Problem 1 —

Consider the following sets of TV shows for networks A, and B, respectively.

$S: [A_1, A_2]$ $T: [B_1, B_2]$

For the case of 2 tv shows as represented in S and T, there are two slots that can be filled by these tv shows.

Let's take the following set of schedules and ratings for the 2 time slots:

Slot 1 — A_1: 4 B_1: 3 where $A_1 > B_1 > A_2 > B_2$ in

Slot 2 — A_2: 2 B_2: 1 term's of the shows ratings.

The pair of schedules (S,T) yields a schedule of A_1 followed by A_2 because A_1 rating was greater than B_1's rating in slot 1, and A_2's rating was greater than B_2's slot in slot 2.

This is not a stable schedule, however, as there is a schedule, call it T', that yields more time slots for network B. That is, $T': [B_2, B_1]$ s.t.

Slot 1 — A_1: 4 B_2: 1 $\Rightarrow$ A_1 wins because $A_1 > B_2$

Slot 2 — A_2: 2 B_1: 3 $\Rightarrow$ B_1 wins because $B_1 > A_2$

In schedule (S,T') we notice that B has won one more game than schedule (S,T). Therefore this set of shows and associated ratings have no stable pairings because B can always change its schedule to win more shows, and for a pairing to be stable, both A and B should be unable to unilaterally change its schedule to win more spots.

~~[illegible]~~

∴ By work of ~~[illegible]~~ a counter-example, we've demonstrated that there is not always a stable pairings ~~[illegible]~~ for every set of shows and associated rating.

Problem 2 —

a) Yes, there always exists a perfect match with no strong stability.

Algo

- Initially all $m \in M$ and $w \in W$ are free
- order each $m \in M$ and $w \in W$ s.t. if m is indifferent between w and w'..., then choose w, and vice versa. (Ranks ordered list on first available element)
- every woman for which $(m,w) \notin F$
  - choose such man m
    - let w be highest ranked women in m's pref list that is not already matched
    - if w free
      - (m,w) matched
    - else w engaged to m'
      - if w prefers m' to m
        - m still free
      - else w prefers ~~m~~ to m'
        - (m,w) matched
        - m' free
      - end if
    - endif
- endwhile

Proof $m, m' \in M$ and $w, w' \in W$ and are free

Consider ~~2 arbitrary elements in W~~. By way of contradiction, we know that (m,w') could not have been a pair. The ordering on line 2 of the algorithm determines that $w > w'$ in preference list. That means for the following 2 cases:

1) if m proposed to w', that means $w' > w$ which is a contradiction and

~~2) if m was accepted by w'', a third woman $\in W$,~~

2) if w had to prefer a third $m'' \in M$, then $m'' > m$ its clear that

$m' > m''$ because m' would have proposed to w', which by transivity means $m' > m$, which is also a contradiction.

Hence, the solution has no strong stabilities with the issue of indifferences as long as those indifferences are given an arbitrary ordering, like that of choosing $w > w' > w''$... and same for m.

b) Suppose that both m and m' are indifferent to w, and w', but w prefers $m > m'$ and so does w'.

In this case, we have 2 perfect matchings where:

(m, w) and (m', w') are matchings

(m, w') and (m', w) are matchings

Both of these matchings are unstable because w prefers m to m' and m is indifferent in both cases, hence both are weak instabilities. Therefore all perfect matchings are weak instabilities, meaning there does not always exist a perfect matching w/o weak instabilities.

Problem 3 —

· Let I be the set of all inputs, and O the set of all outputs, with an arbitrary Pattern in which the input wires and output wires meet.

· order all preference lists of $i \in I$ such that each input wire i prefer output wires in the order that the stream meets each output wire from source to terminus

· order all preference lists of $o \in O$ s.t. each output wire o prefer input lines in the order of downstream junctions to upstream junctions, without any ties.

· all $i \in I$ and $o \in O$ are unswitched and haven't proposed switching to each $o \in O$

· while input wire i is unswitched

  · choose input wire i

    · let o be output wire with highest preference for i to which i hasn't already matched

    · if o free

      · (i,o) is new switched ordering

    · else o matched with i'

      · if o prefers i' over i

        · i remains unswitched

      · else o prefers i over i'

        · (i,o) switched

        · i' free

      · endif

    · endif

· endwhile

By way of contradiction, suppose that there exists two stable matchings (i,o) and (i',o') that meet at some junction. This junction could either be on line i or o for (i,o) and on line i' or o' for (i',o').

This would imply that the junction is not on both i and i', and similarly, not on o and o', because that is contrary to the problem statement on the definition of a junction.

