a) Prove that there always exists a perfect matching that is weakly stable.

3 4
1
0
2 3
1 1 2 2
2 1 1 1
1 1 1 1
2 3 1 1
1 2 2 2
1 1 1 1
1 2 1 1

Suppose there does not exists a perfect matching that is weakly stable pair and tenant 0 appartment2 is not a weakly stable pair

Case 1:
if tenant 0 have no reference to which apartment is require then tenant 0 can switch to an other apartment and still be a stable pair

case 2:
if apartment 2 have no reference to which tenant to lease then apartment can switch to another tenant and still be a stable pair

in either case, since both the apartment and the tenant can switch without affect the stability of the algorithm, then there is a contraction.

(b) Give an algorithm in pseudocode (either an outline or paragraph works) to find a stable assignment.

Initially all m ∈ M and w ∈ W are free
while ∃ m who is free and hasn't proposed to every w ∈ W
do Choose such a man m
Let w be the highest ranked in m's preference list to whom m has not yet proposed
if w is free
then (m, w) become engaged
else w is currently engaged to m
if w prefers m to m
then m remains free
if m already taken or have no reference
then W must look for his next reference
else w prefers m to m
(m, w) become engaged
m becomes free back to the queue

(c) Give a proof of your algorithm's correctness. Remember that you must prove both that your algorithm terminates and gives a correct result.

I used Gale-Shapley algorithms. Since we proofed this in class, the algorithm is correct. Assume thereis an instability. Then there exist two pairs (m, w) and (m , w) in S s.t. m prefers w to w and w prefers m to m . During execution, m's last proposal must have been to w. Had m proposed to w at some earlier time? If not, w must be higher than w on m's preference list (which is a contradition to our assumption that m prefers w to w). If yes, then he was rejected by w in favor of some other guy m. Either m = m or w prefers m to m (since the quality of her match only goes up). Either way, this is a contradiction to our founding assumption.

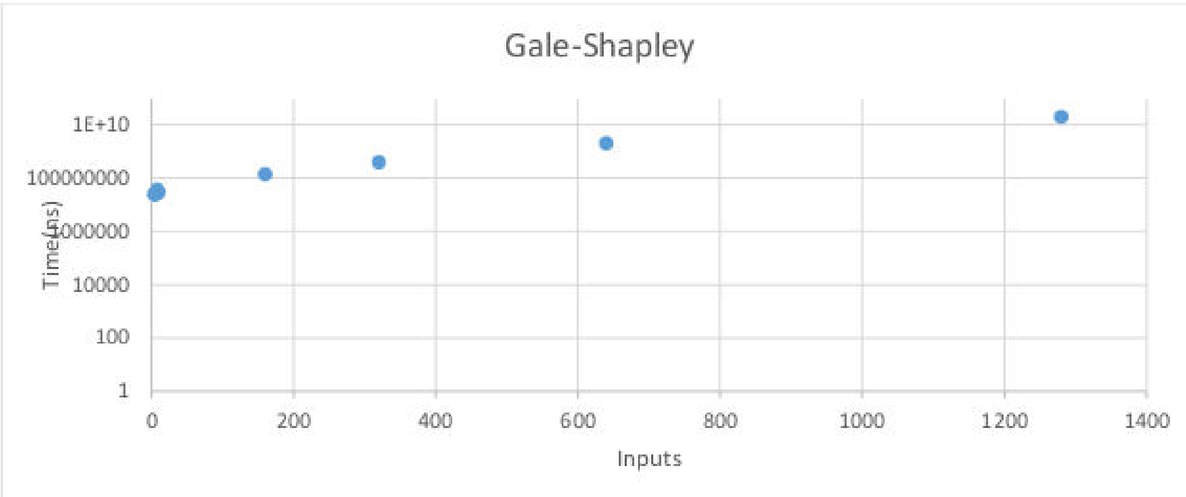
(d) Give the runtime complexity of your algorithm in Big O notation and explain why your proposal accurately captures the runtime of your algorithm.

The Big O for my efficient algorithm is O(n 2) because you have n tenants and n apartment and you have to compare each tenant to each of the apartment

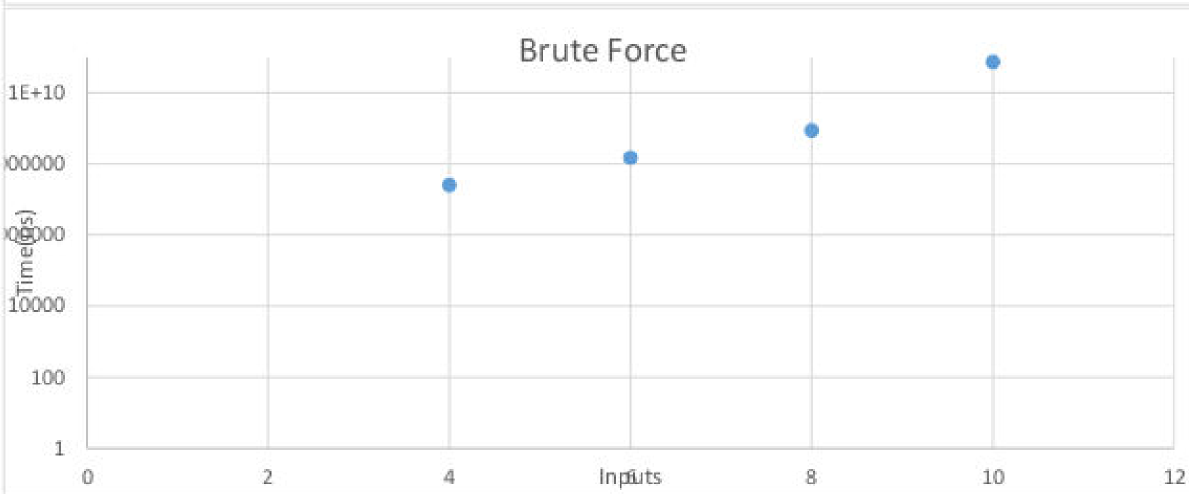
(e) Consider a Brute Force Implementation of the algorithm where you find all combinations of possible matchings and verify whether they form a weakly stable match one by one. Give the runtime complexity of this brute force algorithm in Big O notation and explain why.

The runtime for Brute Force Algorithms is Big O(n 2 n!) because you finding every permutation for every single pair.

(f)



Gale-Shapley
1E+10
100000000
1000000
10000
100
1
Time(ns)
0
200
400
600
800
1000
1200
1400
Inputs



Brute Force
1E+10
10000
100
1
Time(ns)
0
2
4
6
8
10
12
Inputs