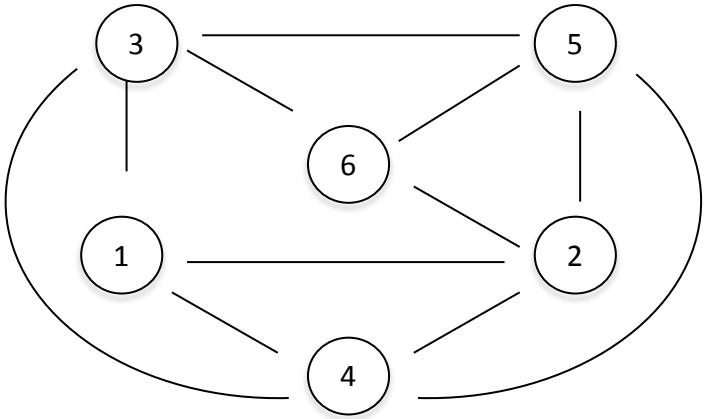
Εργαστήριο Τεχνητή Νοημοσύνη ΙΙ

Παύλος Πέππας

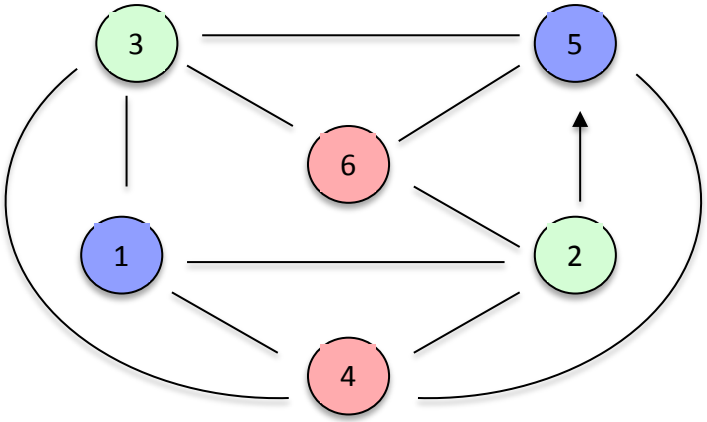
Τμήμα Ηλεκτρολόγων Μηχανικών

και Τεχνολογίας Υπολογιστών

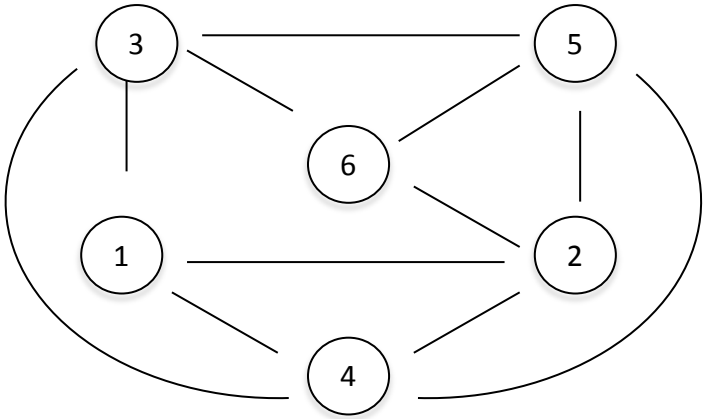
Χρωματισμός Γράφου



Χρωματισμός Γράφου



Χρωματισμός Γράφου



Βάση Γνώσης

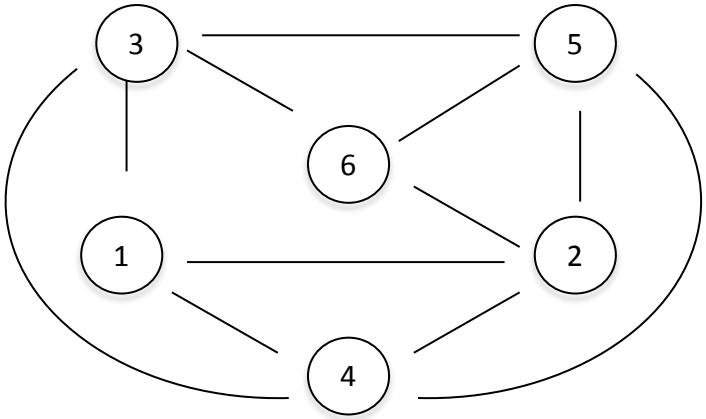
$node(1) \wedge node(2) \wedge node(3) \wedge node(4) \wedge node(5) \wedge node(6)$

