1. *Show that there is always a stable assignment of users to servers.*

Proof By Contradiction

Let us assume that there is an instability within the outcome of the algorithm.

The instability would contain a pair (*S, U*) and (*S’, U’*) such that *S* prefers *U’* and *S’* prefers *U*.

During the execution of the algorithm, *S*’s last connection was to *U*. The while loop terminated shortly after this connection was made. Since *S* prefers *U’* to *U*, this would suggest that during the execution of the algorithm, *S* connected with *U’* but *U’* rejected the connection. But this is a contradiction, since we stated that *U’* preferred *S* to *S’*. Thus by contraction, there is no instability in the outcome of the algorithm.

1. *Give an algorithm in pseudocode (either an outline or paragraph works) to find a stable assignment that is server optimal. Hint: it should be very similar to the Gale-Shapley algorithm, with servers taking the role of the men, and users of the women.*

Initially all servers *S* are free and no user *U* is assigned a server.

While there are servers *S* with slots *s* that are free.

Choose a server *S* with free slots

Let *F* be the highest-ranked user in *S*’s preference list to whom *F* is unassigned

If *F* is free then

(*F, S*) link together

Else *F* is already assigned to server *S*’

If *F* prefers *S*’ to *S* then

*F* remains free

Else *F* prefers *S* to *S*’

(*F,S*) link together

*S*’ becomes free

Endif

Endif

Endwhile

Return the set of linked pairs

1. *Give the runtime complexity of your algorithm in Big O notation and explain why.*

The runtime complexity of this algorithm is O(n2). This is because if every server had a preference list that is exactly opposite of the user’s list, it would then iterate through every server (size n) and iterate through ever user preference list (size n) for every iteration of the server. Thus, the worst runtime complexity of this algorithm is O(n2).

1. *Give a proof of your algorithm’s correctness. Remember that you must prove both that your algorithm terminates and gives a correct result.*
2. *Consider a Brute Force Implementation of the algorithm where you find all combinations of possible matchings and verify if they are a stable marriage one by one. Give the runtime complexity of this brute force algorithm in Big O notation and explain why.*

The runtime complexity would be O(n!). Since for every server (n), every user is assigned to it (n). The algorithm will then need to check every combinations of slots on the server(n!), thus O(n) + O(n) + O(n!) = O(n!).

1. *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 servers (x-axis) against the time in ms it takes for your code to run (y-axis). There are four small data files and four large data files included in the input provided. The large data files may be too large for the brute force algorithm to finish running on your machine. If that is the case, do not worry about plotting the brute force results for the large data files. Your plot should therefore contain 8 points from your efficient algorithm and 4-8 points from the brute force algorithm.*

*Please make sure the points from different algorithms are distinct so that you can easily compare the runtimes from the brute force algorithm and your efficient algorithm. Scale the plot so that the comparisons are easy to make (we recommend a logarithmic scaling). Also take note of the trend in run time as the number of servers increases.*