ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Τεχνητή Νοημοσύνη Ι

Διαφάνειες Εργαστηρίου

Σγάρμπας Κυριάκος

Πολυτεχνική Σχολή

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

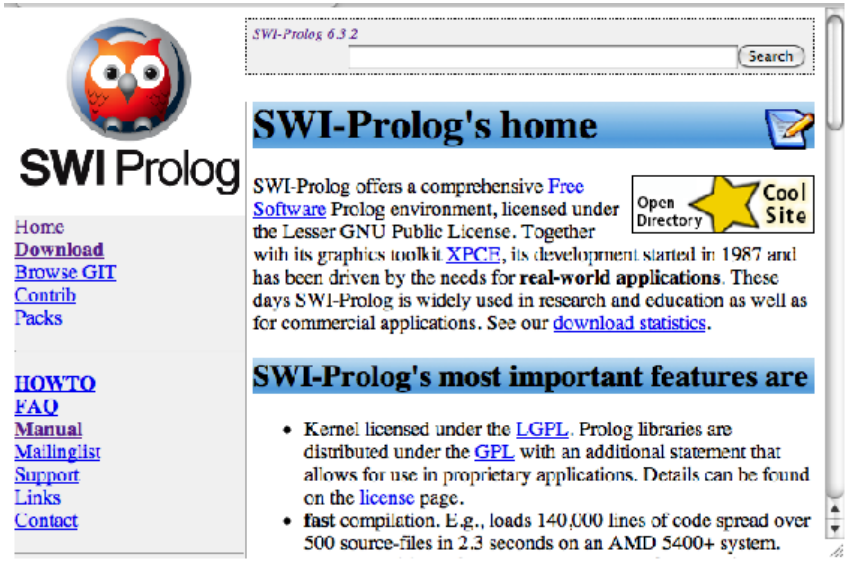
Σκοποί ενότητας

Εργαστήριο Μαθήματος Τεχνητής Νοημοσύνης Ι (Prolog)

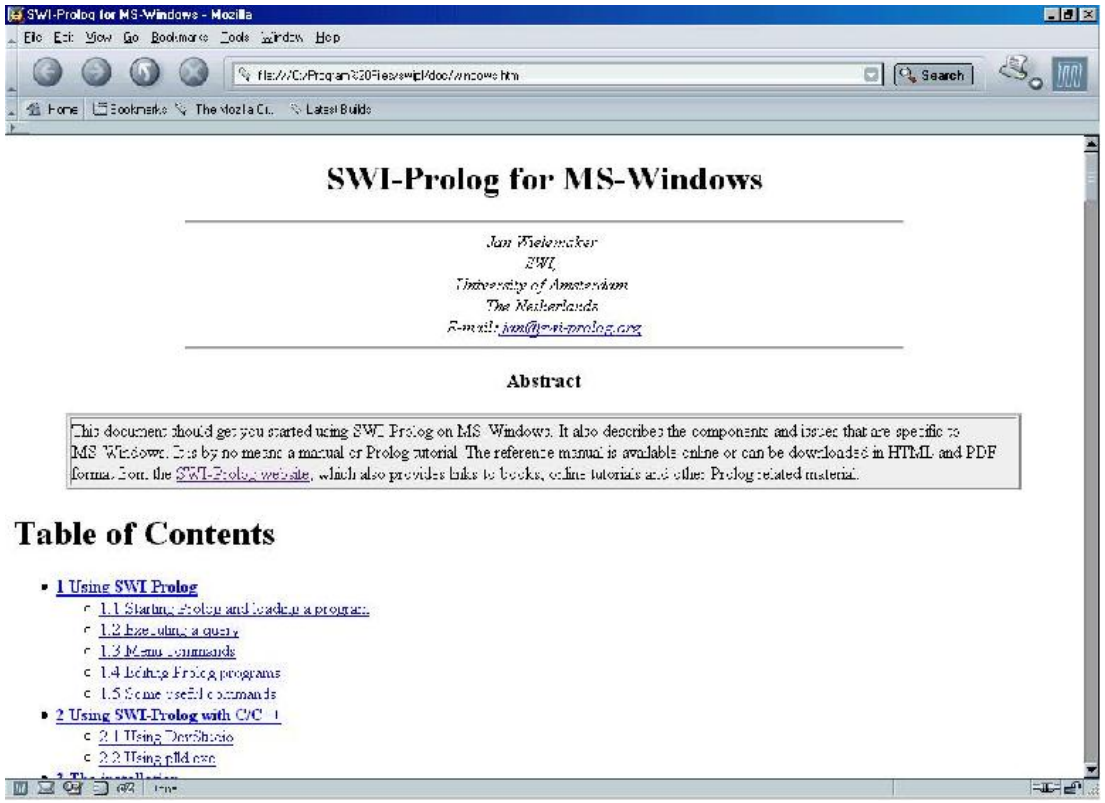
Περιεχόμενα ενότητας

Εργαστήριο Μαθήματος Τεχνητής Νοημοσύνης Ι (Prolog)

SWI-Prolog(Free Compiler)
SWI-Prolog Editor



Εκμάθηση Prolog



SWI-Prolog for MS-Windows - Mozilla

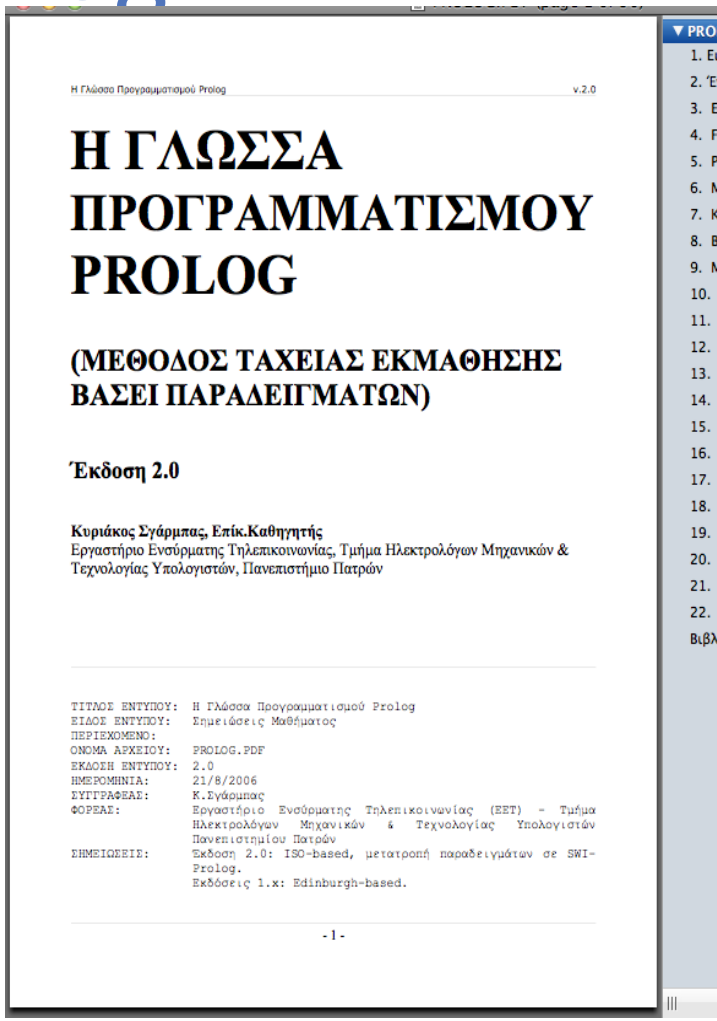
SWI-Prolog for MS-Windows

Jan Wielemaker
SWI
University of Amsterdam
The Netherlands
E-mail: jan@swi-prolog.org

Abstract

This document should get you started using SWI-Prolog on MS-Windows. It also describes the components and issues that are specific to MS-Windows. It is by no means a manual or Prolog tutorial. The reference manual is available online or can be downloaded in HTML and PDF format. See the SWI-Prolog website, which also provides links to books, online tutorials and other Prolog related material.

