CSCE 586 - Design and Analysis of Algorithms
Date: Wednesday 10th October, 2018
Name: Micah Hayden
Documentation: I utilized stack overflow for information on some LaTeX syntax. I used the Desmos online graphing calculator to visualize the six functions for Chapter 2 Problem #3. I discussed the different ways of assigning the priority lists to the port/ship problem with 2Lt Mireles.

Chapter 1, Problem #4

Problem Statement:

Develop an algorithm to assign at most *n* students to *m* hospitals where each hospital has ≥ 1 students.

Description of Algorithm:

This algorithm utilizes the standard Gale-Shapley algorithm with two small differences: it uses a set of hospital slots as opposed to a set of hospitals; and it terminates without regard to whether or not all students are matched (if there are fewer slots than students, not all students will have a matching hospital)

Algorithm 1 Stable Matching Algorithm with multiple students per hospital

```
Let M be the set of hospital slots
Let N be the set of available students.
Initially, all m ∈ M and n ∈ N are free.
while there is a hospital slot m free do:
    Choose the first hospital slot m in M
    Let n be the highest preferred student on m's list.
    if n is free then:
        Match (m, n)
    else                                          ▷ n is currently matched with m'
        if n prefers m to m' then:
            (m, n) become matched
            m' becomes free
        else
            m remains free.
        end if
    end if
end while
```

This algorithm terminates once all hospital slots are filled (when there are no hospital slots *m* free), and does not care if there are unassigned students remaining. This solves the case in which $n \geq m$.

Asymptotic Analysis:

There are *m* hospital slots available, and for each slot there are *n* students available to extend an offer to. There are at most $m \cdot n$ selections to attempt: n for m1, n for m2, etc. until the final hospital slot. Every other operation can occur in constant time. Thus, this algorithm will execute in $O(m \cdot n)$.

Proof:

Instability #1: there are students s and s', and a hospital h such that:
- s is assigned to h
- s' is free

- h prefers s' to s

Because *h* prefers *s'* to *s*, *h* must have offered *s'* first. *s'* must accept because *s'* is free.

Instability #2: There are students s and s', and hospitals h and h' such that:

- s is assigned to h
- s' is assigned to h'
- h prefers s' to s
- s' prefers h to h'

Because h prefers s' to s, h would have offered a spot to s' first.
if s' was free **then**
    h and s' would be matched.
end if
if s' was matched with h' **then**
    s' would break with h' and accept its preferred school h.
    h' then becomes free, extending an offer to its next highest student, s.
    s accepts the invitation because it was free.
end if

Difficulty and Time:

I would rate this problem as a difficulty 3, it followed the main G-S algorithm with only small modifications. I spent about 30 minutes between solving and presenting the problem.

Chapter 1, Problem #6

Problem Statement:

Given the schedule for each ship $s \in S$, that visits each port $p \in P$, find a truncation of each so that condition (†) continues to hold: no two ships are ever in the same port on the same day. Show that such a set of truncations can always be found, and give an algorithm to find them.

Description of Algorithm:

This algorithm utilizes the standard Gale-Shapley algorithm with two sets of priority lists. Each ship s prioritizes the ports in chronological order. Each port p prioritizes the ships in reverse-chronological order. A ship will propose to the first port it visits, after which the port will "trade up" to ships that appear later. If the ship is rejected or the port trades the ship away, the ship will then propose to its next destination. Thus, there will never be an occurrence where two ships are at the same port.

Algorithm 2 Stable Matching Algorithm for Ship Route Truncation

```
Let S be the set of ships
Let P be the set of ports.
Initially, all s ∈ S and p ∈ P are free.
s prioritizes ports p in chronological order
p prioritizes ships s in reverse-chronological order
while there is a ship s without an assigned stopping port p do:
    Let p be the highest preferred port on s's list.
    if p is free then:
        Match (s, p)
    else                                          ▷ p is currently matched with s'
        if p prefers s to s' then:
            (s, p) become matched
            s' becomes free
        else
            s remains free.
        end if
    end if
end while
```

This algorithm will terminate once all ships s have a matching port p at which to stop.