This would mean that the junction is either on the stream where i' hits o (i',o) or the stream where i hits o', (i,o').

These are contradictions:

~~if (i,o) hits junction, then matching (i',o) indicates that i' prefers $o > o'$ when it should prefer $o' > o$ because the junction (i',o) is upstream to the junction (i,o) [illegible]~~

~~if (i',o') hits junction, then matching (i,o) indicates that o' prefers~~

1. $J(i,o')$

- (i,o) passes J
- J upstream to junction $(i,o) \Rightarrow i$ prefers $o' > o$
- (i',o') passes J, J is downstream to $(i',o') \Rightarrow o'$ prefers $i > i'$

since o' prefers i, this in instability

~~because~~

2. $J(i',o)$

- (i',o') passes J
- J upstream to junction $(i',o') \Rightarrow$ i' prefers $o > o'$
- J downstream (i,o) when (i',o') passes J so o prefers $i' > i$
- since o prefers i', this is an instability

This means that after switching, there cannot be a case where two pairs meet at the same junction. Therefore, by proof of contradiction there can only exist stable pairings.

Problem 4. —

First let's find the number of total operations the computer can perform in an hour.

$$10^{10} \text{ per second} = \frac{10^{10}}{1s} \cdot \frac{60s}{1min} \cdot \frac{60min}{1hr} = \frac{3.6 \cdot 10^{13}}{hr}$$

* I assume that n must be a whole number.

a) The input size upper bound is that when input size = num of total operations done in an hour

Thus,

$$n^2 = 3.6 \cdot 10^{13} \implies n = \sqrt{3.6 \cdot 10^{13}}$$

$$\boxed{\text{input size} = 6 \cdot 10^6}$$

b)

$$n^3 = 3.6 \cdot 10^{13} \implies n = \sqrt[3]{3.6 \cdot 10^{13}}$$

$$\boxed{\text{input size} = 33019}$$

c)

$$100n^2 = 3.6 \cdot 10^{13} \implies n = \sqrt{3.6 \cdot 10^{11}}$$

$$\boxed{\text{input size} = 6 \cdot 10^5}$$

d)

$$n \log n = 3.6 \cdot 10^{13} \implies \cancel{\log n = \frac{3.6 \cdot 10^{13}}{n}}$$

$$\cancel{n^2 = 10^{3.6 \cdot 10^{13}}}$$

$$\cancel{n \ln(n) = 3.6 \cdot 10^{13}}$$

$$\cancel{\ln(n^n) = 3.6 \cdot 10^{13}}$$

$$\cancel{n^n = e^{3.6 \cdot 10^{13}}}$$

$$\cancel{n = \sqrt[n]{e^{3.6 \cdot 10^{13}}}}$$

~~input size =~~

$\implies n \log(n) = 3.6 \cdot 10^{13}$

this question is unsolvable with any methods I've learned from earlier courses. So, using wolfram alpha I found that $\boxed{\text{input size} = 1.29 \cdot 10^{12}}$

e) $2^n = 3.6 \cdot 10^{13} \Rightarrow \ln(2^n) = \ln(3.6 \cdot 10^{13})$

$\cancel{\ln(2^n) = 3.6 \cdot 10^{13}}$ $\qquad n \ln(2) = \ln(3.6 \cdot 10^{13})$

$\cancel{n = \frac{3.6 \cdot 10^{13}}{\ln(2)}}$ $\qquad n = \frac{\ln(3.6 \cdot 10^{13})}{\ln(2)}$

$\cancel{n = 5 \cdot 10^{13}}$ $\qquad \boxed{\text{input size} = 45}$

f) $2^{2^n} = 3.6 \cdot 10^{13} \Rightarrow \ln(2^{2^n}) = \ln(3.6 \cdot 10^{13})$

$2^n \ln(2) = \ln(3.6 \cdot 10^{13})$

$\ln(2^n) = \ln\left(\frac{\ln(3.6 \cdot 10^{13})}{\ln(2)}\right)$

$n = \frac{\ln\left(\frac{\ln(3.6 \cdot 10^{13})}{\ln(2)}\right)}{\ln(2)}$ $\qquad$ from e) ≈ 45

$n \approx 5 \Rightarrow \boxed{\text{input size} = 5}$

Problem 5 —

Let $P(n) = \frac{n(n+1)}{2} = 1 + 2 + \ldots + n$

1. Base case.

For $n = 1$, $P(1) = 1 = \frac{1(1+1)}{2}$

$$1 = \frac{2}{2}$$

$$1 = 1 \Rightarrow LHS = RHS$$

Therefore $P(n)$ is true for the base case $n = 1$.