Table of Contents



Η Γλώσσα Προγραμματισμού Prolog v.2.0

Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PROLOG

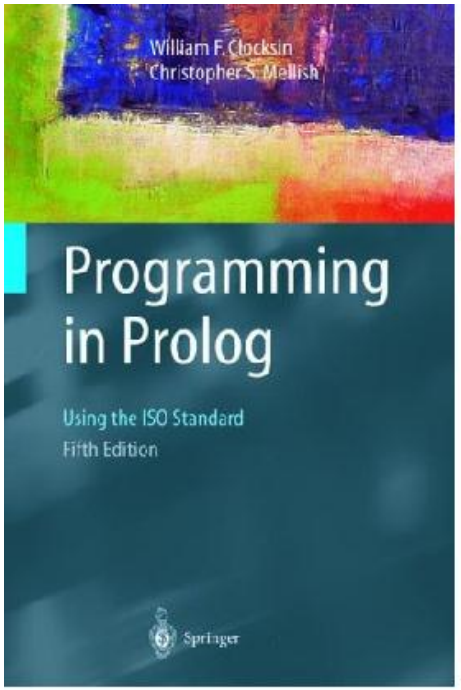
(ΜΕΘΟΔΟΣ ΤΑΧΕΙΑΣ ΕΚΜΑΘΗΣΗΣ ΒΑΣΕΙ ΠΑΡΑΔΕΙΓΜΑΤΩΝ)

Έκδοση 2.0

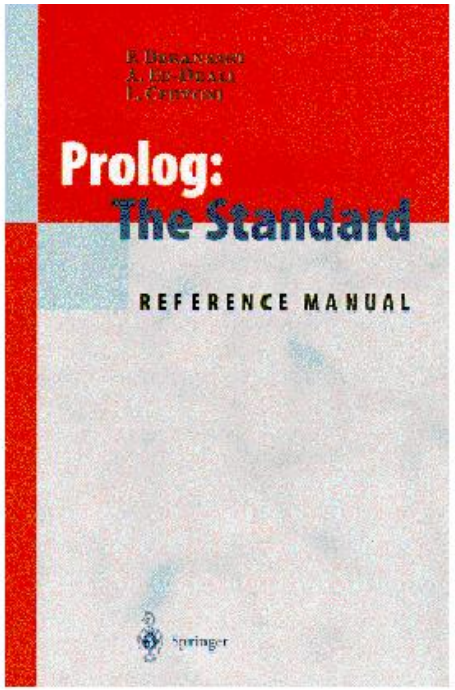
Κυριάκος Σγάρμπας, Επίκ.Καθηγητής
Εργαστήριο Ενσύρματης Τηλεπικοινωνίας, Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών

ΤΙΤΛΟΣ ΕΝΤΥΠΟΥ: Η Γλώσσα Προγραμματισμού Prolog
ΕΙΔΟΣ ΕΝΤΥΠΟΥ: Σημειώσεις Μαθήματος
ΠΕΡΙΕΧΟΜΕΝΟ:
ΟΝΟΜΑ ΑΡΧΕΙΟΥ: PROLOG.PDF
ΕΚΔΟΣΗ ΕΝΤΥΠΟΥ: 2.0
ΗΜΕΡΟΜΗΝΙΑ: 21/8/2006
ΣΥΓΓΡΑΦΕΑΣ: Κ.Σγάρμπας
ΦΟΡΕΑΣ: Εργαστήριο Ενσύρματης Τηλεπικοινωνίας (ΕΕΤ) - Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών Πανεπιστημίου Πατρών
ΣΗΜΕΙΩΣΕΙΣ: Έκδοση 2.0: ISO-based, μετατροπή παραδειγμάτων σε SWI-Prolog.
Εκδόσεις 1.x: Edinburgh-based.

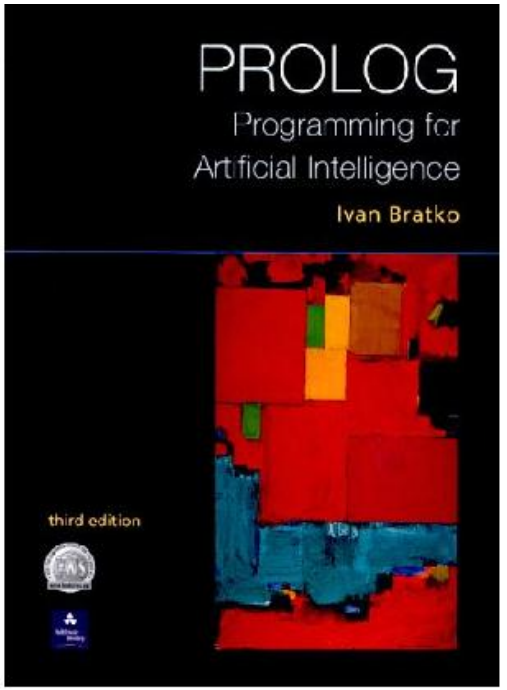
Προτεινόμενα βιβλία



"Programming in Prolog"
W.F. Clocksin and C.S. Mellish
5th edition, Springer Verlag, 2003



"Prolog: The Standard - Reference Manual"
P. Deransart, A. Ed-Dbali, L. Cervoni
Springer-Verlag, 1996



"Prolog Programming for Artificial Intelligence"
Ivan Bratko
3rd edition, Addison-Wesley, 2000