$edge(1,2) \wedge edge(1,3) \wedge edge(1,4) \wedge edge(2,4) \wedge edge(2,5) \wedge edge(2,6) \wedge edge(3,4) \wedge edge(3,5) \wedge edge(3,6) \wedge edge(4,5) \wedge edge(5,6)$

$\forall x \forall y(edge(x,y) \Rightarrow edge(y,x)$

$color(red) \wedge color(green) \wedge color(blue)$

Χρωματισμός Γράφου



Βάση Γνώσης

$node(1) \wedge node(2) \wedge node(3) \wedge node(4) \wedge node(5) \wedge node(6)$

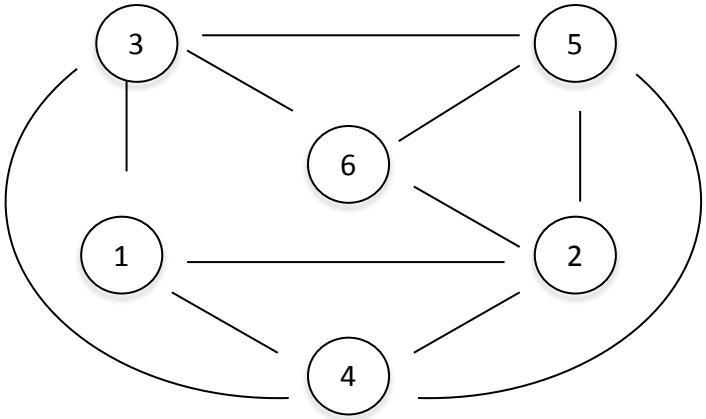
$edge(1,2) \wedge edge(1,3) \wedge edge(1,4) \wedge edge(2,4) \wedge edge(2,5) \wedge edge(2,6) \wedge edge(3,4) \wedge edge(3,5) \wedge edge(3,6) \wedge edge(4,5) \wedge edge(5,6)$

$\forall x \forall y(edge(x,y) \Rightarrow edge(y,x)$

$color(red) \wedge color(green) \wedge color(blue)$

$\forall x (node(x) \Rightarrow \exists z(color(z) \wedge paint(x,z))$

Χρωματισμός Γράφου



Βάση Γνώσης

node(1) ∧ node(2) ∧ node(3) ∧ node(4) ∧ node(5) ∧ node(6)

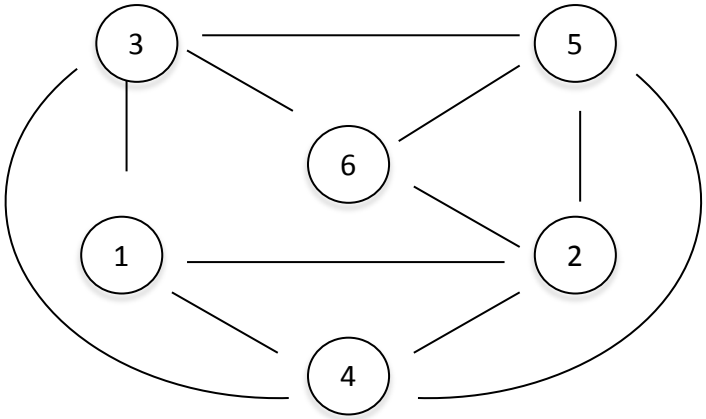
edge(1 ,2) ∧ edge(1 ,3) ∧ edge(1 ,4) ∧ edge(2 ,4) ∧ edge(2 ,5) ∧ edge(2 ,6) ∧ edge(3 ,4) ∧ edge(3 ,5) ∧ edge(3 ,6) ∧ edge(4 ,5) ∧ edge(5 ,6)

∀x ∀y(edge(x,y) ⇒ edge(y,x)

color(red) ∧ color(green) ∧ color(blue)

∀x (node(x) ⇒ ∃z(color(z) ∧ paint(x,z))
∀x ∀y ∀z (paint(x,y) ∧ paint(x,z) ⇒ y=z)

