

# Εργαστήριο Τεχνητή Νοημοσύνη II

Παύλος Πέππας

Τμήμα Ηλεκτρολόγων Μηχανικών  
και Τεχνολογίας Υπολογιστών

# Aggregates

student(john). student(mary). student(helen).

course(db). course(ai1). course(ai2). course(thesis). course(java).

credits(db,6). credits(ai1,5). credits(ai2,5). credits(thesis,10). credits(java,4).

{ select(X,Y) : course(Y) } :- student(X).

# Aggregates

student(john). student(mary). student(helen).

course(db). course(ai1). course(ai2). course(thesis). course(java).

credits(db,6). credits(ai1,5). credits(ai2,5). credits(thesis,10). credits(java,4).

{ select(X,Y) : course(Y) } :- student(X).

:- student(X), **#sum**{ Z,Y : select(X,Y), credits(Y,Z) } < 10.

:- **#count**{ X : select(X,thesis) } > 1.

#show select/2.

# Aggregates

student(john). student(mary). student(helen).

course(db). course(ai1). course(ai2). course(thesis). course(java).

credits(db,6). credits(ai1,5). credits(ai2,5). credits(thesis,10). credits(java,4).

{ select(X,Y) : course(Y) } :- student(X).

:- student(X), **#sum**{ Z,Y : select(X,Y), credits(Y,Z) } < 10.

:- **#count**{ X : select(X,thesis) } > 1.

#show select/2.

---

\$ clingo aggregates.lp

clingo version 5.5.0

Reading from aggregates.lp

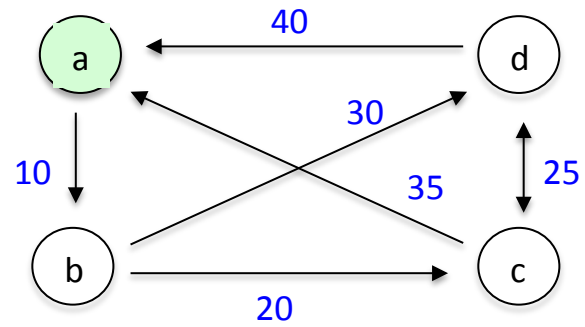
Solving...

Answer: 1

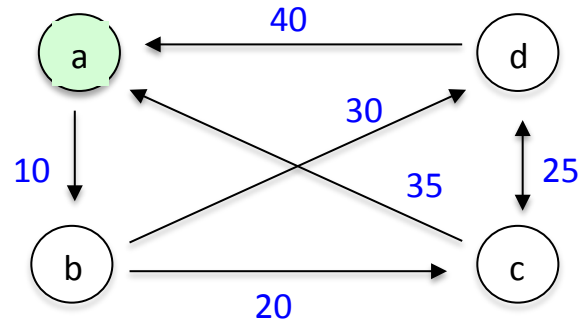
select(john,ai1) select(john,ai2) select(mary,thesis) select(helen,ai1) select(helen,ai2)

SATISFIABLE

# Travelling Salesman



# Travelling Salesman



```

#const n=4.
city(a). city(b). city(c). city(d).
road(a,b,10). road(b,c,20). road(c,d,25). road(d,a,40).
road(b,d,30). road(d,c,25). road(c,a,35).
at(a,0).
  
```

```

% Στοχος
visited(C) :- at(C,T), T=0..n-1.
:- city(C), not visited(C).
  
```

```

% Επιλογή Κινήσεων
1 { go(X,T): city(X) } 1 :- T=0..n-2.
go(a,n-1).
  
```

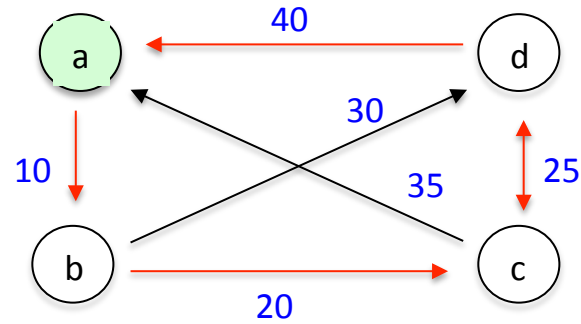
% Effect Axiom

```
at(Y,T+1) :- at(X,T), road(X,Y,_), go(Y,T).
```

```
#minimize { D,X,Y,T : at(X,T), go(Y,T), road(X,Y,D) }.
```

```
#show go/2.
```

# Travelling Salesman



```
$ clingo salesman3.lp
```

```
clingo version 5.5.0
```

```
Reading from salesman3.lp
```

```
Solving...
```

```
Answer: 1
```

```
go(a,3) go(b,0) go(d,1) go(c,2)
```

```
Optimization: 100
```