Asymptotic Analysis:

Similarly to problem 4, there is a loop of all the ships, s, which can consider at most each port p. Thus, there are a total of $s \cdot p$ combinations. Because all other operations can occur in constant time, this algorithm executes in $O(s \cdot p)$.

Proof:

Completion: To end, each ship must be matched to a port; each port can have a most one ship assigned to it.
Incompatibility #1: † is violated, 2 ships are in the same port p at once. This can only occur if a ship truncates its path, and another continues due to the original scheduling constraints.

Assume s is paired with port p.

- s' has not found a match, thus it proposes to p.
- Because s' would arrive later than s, p prefers s' to s.
- Thus, s would not stop at p because it would have been freed.

The reverse-ordering of the ports solves the incompatibility: each port will accept the latest ship to arrive, respectively. Each ship will either stop, and is the last to arrive at its stopping point; or it will propose to a future destination.

Difficulty and Time:

I would rate this problem as a difficulty 7, it followed the main G-S algorithm with only small modifications. The main cognitive shift was figuring out the prioritization of the list of ships and ports. I spent about 45 minutes, the majority of which was spent on finding how to prove that the algorithm does not violate the condition †.

Chapter 2, Problem # 3

Problem Statement:

Arrange the list of functions in ascending order of growth rate. That is, if $g(n)$ follows $f(n)$, then $f(n)$ should be $O(g(n))$.

$$f_1(n) = n^{2.5}$$
$$f_2(n) = \sqrt{2 \cdot n}$$
$$f_3(n) = n + 10$$
$$f_4(n) = 10^n$$
$$f_5(n) = 100^n$$
$$f_6(n) = n^2 \cdot \log n$$

Solution:

The first step in this solution is to inspect the functions to see their rough growth rate. Functions 1-3 all grow in polynomial time, $f(n) = n^x$ where $x = 2.5,\ 0.5,$ and 1, respectively; functions 4-5 grow in exponential time, $f(n) = x^n$, with $n = 10$ and 100, respectively; and function 6 has polynomial and logarithmic elements.
The slowest growth rate of these functions are those with polynomial growth, and the fastest growth rate is exponential. Finally, function 6 needs to be compared to the polynomial functions because of its polynomial term.

Polynomial Comparisons:
$f_2(n)$ and $f_3(n)$ are both $O(f_1(n))$ because $\sqrt{n} \leq n^1 \leq n^{2.5}$ for $n \geq 1$
$f_2(n)$ is $O(f_3(n))$ because $n^{0.5} \leq n^1$ for all $n \geq 1$
$f_2(n)$ is $O(f_6(n))$ because $n^{0.5} \leq n^2$ for all $n \geq 1$.
$f_3(n)$ is $O(f_6(n))$ because $n \leq n^2$ for all $n \geq 1$.

To compare $f_1(n)$ and $f_6(n)$, divide both functions by n^2. We are left with $f_1(n) = n^{0.5}$, and $f_6(n) = \log n$. $\log n \leq n^{0.5}$ for all $n \geq 0$. Thus, $f_6(n)$ is $O(f_1(n))$.
These can be ordered as follows: $f_2(n), f_3(n), f_6(n), f_1(n)$.

Exponential Comparison: $f_4(n) \leq f_5(n)$ for all $n \geq 0$.

Final Ordering:

$$f_2(n) = \sqrt{2 \cdot n}$$
$$f_3(n) = n + 10$$
$$f_6(n) = n^2 \cdot \log n$$
$$f_1(n) = n^{2.5}$$
$$f_4(n) = 10^n$$
$$f_5(n) = 100^n$$

Difficulty and Time:

I would rate this as a difficulty of 1. The functions can be ordered relatively simply, with the main challenge being writing it in an adequate form. I spent roughly 15 minutes on this problem.