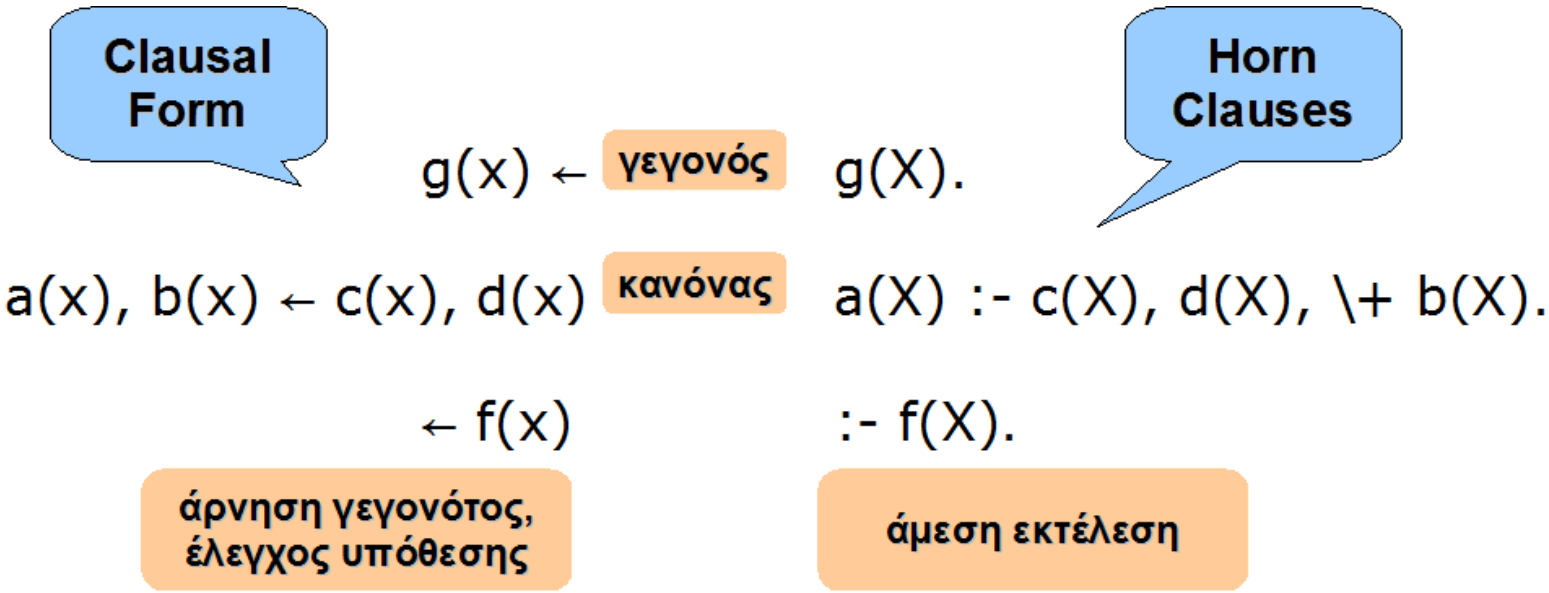
Free Prolog Books & Tutorials

- http://www.freeprogrammingresources.com/prologbook.html
- http://www.amzi.com/articles/prolog_books_tutorials.htm

Prolog

- Γλώσσα λογικού προγραμματισμού (Prolog = Programming in logic)
- Δεν προγραμματίζουμε τη διαδικασία επίλυσης. Αντίθετα, ορίζουμε το πρόβλημα και η γλώσσα προσπαθεί να το λύσει μόνη της!
- Διαθέτει:
- έναν μηχανισμό αναζήτησης εις βάθος (depth-first search)
- έναν μηχανισμό ταιριάσματος (matching)
- μια βάση γνώσης (knowledge base) στην οποία επεμβαίνουμε στατικά ή δυναμικά
- Μοιάζει σαν βάση δεδομένων, μόνο που εκτός από γεγονότα διαχειρίζεται και κανόνες (βάση γνώσης).

Αναλογία με κατηγορηματική λογική



Γεγονότα & Ερωτήσεις

```
man(peter).
man(jimmy).
woman(helen).
```

Πρόγραμμα
(Βάση Γνώσης)

```
?- man(peter).
   Yes
?- woman(helen).
   Yes
?- man(helen).
   No
?- woman(jimmy).
   No
?- woman(jenny).
   No
```

Yes = True
No = False

Ερωτήσεις με λογικούς τελεστές

```
man(peter).
man(jimmy).
woman(helen).
```

Στον κώδικα
δεν επιτρέπονται:

```
\+ man(helen).
man(peter); man(jimmy).
```

```
?- \+ man(peter).
   No
?- \+ woman(peter).
   Yes
?- \+ woman(jenny).
   Yes
?- \+ man(jenny).
   Yes
?- man(peter), man(jimmy).
   Yes
?- man(peter), \+ man(jimmy).
   No
?- man(peter); \+ man(jimmy).
   Yes
```

Ερωτήσεις με μεταβλητές

```
?- man(X).
   X = peter ;
   X = jimmy ;
   No

?- man(X).
   X = peter □
   Yes

?- man(X).
   X = peter ;
   X = jimmy □
   Yes

?- man(X).
   X = peter ;
   X = jimmy ;
   No
```

```
man(peter).
man(jimmy).
woman(helen).
```

```
?- man(X); woman(X).
   X = peter ;
   X = jimmy ;
   X = helen ;
   No

?- woman(X); man(X).
   X = helen ;
   X = peter ;
   X = jimmy ;
   No

?- man(X), \+ woman(X).
   X = peter ;
   X = jimmy ;
   No

?- \+ woman(X), man(X).
   No
```

Γεννήτριες και ελεγκτές

```
man(peter).
man(jimmy).
woman(helen).
```

```
?- \+ woman(X).
   No

?- man(X), \+ woman(X).
   X = peter ;
   X = jimmy ;
   No

?- \+ woman(X), man(X).
   No

?- \+ woman(X); man(X).
   X = peter ;
   X = jimmy ;
   No

?- \+ \+ woman(helen).
   Yes

?- woman(X).
   X = helen ;
   No
```

Γεγονότα με περισσότερα ορίσματα

```
mother_of(helen,peter).
mother_of(helen,jimmy).
mother_of(jenny,george).
```

```
?- mother_of(helen,peter).
   Yes
?- mother_of(helen,X).
   X = peter ;
   X = jimmy ;
   No
?- mother_of(X,peter).
   X = helen ;
   No
```

```
?- mother_of(X,Y).
   X = helen
   Y = peter ;
   X = helen
   Y = jimmy ;
   X = jenny
   Y = george ;
   No
?- mother_of(X,X).
   No
?- mother_of(X,_).
   X = helen ;
   X = helen ;
   X = jenny ;
   No
?- mother_of(_,_).
   Yes
```

Σύνθετες ερωτήσεις

```
?- mother_of(X,Y1), mother_of(X,Y2).
   X = helen
   Y1 = peter
   Y2 = peter ;

   X = helen
   Y1 = peter
   Y2 = jimmy ;

   X = helen
   Y1 = jimmy
   Y2 = peter ;

   X = helen
   Y1 = jimmy
   Y2 = jimmy ;

   X = jenny
   Y1 = george
   Y2 = george ;
   No
```

```
mother_of(helen,peter).
mother_of(helen,jimmy).
mother_of(jenny,george).
```

...με έλεγχο τιμών

```
?- mother_of(X,Y1), mother_of(X,Y2), \+ Y1==Y2.
   X = helen
   Y1 = peter
   Y2 = jimmy ;

   X = helen
   Y1 = jimmy
   Y2 = peter ;
   No
```

```
mother_of(helen,peter).
mother_of(helen,jimmy).
mother_of(jenny,george).
```

Ανώνυμη Μεταβλητή

```
?- mother_of(_,Y1), mother_of(_,Y2), \+ Y1==Y2.
   Y1 = peter
   Y2 = jimmy ;

   Y1 = peter
   Y2 = george ;

   Y1 = jimmy
   Y2 = peter ;

   Y1 = jimmy
   Y2 = george ;

   Y1 = george
   Y2 = peter ;

   Y1 = george
   Y2 = jimmy ;
   No
```

```
mother_of(helen,peter).
mother_of(helen,jimmy).
mother_of(jenny,george).
```

