CSE 5311 Notes 14: Stable Marriages

STABLE MARRIAGES PROBLEM

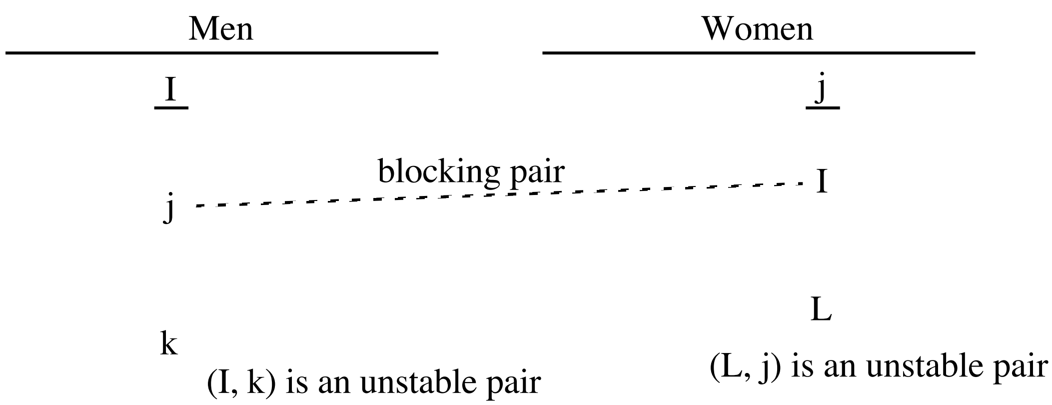
Classical Problem Instance:

n men (A, B, C, . . .) with preference lists

n women (1, 2, 3, . . .) with preference lists

Goal: Produce list of *n* stable marriages.

A set of marriages is *unstable* if there is a *blocking pair*:



Applications:

Matching new M.D.s to internships (many-to-one)

Matching lawyers to federal clerkships (one-to-one)

Matching students to classes (many-to-many)

Centralized admissions decisions for universities (many-to-one)

GALE-SHAPLEY ALGORITHM

Corresponds to most societies.

Men propose from the beginning of their lists.

Women always accept the first proposal, but may break the engagement later.

Example (from Sedgewick)

<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
2	1	2	1	5	E	D	A	C	D
5	2	3	3	3	A	E	D	B	B
1	3	5	2	2	D	B	B	D	C
3	4	4	4	1	B	A	C	A	E
4	5	1	5	4	C	C	E	E	A

Observations:

1. There is at least one stable solution.

   (Once engaged, a woman is always engaged. A man could eventually propose to all women and can't be rejected by all of them.)

2. The set of currently engaged couple is stable.

3. Gale-Shapley algorithm gives *male-optimal* matching. Switching roles in algorithm gives *female-optimal* matching.

4. If male-optimal solution is the same as female-optimal solution, the solution is unique.

5. The order of proposals by the available men *makes no difference* in the outcome.

Also possible to maintain n^2 nodes in data structure instead of $2n^2$ nodes.

ROTATIONS AND LATTICE OF STABLE MARRIAGE SOLUTIONS

A *rotation* takes two or more men, breaks their engagements, and engages them with the next (remaining) choice on their preference lists.

Stability is maintained since the women become matched with more preferable men.

Example:

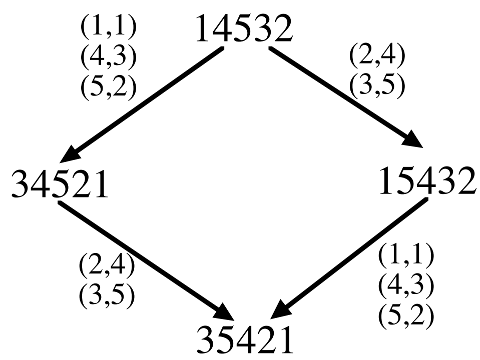
```
cat sedgewick.dat
5
2 5 1 3 4
1 2 3 4 5
2 3 5 4 1
1 3 2 4 5
5 3 2 1 4
5 1 4 2 3
4 5 2 1 3
1 4 2 3 5
3 2 4 1 5
4 2 3 5 1
a.out<sedgewick.dat
male preference lists are:
1: 1 3
2: 4 5
3: 5 4
4: 3 2 5
5: 2 1
female preference lists are:
1: 5 1
2: 4 5
3: 1 4
4: 3 2
5: 4 2 3
Male optimal solution:
1 1
2 4
3 5
4 3
5 2
Found a rotation:
(1,1)
(4,3)
(5,2)
Delete male=1 female=1
Delete male=4 female=3
Delete male=5 female=2
```

```
Revised preference lists:
male preference lists are:
1: 3
2: 4 5
3: 5 4
4: 2 5
5: 1
female preference lists are:
1: 5
2: 4
3: 1
4: 3 2
5: 4 2 3
Next matching:
1 3
2 4
3 5
4 2
5 1
Found a rotation:
(2,4)
(3,5)
Delete male=2 female=4
Delete male=3 female=5
Revised preference lists:
male preference lists are:
1: 3
2: 5
3: 4
4: 2 5
5: 1
female preference lists are:
1: 5
2: 4
3: 1
4: 3
5: 4 2
Next matching:
1 3
2 5
3 4
4 2
5 1
```

