Algorithm_HW1_1826630_Yao-Chung Liang

1.

False.

Using counterexample. If exists m belongs to M which proposed to w belongs to W and w is ranked first on man's preference list, but the man is not ranked first on the woman's preference list. Then result contradicts with the statement.

2.

True.

In every stable matching S for this instance, once they get engaged, no one can tear them apart (no man m' can replace man m).

3.

Algorithm:

```
# define s to be name of student
# define s' to be name of another student
# define h to be current name of hospital
# define h' to be another name of hopital
# define p to be positions available in the hospital
# status of each student:          free -> sign a temporary contract with hospital but
hospital can cancel it unilaterally -> get a job in a hospital or not
# define S to be the final result of some students who get job in certain hospital
function stable_match_student_hospital{
    initialize s belongs to student set ,h belongs to hospital set to be free (students
have no job, hospitals haven't recruited anyone)
    while (student s is free and not yet contract to every hospital before){
        s = the highest ranked student in hospital's preference list
        if (h still have positions){
            (s,h) sign a temporary contract
        }
        else(h have no more position available){
            if (h prefer s to s') do {
                (s,h) sign a temporary contract
                s be free
            }
            else(h still like student s){
                (s,h) remain the same
            }
```

```
        }
    }
    return all (s,h) to S
}
```

Show that there is always a stable assignment of students to hospital:

I think I need to prove two thing :

First is to prove the algorithm will result in stable matching. Second is to prove it always yeild the same matching.

First prove:

Assume my algorithm will result in unstable matching S.

Generally, there are two types of instability, I first consider the first type of instability. If h prefers s' to s, then from my algorithm h will have contract with s' and thus the first type of instability will no longer exist. Secondly, if second type of instabbility happens. If h prefers s' to s and s' prefers h to h' and s' is assigned to h, then in the beginning of the execution, s' would be assigned to h' already. Because s' selection of hospitals is in a descending order, so if he didn't picked by h', then this means hospital h' don't prefer s' to s and thus contradict with our assumption, thus second type of instability won't even exist. In the end, we know that this algorithm will result in stable matching.

Second prove :

Assume we already know that every student can have their best match of hospitals. However, in some execution E, s doesn't match with best(s) which I will say h. And s match with h'. But in this situation, another student s' must be kicked out by hospital h'. But because they can match with hospital only with descending order, thus this situation will occur during the execution E and result in a position from some hospital h'' would still be available and no student can get that position, which is a contradiction to the algorithm that it will stop and get a stable matching. Thus, I know every student should have their best matching. And by the time, I know there will only one situation of best matching, thus this means the algorithm will always yeild the same matching.

Finally, with the proves above, I can comfirm that there is always a stable ssignment of student to hospital.

There is always a stable assignment of students to hospitals,
The first type of instability won't exist because in my algorithm, if h prefers s' to s, then h will sign contract with s' rather than with s.
The second tye of instability won't exist because in my algorithm, if h prefers s' to s, then h will sign contract with s' rather than with s.
Since all type of instability won't arise, thus there is always a stable assignment.

----

runtime-Analysis:
Assume n students and n hospitals.
Since every student can talk to hospital and get a chance to sign a contract for one time, thus there will be n*n precedures.
Which means this is $O(n^2)$.

The first type of instability won't exist because in my algorithm, if h prefers s' to s, then h will sign contract with s' rather than with s.
The second tye of instability won't exist because in my algorithm, if h prefers s' to s, then h will sign contract with s' rather than with s.
Since all type of instability won't arise, thus my algorithm is correct.

4.
(a) n^2 : n= $6*10^6$
(b) n^3 : n= 33019
(c) $100*n^2$: n= $6*10^5$
(d) n log n : n=$2^{(18*10^{(12)})}$
(e) 2^n : n=45
(f) $2^{(2^n)}$: n=5

5.
$100^n > 10^n > n^{2.5} > n^2*\log(n) > n+10 > \sqrt{2*n}$
f5 > f4 > f1 > f6 > f3 > f2
proof is in the pdf file

6.
(a) True
$\log_2(f(n))=\log(f(n))/\log(2)$ is $O(\log(f(n))) = O(\log(g(n)))$
(b) True
$2^{(f(n))}$ transform to $f(n)*\log_2(2) = f(n)$ is $O(g(n))$
thus $2^{(f(n))}$ is $O(2^{(g(n))})$

(c) True

f(n)^2 trnsform to 2*log(f(n)) is O(log(g(n)))

thus f(n)^2 is O(g(n)^2)

Algorithm

HW 1: 1826630. Yao-Chung Liang

5. ⓐ $f_1(n) = n^{2.5}$

$n^{2.5} \leq 2 \cdot n^{2.5}$ so $n^{2.5}$ is $O(n^{2.5})$

$n^{2.5} \geq \frac{1}{2} \cdot n^{2.5}$ so $n^{2.5}$ is $\Omega(n^{2.5})$

$\Rightarrow$ so $n^{2.5}$ is $\Theta(n^{2.5})$

ⓑ $f_2(n) = \sqrt{2n} = \sqrt{2} \cdot n^{\frac{1}{2}}$

$\sqrt{2} \cdot n^{\frac{1}{2}} \leq 2 \cdot n^{\frac{1}{2}}$ so $\sqrt{2n}$ is $O(n^{\frac{1}{2}})$

$\sqrt{2} \cdot n^{\frac{1}{2}} \geq 1 \cdot n^{\frac{1}{2}}$ so $\sqrt{2n}$ is $\Omega(n^{\frac{1}{2}})$

$\Rightarrow$ so $\sqrt{2n}$ is $\Theta(n^{\frac{1}{2}})$

ⓒ $f_3(n) = n + 10$

$n + 10 \leq 2n$ $(\forall n \geq 10)$ so $n+10$ is $O(n)$

$n + 10 \geq n$ so $n+10$ is $\Omega(n)$

$\Rightarrow$ so $n+10$ is $\Theta(n)$

ⓓ $f_4(n) = 10^n$

$10^n \leq 2 \cdot 10^n$ so 10^n is $O(10^n)$

$10^n \geq \frac{1}{2} \cdot 10^n$ so 10^n is $\Omega(10^n)$

$\Rightarrow$ so 10^n is $\Theta(10^n)$

ⓔ $f_5(n) = 100^n$

$100^n \leq 2 \cdot 100^n$ so 100^n is $O(100^n)$

$100^n \geq \frac{1}{2} \cdot 100^n$ so 100^n is $\Omega(100^n)$

$\Rightarrow$ so 100^n is $\Theta(100^n)$

ⓕ $f_6(n) = n^2 \log n$

$n^2 \log n \leq 2 \cdot n^2 \log n$ so $n^2 \log n$ is $O(n^2 \log n)$

$n^2 \log n \geq \frac{1}{2} n^2 \log n$ so $n^2 \log n$ is $\Omega(n^2 \log n)$

$\Rightarrow$ so $n^2 \log n$ is $\Theta(n^2 \log n)$