Objects (Αντικείμενα) στην Prolog

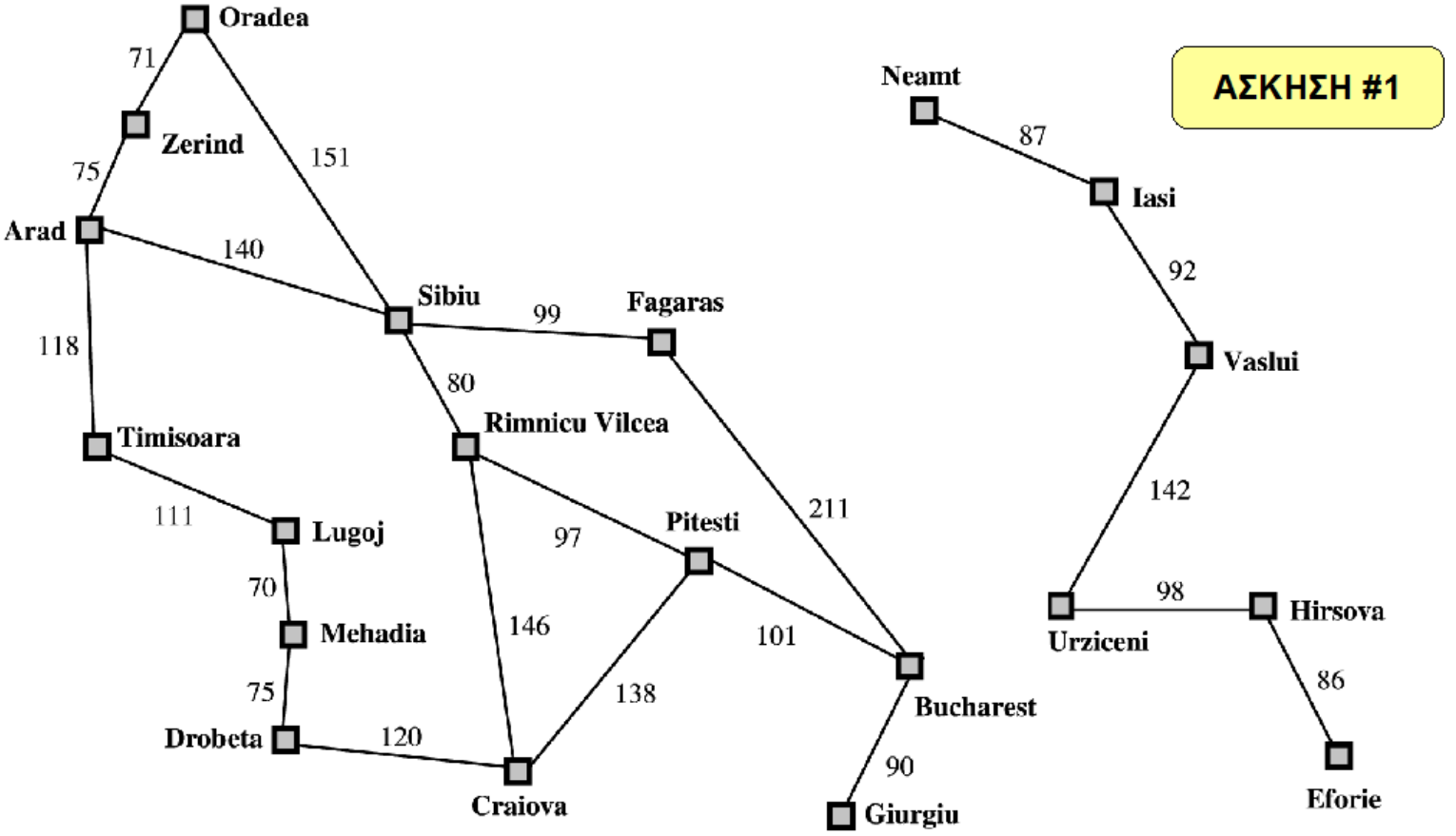
- Constants (Σταθερές):
  - Atoms: john jOHN 'JOHN' b019 tell_mE 'c d' (σε αντίθεση με: 18ab John _alpha tell-me)
  - Integers: 0, 1, 2, -41, 6221
- Variables (Μεταβλητές):

  Answer Tell_me WHAT _get _ _45E8
- Compound Objects (Σύνθετα Αντικείμενα)

  date(wednesday,16,8,2006)

  now(date(wednesday,16,8,2006),time(23,15,20))
- Lists (Λίστες): [1,2,f(3,h),go]

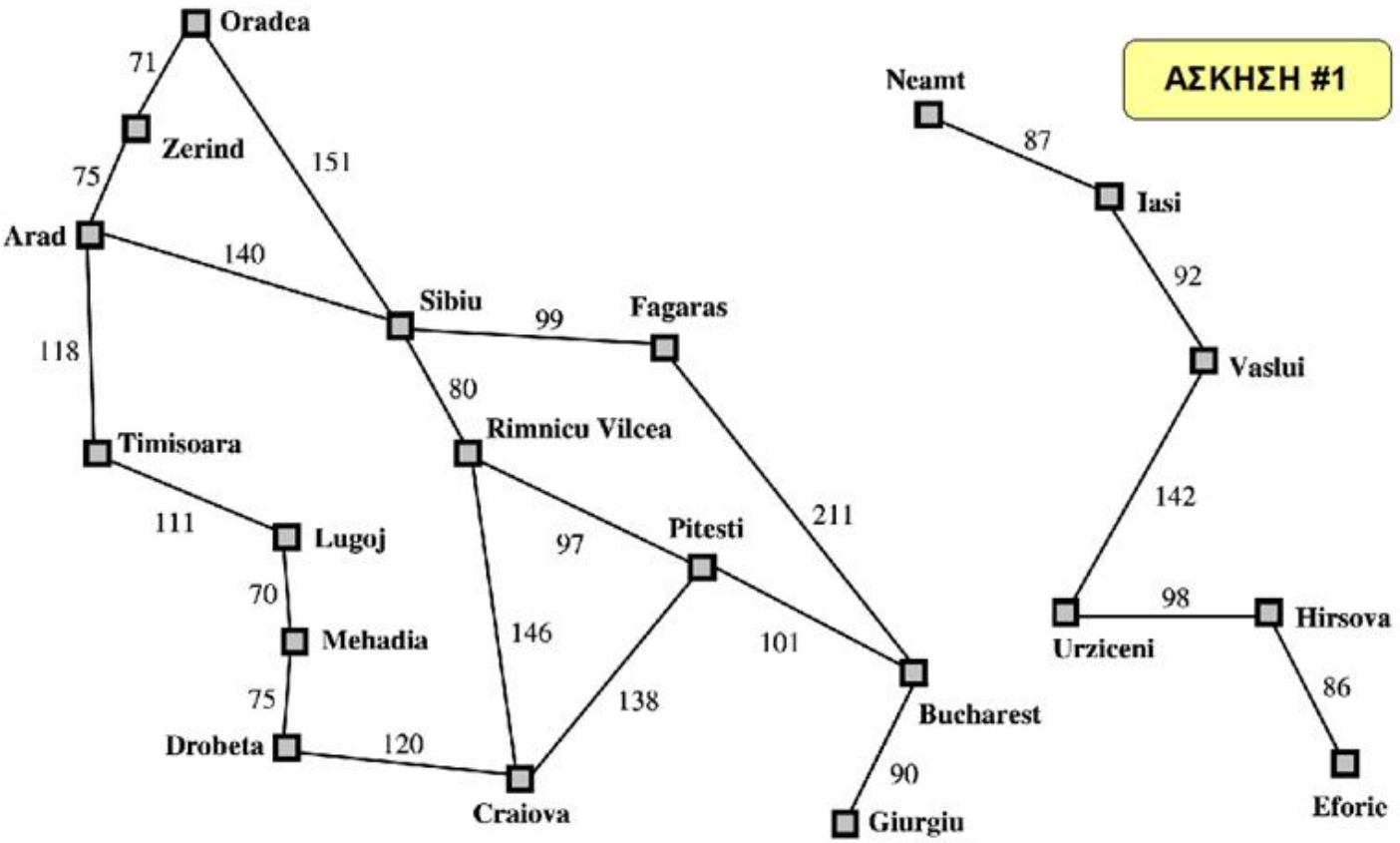
Άσκηση 1



1. Εκφράστε τις συνδέσεις μεταξύ των πόλεων ως γεγονότα στην Prolog. Πχ.:

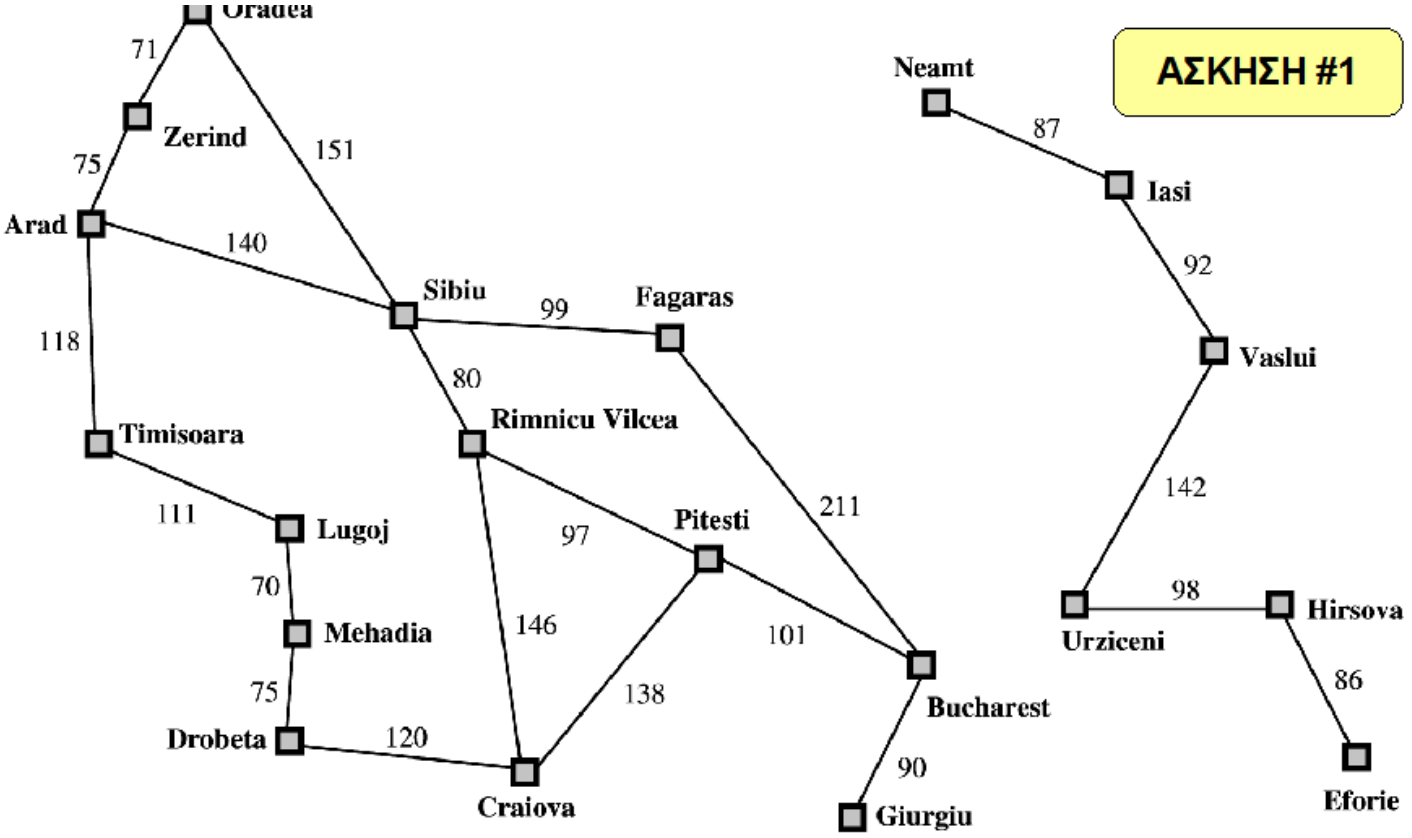
```
link(d,c,120).
link(s,f,99).
...
link(h,e,86).
```

Άσκηση 1



2. Κάντε κατάλληλη ερώτηση στην Prolog ώστε να βρείτε:
   (a) όλες τις πόλεις που συνδέονται απευθείας με το Sibiu.
   (b) όλες τις πόλεις που συνδέονται απευθείας με το Sibiu και απέχουν λιγώτερο από 100 χιλιόμετρα από αυτό.

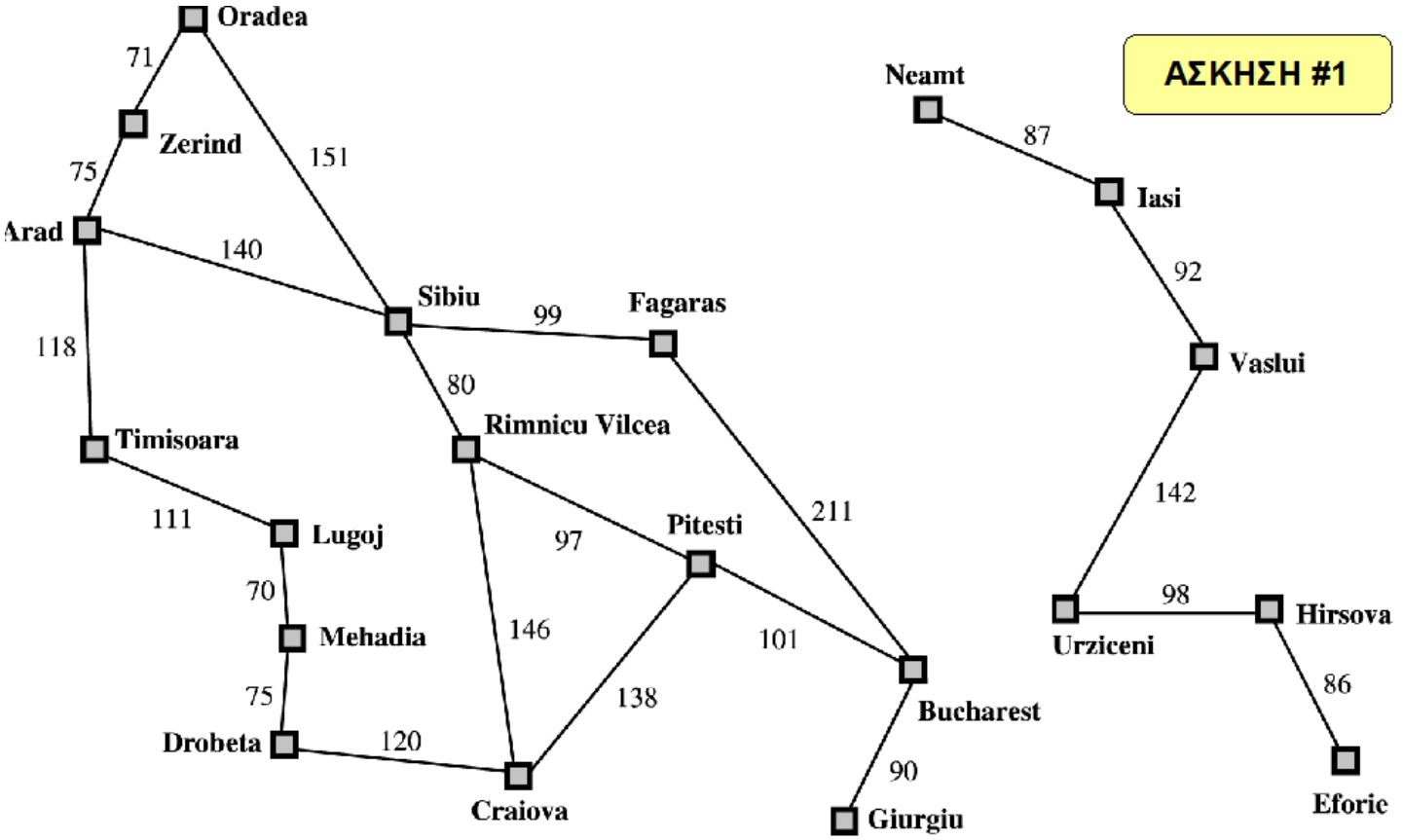
Άσκηση 1



(c) όλες τις πόλεις που συνδέονται απευθείας:

i. και με το Arad και με την Oradea (AND).
ii. με το Arad ή με την Oradea (OR).
iii. ή με το Arad ή με την Oradea (XOR).