Given any pair of stable marriage matchings, another stable matching may be found by taking either:

1. The more preferred woman for every man.
2. The less preferred woman for every man.

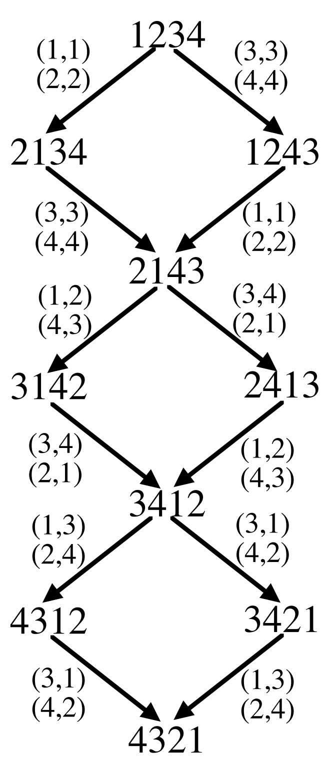
Mathematically, the result is a *distributive lattice*:



Example:

```
cat sm22.dat
4
1 2 3 4
2 1 4 3
3 4 1 2
4 3 2 1
4 3 2 1
3 4 1 2
2 1 4 3
1 2 3 4
a.out<sm22.dat
male preference lists are:
1: 1 2 3 4
2: 2 1 4 3
3: 3 4 1 2
4: 4 3 2 1
female preference lists are:
1: 4 3 2 1
2: 3 4 1 2
3: 2 1 4 3
4: 1 2 3 4
Male optimal solution:
1 1
2 2
3 3
4 4
Found a rotation:
(1,1)
(2,2)
Delete male=1 female=1
Delete male=2 female=2
Revised preference lists:
male preference lists are:
1: 2 3 4
2: 1 4 3
3: 3 4 1 2
4: 4 3 2 1
female preference lists are:
1: 4 3 2
2: 3 4 1
3: 2 1 4 3
4: 1 2 3 4
Next matching:
1 2
2 1
3 3
4 4
Found a rotation:
(3,3)
(4,4)
Delete male=3 female=3
Delete male=4 female=4
Revised preference lists:
male preference lists are:
1: 2 3 4
2: 1 4 3
3: 4 1 2
4: 3 2 1
female preference lists are:
1: 4 3 2
2: 3 4 1
3: 2 1 4
4: 1 2 3
Next matching:
1 2
2 1
3 4
4 3
Found a rotation:
(1,2)
(4,3)
Delete male=1 female=2
Delete male=4 female=3
Revised preference lists:
male preference lists are:
1: 3 4
2: 1 4 3
3: 4 1 2
4: 2 1
female preference lists are:
1: 4 3 2
2: 3 4
```