Χρωματισμός Γράφου



Βάση Γνώσης

$node(1) \wedge node(2) \wedge node(3) \wedge node(4) \wedge node(5) \wedge node(6)$

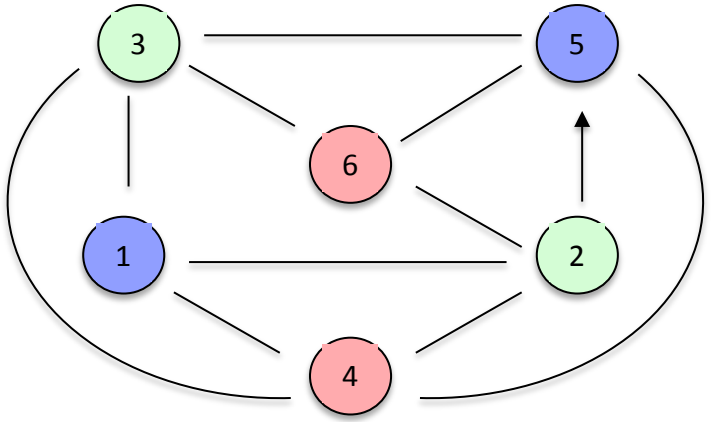
$edge(1,2) \wedge edge(1,3) \wedge edge(1,4) \wedge edge(2,4) \wedge edge(2,5) \wedge edge(2,6) \wedge edge(3,4) \wedge edge(3,5) \wedge edge(3,6) \wedge edge(4,5) \wedge edge(5,6)$

$\forall x \forall y (edge(x,y) \Rightarrow edge(y,x)$

$color(red) \wedge color(green) \wedge color(blue)$

$\forall x (node(x) \Rightarrow \exists z (color(z) \wedge paint(x,z))$
$\forall x \forall y \forall z (paint(x,y) \wedge paint(x,z) \Rightarrow y=z)$
$\neg(\exists x \exists y \exists z (edge(x,y) \wedge color(x,z) \wedge color(y,z))$

Χρωματισμός Γράφου



$paint^M$ = { (1,blue), (2,green), (3,green), (4,red), (5,blue), (6,red) }

Βάση Γνώσης

$node(1) \wedge node(2) \wedge node(3) \wedge node(4) \wedge node(5) \wedge node(6)$

$edge(1,2) \wedge edge(1,3) \wedge edge(1,4) \wedge edge(2,4) \wedge edge(2,5) \wedge edge(2,6) \wedge edge(3,4) \wedge edge(3,5) \wedge edge(3,6) \wedge edge(4,5) \wedge edge(5,6)$

$\forall x \forall y (edge(x,y) \Rightarrow edge(y,x)$

$color(red) \wedge color(green) \wedge color(blue)$

$\forall x (node(x) \Rightarrow \exists z (color(z) \wedge paint(x,z))$

$\forall x \forall y \forall z (paint(x,y) \wedge paint(x,z) \Rightarrow y=z)$

$\neg(\exists x \exists y \exists z (edge(x,y) \wedge color(x,z) \wedge color(y,z))$

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

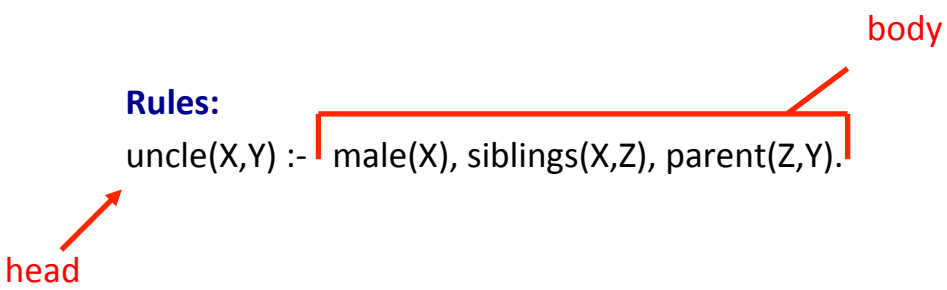
Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).



Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X, _).

Ανώνυμη μεταβλητή

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

commonFriend(X,Y) :- friends(X,_), friends(_,Y).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

commonFriend(X,Y) :- friends(X,_), friends(Y,_).

