Example

```
FreeM <- M
FreeW <- W
Invitations <- {}
Tentative <- {}
While there is an m in FreeM s.t. there is a w s.t. (m,w) not in Invitations
    choose such an m
    let w be m's highest ranked s.t. (m,w) not in Invitations

    add (m,w) to Invitations

    if w in FreeW then
        remove w from FreeW
        remove m from FreeM
        add (m,w) to Tentative
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
            remove m from FreeM
            add m' to FreeM
            remove (m', w) from Tentative
            add(m, w) to Tentative
return Tentative
```

— HashMap or FreeM

1	2	3	4	5
T	F	F	T	T

FreeQ

2	3	2	

FreeW

1	2	3	4	5

$O(n^2)$ ← preprocess to create Rank s.t. Rank[i][j] = m_i's rank of w_j

does V prefer D to C?

$O(n^2)$ iterations

$O(1)$

$O(1)$

	m	PrefM		w	PrefW
		3 4 1 2 5			
1	A	~~X~~ Y V W Z	1	~~V~~	A D C E B
2	B	V X W Y Z	2	W	A B D C E
3	C	V Z W Y X	3	~~X~~	D E C A B
4	D	W V X Z Y	4	Y	C B A E D
5	E	X Y V W Z	5	Z	A B D E C

~~(A,X)~~ ~~(B,V)~~ ~~(C,V)~~ (D,V) (E,X)

(A,Y) (B,W) (C,Z)

(A,V) (B,W) (D,X) (C,Y) (E,Z)

stable

V (w_1)'s pref

0 = A	1 = B	2 = C	3 = D	4 = E
5	3	2	4	1

E C B D A

does V prefer D to B NO

$O(n^2)$ overall

MatchM = hash table

MatchM (m) = welder matched with (or NIL)

Invariant

a) $\forall m,\ m \notin FreeM \longleftrightarrow \exists w$ s.t. $(m,w) \in Tent$
$\forall w,\ w \notin FreeW \longleftrightarrow \exists m$ s.t. $(m,w) \in Tent$ — FreeM, FreeW keep track of unmatched machinists, welders

b) $\forall w,\ w \in FreeW \longleftrightarrow \sim\exists m$ s.t. $(m,w) \in Invites$ — Free welders are exactly those who have received no invitations

c) Tent is a matching and stableish → stable when ignoring unmatched machinists, welders

d) $|Invites| = k$

MatchW(w) after iteration j

once a welder receives an invitation, that welder is never free again

e) $\forall w,\ j<k,\ \underbrace{MatchW_j(w) \neq NIL}$ $\rightarrow MatchW_{j+1}(w), \ldots, MatchW_k(w)\ != NIL$

f) $\forall w,\ MatchW(w) = \max_{m}{}_{w}\ (m,w) \in Invites$ — w is matched with their most preferred machinist they've received an invitation from

g) $\forall m, w, w'$ if $(m,w) \in Invites$ and m prefers w' to w then $(m,w') \in Invites$ — machinists send invitations in order of ↓ preference

h)

Maintenance (easy parts)

Suppose INV is T before loop and $\exists m \in FreeM, w$ s.t. $(m,w) \notin Invites$

```
FreeM <- M
FreeW <- W
Invitations <- {}
Tentative <- {}
k <- 0

while there is an m in FreeM s.t. there is a w s.t. (m,w) not in Invitations
    choose such an m
    let w be m's highest ranked s.t. (m,w) not in Invitations

    add (m,w) to Invitations

    if w in FreeW then
        remove w from FreeW
        remove m from FreeM
        add (m,w) to Tentative
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
            remove m from FreeM
            add m' to FreeM
            remove (m', w) from Tentative
            add(m, w) to Tentative
    k <- k+1
return Tentative
```

a) $\forall m,\ m \notin FreeM \longleftrightarrow \exists w$ s.t. $(m,w) \in Tent$

$\forall w,\ w \notin FreeW \longleftrightarrow \exists m$ s.t. $(m,w) \in Tent$

Only M changed are m, m'

in cases 1,2 m removed from FreeM, (m,w) added to Tent

in 2 m' added to FreeM, (m',w) removed from Tent and no other $(m',w') \in Tent$

in case 3 no changes

Only W changed is w

in case 1, w removed from FreeW, (m,w) added to Tent $F \leftrightarrow F$

in cases 2, $w \notin FreeW$

(m,w) added to Tent $T \leftrightarrow T$

in case 3, no change

b) $\forall w,\ w \in FreeW \longleftrightarrow \sim\exists m$ s.t. $(m,w) \in Invites$