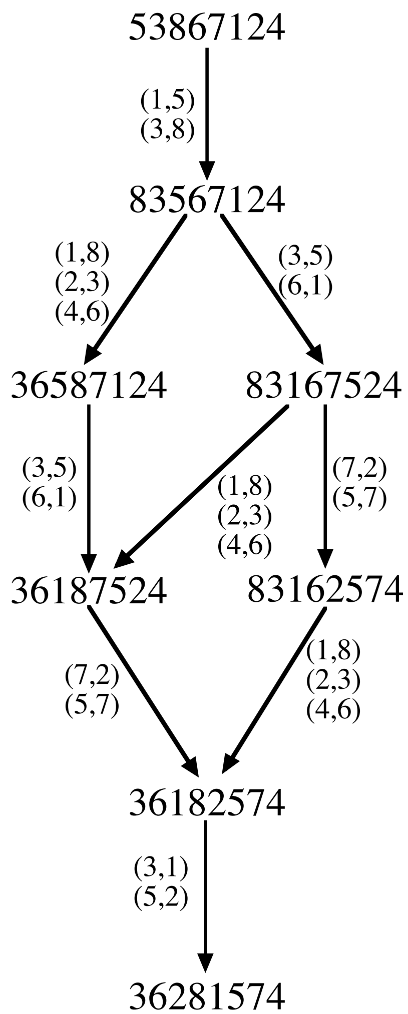
```
3: 2 1
4: 1 2 3
Next matching:
1 3
2 1
3 4
4 2
Found a rotation:
(3,4)
(2,1)
Delete male=3 female=4
Delete male=2 female=1
Revised preference lists:
male preference lists are:
1: 3 4
2: 4 3
3: 1 2
4: 2 1
female preference lists are:
1: 4 3
2: 3 4
3: 2 1
4: 1 2
Next matching:
1 3
2 4
3 1
4 2
Found a rotation:
(1,3)
(2,4)
Delete male=1 female=3
Delete male=2 female=4
Revised preference lists:
male preference lists are:
1: 4
2: 3
3: 1 2
4: 2 1
female preference lists are:
1: 4 3
2: 3 4
3: 2
4: 1
Next matching:
1 4
2 3
3 1
4 2
Found a rotation:
(3,1)
(4,2)
Delete male=3 female=1
Delete male=4 female=2
Revised preference lists:
male preference lists are:
1: 4
2: 3
3: 2
4: 1
female preference lists are:
1: 4
2: 3
3: 2
4: 1
Next matching:
1 4
2 3
3 2
4 1
```



Example:

```
cat sm12.dat
8
5 7 1 2 6 8 4 3
2 3 7 5 4 1 8 6
8 5 1 4 6 2 3 7
3 2 7 4 1 6 8 5
7 2 5 1 3 6 8 4
1 6 7 5 8 4 2 3
2 5 7 6 3 4 8 1
3 8 4 5 7 2 6 1
5 3 7 6 1 2 8 4
8 6 3 5 7 2 1 4
1 5 6 2 4 8 7 3
8 7 3 2 4 1 5 6
6 4 7 3 8 1 2 5
2 8 5 3 4 6 7 1
7 5 2 1 8 6 4 3
7 4 1 5 2 3 6 8
male preference lists are:
1: 5 8 3
2: 3 8 6
3: 8 5 1 6 2
4: 6 8 5
5: 7 2 1 3 6 8
6: 1 5 2 3
7: 2 5 7 8 1
8: 4 5 2 6
female preference lists are:
1: 5 3 7 6
2: 8 6 3 5 7
3: 1 5 6 2
4: 8
5: 6 4 7 3 8 1
6: 2 8 5 3 4
7: 7 5
8: 7 4 1 5 2 3
Male optimal solution:
1 5
2 3
3 8
4 6
5 7
6 1
7 2
8 4
Found a rotation:
(1,5)
(3,8)
Delete male=1 female=5
Delete male=8 female=5
Delete male=3 female=8
Delete male=2 female=8
Delete male=5 female=8
Revised preference lists:
male preference lists are:
1: 8 3
2: 3 6
3: 5 1 6 2
4: 6 8 5
5: 7 2 1 3 6
6: 1 5 2 3
7: 2 5 7 8 1
8: 4 2 6
female preference lists are:
1: 5 3 7 6
2: 8 6 3 5 7
3: 1 5 6 2
4: 8
5: 6 4 7 3
6: 2 8 5 3 4
7: 7 5
8: 7 4 1
Next matching:
1 8
2 3
3 5
4 6
5 7
6 1
7 2
8 4
Found a rotation:
(1,8)
(2,3)
(4,6)
Delete male=1 female=8
Delete male=2 female=3
Delete male=6 female=3
Delete male=5 female=3
Delete male=4 female=6
Delete male=3 female=6
Delete male=5 female=6
Delete male=8 female=6
Revised preference lists:
male preference lists are:
1: 3
2: 6
3: 5 1 2
4: 8 5
5: 7 2 1
6: 1 5 2
7: 2 5 7 8 1
8: 4 2
female preference lists are:
1: 5 3 7 6
2: 8 6 3 5 7
3: 1
4: 8
5: 6 4 7 3
6: 2
7: 7 5
8: 7 4
Next matching:
1 3
2 6
3 5
4 8
5 7
```

```
6 1
7 2
8 4
Found a rotation:
(3,5)
(6,1)
Delete male=3 female=5
Delete male=7 female=5
Delete male=4 female=5
Delete male=6 female=1
Delete male=7 female=1
Revised preference lists:
male preference lists are:
1: 3
2: 6
3: 1 2
4: 8
5: 7 2 1
6: 5 2
7: 2 7 8
8: 4 2
female preference lists are:
1: 5 3
2: 8 6 3 5 7
3: 1
4: 8
5: 6
6: 2
7: 7 5
8: 7 4
Next matching:
1 3
2 6
3 1
4 8
5 7
6 5
7 2
8 4
Found a rotation:
(7,2)
(5,7)
Delete male=7 female=2
Delete male=5 female=7
Revised preference lists:
male preference lists are:
1: 3
2: 6
3: 1 2
4: 8
5: 2 1
6: 5 2
7: 7 8
8: 4 2
female preference lists are:
1: 5 3
2: 8 6 3 5
3: 1
4: 8
5: 6
6: 2
7: 7
8: 7 4
Next matching:
1 3
2 6
3 1
4 8
5 2
6 5
7 7
8 4
Found a rotation:
(3,1)
(5,2)
Delete male=3 female=1
Delete male=5 female=2
Revised preference lists:
male preference lists are:
1: 3
2: 6
3: 2
4: 8
5: 1
6: 5 2
7: 7 8
8: 4 2
female preference lists are:
1: 5
2: 8 6 3
3: 1
4: 8
5: 6
6: 2
7: 7
8: 7 4
Next matching:
1 3
2 6
3 2
4 8
5 1
6 5
7 7
8 4
```