Διαφορετικά instances της ανώνυμης μεταβλητή δεν λαμβάνουν απαραίτητα την ίδια τιμή.

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2
Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

Integrity Constraints:

:- siblings(X,Y), X=Y.

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2
Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).
hasSibling(X) :- siblings(X,_).

Integrity Constraints:

:- siblings(X,Y), X=Y.

Choices:

1 { paint(X, Y): color(Y) } 1 :- node(X).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

Integrity Constraints:

:- siblings(X,Y), X=Y.

Choices:

1 { paint(X, Y): color(Y) } 1 :- node(X).

Ελάχιστο πλήθος

Μέγιστο πλήθος

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Κάθε «***ελάχιστο»*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***answer set*** του Π

Παράδειγμα

```
mother(vicky,jim).
parent(X,Y) :-  mother(X,Y).
```

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Κάθε «***ελάχιστο»*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***answer set*** του Π

Παράδειγμα

```
mother(vicky,jim).
parent(X,Y) :-  mother(X,Y).
```

Answer set

```
mother(vicky,jim)
```

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Κάθε «***ελάχιστο»*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***answer set*** του Π

Παράδειγμα

```
mother(vicky,jim).
parent(X,Y) :-  mother(X,Y).
```

Answer set

```
mother(vicky,jim),
parent(vicky,jim)
```

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Κάθε «***ελάχιστο»*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***answer set*** του Π

Παράδειγμα

```
mother(vicky,jim).
parent(X,Y) :-  mother(X,Y).
```

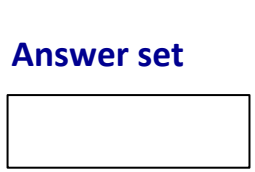
Answer set

```
mother(vicky,jim),
parent(vicky,jim)
```

Παράδειγμα

```
a :- b.
b :- a.
```

Answer set



Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Κάθε «***ελάχιστο»*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***answer set*** του Π

Παράδειγμα

```
mother(vicky,jim).
parent(X,Y) :-  mother(X,Y).
```

Answer set

mother(vicky,jim),
parent(vicky,jim)

Παράδειγμα

```
a :- b.
b :- a.
```

Answer set

Παράδειγμα

```
block(a). block(b). block(c).
1 { onTable(X) : block(X) }.
onTable(b) :- onTable(a).
```

Answer set

block(a), block(b), block(c),
onTable(c)

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Κάθε «***ελάχιστο»*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***answer set*** του Π

Παράδειγμα

```
mother(vicky,jim).
parent(X,Y) :-  mother(X,Y).
```

Answer set

mother(vicky,jim),
parent(vicky,jim)

Παράδειγμα

```
a :- b.
b :- a.
```

Answer set

Παράδειγμα

```
block(a). block(b). block(c).
1 { onTable(X) : block(X) }.
onTable(b) :- onTable(a).
```

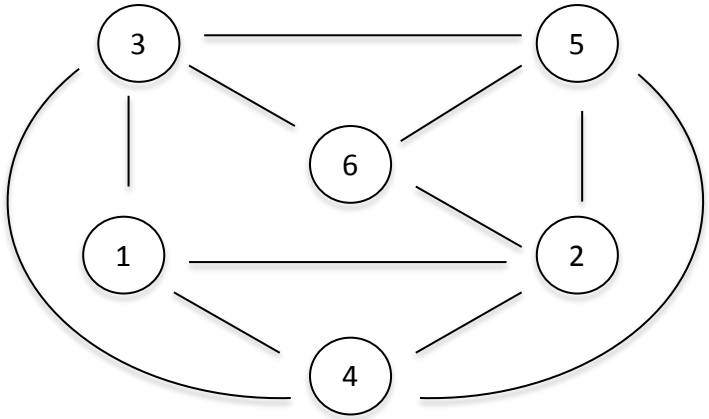
Answer set

block(a), block(b), block(c),
onTable(c)

block(a), block(b), block(c),
onTable(b)

.
.
.