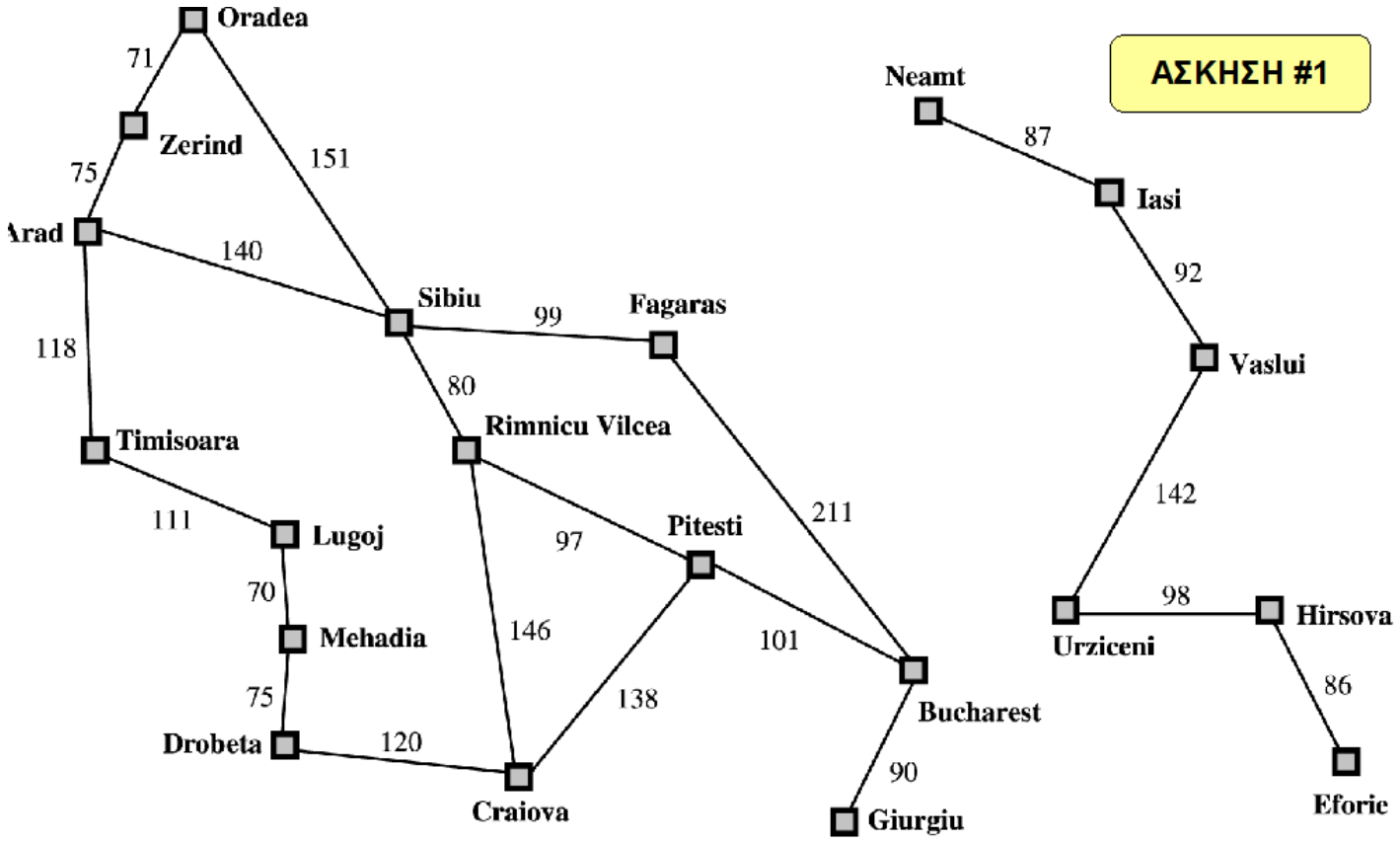
Άσκηση 1



(d) σε ποιές πόλεις μπορείτε να ταξιδέψετε από το Pitesti:

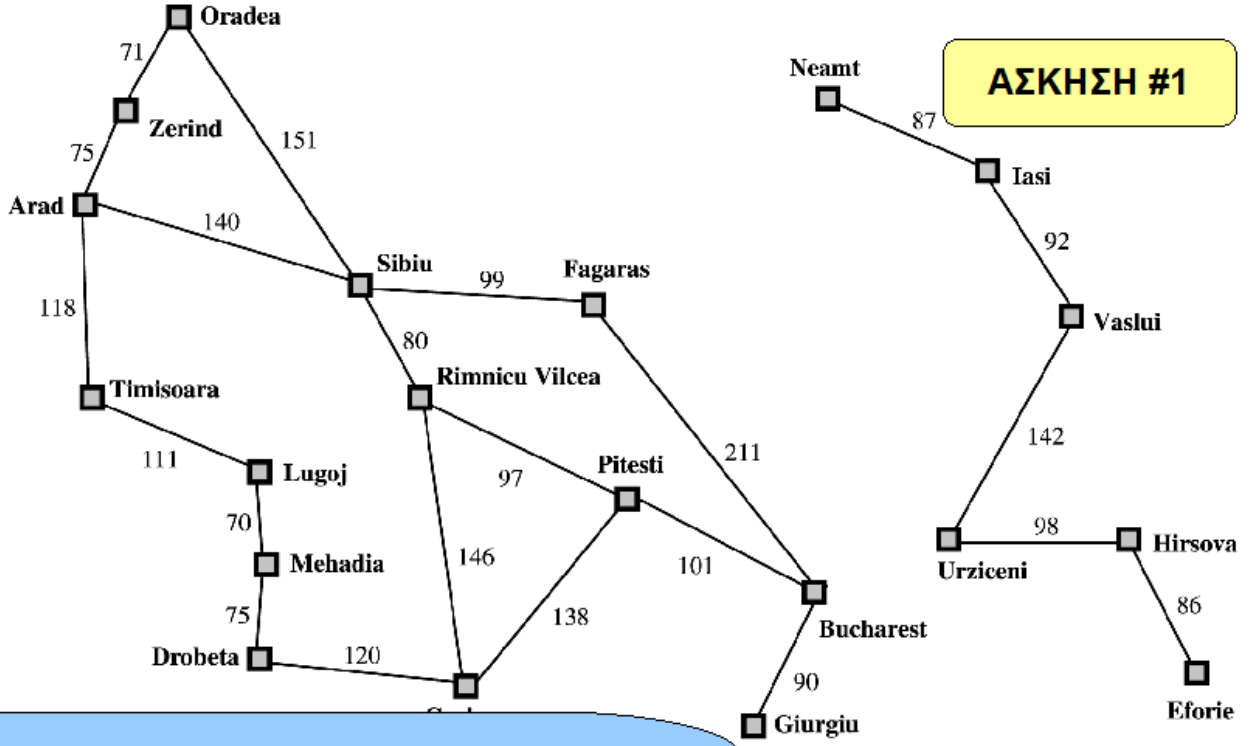
i. κάνοντας ακριβώς έναν ενδιάμεσο σταθμό,
ii. ..., χωρίς όμως να καταλήξετε στο Pitesti,
iii. ..., ούτε σε πόλεις οι οποίες συνδέονται απευθείας με αυτό.

