^2}{4}$$

$$1^3 + 2^3 + 3^3 + \cdots + k^3 + (k+1)^3 = \left[\frac{(k+1)(k+2)}{2}\right]^2$$

Therefore, since we have proved that the equation holds for base case $n = 1$, and that if we assume it holds for $n = k$, it also holds for $n = k + 1$, this holds for all $n \geq 1$.

6. How many tries do you need (in the worst case) in the two egg problem when there are 200 steps? what about n steps?

Since we have two eggs instead of one, we can use our first egg to narrow down the problem and then use our second egg to do a linear search on a smaller problem set.

A proposed algorithm works as follows:

```
let x be the initial block size

while egg1 hasn't broken

    drop from current floor + x

    decrement x

do a linear search of egg2 within the range (current floor, current floor – x)
```

This algorithm ensures that the worst-case runtime stays the same, even when the egg breaks near the top floors. This is accomplished by reducing the block size by 1 each time the egg survives the drop. This counteracts the additional step as we move up a block. Since we see block size decreasing by 1, we can set up an equation to represent what our starting block size will be and how many steps it will take to find in the worst case.

The following equation represents the number of floors covered by our algorithm if we assume a block size of x. Since we want to cover 200 floors, we want our summation to be greater than or equal to 200.

$$x + (x - 1) + (x - 2) + \cdots + 2 + 1 \geq 200$$

Using the Gauss equation, we can represent the sum of consecutive terms as the following

$$\frac{x(x + 1)}{2} \geq 200$$

Solving for x in the case of 200 floors and 2 eggs gives that $x = 20$. This means that in (one of) the worst cases, we drop the 1st egg at floor 20, it breaks, and we must do a linear search from floor 1-19, which would take 20 steps, which is the same as the block size.

For a general solution, we consider the case of 2 eggs and n floors. The equation below represents the solution.

$$n \leq \frac{x(x + 1)}{2}$$

$$0 \leq \frac{x^2 + x - 2n}{2}$$

Solving by the quadratic formula gives the following. I have added the ceiling function to ensure we are considering an algorithm to cover $\geq n$ floors.

$$x = \left\lceil \frac{-1 \pm \sqrt{1 - 4 * 1 * -2n}}{2} \right\rceil = \left\lceil \frac{-1 \pm \sqrt{1 + 8n}}{2} \right\rceil$$

The worse-case runtime of this algorithm is equal to the block size, x, same as above.