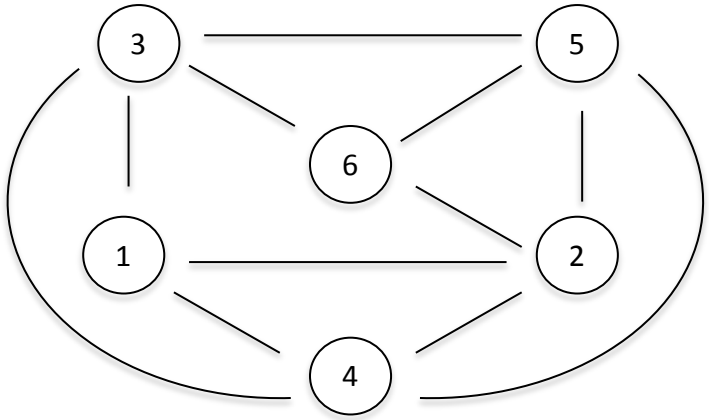
Χρωματισμός Γράφου



graphColoring.lp

```
node(1).  node(2).  node(3).
node(4).  node(5).  node(6).

edge(1,2). edge(1,3). edge(1,4).
edge(2,4). edge(2,5). edge(2,6).
edge(3,4). edge(3,5). edge(3,6).
edge(4,5). edge(5,6).

edge(Y,X) :- edge(X,Y).

color(red).  color(green). color(blue).
```

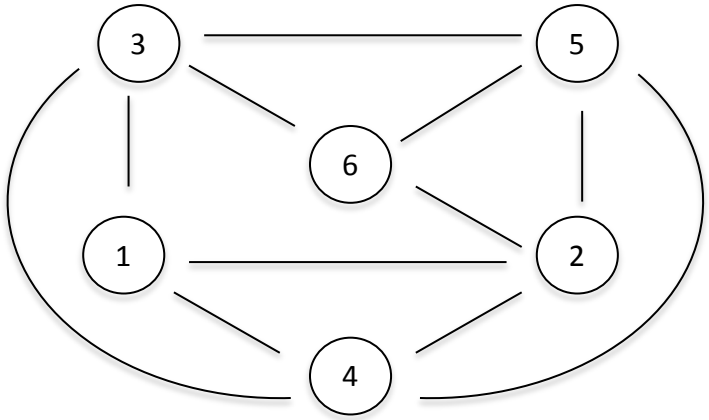
Χρωματισμός Γράφου



graphColoring.lp

```
node(1..6).

edge(1,2). edge(1,3). edge(1,4).
edge(2,4). edge(2,5). edge(2,6).
edge(3,4). edge(3,5). edge(3,6).
edge(4,5). edge(5,6).

edge(Y,X) :- edge(X,Y).

color(red).  color(green). color(blue).
```

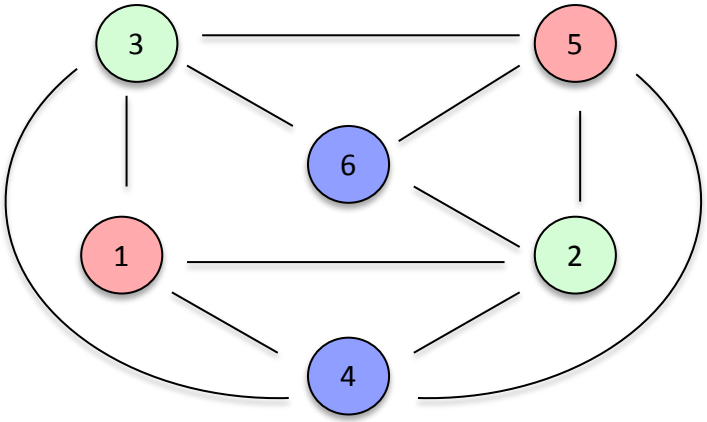
Χρωματισμός Γράφου



graphColoring.lp

```
node(1..6).

edge(1,2). edge(1,3). edge(1,4).
edge(2,4). edge(2,5). edge(2,6).
edge(3,4). edge(3,5). edge(3,6).
edge(4,5). edge(5,6).

edge(Y,X) :- edge(X,Y).

color(red).  color(green). color(blue).

1 { paint(X,Z): color(Z) }  1 :- node(X).

:- paint(X,Z), paint(Y,Z), edge(X,Y).

#show paint/2.
```