Άσκηση 1



(e) όλες τις πόλεις που συνδέονται απευθείας με τουλάχιστον τρεις άλλες πόλεις.

Άσκηση 1



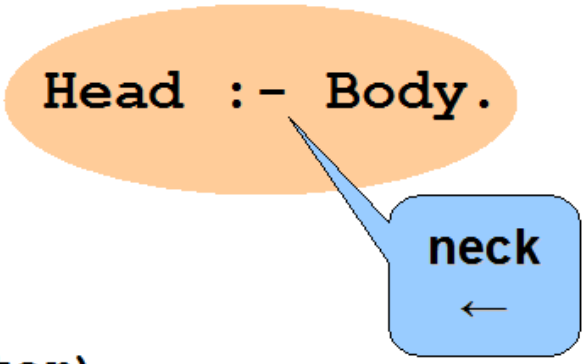
Παρατηρήστε πόσο περιπλέκονται οι απευθείας ερωτήσεις. Και ότι δεν μπορούμε να βάλουμε συνθήκες μεταξύ διαφορετικών επιστροφών. Τα πράγματα διευκολύνονται αν ορίσουμε βοηθητικές συναρτήσεις.

υν λιγώτερο

το Arad ή με την Oradea (OR).
το Arad ή με την Oradea (XOR).
λεις μπορείτε να ταξιδέψετε από το Pitesti:
τας ακριβώς έναν ενδιάμεσο σταθμό,
ίς όμως να καταλήξετε στο Pitesti,
iii. ..., ούτε σε πόλεις οι οποίες συνδέονται απευθείας με αυτό.
(e) όλες τις πόλεις που συνδέονται απευθείας με τουλάχιστον τρεις άλλες πόλεις.

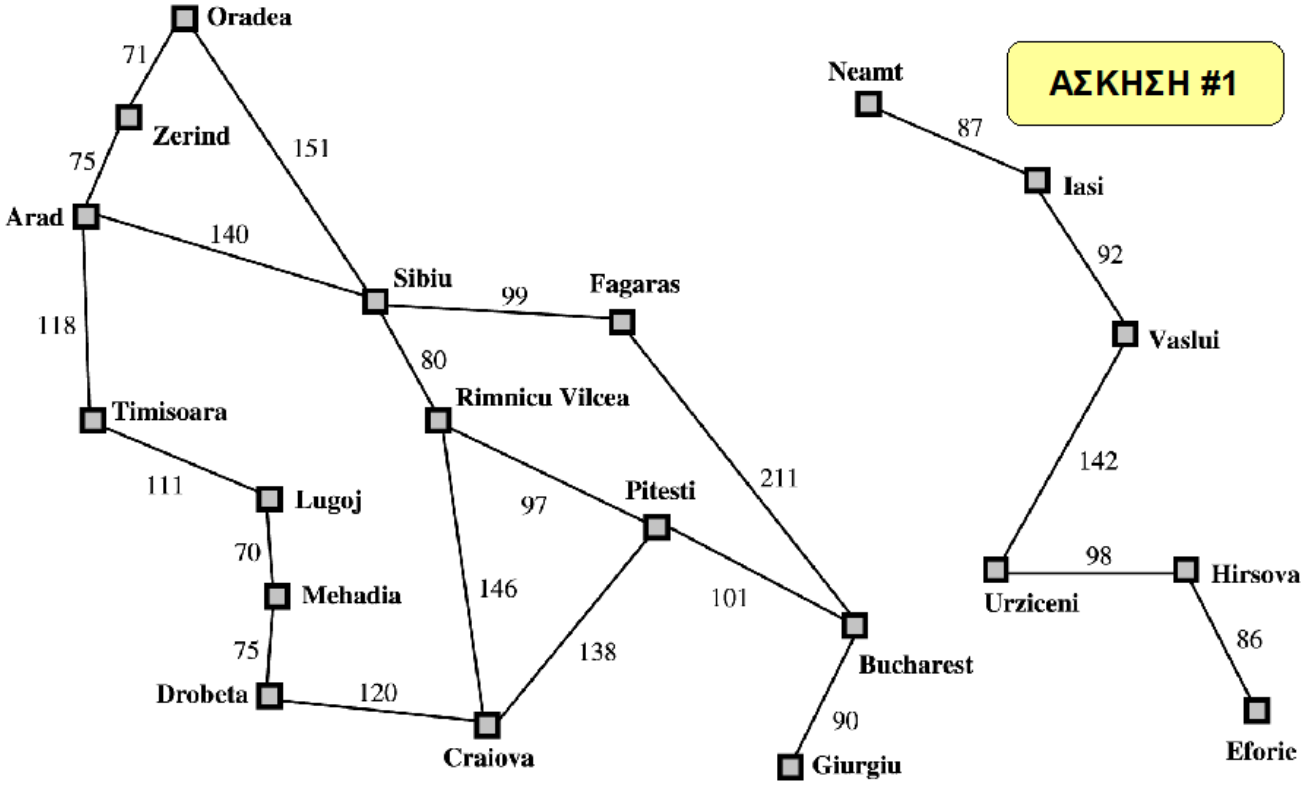
Κανόνες

```
fact.
fact :- true.
```




```
mother_of(helen,peter).
mother_of(helen,jimmy).
mother_of(jenny,george).
brothers(X,Y):- mother_of(Z,X), mother_of(Z,Y), \+ X==Y.
brothers2:- brothers(X,Y).
brothers3:- brothers(X,Y), write(X), nl, write(Y), nl.
brothers4:- brothers3.
```

