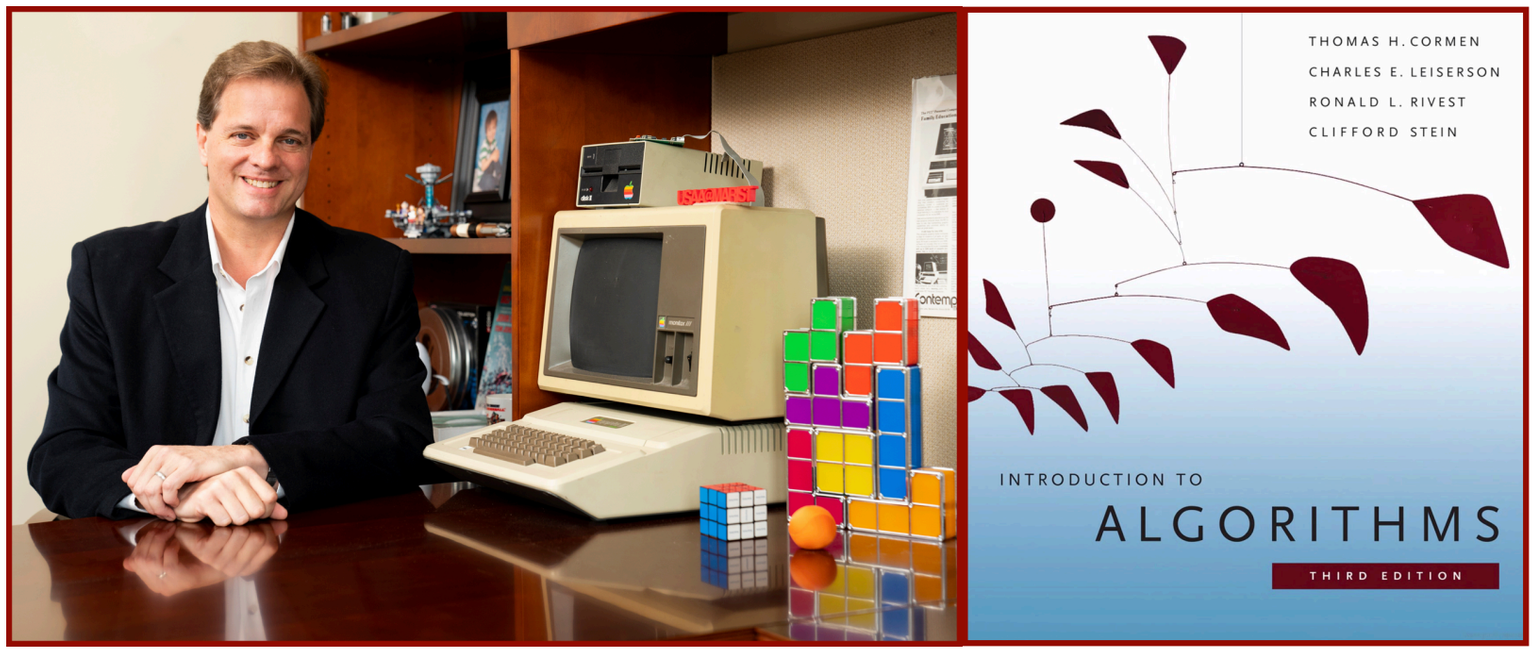
~~Final~~ Project

The *Stable Marriage* / *Stable Matching* / *Matching Markets* Problem



Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu

Background, from Dr. Reid

The *Stable Marriage* / *Stable Matching* / *Matching Markets* Problem

The Stable Marriage Problem

Elizabeth Reid

Marist College Department of Mathematics Seminar Series

November 1, 2019

Background

The *Stable Marriage* / *Stable Matching* / *Matching Markets* Problem

Classic Set-up:

- n men and n women
- each man lists the women according to preference
- each woman lists the men according to preference
- arrange marriages so no unmatched man and woman prefer each other to their assigned partners

Background

The *Stable Marriage / Stable Matching / Matching Markets* Problem

tiny Example

Two men: Alex and Bob;
Two women: Carol and Dana

- $A : C > D$
- $B : C > D$
- $C : A > B$
- $D : B > A$

Background

The *Stable Marriage / Stable Matching / Matching Markets* Problem

tiny Example

Two men: Alex and Bob;
Two women: Carol and Dana

- $A : C > D$
- $B : C > D$
- $C : A > B$
- $D : B > A$

Possible Matchings:

