CS180 Fall 19 Homework 1

1. Exercise 3 page 22

Prove “For every set of TV shows and ratings, there is not always a stable pair of schedules”. We will show this by giving an example of a set of TV shows and associated ratings for which there is no stable pair of schedules. (i.e. (b))

Suppose Network A has two TV shows with ratings 1 and 3, and Network B has two TV shows with ratings 2 and 4. Possible pairs of schedule S and T are

1. S=[1,3], T=[2,4]
2. S=[1,3], T=[4,2]
3. S=[3,1], T=[2,4]
4. S=[3,1], T=[4,2]

Case 1 (S=[1,3], T=[2,4]):
Network A can switch 1 and 3 (i.e. S’=[3,1]).
Then Network A wins B one more slot.
Thus, this is not a stable pair

Case 2 (S=[1,3], T=[4,2]):
Network B can switch 2 and 4 (i.e. T’=[2,4])
Then Network B wins A one more slot.
Thus, this is not a stable pair.

Case 3 (S=[3,1], T=[2,4]):
Network B can switch 2 and 4 (i.e. T’=[4,2])
Then Network B wins A one more slot.
Thus, this is not a stable pair.

Case 4 (S=[3,1], T=[4,2]):
Network A can switch 1 and 3 (i.e. S’=[1,3]).
Then Network A wins B one more slot.
Thus, this is not a stable pair.

Therefore, any possible pairs of S and T are not stable.
Q.E.D.

2. Exercise 4 page 22

There is always a stable assignment of students to hospitals because both types of instability can be resolved.

Type 1:
Remove assignment between s and h
Add assignment between s’ and h

Type 2:
Remove assignment between s and h
Remove assignment between s’ and h’
Add assignment between s and h’
Add assignment between s’ and h

Now in both cases, the assignments are stable.

The algorithm is the following:

```
While there exists a hospital h that has open positions:
    s is the next student on h’s preference list
    If s has no assignment:
        Add assignment between s and h
    Else:
        h’ is the hospital s is currently assigned to
        If s prefers h’ to h:
            Do nothing
        Else:
            Remove assignment between s and h’
            Add assignment between s and h
```

<u>3. Exercise 6 page 25</u>
We can translate the problem into the Stable Matching Problem (i.e. Finding a stable matches between ships and ports, where a match between ship s and port p indicated s stops at p.)

We can construct a preference list of a ship by the order of its visiting ports. For example, if the schedule of a ship is [port p1, at sea, port p2, at sea], then the preference list of the ship is [p1, p2].

From a port's point of view, it wants to have the ship that come at last to stop at its place. Then the port does not need to worry about having another ship visiting after it already has a ship stopping. Thus, we can construct a preference list of a port by reverse-order of visiting ships. For example, suppose there are ship s1, s2, and s1 visits p1 first before s2 visits p1. Then the preference list of p1 is [s2, s1].

Now we need to show this stable matching problem we just defined does not violate the requirement (i.e. No two ships can be in the same port on the same day). We will prove this by contradiction.

Suppose there exists a match that violates the requirement.
(i.e. There exists a ship s that visits port p, at which ship s' is already stopping)
Then p's preference list is [..., s, …, s', ...] since s' visits p before s does.
Then s's preference list is [..., p, …, p', ...] where p' is where s will actually stop.
This can't happen a stable match would never connect s' and p given p's preference list is [..., s, …, s', ...] and s's preference list is [..., p, …, p', ...].
Thus contradiction.
Q.E.D.

The algorithm is the following:

```
Construct preference lists of ships by order of visiting ports
Construct preference lists of ports by reverse-order of visiting ships
While there exists a ship s that has no match:
    p is the next port s will visit
    If p is not matched yet:
        Connect s and p
    Else:
        p is matched with a ship s'
        If p prefers s' to s:
            do nothing
        Else:
            Disconnect s' and p
            Connect s and p
```

4. Exercise 4 on page 67

$$2^{\sqrt{logn}} < n(logn)^3 < n^{\frac{4}{3}} < n^{logn} < 2^n < 2^{n^2} < 2^{2^n}$$

Proof:

$2^{\sqrt{logn}} < n(logn)^3$ since $log(2^{\sqrt{logn}}) = \sqrt{logn}log2 < logn + 3log(logn) = log(n(logn)^3)$

$n(logn)^3 < n^{\frac{4}{3}}$ since $log(n(logn)^3) = logn + 3log(logn) = x + 3logx < x + \frac{1}{3}x = \frac{4}{3}x = \frac{4}{3}logn = log(n^{\frac{4}{3}})$
where $x = logn$

$n^{\frac{4}{3}} < n^{logn}$ since $log(n^{\frac{4}{3}}) = \frac{4}{3}logn < (logn)^2 = log(n^{logn})$

$n^{logn} < 2^n$ since $log(n^{logn}) = (logn)^2 < nlog2 = log(n^2)$

$2^n < 2^{n^2}$ since $n < n^2$

$2^{n^2} < 2^{2^n}$ since $n^2 < 2^n$

5 (a)

Prove $P(n) : 1 + 2 + ... + n = \frac{n(n+1)}{2}$ by induction

Base Case (n=1):

$1 = \frac{1(1+1)}{2}$

∴P(1) holds

Inductive Step:

Suppose P(n) holds ($n \geq 1$) … (*)

(i.e. $1 + 2 + ... + n = \frac{n(n+1)}{2}$)

We want to show P(n+1) holds (∵ (*))

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

∴ P(n+1) holds

Q.E.D.

5 (b)

Prove $P(n) : 1^3 + 2^3 + ... + n^3 = (\frac{n(n+1)}{2})^2$ by induction

Base Case (n=1):

$1^3 = (\frac{1}{2} \cdot 1 \cdot (1+1))^2$

∴ P(1) holds

Inductive Step:

Suppose P(n) holds

i.e. $1^3 + 2^3 + ... + n^3 = (\frac{n(n+1)}{2})^2$ … (*)

We want to show P(n+1) holds

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

∴ P(n+1) holds

Q.E.D.

6. Two Egg Problem

The idea of 2-egg problem is that we want to minimize the number of tries in the worst case. A simple idea is to binary search with the first egg, then perform linear search with the second egg after the first egg breaks. We can modify this algorithm so that it distributes number of tries across all possible cases. We will drop the egg from bottom to up. The question is what the interval should be. We decide the interval such that the number of linear search will be reduced by 1 every time we drop the first egg. For example, we drop the first egg at floor x. If it breaks we do x-1 linear searches. If it doesn't break, we drop the egg at x+(x-1) floor. With this approach, the number of drops are the same no matter when the first egg breaks (always x drops). Thus, we want to divide the floors into x, x-1, …, 1. The sum should be the number of floors.

$$x + (x-1) + ... + 1 = 200$$
$$\tfrac{1}{2}x(x+1) = 200$$
$$x^2 + x - 400 = 0$$
$$x = \tfrac{-1+\sqrt{1601}}{2} = \tfrac{39.01...}{2} = 19.5...$$

Therefore, we need 20 tries in the worst case with 200 steps. To find out how many tries we need in general, just replace 200 with n.

$$x + (x-1) + ... + 1 = n$$
$$\tfrac{1}{2}x(x+1) = n$$
$$x^2 + x - 2n = 0$$
$$x = \tfrac{-1+\sqrt{1+8n}}{2}$$

Therefore, we need $\frac{-1+\sqrt{1+8n}}{2}$ tries in the worst case with n steps.