Lab 1 Report

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

This problem can be mapped to the stable marriage problem.

First, we can map the tenants to the men, each already with their own sets of preferences. The apartments can be mapped to the women, each with their list of preferences represented by that of their landlord. The definition of marriage is represented by the renting of an apartment.

At this point, the problem is identical to the stable marriage problem with the exception of indifferences or ties in preference. To resolve this, we simply choose a random order among the ties and set that as the new preference list if there are ties within a set of preferences

Proof by contradiction

Let S be the set of matched pairs of tenants and apartments. Assume there is an instability with respect to S. Such an instability requires two pairs (T1, A1) and (T2, A2) in S such that:

- T1 prefers A2 to A1
- the landowner of A2 prefers T1 to T2

T1's last application is, by definition, included in S. There are two possible scenarios that resulted in this pair (T1, A1).

a. T1 applied to A2 earlier. If so, then T1 was rejected by the landowner of A2 because he prefers another tenant A3. Either T2 = T3 or A2's landowner prefers T2 to T3.
b. T1 has not applied to A2. If so, A1 must occur higher on T1's preferences.

It follows that S is a stable matching

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

There are two parts to solving this problem. The first will be to convert the current problem into a stable marriage problem. This means converting the preference of tenants and landlords from partial order to total order. The second part will be to execute the standard Gale–Shapley algorithm to find a stable set of matches between tenants and apartments.

```
Algorithm():
    Phase 1:
    For t in tenants:
        Convert preferences from partial order to total order
    For a in apartments:
        Convert preferences from partial order to total order
```

Phase 2:

Execute standard Gale-Shapley algorithm on tenants and apartments

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

Proof of termination

We let denote P(t) the set of pairs (T, A) such that T has applied to A by the end of iteration t. We see that for all T, the size of P(t+1) is strictly greater than the size of P(t). The value of P(t) can increase at most n^2 times (once for each possible pair of tenants and apartments).

Proof of correctness by contradiction

Assertion 1: If T is homeless at some point in the execution of the algorithm, then there is apartment to which he has not yet applied.

By contradiction, suppose tenant T is homeless, but has applied to all apartments. Each of the n apartments must be inhabited. Since each inhabited apartment has a corresponding tenant, all n tenants must be not homeless. This forms a contradiction

Assertion 2: The set of paired tenants and apartments always forms a matching. All tenants and apartments are present within the set.

By contradiction, assume a homeless tenant T has applied to every apartment, hence, terminating the algorithm. This directly contradicts Assertion 1, which says there cannot be a homeless tenant who has applied to every apartment.

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

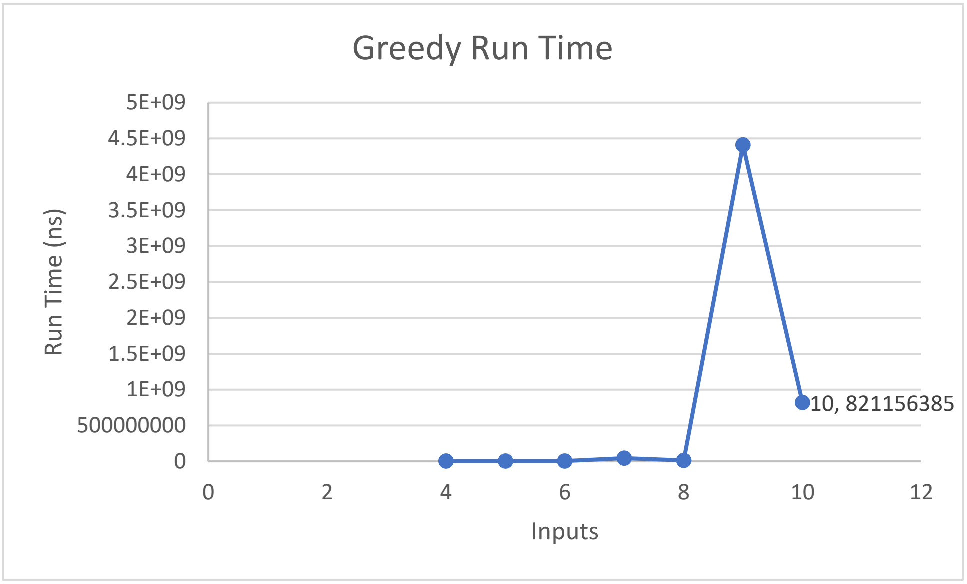
The runtime complexity of the proposed algorithm can be represented by the sum of the two phases in pseudocode. Phase 1 must examine all n tenants and apartments and sort their preference list to create a total order. Efficient sorting can be achieved in O (n log(n)). Phase 2 is done with the conventional Gale-Shapley algorithm, which is known to have a time complexity of O (n^2). Since phase 2 will dominate the runtime complexity of the algorithm, it can be assumed that the proposed algorithm will run at O (n^2).