1. $\{(A, D), (B, C)\}$
2. $\{(A, C), (B, D)\}$

Background

The *Stable Marriage / Stable Matching / Matching Markets* Problem

tiny Example

Two men: Alex and Bob;
Two women: Carol and Dana

- $A : C > D$
- $B : C > D$
- $C : A > B$
- $D : B > A$

Possible Matchings:

1. $\{(A, D), (B, C)\} \longrightarrow$ unstable (A, C)
2. $\{(A, C), (B, D)\}$

Background

The *Stable Marriage / Stable Matching / Matching Markets* Problem

tiny Example

Two men: Alex and Bob;
Two women: Carol and Dana

- $A : C > D$
- $B : C > D$
- $C : A > B$
- $D : B > A$

Possible Matchings:

1. $\{(A, D), (B, C)\} \longrightarrow$ unstable (A, C)
2. $\{(A, C), (B, D)\} \longrightarrow$ stable

Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

Gale-Shapley Algorithm:

- *Input.*
  - A set of n men, a set of n women, a ranked list of the n women for each man, and a ranked list of the n men for each woman.

	1	2	3	4
Alex	H	F	E	G
Bob	F	G	H	E
Carl	E	F	H	G
Dave	H	F	G	E

	1	2	3	4
Erin	B	A	D	C
Fran	D	C	B	A
Grace	A	D	B	C
Holly	C	D	A	B

Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

Gale-Shapley Algorithm:

- *Input.*
  - A set of n men, a set of n women, a ranked list of the n women for each man, and a ranked list of the n men for each woman.

	1	2	3	4
Alex	H	F	E	G
Bob	F	G	H	E
Carl	E	F	H	G
Dave	H	F	G	E

	1	2	3	4
Erin	B	A	D	C
Fran	D	C	B	A
Grace	A	D	B	C
Holly	C	D	A	B

- *Output.*
  - A stable matching that pairs the n men and n women.

Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

Gale-Shapley Algorithm:

- *Input.*
  - A set of n men, a set of n women, a ranked list of the n women for each man, and a ranked list of the n men for each woman.

	1	2	3	4
Alex	H	F	E	G
Bob	F	G	H	E
Carl	E	F	H	G
Dave	H	F	G	E

	1	2	3	4
Erin	B	A	D	C
Fran	D	C	B	A
Grace	A	D	B	C
Holly	C	D	A	B

- *Output.*
  - A stable matching that pairs the n men and n women.

When Men Propose: $\{(A, E), (B, G), (C, F), (D, H)\}$

Does it make a difference who proposes?

Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

Gale-Shapley Algorithm:

- *Input.*
  - A set of n men, a set of n women, a ranked list of the n women for each man, and a ranked list of the n men for each woman.

	1	2	3	4
Alex	H	F	E	G
Bob	F	G	H	E
Carl	E	F	H	G
Dave	H	F	G	E

	1	2	3	4
Erin	B	A	D	C
Fran	D	C	B	A
Grace	A	D	B	C
Holly	C	D	A	B

- *Output.*
  - A stable matching that pairs the n men and n women.

When Men Propose: $\{(A, E), (B, G), (C, F), (D, H)\}$

When Women Propose: $\{(A, G), (B, E), (C, H), (D, F)\}$

Variation: Hospitals and Residents

The *Stable Marriage* / *Stable Matching* / *Matching Markets* Problem

Applications: Hospitals and Residents

- each medical student ranks hospital residency programs
- each hospital ranks the candidates

	1	2	3
Alex	X	Y	Z
Bob	Y	Z	X
Carl	Z	Y	X
Dana	X	Z	Y
Erin	X	Y	Z
Fran	Y	X	Z

	1	2	3	4	5	6
X	A	C	D	B	E	F
Y	E	F	D	A	C	B
Z	B	D	F	A	C	E

Variation: Hospitals and Residents — Algorithm

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

Figure 1.16: Resident-oriented algorithm

From page 42 of *The Stable Marriage Problem* by Gusfield and Irving. MIT Press. 1989

Variation: Hospitals and Residents — Algorithm

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

Let's do it!

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

Variation: Hospitals and Residents — Algorithm

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

Initialize residents and hospitals.

All residents are free.

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r1:
Provisionally assigning r1 to hospital h3.

Processing resident r2:
Provisionally assigning r2 to hospital h1.

Processing resident r3:
Provisionally assigning r3 to hospital h4.

Processing resident r4:
Provisionally assigning r4 to hospital h3.

Processing resident r5:
Provisionally assigning r5 to hospital h1.
```

```
Match:
(r1, h3)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r1:
Provisionally assigning r1 to hospital h3.

