Gale–Shapley deferred acceptance algorithm

An intuitive method that guarantees to find a stable matching.

Gale–Shapley (preference lists for hospitals and students)

\[ M = \{ \} \]

1. Initialize \( M \) to empty matching.
2. While (some hospital \( h \) is unmatched and hasn’t proposed to every student)
3. \( s \) ← first student on \( h \)’s list to whom \( h \) has not yet proposed.
4. If (\( s \) is unmatched)
5. Add \( h \rightarrow s \) to matching \( M \).
6. Else if (\( s \) prefers \( h \) to current partner \( h' \))
7. Replace \( h' \rightarrow s \) with \( h \rightarrow s \) in matching \( M \).
8. Else
9. \( s \) rejects \( h \).
10. RETURN stable matching \( M \).

\[ \text{Output} \]

\[ n \times \text{complexity of iteration body} = O(n^3) \]
Proof of correctness: termination

Observation 1. Hospitals propose to students in decreasing order of preference.

Observation 2. Once a student is matched, the student never becomes unmatched; only “trades up.”

Claim. Algorithm terminates after at most $n^2$ iterations of while loop.

Pf. Each time through the while loop a hospital proposes to a new student. There are only $n^2$ possible proposals.

\[
\begin{array}{ccccc}
\text{1st} & \text{2nd} & \text{3rd} & \text{4th} & \text{5th} \\
\hline
\text{Atlanta} & A & B & C & D & E \\
\text{Boston} & B & C & D & A & E \\
\text{Chicago} & C & D & A & B & E \\
\text{Dallas} & D & A & B & C & E \\
\text{Eugene} & A & B & C & D & E \\
\end{array}
\]

\[
\begin{array}{ccccc}
\text{1st} & \text{2nd} & \text{3rd} & \text{4th} & \text{5th} \\
\hline
\text{Val} & W & X & Y & Z & V \\
\text{Wayne} & X & Y & Z & V & W \\
\text{Xavier} & Y & Z & V & W & X \\
\text{Yolanda} & Z & V & W & X & Y \\
\text{Zeus} & V & W & X & Y & Z \\
\end{array}
\]

$n(n-1) + 1$ proposals required
Proof of correctness: perfection

Gale–Shapley produces a matching.

Claim. In Gale–Shapley matching, all hospitals get matched.

Pf. Hospital proposes only if unmatched; student

Proof sketch

Claim. In Gale–Shapley matching, all hospitals get matched.

Pf. By contradiction, there exists a hospital $h \in H$ not matched upon termination.

If $s_1 \sim h$ then $s_2 \sim h$.

But, by Observation 2, $s_1$ was never proposed to $h$.

Then some student, say $s \in S$, is not matched upon termination.

Claim. In Gale–Shapley matching, all students get matched.

Pf. By previous claim, all $n$ hospitals get matched.

Thus, all $n$ students get matched.
Proof of correctness: stability

Claim. In Gale–Shapley matching $M^*$, there are no unstable pairs.
Pf. Suppose that $M^*$ does not contain the pair $h-s$.

- **Case 1:** $h$ never proposed to $s$.
  $\Rightarrow$ $h$ prefers its Gale–Shapley partner $s'$ to $s$.
  $\Rightarrow$ $h-s$ is not unstable pair.

- **Case 2:** $h$ proposed to $s$.
  $\Rightarrow$ $s$ rejected $h$ (right away or later)
  $\Rightarrow$ $s$ prefers Gale–Shapley partner $h'$ to $h$.
  $\Rightarrow$ $h-s$ is not unstable pair.

- In either case, the pair $h-s$ is not unstable. ■
Summary

Stable matching problem. Given \( n \) hospitals and \( n \) students, and their preferences, find a stable matching if one exists.


Q. How to implement Gale–Shapley algorithm efficiently?
Q. If multiple stable matchings, which one does Gale–Shapley find?

COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of \( n \) applicants of which it can admit a quota of only \( q \). Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the \( q \) best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept. Accordingly, in order for a college to receive \( q \) acceptances, it will generally have to offer to admit more than \( q \) applicants. The problem of determining how many and which ones to admit requires some rather involved guesswork. It may not be known (a) whether a given applicant has also applied elsewhere; if this is known it may not be known (b) how he ranks the colleges to which he has applied; even if this is known it will not be known (c) which of the other colleges will offer to admit him. A result of all this uncertainty is that colleges can expect only that the entering class will come reasonably close in numbers to the desired quota, and be reasonably close to the attainable optimum in quality.
Efficient implementation

Efficient implementation. We describe an $O(n^2)$ time implementation.

Representing hospitals and students. Index hospitals and students 1, ..., n.

Representing the matching.

- Maintain a list of free hospitals (in a stack or queue).
- Maintain two arrays $student[h]$ and $hospital[s]$.
  - if $h$ matched to $s$, then $student[h] = s$ and $hospital[s] = h$
  - use value 0 to designate that hospital or student is unmatched

Hospitals proposing.

- For each hospital, maintain a list of students, ordered by preference.
- For each hospital, maintain a pointer to students in list for next proposal.
Students rejecting/accepting.

- Does student $s$ prefer hospital $h$ to hospital $h'$?
- For each student, create inverse of preference list of hospitals.
- Constant time access for each query after $O(n)$ preprocessing.

\[
\begin{array}{cccccccc}
\text{pref[]} & 1\text{st} & 2\text{nd} & 3\text{rd} & 4\text{th} & 5\text{th} & 6\text{th} & 7\text{th} & 8\text{th} \\
8 & 3 & 7 & 1 & 4 & 5 & 6 & 2 \\
\text{inverse[]} & 4\text{th} & 8\text{th} & 2\text{nd} & 5\text{th} & 6\text{th} & 7\text{th} & 3\text{rd} & 1\text{st} \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

\(O(n)\)


\[\text{for } i = 1 \text{ to } n, \quad \text{inverse[pref[i]]} = i\]