Only w changed is w

(m,w) added to Invites

in case 1, w removed from FreeW

in cases 2,3 $w \notin FreeW$ to start with and not changed so $F \leftrightarrow F$ at end of loop

c) $\forall w, j<k,\ MatchW_j(w) \neq NIL \rightarrow MatchW_{j+1}(w), \ldots, MatchW_k(w)$!= NIL

Only w changed, and w will always have $(m,w) \in Tent$ or $(m',w) \in Tent$

d) $|Invites| = k$

$|Invites_{old}| = k_{old}$

$k_{new} = k_{old} + 1$

$Invites_{new} = Invites_{old} \cup \{(m,w)\}$

$(m,w) \notin Invites$

$|Invites_{new}| = |Invites_{old}| + |\{(m,w)\}|$
$= k+1$

Maintenance (matches improve for welders)

```
FreeM <- M
FreeW <- W
Invitations <- {}
Tentative <- {}
k <- 0

while there is an m in FreeM s.t. there is a w s.t. (m,w) not in Invitations
    choose such an m
    let w be m's highest ranked s.t. (m,w) not in Invitations

    add (m,w) to Invitations

    if w in FreeW then
        remove w from FreeW
        remove m from FreeM
        add (m,w) to Tentative
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
            remove m from FreeM
            add m' to FreeM
            remove (m', w) from Tentative
            add(m, w) to Tentative
    k <- k+1
return Tentative
```

Suppose INV is T before loop and $\exists m \in \text{FreeM}, w$ s.t. $(m,w) \notin \text{Invites}$

Notation: m s.t. $(m,w) \in \text{Tent}$ (or NIL) w's highest pref $\hat{m}$ s.t. $(\hat{m},w) \in \text{Invites}$

f) $\forall w,\ \text{MatchW}(w) = \underset{(\hat{m},w)\in \text{Invites}}{\max\nolimits_w} \hat{m}$

every w is matched with their favorite machinist they've received an invitation from (best inviter)

Only change to Tent / Invites is for w (so if true before loop for other welders, still true after)

$\text{MatchW}_{old}(w) = \underset{(m,w)\in \text{Invites}_{old}}{\max\nolimits_w} m$ (INV)

case 1: $w \in \text{FreeW}_{old}$

$\neg \exists m$ s.t. $(m,w) \in \text{Invites}_{old}$ (INV b)

$\text{MatchW}_{new}(w) = m$ (code - add (m,w) to Tent)

$\underset{(\hat{m},w)\in \text{Invits}_{new}}{\max\nolimits_w} \hat{m} = \underset{(\hat{m},w)\in \text{Invits}_{old} \cup \{(m,w)\}}{\max\nolimits_w} \hat{m}$

$= m$ (m is only term in)

case 2: $w \notin \text{FreeW}_{old}$

$(m',w) \in \text{Tent}_{old}$

case a) w prefers m to m'

m is better than prev. best inviter

m is new best inviter

w is paired with m

$m \overset{\text{preferred by } w}{>_w} m' = \underset{(\hat{m},w)\in \text{Inv}_{old}}{\max\nolimits_w} \hat{m}$

$m = \underset{(\hat{m},w)\in \text{Inv}_{old} \cup \{(m,w)\}}{\max\nolimits_w} \hat{m} = \underset{(\hat{m},w)\in \text{Inv}_{new}}{\max\nolimits_w} \hat{m}$

$\underbrace{(m,w) \in \text{Tent}_{new}}_{\text{code}}$, so $\text{MatchW}_{new}(w) = m$ ✓

b) w prefers m' to m

m is not better than prev. best inviter

m' is still best inviter

w still paired with m'

$m <_w m' = \underset{(\hat{m},w)\in \text{Inv}_{old}}{\max\nolimits_w} \hat{m}$

$m' = \underset{(\hat{m},w)\in \text{Inv}_{old} \cup \{(m,w)\}}{\max\nolimits_w} \hat{m} = \underset{(\hat{m},w)\in \text{Inv}_{new}}{\max\nolimits_w} \hat{m}$