STABLE ROOMMATES - introductory concepts

Classical Problem Instance:

n persons with preference lists including the other *n* - 1 persons

n is assumed to be even

Goal: Produce list of $\frac{n}{2}$ stable pairs.

Generalizes stable marriages. An instance of S.M. is easily translated to an instance of S.R.

Two-phase algorithm is more complicated, since a solution is *not guaranteed.*

Phase 1: Like Gale-Shapley, but based on asymmetric *semi-engagements*.

x is semi-engaged to y means that x has issued a proposal to y, which y accepted.
(`semiEngaged[i]=j` means that `j` is semi-engaged to `i`)

Phase 2: Uses rotations to assure that x is semi-engaged to y iff y is semi-engaged to x.

In rare cases of there being exactly one solution, the second phase may be skipped:

```
a.out<fig4.4.dat
Input:
1: 4 2 3
2: 1 3 4
3: 2 4 1
4: 2 1 3
debug: semiEngaged[2]=4
debug:  after processing semiengagement
1: 4 2 3
2: 1 3 4
3: 2 4 1
4: 2 1 3
debug: semiEngaged[2]=3
debug: delete {2 4} from lists
debug:  after processing semiengagement
1: 4 2 3
2: 1 3
3: 2 4 1
4: 1 3
debug: semiEngaged[1]=4
debug: delete {1 2} from lists
debug: delete {1 3} from lists
debug:  after processing semiengagement
1: 4
2: 3
3: 2 4
4: 1 3
debug: semiEngaged[3]=2
debug: delete {3 4} from lists
debug:  after processing semiengagement
1: 4
2: 3
3: 2
4: 1
debug: semiEngaged[4]=1
debug:  after processing semiengagement
1: 4
2: 3
3: 2
4: 1
After phase 1:
1: 4
2: 3
3: 2
4: 1
phase 2 not needed
```