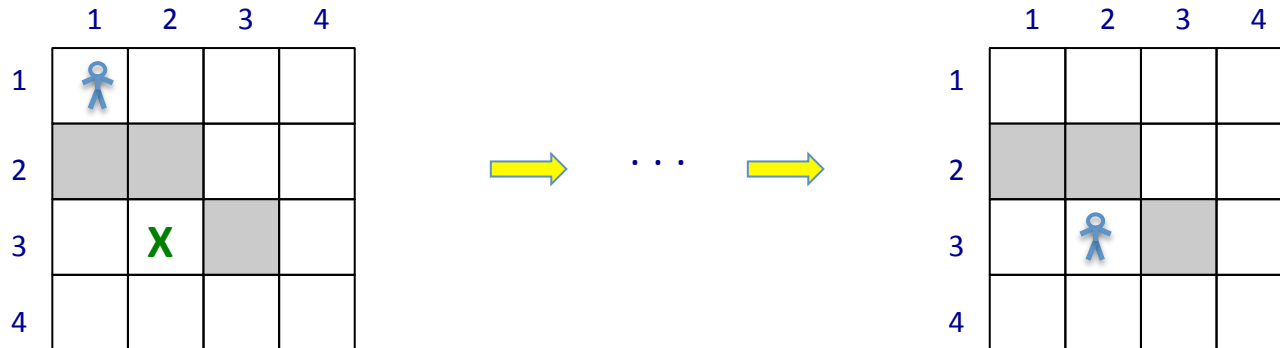
```
Answer: 2
```

```
go(a,3) go(b,0) go(c,1) go(d,2)
```

```
Optimization: 95
```

```
OPTIMUM FOUND
```

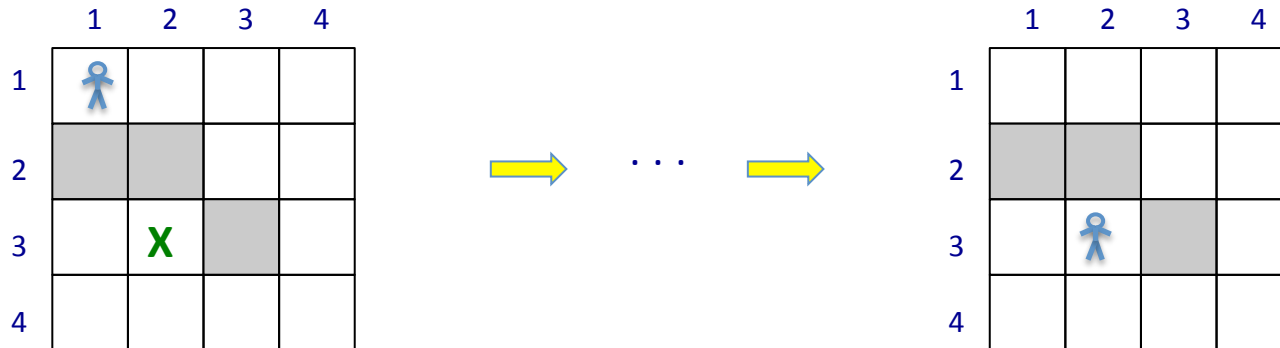
# Άσκηση



Το robot μπορεί να μετακινηθεί πάνω στο πλέγμα -- κατά μια θέση την φορά -- πάνω, κάτω, δεξιά, αριστερά, με την προϋπόθεση πως δεν βγαίνει εκτός πλέγματος, και δεν υπάρχει εμπόδιο στον προορισμό.



# Άσκηση



Το robot μπορεί να μετακινηθεί πάνω στο πλέγμα -- κατά μια θέση την φορά -- πάνω, κάτω, δεξιά, αριστερά, με την προϋπόθεση πως δεν βγαίνει εκτός πλέγματος, και δεν υπάρχει εμπόδιο στον προορισμό.

```
#const rows=4.
#const cols=4.
#const n=rows*cols.
blocked(2,1). blocked(2,2). blocked(3,3).
direction(up; down; left; right).
```

```
% Αρχική Κατάσταση
robot(1,1,0).
```

```
% Στόχος
goal :- robot(3,2,n).
:- not goal.
```

```
% Επιλογή Κινήσεων
0 { move(D,T): direction(D) } 1 :- T = 0..n-1.
```


```
% Ελαχιστοποίηση Κινήσεων
action(T) :- move(_,T).
lastAction(T) :- T = #max{ X : action(X) }.
#minimize { T: lastAction(T) }.
```

```
·
·
·
```

# Λύση

```
#const rows=4.  
#const cols=4.  
#const n=rows*cols.  
  
% Εμπόδια - κατευθύνσεις.  
blocked(2,1). blocked(2,2). blocked(3,3).  
direction(up;down;left;right).  
  
% Αρχική Κατάσταση  
robot(1,1,0).  
  
% Στοχος  
goal :- robot(3,2,n).  
:- not goal.  
  
% Επιλογή Κινήσεων  
0 { move(D,T): direction(D) } 1 :- T = 0..n-1.  
  
% Ελαχιστοποίηση Κινήσεων  
action(T) :- move(_,T).  
lastAction(T) :- T = #max{ X : action(X) }.  
#minimize { T: lastAction(T) }.
```

```
% Effect Axioms  
robot(X-1,Y,T+1) :- robot(X,Y,T), X>1,  
                        not blocked(X-1,Y), move(up,T).  
robot(X+1,Y,T+1) :- robot(X,Y,T), X<rows,  
                        not blocked(X+1,Y), move(down,T).  
robot(X,Y-1,T+1) :- robot(X,Y,T), Y>1,  
                        not blocked(X,Y-1), move(left,T).  
robot(X,Y+1,T+1) :- robot(X,Y,T), Y<cols,  
                        not blocked(X,Y+1), move(right,T).  
  
% Frame Axiom  
robot(X,Y,T+1) :- robot(X,Y,T), not move(_,T), T=0..n-1.  
  
#show move/2.
```

	1	2	3	4
1				
2				
3		X		
4				