Processing resident r2:
Provisionally assigning r2 to hospital h1.

Processing resident r3:
Provisionally assigning r3 to hospital h4.

Processing resident r4:
Provisionally assigning r4 to hospital h3.

Processing resident r5:
Provisionally assigning r5 to hospital h1.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r1:
Provisionally assigning r1 to hospital h3.

Processing resident r2:
Provisionally assigning r2 to hospital h1.

Processing resident r3:
Provisionally assigning r3 to hospital h4.

Processing resident r4:
Provisionally assigning r4 to hospital h3.

Processing resident r5:
Provisionally assigning r5 to hospital h1.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r1:
Provisionally assigning r1 to hospital h3.

Processing resident r2:
Provisionally assigning r2 to hospital h1.

Processing resident r3:
Provisionally assigning r3 to hospital h4.

Processing resident r4:
Provisionally assigning r4 to hospital h3.

Processing resident r5:
Provisionally assigning r5 to hospital h1.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r1:
Provisionally assigning r1 to hospital h3.

Processing resident r2:
Provisionally assigning r2 to hospital h1.

Processing resident r3:
Provisionally assigning r3 to hospital h4.

Processing resident r4:
Provisionally assigning r4 to hospital h3.

Processing resident r5:
Provisionally assigning r5 to hospital h1.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
```

Wow... this is really easy! I bet it never...

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h4.
Hospital h4 is full. Finding worst resident assigned
to hospital h4.
   Considering r8. (no), Considering r6. (yes)
   Removing all residents after r6 from
   hospital h4's list:
   Deleting (h4, r8): Removing hospital h4
      from resident r8's list.
      Removing r8 from hospital h4's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
```

Oh.

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h4.
Hospital h4 is full. Finding worst resident assigned
to hospital h4.
   Considering r8. (no), Considering r6. (yes)
   Removing all residents after r6 from
   hospital h4's list:
   Deleting (h4, r8): Removing hospital h4
      from resident r8's list.
      Removing r8 from hospital h4's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
```

Never mind. I see it now.

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h4.
Hospital h4 is full. Finding worst resident assigned
to hospital h4.
   Considering r8. (no), Considering r6. (yes)
   Removing all residents after r6 from
   hospital h4's list:
   Deleting (h4, r8): Removing hospital h4
      from resident r8's list.
      Removing r8 from hospital h4's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r7:
Provisionally assigning r7 to hospital h2.

Processing resident r8:
Provisionally assigning r8 to hospital h1.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r7, h2)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r7:
Provisionally assigning r7 to hospital h2.

Processing resident r8:
Provisionally assigning r8 to hospital h1.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r7, h2)
(r8, h1)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r9: Hospital h4 is full.
Finding worst resident assigned to hospital h4.
   Considering r6. (yes)
   Bumping r6 from hospital h4.
Provisionally assigning r9 to hospital h4.
Hospital h4 is (still) full. Finding worst resident
assigned to hospital h4.
   Considering r6. (no) Considering r3. (yes)
   Removing all residents after r3 from
   hospital h4's list:
      Deleting (h4, r6): Removing hospital h4
      from resident r6's list.
   Removing r6 from hospital h4's list.
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r7, h2)
(r8, h1)
```

(make room in h4 for r9)