Χρωματισμός Γράφου

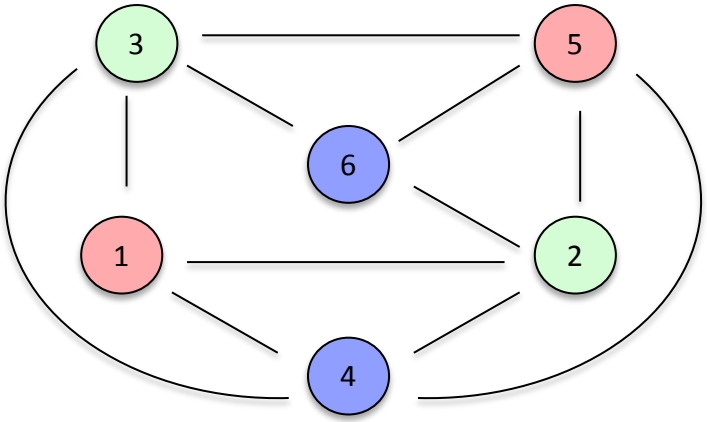


graphColoring.lp

```
node(1..6).

edge(1,2). edge(1,3). edge(1,4).
edge(2,4). edge(2,5). edge(2,6).
edge(3,4). edge(3,5). edge(3,6).
edge(4,5). edge(5,6).

edge(Y,X) :- edge(X,Y).

color(red).  color(green). color(blue).

1 { paint(X,Z): color(Z) }  1 :- node(X).

:- paint(X,Z), paint(Y,Z), edge(X,Y).

#show paint/2.
```

```
Labs — -bash — 57×19
<pavlos>$ clingo graphColoring.lp
clingo version 5.5.0
Reading from graphColoring.lp
Solving...
Answer: 1
paint(1,red) paint(2,green) paint(3,green) paint(4,blue)
paint(5,red) paint(6,blue)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.010s (Solving: 0.00s 1st Model: 0.00s Un
sat: 0.00s)
CPU Time     : 0.004s
```

Χρωματισμός Γράφου



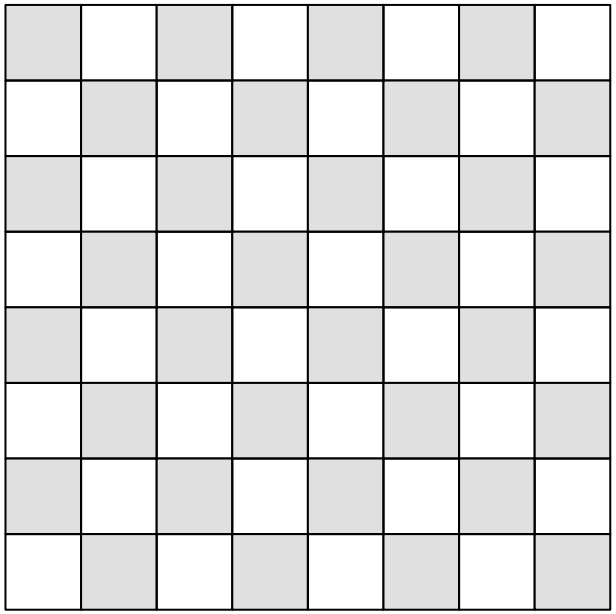
graphColoring.lp

```
node(1..6).

edge(1,2). edge(1,3). edge(1,4).
edge(2,4). edge(2,5). edge(2,6).
edge(3,4). edge(3,5). edge(3,6).
edge(4,5). edge(5,6).

edge(Y,X) :- edge(X,Y).

color(red).  color(green). color(blue).

1 { paint(X,Z): color(Z) }  1 :- node(X).

:- paint(X,Z), paint(Y,Z), edge(X,Y).

#show paint/2.
```

```
Labs — -bash — 57×19
<pavlos>$ clingo graphColoring.lp
clingo version 5.5.0
Reading from graphColoring.lp
Solving...
Answer: 1
paint(1,red) paint(2,green) paint(3,green) paint(4,blue)
paint(5,red) paint(6,blue)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.010s (Solving: 0.00s 1st Model: 0.00s Un
sat: 0.00s)
CPU Time     : 0.004s
```

