Yonatan Golan, 206387383

NOTE: all the algorithms that include random inside them sometimes don't find a solution, just re-run it.

<u>Problem 2: CSP I</u>

The 4-queens problem as a CSP:

$\boldsymbol{Variables}$:

- $Q_i - the\ place\ of\ queen_i\ on\ the\ board,\ \forall i \in [1,4]$

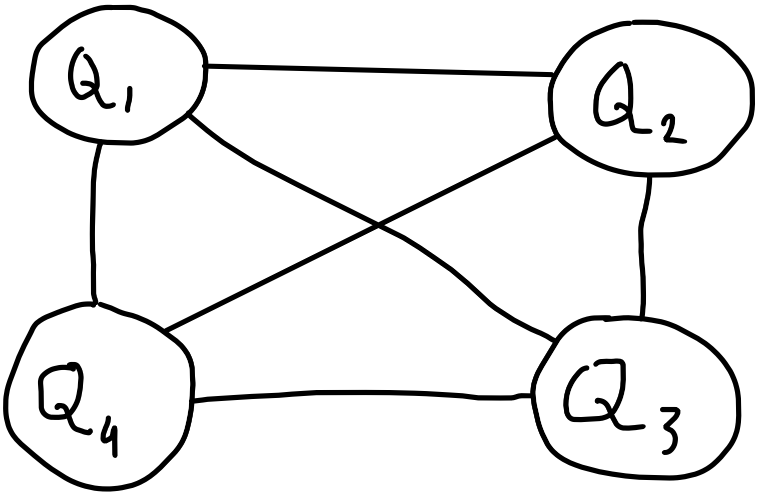
$\boldsymbol{Domains}$:

- $D_i - the\ domain\ of\ places\ for\ queen_i,\ \forall i \in [1,4]$
  $D_i \in \{(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),(2,3),(2,4),$
  $(3,1),(3,2),(3,3),(3,4),(4,1),(4,2),(4,3),(4,4)\}$

$\boldsymbol{Constraints}$:

- $\forall i,j \in [1,4] \cap (i \neq j)$:
- $Q_i \neq Q_j$: $the\ place\ of\ two\ queens\ can't\ be\ the\ same$
- $row(Q_i) \neq row(Q_j)$: $two\ queens\ can't\ be\ in\ the\ same\ row$
- $col(Q_i) \neq col(Q_j)$: $two\ queens\ can't\ be\ in\ the\ same\ column$
- $|row(Q_{i)} - row(Q_j)| \neq |col(Q_i) - col(Q_j)|$: $two\ queens\ can't\ be\ in\ the\ same\ diagonal$

$\boldsymbol{CSP\ graph}$: $an\ arc\ between\ two\ nodes\ resembles\ that\ node_i\ has\ a\ constraint\ on\ node_j$



Problem 3: CSP II

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
inputs: *csp*, a binary CSP with components (X, D, C)
local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**
(X_i, X_j) ← REMOVE-FIRST(*queue*)
if REVISE(*csp*, X_i, X_j) **then**
if size of D_i = 0 **then return** *false*
for each X_k in X_i.NEIGHBORS − {X_j} **do**
add(X_k, X_i) to *queue*
return *true*

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i

revised ← *false*
for each *x* **in** D_i **do**
if no value *y* in D_j allows (*x*, *y*) to satisfy the constraint
between X_i and X_j **then**
delete *x* from D_i
revised ← *true*
return *revised*

d – maximum domain size

e – number of arcs

Complexity of Revise:
the revise function checks a specific arc, it checks if the domain x in Di stands with any other domain y in Dj to satisfy the constraint. That means that the function could possibly examin all the values of the second domain (Dj) – and it does it for every domain x in Di.
so if lets say that all domains are in the same length, revise could possible run d^2 times.

Complexity of AC3:

The initial queue has e arcs, for each one of them we call to the revise function, so to that point we're calling the revise function **e** number of times
The thing is that we can revise the domains, and that leads to adding arcs once again to the queue. Every arc validates only one domain at a time, so I can add the specific arc back to the queue only as many times as |D| (== d).

so in the worst case I'd add a specific arc back to the queue **ed** times (for every domain I'd check revise and lets say I delete every x in Di and then add the arc once again to check it with the new revised domain)
All in all – I potentially could call revise (e + ed) times -> O((e+ed)d^2)) ⇔ O(ed^3).