Empirical Analysis of Gale-Shapely Algorithm

# Abstract

The stable matching problem (or SMP) is the problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences for each element. The runtime complexity of this algorithm is O(n2). This means its performance is directly proportional to the square of the size of the input data set. This paper has been structured to describe the algorithm using pseudocode and showing results of experiments with different input sizes. Finally, after analyzing the data, we will see the data is consistent with the theory.

# Description of "Gale-Shapely" Algorithm

The **stable matching problem,** a practical execution of "Gale-Shapely" algorithm is the problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences for each element. A [matching](https://en.wikipedia.org/wiki/Matching_(graph_theory)) is a mapping from the elements of one set to the elements of the other set. A matching is *not* stable if:

1. There is an element *A* of the first matched set which prefers some given element *B* of the second matched set over the element to which *A* is already matched, and
2. *B* also prefers *A* over the element to which *B* is already matched.

In other words, a matching is stable when there does not exist any match (*A*, *B*) by which *both* *A* and *B* would be individually better off than they are with the element to which they are currently matched.

The Gale–Shapley algorithm involves a number of iterations:

1. In the first round, first *a*) each free man proposes to the woman he prefers most, and then *b*) each woman tentatively engages to her suitor she most prefers and rejects all other suitors. She is then provisionally "engaged" to the suitor she most prefers so far, and that suitor is likewise provisionally engaged to her.
2. In each subsequent round, first *a*) each free man proposes to the most-preferred woman to whom he has not yet proposed (regardless of whether the woman is already engaged), and then *b*) each woman tentatively engages if she is currently not engaged or if she prefers this guy over her current partner (in this case, she dumps her current provisional partner who becomes free again). The provisional nature of engagements preserves the right of an already-engaged woman to "trade up" (and, in the process, to "reject/dump" her until-then partner).
3. This process is repeated until everyman is engaged to every woman.

**Runtime complexity of the algorithm:** The runtime complexity of this algorithm is O(n2) where n is number of men or women.

### Pseudocode

A pseudocode of the stable matching problem is given below.

Initially all *m ∈ M* and *w ∈ W* are free

While there is a man *m* who is free and hasn’t proposed to

every woman

Choose such a man *m*

Let *w* be the highest-ranked woman 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

Else *w* prefers *m* to *m*

(*m*, *w*) become engaged

*m'* becomes free

Endif

Endif

Endwhile

Return the set *S* of engaged pairs

The algorithm ensures that:

1. The set *S* returned at termination is a perfect matching.
2. The set *S* returned at termination is a stable matching.
3. Every execution of the G-S algorithm results in the set *S\**, where *S\* = {(m, best(m)):m ∈ M}*
4. Everyone gets married.
5. The marriages are stable.

# Experiment

The following table is the summarized version of the *fit.log* file generated by program *gnuplot.* The table shows how many iterations have been used for different sizes (ranging from 1 to 100, 1000, 2000, 5500, etc.) and the initial set of parameter values used in '*guess.txt'.* The table gives a clear idea of final sum of squares of residuals and degrees of freedom for each execution.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of Iterations | Input sizes | Number of Iterations to converge the fit | Initial set of free parameter values | Final sum of squares of residuals | Degrees of Freedom |
| 20 | 1 - 5500 | 10 | a=4, b=0.2, c=1 | 152.67 | 17 |
| 13 | 1 - 1000 | 9 | a=0.1, b=0.2, c=0.2 | 0.0486497 | 10 |
| 20 | 1 - 2000 | 9 | a=0.1, b=0.2, c=0.2 | 1.36867 | 7 |
| 18 | 1 - 50 | 7 | a=1, b=1, c=1 | 6.10438e-06 | 15 |
| 10 | 1 - 100 | 7 | a=1, b=1, c=1 | 6.86497e-06 | 7 |
| 50 | 1 - 80 | 7 | a=1, b=1, c=1 | 7.53407e-05 | 41 |
| 40 | 1 - 50 | 7 | a=1, b=1, c=1 | 5.50137e-06 | 21 |
| 10 | 1 – 100 | 8 | a=0.1, b=0.2, c=0.2 | 5.91213e-05 | 7 |

# Conclusion

The runtime complexity of the Gale-shapely algorithm in O(n2), which means its performance is directly proportional to the square of the size of the input data set. Analyzing the data listed in the table above, we can see for sufficiently large enough input size (1000), calculated final sum of squares of residuals is 0.0486497, which is very small. So, we can claim the empirical analysis is consistent with the theory.