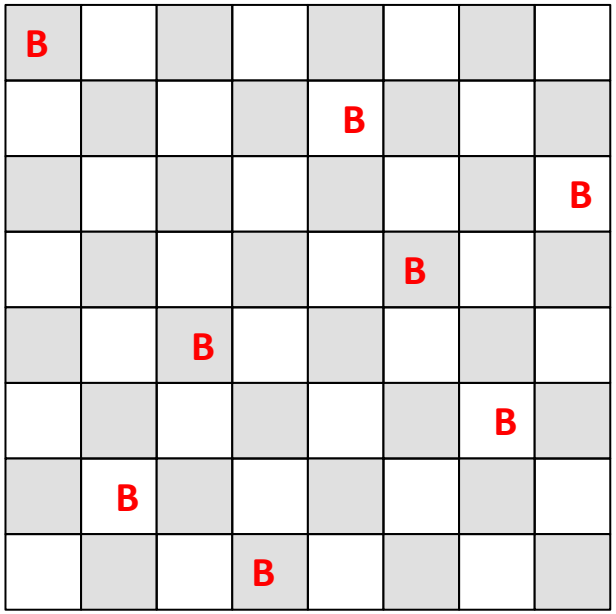
https://potassco.org/clingo/run/

Άσκηση



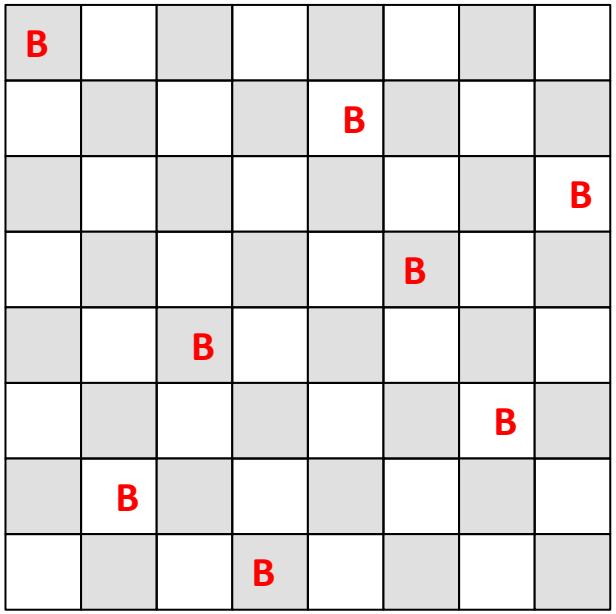
Τοποθετήσετε 8 βασίλισσες στην σκακιέρα χωρίς να αλληλο-απειλούνται.

Άσκηση



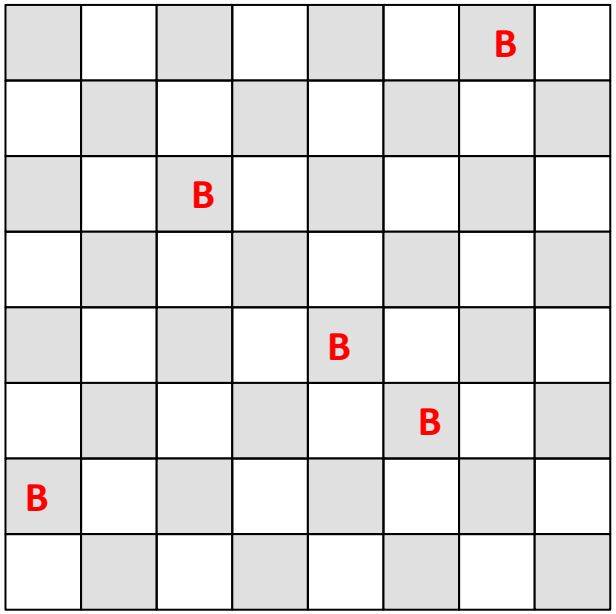
Τοποθετήσετε 8 βασίλισσες στην σκακιέρα χωρίς να αλληλο-απειλούνται.