2. Induction.

Assume for induction that for some arbitrary number k, $n = k$ s.t.

$$P(n) = P(k) = 1 + 2 + \ldots + k = \frac{k(k+1)}{2}$$

is true.

We must prove $n = k+1$. We notice

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

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

$$= \frac{k^2 + k + 2k + 2}{2}$$

$$= \frac{k^2 + 3k + 2}{2}$$

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

We see that adding a term $k+1$ gives us $\frac{(k+1)(k+2)}{2}$, which is what we get if we plug $n = k+1$ into $P(n)$. Therefore, by induction, this retains true for $n = k$.

$$\therefore 1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

Problem 6 —

$$1^2+2^2+3^2+\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$$

Let $P(n) = \frac{n(n+1)(2n+1)}{6}$

Base case.

For $n=1$,

$$1^2 = P(1)$$
$$= \frac{1(1+1)(1\cdot 2+1)}{6}$$
$$= \frac{6}{6}$$
$$1 = 1$$

Therefore, $P(n)$ is true for base case $n=1$.

Induction step.

Assume for an arbitrary number k that $P(n)$ is true for $P(k)$ s.t.

$$1^2+2^2+3^2+\dots+k^2 = \frac{k(k+1)(2k+1)}{6}$$

Then for some $n=k+1$, $P(k+1)$ must also be true. We notice

$$1^2+2^2+3^2+\dots+k^2+(k+1)^2 = \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$
$$= (k+1)\left[\frac{k(2k+1)}{6} + k+1\right]$$
$$= (k+1)\left[\frac{2k^2+k}{6} + \frac{6(k+1)}{6}\right]$$
$$= (k+1)\left[\frac{2k^2+7k+6}{6}\right]$$
$$= (k+1)\left[\frac{(k+2)(2k+3)}{6}\right]$$
$$= \frac{k+1(k+2)(2k+3)}{6}$$

We notice that $P(k+1) = \frac{(k+1)(k+2)(2k+3)}{6} =$

$\therefore$ By induction, $P(k+1)$ remains true. Hence, $1^2+2^2+3^2+\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$

Problem 7 →

Given 2 eggs, we can use the eggs to calculate the number of tries it would take to guess what floor it'll break at.

First, lets take the case of 1 egg. The only way to use 1 egg to find the num of floors it'll take to break is by starting from the bottom one floor at a time. This means that our second egg will <u>have</u> to use this method to determine the floor the egg will break at. That means the first egg will have to be strategically dropped in order to maximize the risk of where to the egg.

We could set a fixed number of increments to drop the egg from: i.e.

if 200 eggs →

200
⋮
20
10 } 10 floors

and then once the egg drops it has to be one of the 10 floors from the floor dropped at and under.

The question now remains how to maximize the above algorithm, as it's worst case for 200 steps is $(200)/10 + 9 = 29$. tries, ~~or to generalize, if f is the number of floors~~ and ~~then f f~~

We can notice that if we had 10 floors

10
9
8] 1 egg
7
6
5] 1 egg
4
3
2
1] 1 egg

Worst num of tries 3 tries or if we had used a fixed number like 3 or 2, we'd get 4 tries and 5 tries respectively.

This shows that we have the decrement the search range by -1 for every increment of floors the worse eggs.

This means that for m increments, we decrement -1 ⟹ $m + (m-1) + (m-2) + \dots + (1)$ = worst

As shown in problem 5,

$$m + (m-1) + (m-2) + \ldots + 1 \Rightarrow 1^2 + 2^2 + \ldots + m^2 = \frac{m(m+1)}{2} \geq \text{floors}$$

Hence, the ~~best~~ worst number of tries will be $\frac{m(m+1)}{2} \geq n$

For 200 tries,

$$\frac{m(m+1)}{2} \geq 200 \Rightarrow \frac{m^2 + m - 400}{2}$$

$$\Rightarrow \frac{-1 \pm \sqrt{1^2 - 4(1)(-400)}}{2}$$

$$\Rightarrow m \approx 19.5 \Rightarrow m = 20 \text{ floors}$$

With this algorithm:

- let n be the number of floors
- drop egg 1 in m increments where $n \geq \frac{m(m+1)}{2}$
- ~~drop egg~~ while egg 1 doesn't break
  - move egg 1 up $m-1$ floors
- if egg 1 does break
  - while egg 2 has not broken
    - move egg 2 up 1 floor
    - if egg 2 does not break

      continue
  - else, ~~th~~
    - solution is num floors, exit

Therefore, with 200 tries, the worst case is 20 floors and for n floors, the worse case is $\frac{-1 \pm \sqrt{1^2 - 4(1)(n)}}{2}$.