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.

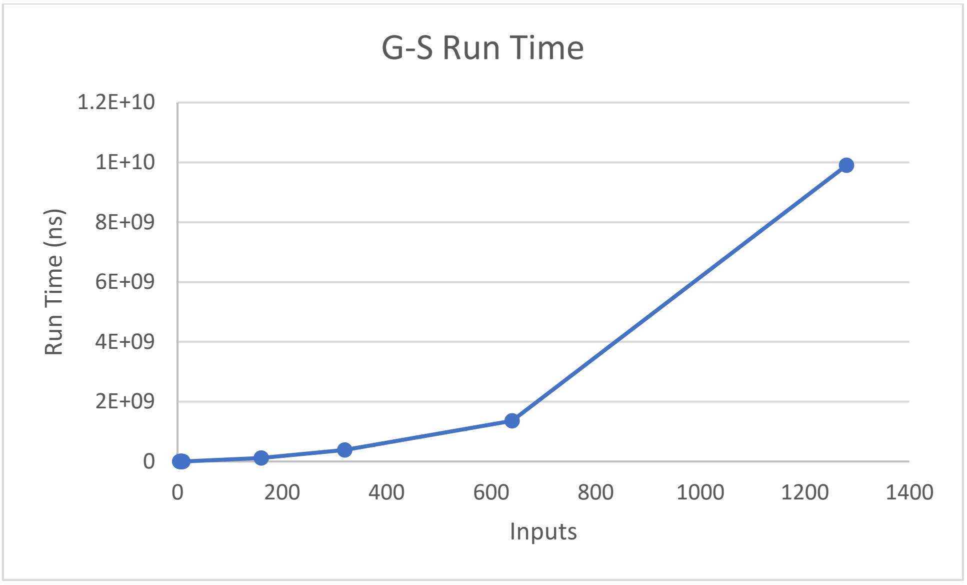
In the worst case, the brute force solution will examine every possible matchings and test to see if each is a stable matching. There are n! possible matchings and checking for a stable matching can at the worst case, require the examination of all pairs of tenant and apartment. Therefore, the runtime complexity of the brute force method will be O (n^2 * n!).

f) In the following two sections you will implement code for a brute force solution and an efficient solution. In your report, use the provided data files to plot the number of couples (x-axis) against the time in ms it takes for your code to run (y-axis).

The graph below shows the runtime of the Brute Force algorithm against random stable marriage problems of arbitrary n. It can be noted that the algorithm's runtime can fluctuate dramatically between runs. Though the maximum runtime can be O (n^2 * n!), the amount of tries that it takes to find a stable matching set can be arbitrary.



The graph below shows the runtime of the Gale-Shapley algorithm against random stable marriage problems of arbitrary n. The parabolic nature of the algorithm can be observed from the graph, but since the algorithm rarely takes anywhere close to the maximum amount of tries it takes to solve a given set, the realistic runtime of the algorithm can be places somewhere between linear and parabolic time.


G-S Run Time
Run Time (ns)
1.2E+10
1E+10
8E+09
6E+09
4E+09
2E+09
0
0
200
400
600
800
1000
1200
1400
Inputs