Άσκηση




```
#const n=8.
1 { queen(I,J): J=1..n } 1 :- I=1..n.

:- queen(I1,J), queen(I2,J), I1 != I2.
:- queen(I1,J1), queen(I2,J2), I1 != I2, |I1-I2| = |J1-J2|.
```

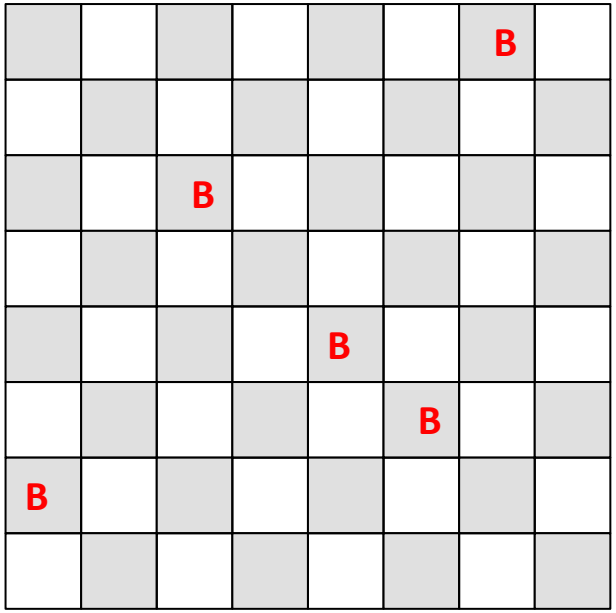
Άσκηση



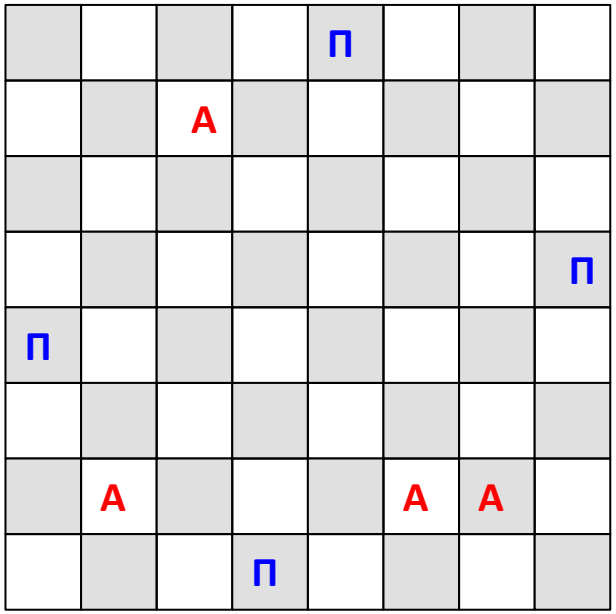
Τοποθετήσετε 5 βασίλισσες στην σκακιέρα έτσι ώστε να απειλούνται όλες οι θέσεις της σκακιέρας.

Λύση

```
#const n=8.
5 { queen(I,J): I=1..n, J=1..n } 5.

threaten(I,J) :- queen(I,_), J=1..n.
threaten(I,J) :- queen(_,J), I=1..n.
threaten(I,J) :- queen(X,Y), |I-X| = |J-Y|, I=1..n,
J=1..n.

:- not threaten(I,J), I=1..n, J=1..n.

% #minimize{1,I,J: queen(I,J)}.
#show queen/2.
```



Άσκηση



Τοποθετήσετε 4 πύργους και 4 αξιωματικούς στην σκακιέρα χωρίς να αλληλο-απειλούνται. Επιπλέον όλες οι στήλες θα πρέπει να έχουν ένα πύργο ή αξιωματικό.

Λύση

```
#const n=8.
n/2 { tower(X,Y): X=1..n, Y=1..n } n/2.
n/2 { bishop(X,Y): X=1..n, Y=1..n } n/2.

piece(X,Y) :- tower(X,Y).
piece(X,Y) :- bishop(X,Y).

threaten(X,Y) :- piece(X,Y), tower(X,Y1), Y != Y1.
threaten(X,Y) :- piece(X,Y), tower(X1,Y), X != X1.
threaten(X,Y) :- piece(X,Y), bishop(I,J), |X-I| = |Y-J|,
X != I.
threaten(X,Y) :- piece(X,Y), bishop(I,J), |X-I| = |Y-J|,
Y != J.

:- tower(X,Y), bishop(X,Y).
:- threaten(X,Y), X=1..n, Y=1..n.

:- not tower(_,Y), not bishop(_,Y), Y=1..8.

#show tower/2.
#show bishop/2.
```