Jeremy Cristobal

604982952

CS 180 Midterm

Discussion 1B

Problem 1

Variable key:

P, R, T = set of universes
Q = set of ordered pairs of costs and connections
S, V = set of connections   A, B = cost

```
P, Q, R, S = empty set
while(some universe in W is not in P){
  Choose such universe u
  Add u to set P
  for(each universe not in P){
    Choose such universe v
    Use estimated cost function for u & v
    Add (edge, cost) to Q if it exists
  }
}
A = 0
while(P is not empty){
  T, V = empty set
  B = 0
  Remove a universe u from P and add u to T
  while(T contains a universe with a connection that
                would not form a cycle){
    Choose cheapest valid connection from universes in T
    Add cost to B
    Add connection to V and remove it from Q
    Add corresponding universe to T and remove it from P
  }
  if(T has as many or more elements than R){
    if(|T| = |R| and A < B){
      next loop}
    else{ Set R equal to T
          Set S equal to V
          Set A equal to B }
  }
  if(R has more elements than P)
    exit loop
}
```

Problem 1 (Part II)

This algorithm is correct, so let's walk through it. After creating our sets, we first pick any universe and throw it into our set. We estimate the cost of every connection then save the connection and cost in an ordered pair in our other set (Q). We do this for every universe and remove repeats through the use of our first set (P).

After this, our set P now contains all the universes we want to explore. So, we take a universe from P, remove it, and add it to our new set T. From there, we find the cheapest connection for u, add the universe to T and add the connection to V. We remove these from their previous sets because a universe or connection cannot be a part of two distinct disconnected subgraphs. So, we repeat this process until there are no more valid connections.

After exiting our nested loop, we compare our sets T and ~~$~~ V to our previously biggest sets; R and S. If T and V are bigger, we set R and S to T and V, ~~then empty~~ Then, if R has more elements than P, we are done. If not, we ~~pe~~ empty T and V and repeat the process. And if R = P, we take the cheaper one.

Our algorithm has two nested loops, meaning our complexity is about $T_A(x) = 2x^2$, which comes out to $\boxed{O(n^2)}$

It is not always possible to connect all universes in W because we are not ~~gaur~~ guaranteed that a connected graph exists. We could, for example, have two disconnected graphs, which would mean not all universes are connected.

Problem 2

The given statement is <u>false</u>.

Let's take a look at the following example:

Captains (A, B, C, D):

	1st	2nd	3rd	4th
A	E	G	H	F
B	F	E	H	G
C	G	F	E	H
D	H	G	E	F

Spaceships (E, F, G, H):

	1st	2nd	3rd	4th
E	A	C	D	B
F	B	A	D	C
G	C	A	B	D
H	D	B	C	A

As we can see, there is only one stable matching:
{(A, E), (B, F), (C, G), (D, H)} ~~This is because~~

All choices here are first choices, and therefore ~~one~~ none is the third ranked preference of another. Because this <u>is</u> an instance of Stable Spaceship matching, the statement is false.

In fact, if every captain is the top choice of said captain's top choice, then the statement will be false for any number n. ~~Furthermore~~

Furthermore, we see this happens because if a captain and a spaceship are each other's top choice, they are the only stable pairing for one another.

Problem 3

$\sqrt[3]{n} \Rightarrow n^{1/3} \Rightarrow O(n^{1/3})$ 6

$\log_{10} 2^n \Rightarrow n \log_{10} 2 \Rightarrow O(n)$ 4 (Tied)

$\sqrt[4]{n^3} \Rightarrow n^{3/4} \Rightarrow O(n^{3/4})$ 5

$\sqrt{n^n} \Rightarrow n^{n/2} \Rightarrow O(n^{n/2})$ 1

$\log_{10}^2 n + 5n \Rightarrow O(n)$ 4 (Tied)

$5n + 5^n \Rightarrow O(5^n)$ 2

$n^{7/2} \Rightarrow O(n^{7/2})$ 3

$\log_{10}(n^5 + 5n) \Rightarrow O(\log_{10}(n))$ 7

Descending Order:

1. $\sqrt{n^n}$ — Highest growth rate
2. $5n + 5^n$
3. $n^{7/2}$
4. Tie! $\log_{10} 2^n$ & $\log_{10}^2 n + 5n$
5. $\sqrt[4]{n^3}$
6. $\sqrt[3]{n}$
7. $\log_{10}(n^5 + 5n)$ — Lowest growth rate