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r9: h4 is full
Finding worst resident assigned to hospital h4.
   Considering r6. (yes)
   Bumping r6 from hospital h4.
Provisionally assigning r9 to hospital h4.
Hospital h4 is (still) full. Finding worst resident
assigned to hospital h4.
   Considering r6. (no) Considering r3. (yes)
   Removing all residents after r3 from
   hospital h4's list:
      Deleting (h4, r6): Removing hospital h4
      from resident r6's list.
   Removing r6 from hospital h4's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r7, h2)
(r8, h1)
(r9, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r9: h4 is full
Finding worst resident assigned to hospital h4.
   Considering r6. (yes)
   Bumping r6 from hospital h4.
Provisionally assigning r9 to hospital h4.
Hospital h4 is (still) full. Finding worst resident
assigned to hospital h4.
   Considering r6. (no) Considering r3. (yes)
   Removing all residents after r3 from
   hospital h4's list:
      Deleting (h4, r6): Removing hospital h4
      from resident r6's list.
      Removing r6 from hospital h4's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)   (r6 is still unassigned)
(r7, h2)
(r8, h1)
(r9, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r9: h4 is full
Finding worst resident assigned to hospital h4.
   Considering r6. (yes)
   Bumping r6 from hospital h4.
Provisionally assigning r9 to hospital h4.
Hospital h4 is (still) full. Finding worst resident
assigned to hospital h4.
   Considering r6. (no) Considering r3. (yes)
   Removing all residents after r3 from
   hospital h4's list:
      Deleting (h4, r6): Removing hospital h4
      from resident r6's list.
      Removing r6 from hospital h4's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)   (r6 is still unassigned)
(r7, h2)
(r8, h1)
(r9, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h3.
Hospital h3 is full. Finding worst resident assigned
to hospital h3.
   Considering r10. (no), Considering r1. (yes)
   Removing all residents after r1 from
   hospital h3's list:
      Deleting (h3, r10): Removing hospital h3
      from resident r10's list.
      Removing r10 from hospital h3's list.
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)  (r6 now reassigned)
(r7, h2)
(r8, h1)
(r9, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h3.
Hospital h3 is full. Finding worst resident assigned
to hospital h3.
   Considering r10. (no), Considering r1. (yes)
   Removing all residents after r1 from
   hospital h3's list:
      Deleting (h3, r10): Removing hospital h3
      from resident r10's list.
      Removing r10 from hospital h3's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h3.
Hospital h3 is full. Finding worst resident assigned
to hospital h3.
   Considering r10. (no), Considering r1. (yes)
   Removing all residents after r1 from
   hospital h3's list:
      Deleting (h3, r10): Removing hospital h3
      from resident r10's list.
      Removing r10 from hospital h3's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r10:
Provisionally assigning r10 to hospital h1.
Hospital h1 is full. Finding worst resident assigned
to hospital h1.
   Considering r2. (yes)
   Removing all residents after r2
   from hospital h1's list:
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r10:
Provisionally assigning r10 to hospital h1.
Hospital h1 is full. Finding worst resident assigned
to hospital h1.
   Considering r2. (yes)
   Removing all residents after r2
   from hospital h1's list:
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r10:
Provisionally assigning r10 to hospital h1.
Hospital h1 is full. Finding worst resident assigned
to hospital h1.
   Considering r2. (yes)
   Removing all residents after r2
   from hospital h1's list:
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r11:
Provisionally assigning r11 to hospital h5.
Hospital h5 is full. Finding worst resident assigned
to hospital h5.
   Considering r9. (no), Considering r11. (yes)
   Removing all residents after r11
   from hospital h5's list:
      Deleting (h5, r9): Removing
      hospital h5 from resident r9's list.
      Removing r9 from hospital h5's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r11:
Provisionally assigning r11 to hospital h5.
Hospital h5 is full. Finding worst resident assigned
to hospital h5.
   Considering r9. (no), Considering r11. (yes)
   Removing all residents after r11
   from hospital h5's list:
      Deleting (h5, r9): Removing
      hospital h5 from resident r9's list.
      Removing r9 from hospital h5's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
Processing resident r11:
Provisionally assigning r11 to hospital h5.
Hospital h5 is full. Finding worst resident assigned
to hospital h5.
   Considering r9. (no), Considering r11. (yes)
   Removing all residents after r11
   from hospital h5's list:
      Deleting (h5, r9): Removing
      hospital h5 from resident r9's list.
      Removing r9 from hospital h5's list.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

Variation: Hospitals and Residents — Example

The *Stable Marriage / Stable Matching / Matching Markets* Problem

```
assign all residents to be free ;
assign all hospitals to be totally unsubscribed ;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; {r "proposes" to h}
    if h is fully subscribed then
        begin
            r' := worst resident provisionally assigned to h ;
            assign r' to be free
        end ;
    provisionally assign r to h ;
    if h is fully subscribed then
        begin
            s := worst resident provisionally assigned to h ;
            for each successor s' of s on h's list do
                remove s' and h from each other's lists
        end
end
```

```
No residents are free.
```

```
Resident Preferences:
   r1: h3 h1 h5 h4          r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5       r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2       r9: h4 h1 h5
   r4: h3 h4 h1 h5          r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2             r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5

Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Final Stable Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

Final Project

The *Stable Marriage* / *Stable Matching* / *Matching Markets* Problem

Part one: (70 points)
Implement the Hospital and Residents variation as shown here.

- Test your code on this data, Make sure your results match mine.
- Then test your code on your own data and make a case that it's correct.

Part two: (20 points)
Implement a variation on the Hospital and Residents variation where to hospitals do not rank residents but residents still rank the hospitals.

- Define what "stability" means in this context.
- Then test your code on your own data and make a case that it's correct.

See the final project assignment on our class web site for additional details (like the remaining 10 points).