In some cases there is no solution:

```
a.out<fig4.3.dat
Input:
1: 3 2 4
2: 1 3 4
3: 2 1 4
4: 1 2 3
debug: semiEngaged[1]=4
debug:  after processing semiengagement
1: 3 2 4
2: 1 3 4
3: 2 1 4
4: 1 2 3
debug: semiEngaged[2]=3
debug: delete {2 4} from lists
debug:  after processing semiengagement
1: 3 2 4
2: 1 3
3: 2 1 4
4: 1 3
debug: semiEngaged[1]=2
debug: delete {1 4} from lists
debug:  after processing semiengagement

1: 3 2
2: 1 3
3: 2 1 4
4: 3
debug: semiEngaged[3]=4
debug:  after processing semiengagement
1: 3 2
2: 1 3
3: 2 1 4
4: 3
debug: semiEngaged[3]=1
debug: delete {3 4} from lists
debug:  after processing semiengagement
1: 3 2
2: 1 3
3: 2 1
4:
phase 1 has empty list for 4 - no
solution exists
1: 3 2
2: 1 3
3: 2 1
4:
```

General case:

```
a.out<fig4.5.dat
Input:
1: 8 2 9 3 6 4 5 7 10
2: 4 3 8 9 5 1 10 6 7
3: 5 6 8 2 1 7 10 4 9
4: 10 7 9 3 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 2 1 8 3 5 10 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 3 4 8 1
10: 3 1 6 5 2 9 8 4 7
debug: semiEngaged[3]=10
debug: delete {3 4} from lists
debug: delete {3 9} from lists
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7 10
2: 4 3 8 9 5 1 10 6 7
3: 5 6 8 2 1 7 10
4: 10 7 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 2 1 8 3 5 10 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 4 8 1
10: 3 1 6 5 2 9 8 4 7
debug: semiEngaged[6]=9
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7 10
2: 4 3 8 9 5 1 10 6 7
3: 5 6 8 2 1 7 10
4: 10 7 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 2 1 8 3 5 10 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 4 8 1
10: 3 1 6 5 2 9 8 4 7
debug: semiEngaged[10]=8
debug: delete {10 4} from lists
debug: delete {10 7} from lists
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7 10
2: 4 3 8 9 5 1 10 6 7
3: 5 6 8 2 1 7 10
4: 7 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 2 1 8 3 5 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 4 8 1
10: 3 1 6 5 2 9 8
debug: semiEngaged[2]=7
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7 10
2: 4 3 8 9 5 1 10 6 7
3: 5 6 8 2 1 7 10
4: 7 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 2 1 8 3 5 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 4 8 1
10: 3 1 6 5 2 9 8
debug: semiEngaged[2]=6
debug: delete {2 7} from lists
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7 10
2: 4 3 8 9 5 1 10 6
3: 5 6 8 2 1 7 10
4: 7 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 1 8 3 5 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 4 8 1
10: 3 1 6 5 2 9 8
debug: semiEngaged[1]=7
debug: delete {1 10} from lists
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7
2: 4 3 8 9 5 1 10 6
3: 5 6 8 2 1 7 10
4: 7 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 7 3 4 10 1 5 9
7: 1 8 3 5 4 6 9
8: 10 4 2 5 6 7 1 3 9
9: 6 7 2 5 10 4 8 1
10: 3 6 5 2 9 8
debug: semiEngaged[7]=5
debug: delete {7 4} from lists
debug: delete {7 6} from lists
debug: delete {7 9} from lists
debug:  after processing semiengagement
1: 8 2 9 3 6 4 5 7
2: 4 3 8 9 5 1 10 6
3: 5 6 8 2 1 7 10
