SelectonSort

```
SelectionSort(A)
  for i = 0 to n-2
    find min among A[i],...,A[n-1]
    swap min with A[i]
```

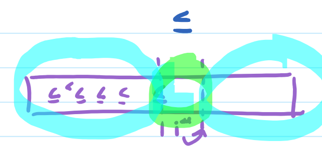
$\forall j,\ 0 \le j \le i-1 \le,\ A[j] \le A[j+1]$

$i = \cancel{3}\ 4$ a) $A[1] \le A[1] \le A[2] \le A[3]$
b) $A[\cancel{2}\ 3] \le A[\cancel{3}\ 4], \ldots$
c) curr A is perm of orig

INV: a) $A[0] \le A[1] \le \cdots \le A[i-1]$

and b) $i = 0$ or $A[i-1] \le A[i], \ldots, A[n-1]$

and c) values in A are a permutation of the original values in A



Init: $(i=0)$ a) vacuous b) $i=0$ c) haven't touched A yet

Maintenance: Suppose INV T after i iterations and $i \le n-2$ [want INV T after next iteration]

$A^{new}[0] \le A^{new}[1] \le \cdots \le A[i^{new}-1]$
$A^{new}[i^{new}-1] \le A[i^{new}], \ldots, A^{new}[n-1]$

min is $\le$ all things considered when choosing min [then $\min \le A^{old}[i^{old}], \ldots, A^{old}[n-1]$ (choice of min)
$\min = A^{old}[j]$ for some j in $i^{old}, \ldots, n-1$

and $A^{new}[i^{old}] = \min$ $\quad A^{new}[j] = A^{old}[i^{old}]$ (result of swap)

min $\le$ all things at or after $i^{old}+1$

so $\min \le A^{old}[i^{old}+1], \ldots, A^{old}[j-1], A^{old}[i^{old}], A^{old}[j+1], \ldots, A^{old}[n-1]$ (reorder)

$A^{new}[i^{old}] \le A^{new}[i^{new}], \ldots, A^{new}[j-1], A^{new}[j], A^{new}[j+1], \ldots, A^{new}[n-1]$ ($i^{new} = i^{old}+1$; result of swap)

$\quad = A^{new}[i^{new}-1]$ so INV b is true

and $A^{new}[i^{new}-2] = A^{new}[i^{old}-1] = A^{old}[i^{old}-1] \le A^{old}[j] = \min = A^{new}[i^{old}] = A^{new}[i^{new}-1]$

$i^{new} = i^{old}+1$ didn't change index $i^{old}-1$ INV b) so INV a true $i^{new} = i^{old}+1$

Termination: terminates when $i = n-1$

$A[i-2] \le A[i-1]$

Post-Condition: INV says $\underbrace{A[0] \le \cdots \le A[i-1]}_{\text{INV a}}$ and $\underbrace{A[i-1] \le A[i]}_{\text{INV b}}$

so $A[0] \le \cdots \le A[n-2]$ and $A[n-2] \le A[n-1]$ $(i = n-1)$

so $A[0] \le \cdots \le A[n-1]$

and A is a permutation of original (INV c)] so A is correctly sorted

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 track unmatched workers

b) $\forall m,w,\ (m,w) \in Tent \longrightarrow MatchM[m] = w \wedge MatchW[w] = m$
$\forall m,\ MatchM[m] \neq NIL \longrightarrow (m, MatchM[m]) \in Tent$
$\forall w,\ MatchW[w] \neq NIL \longrightarrow (MatchW[w], w) \in Tent$

matchM / matchW track current tentative matching

c) $\forall m\ m \in FreeM \longleftrightarrow MatchM[m] = NIL$
$\forall w\ w \in FreeW \longleftrightarrow MatchW[w] = NIL$

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

welder is free if and only if no invitations yet

e) $Invites = \bigcup_{i=1}^{n} \bigcup_{j=1}^{Next[i]-1} \{(m_i, PrefM[i][j])\}$

Next tracks invitations that have been made

f) Tent is a matching

g)

h) $|Invites| = k$

iterations of outer loop

MatchW[w] after jth iteration of loop

i) $\forall w,\ j<k,\ MatchW^{j}[w] \neq NIL \rightarrow MatchW^{j+1}[w], \ldots, MatchW^{k}[w]\ != NIL$

j) $\forall w,\ j<k,\ MatchW^{j}[w] \neq NIL \rightarrow$ w prefers $MatchW^{j+1}[w]$ to $MatchW^{j}[w]$, or they are same

seq of matched machinists improves from point of view of each welder

Bookkeeping for Proofs

```
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
    FreeW^k <- FreeW
    FreeM^k <- FreeM
    Tentative^k <- Tentative
    ...
return Tentative
```

like FreeW^k, etc..