$(m',w) \in \text{Tent}_{new}$, so $\text{MatchW}_{new}(w) = m'$ ✓

$(m',w) \in \text{Tent}_{old}$; code doesn't modify in this case

Maintenance (matches improve for welders)

```
FreeM <- M
FreeW <- W
Invitations <- {}
Tentative <- {}
k <- 0

while there is an m in FreeM s.t. there is a w s.t. (m,w) not in Invitations
    choose such an m
    let w be m's highest ranked s.t. (m,w) not in Invitations

    add (m,w) to Invitations

    if w in FreeW then
        remove w from FreeW
        remove m from FreeM
        add (m,w) to Tentative
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
            remove m from FreeM
            add m' to FreeM
            remove (m', w) from Tentative
            add(m, w) to Tentative
    k <- k+1
return Tentative
```

Suppose INV is T before loop and $\exists m \in FreeM, w$ s.t. $(m,w) \notin Invites$

Notation: m s.t. $(m,w) \in Tent$ (or NIL) w's highest pref $\hat{m}$ s.t. $(\hat{m},w) \in Invites$

f) $\forall w,\ MatchW(w) = \max_{w\ (\hat{m},w)\in Invites} \hat{m}$

every w is matched with their favorite machinist they've received an invitation from (best inviter)

Only change to Tent / Invites is for w (so if true before loop for other welders, still true after)

$MatchW_{old}(w) = \max_{w\ (m,w)\in Invites_{old}} m$ (INV)

case 1: $w \in FreeW_{old}$

$\neg \exists m$ s.t. $(m,w) \in Invites_{old}$ (INV b)

$MatchW_{new}(w) = m$ (code - add (m,w) to Tent)

$\max_{w\ (\hat{m},w)\in Invites_{new}} \hat{m} = \max_{w\ (\hat{m},w)\in Invites_{old} \cup \{(m,w)\}} \hat{m}$

$= m$ (m is only term in)

case 2: $w \notin FreeW_{old}$

$(m',w) \in Tent_{old}$

case a) w prefers m to m'

m is better than prev. best inviter

m is new best inviter

w is paired with m

$m >_w m' = \max_{w\ (\hat{m},w)\in Inv_{old}} \hat{m}$ (preferred by w)

$m = \max_{w\ (\hat{m},w)\in Inv_{old}\cup\{(m,w)\}} \hat{m} = \max_{w\ (\hat{m},w)\in Inv_{new}} \hat{m}$

$\underbrace{(m,w) \in Tent_{new}}_{code}$, so $MatchW_{new}(w) = m$ ✓

b) w prefers m' to m

m is not better than prev. best inviter

m' is still best inviter

w still paired with m'

$m <_w m' = \max_{w\ (\hat{m},w)\in Inv_{old}} \hat{m}$

$m' = \max_{w\ (\hat{m},w)\in Inv_{old}\cup\{(m,w)\}} \hat{m} = \max_{w\ (\hat{m},w)\in Inv_{new}} \hat{m}$

$(m',w) \in Tent_{new}$, so $MatchW_{new}(w) = m'$ ✓

$(m',w) \in Tent_{old}$; code doesn't modify in this case

Maintenance (hard part)

Suppose INV is T before loop and $\exists m \in FreeM, w$ s.t. $(m,w) \notin Invites$

c) Tent is a matching and stablish (restricted to matched m, w)

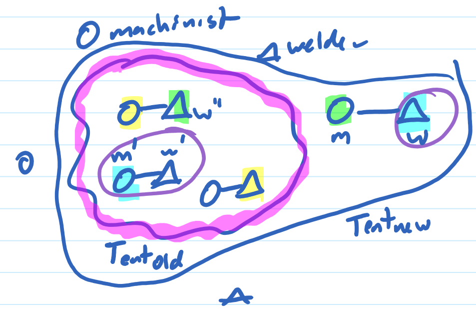
```
FreeM <- M
FreeW <- W
Invitations <- {}
Tentative <- {}
k <- 0

while there is an m in FreeM s.t. there is a w s.t. (m,w) not in Invitations
    choose such an m
    let w be m's highest ranked s.t. (m,w) not in Invitations

    add (m,w) to Invitations

    if w in FreeW then
        remove w from FreeW
        remove m from FreeM
        add (m,w) to Tentative
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
            remove m from FreeM
            add m' to FreeM
            remove (m', w) from Tentative
            add(m, w) to Tentative
    k <- k+1
return Tentative
```

case 1



if $Tent_{new}$ has an instability, it must be between (m, w'')

so new instab must be (m', w)

m' prefers $\underline{w}$ to w'

$w \in FreeW_{old}$

no invitations to w

m' has sent $\underline{w'}$ invitation

m' has sent w invitation $\Rightarrow\Leftarrow$

so m', w is not an instability