4: 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 3 4 10 1 5 9
7: 1 8 3 5
8: 10 4 2 5 6 7 1 3 9
9: 6 2 5 10 4 8 1
10: 3 6 5 2 9 8
debug: semiEngaged[9]=4
debug: delete {9 8} from lists
debug: delete {9 1} from lists
debug:  after processing semiengagement
1: 8 2 3 6 4 5 7
2: 4 3 8 9 5 1 10 6
3: 5 6 8 2 1 7 10
4: 9 1 6 2 5 8
5: 7 4 10 8 2 6 3 1 9
6: 2 8 3 4 10 1 5 9
7: 1 8 3 5
8: 10 4 2 5 6 7 1 3
9: 6 2 5 10 4
```

```
10: 3 6 5 2 9 8
debug: semiEngaged[5]=3
debug: delete {5 1} from lists
debug: delete {5 9} from lists
debug:  after processing semiengagement
1: 8 2 3 6 4 7
2: 4 3 8 9 5 1 10 6
3: 5 6 8 2 1 7 10
4: 9 1 6 2 5 8
5: 7 4 10 8 2 6 3
6: 2 8 3 4 10 1 5 9
7: 1 8 3 5
8: 10 4 2 5 6 7 1 3
9: 6 2 10 4
10: 3 6 5 2 9 8
debug: semiEngaged[4]=2
debug: delete {4 5} from lists
debug: delete {4 8} from lists
debug:  after processing semiengagement
1: 8 2 3 6 4 7
2: 4 3 8 9 5 1 10 6
3: 5 6 8 2 1 7 10
4: 9 1 6 2
5: 7 10 8 2 6 3
6: 2 8 3 4 10 1 5 9
7: 1 8 3 5
8: 10 2 5 6 7 1 3
9: 6 2 10 4
10: 3 6 5 2 9 8
debug: semiEngaged[8]=1
debug: delete {8 3} from lists
debug:  after processing semiengagement
1: 8 2 3 6 4 7
2: 4 3 8 9 5 1 10 6
3: 5 6 2 1 7 10
4: 9 1 6 2
5: 7 10 8 2 6 3
6: 2 8 3 4 10 1 5 9
7: 1 8 3 5
8: 10 2 5 6 7 1
9: 6 2 10 4
10: 3 6 5 2 9 8
After phase 1:
1: 8 2 3 6 4 7
2: 4 3 8 9 5 1 10 6
3: 5 6 2 1 7 10
4: 9 1 6 2
5: 7 10 8 2 6 3
6: 2 8 3 4 10 1 5 9
7: 1 8 3 5
8: 10 2 5 6 7 1
9: 6 2 10 4
10: 3 6 5 2 9 8
DEBUG-rotation: (1,8)(6,2)
debug: delete {2 6} from lists
debug: delete {2 10} from lists
debug: delete {8 1} from lists
debug: delete {8 7} from lists
1: 2 3 6 4 7
2: 4 3 8 9 5 1
3: 5 6 2 1 7 10
4: 9 1 6 2
5: 7 10 8 2 6 3
6: 8 3 4 10 1 5 9
7: 1 3 5
8: 10 2 5 6
9: 6 2 10 4
10: 3 6 5 9 8
DEBUG-rotation: (1,2)(10,3)(9,6)
debug: delete {6 9} from lists
debug: delete {6 5} from lists
debug: delete {6 1} from lists
debug: delete {3 10} from lists
debug: delete {3 7} from lists
debug: delete {2 1} from lists
debug: delete {2 5} from lists
1: 3 4 7
2: 4 3 8 9
3: 5 6 2 1
4: 9 1 6 2
5: 7 10 8 3
6: 8 3 4 10
7: 1 5
8: 10 2 5 6
9: 2 10 4
10: 6 5 9 8
DEBUG-rotation: (1,3)(2,4)
debug: delete {4 2} from lists
debug: delete {4 6} from lists
debug: delete {3 1} from lists
1: 4 7
2: 3 8 9
3: 5 6 2
4: 9 1
5: 7 10 8 3
6: 8 3 10
7: 1 5
8: 10 2 5 6
9: 2 10 4
10: 6 5 9 8
DEBUG-rotation: (8,10)(9,2)
debug: delete {2 9} from lists
debug: delete {10 8} from lists
1: 4 7
2: 3 8
3: 5 6 2
4: 9 1
5: 7 10 8 3
6: 8 3 10
7: 1 5
8: 2 5 6
9: 10 4
10: 6 5 9
DEBUG-rotation: (1,4)(5,7)(9,10)
debug: delete {10 9} from lists
debug: delete {7 5} from lists
debug: delete {4 1} from lists
1: 7
2: 3 8
3: 5 6 2
4: 9
5: 10 8 3
```

```
6: 8 3 10
7: 1
8: 2 5 6
9: 4
10: 6 5
DEBUG-rotation: (2,3)(6,8)
debug: delete {8 6} from lists
debug: delete {8 5} from lists
debug: delete {3 2} from lists
1: 7
2: 8
3: 5 6
4: 9
5: 10 3
6: 3 10
7: 1
8: 2
9: 4
10: 6 5
DEBUG-rotation: (3,5)(10,6)
debug: delete {6 10} from lists
debug: delete {5 3} from lists
phase 2 has solution
After phase 2:
1: 7
2: 8
3: 6
4: 9
5: 10
6: 3
7: 1
8: 2
9: 4
10: 5
```