Let A' be same as algorithm A but with write-only non-output variables removed

Then A' has same output as A (prove using invariant "at each step, the remaining variables have the same values in A, A')

Proof of INV

```
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
```

INV k: $|\text{Invites}| = k$

Init: $(k=0)$ init of Invites makes $\text{Invites} = \emptyset$
so $|\text{Invites}| = 0$ ✓

Maintenance: Suppose INV true after k iterations
and guard is T
[show INV k is true after k+1 iter]

Then $|\text{Invites}^{old}| = k$ (INV k)

$(m,w) \notin \text{Invites}^{old}$

$\text{Invites}^{new} = \text{Invites}^{old} \cup \{(m,w)\}$

$|\text{Invites}^{new}| = |\text{Invites}^{old}| + 1$ (size of disjoint union is sum of sizes)

$= k+1$

Proof of INV

FreeM <- M
FreeW <- W
Invitations <- {}
Tentative <- {}
MatchW[w] ← NIL for all w

```
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
  1)    remove w from FreeW  *
        remove m from FreeM
        add (m,w) to Tentative   MatchW[w] ← m  **
    else
        find m' s.t. (m', w) in Tentative
        if w prefers m to m'
  2)        remove m from FreeM
            add m' to FreeM
            remove (m', w) from Tentative
  3)        add(m, w) to Tentative   MatchW[w] ← m  ***
return Tentative
```

INV c2: $\forall w$, $w \in FreeW \longleftrightarrow MatchW[w] = NIL$

Init: $(k=0)$ initialization makes all $w \in FreeW$ and $MatchW[w] = NIL$ for all w

Maintenance: Suppose INV T after k iterations and guard is T [show INV c2 is T after $k+1^{st}$ iter]

Suppose INV c2 is F after k+1 iter
then $w \in FreeW^{new}$ but $MatchW^{new}[w] \neq NIL$ for some w
or $w \notin FreeW^{new}$ but $MatchW^{new}[w] = NIL$ for some w

and "some w" is the chosen w
(because FreeW, MatchW not changed for any other w)

case 1: $\underbrace{w \notin FreeW^{new}}_{*}$ and $\underbrace{MatchW^{new}[w] = m}_{**} \neq NIL$

so $w \in FreeW^{new} \leftrightarrow MatchW^{new}[w] = NIL$

case 2: $w \notin FreeW^{old} = FreeW^{new}$ and $\underbrace{MatchW^{new}[w] = m}_{***} \neq NIL$

1st if — no changes to FreeW

so $w \in FreeW^{new} \longleftrightarrow MatchW^{new}[w] = NIL$

case 3: $w \notin FreeW^{old} = FreeW^{new}$ and $MatchW^{new}[w] = MatchW^{old}[w]$

no change to MatchW

and $MatchW^{old}[w] \neq NIL$ (INV c2)

so $MatchW^{new}[w] \neq NIL$

and $w \in FreeW^{new} \longleftrightarrow MatchW^{new}[w] = NIL$

in all 3 cases, the code makes $w \in FreeW^{new} \longleftrightarrow MatchW^{new}[w] = NIL$ true

Proof of INV

```
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   MatchW[w] <- m
    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   MatchW[w] <- m
return Tentative
```

INV: $\forall w,\ j < k,\ MatchW^j[w] \neq NIL \rightarrow$ w prefers $MatchW^{j+1}[w]$ to $MatchW^j[w]$, or $MatchW^j[w] = MatchW^{j+1}[w]$

Init: vacuous (initialization sets $MatchW^0[w] = NIL$ for all w)

Maintenance: Suppose INV T after k iterations and guard is T [show INV; T after k+1 iter]

↪ new parts are for $j=k$

For $j \leq k$, $MatchW^j[w]$ doesn't change ($MatchW^j$ is old values of MatchW)

For any w' not picked
$MatchW^{k+1}[w'] = MatchW^k[w]$ (code doesn't change anything for $w' \neq w$)

For w picked

case 1) $MatchW^k[w] = NIL$ ($w \in FreeW$ so $MatchW[w] = NIL$ by INV c2)
if part F, so if/then is T

2) $MatchW^{k+1}[w] = m$, $MatchW^k[w] = m'$ ← ($(m',w) \in Tent$; INV b1)
w prefers m to m' (condition on case 2)
1st part of or in then part, so if/then is T

3) $MatchW^{k+1}[w] = MatchW^k[w]$
2nd part of or in then part, so if/then is T