Άσκηση 1

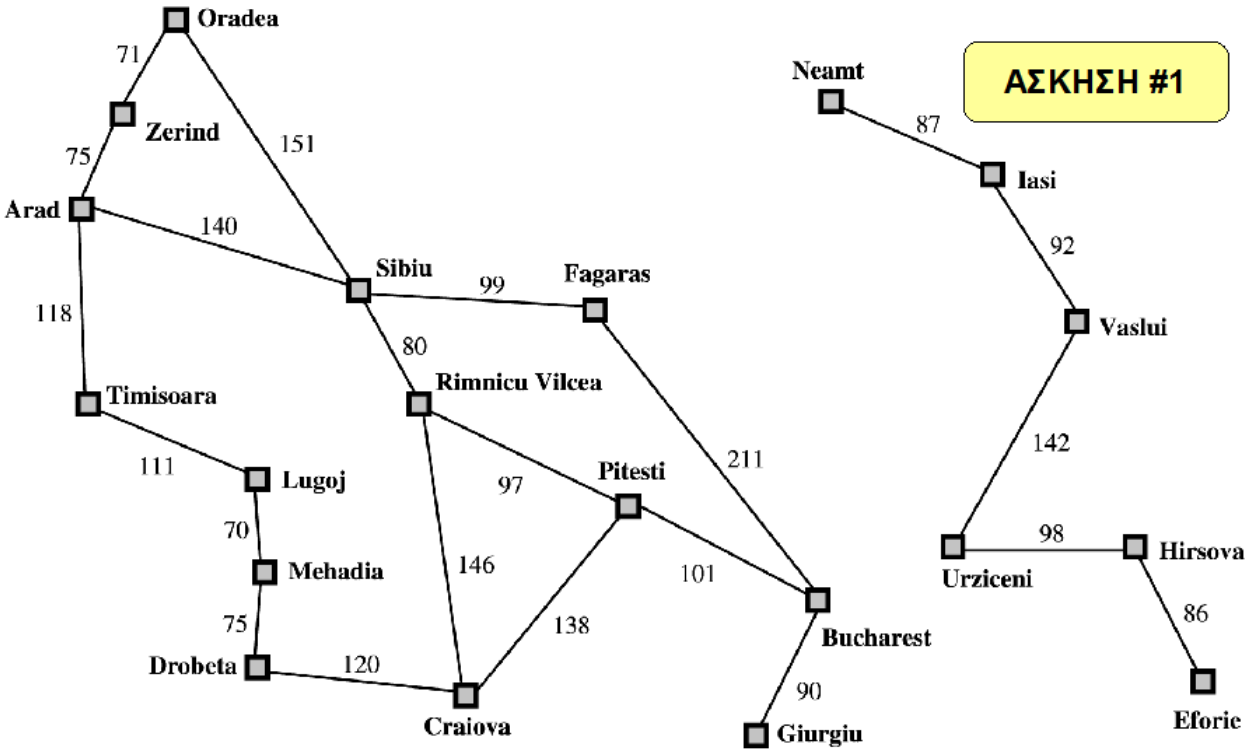


3. Φτιάξτε μια συνάρτηση Prolog με όνομα neighbours/2 η οποία θα δέχεται τα ονόματα δυο πόλεων και θα επαληθεύεται μόνο όταν υπάρχει απευθείας σύνδεση μεταξύ τους.

```
?- neighbours(mehadia,lugoj).
true
? - neighbours(arad,pitesti).
false
```

Σημείωση: Υπάρχουν 2 τρόποι να την ορίσετε.

Άσκηση 1



4. Φτιάξτε μια συνάρτηση Prolog με όνομα next/2 η οποία θα δέχεται το όνομα μιας πόλης και θα επιστρέφει όλες τις πόλεις με τις οποίες η πρώτη έχει απευθείας σύνδεση.

```
?- next(arad,X).
X = zerind ;
X = sibiu ;
X = timisoara ;
false
```

Σχολιάστε την συμμετρία της και την σχέση της με την neighbours/2.

Παραδείγματα

```
?- brothers(X,Y).
   X = peter
   Y = jimmy ;

   X = jimmy
   Y = peter ;
   No
```

```
mother_of(helen,peter).
mother_of(helen,jimmy).
mother_of(jenny,george).
brothers(X,Y):- mother_of(Z,X), mother_of(Z,Y), \+ X==Y.
brothers2:- brothers(X,Y).
brothers3:- brothers(X,Y), write(X), nl, write(Y), nl.
brothers4:- brothers3.
```

```
?- mother_of(Z,X), mother_of(Z,Y), \+ X==Y.
   Z = helen
   X = peter
   Y = jimmy ;

   Z = helen
   X = jimmy
   Y = peter ;
   No
```

```
?- brothers2.
   Yes
```

```
?- brothers2(X,Y).
   no
```

```
?- brothers3.
   peter
   jimmy
   Yes
```

```
?- brothers4.
   peter
   jimmy
   Yes
```

Αν η Βάση Γνώσης περιέχει γεγονότα του τύπου...

man(peter).

man(jimmy).

:

:

woman(helen).

woman(jenny).

:

:

parent_of(peter,jenny).

parent_of(helen,peter).

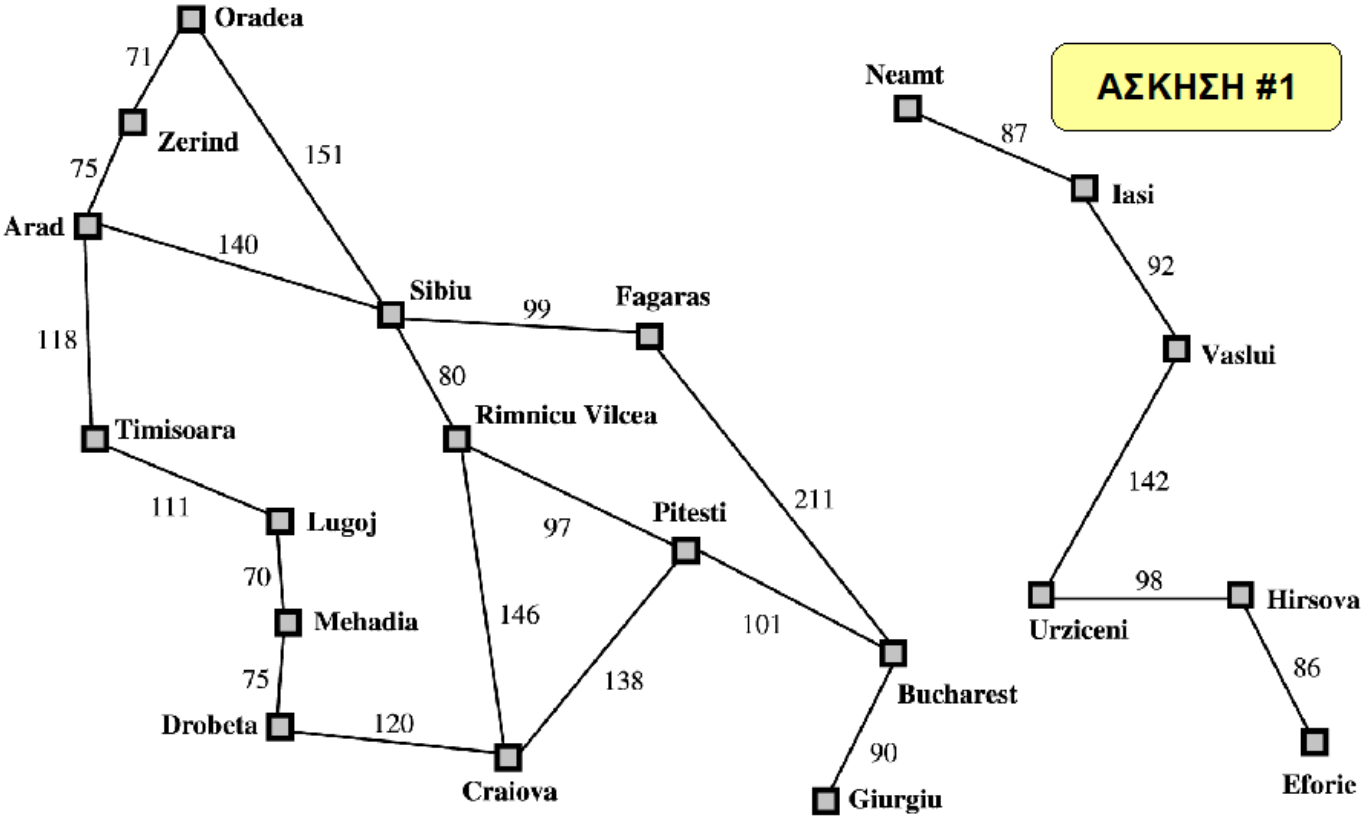
:

:

...ορίζουμε κανόνες (σχέσεις) συγγένειας

```
father_of(X,Y):- parent_of(X,Y), man(X).
mother_of(X,Y):- parent_of(X,Y), woman(X).

son_of(X,Y):- parent_of(Y,X), man(X).
daughter_of(X,Y):- parent_of(Y,X), woman(X).

grandfather_of(X,Y):- parent_of(X,Z),parent_of(Z,Y),man(X).
grandmother_of(X,Y):- mother_of(X,Z), parent_of(Z,Y).

brother_of(X,Y):- parent_of(Z,X), parent_of(Z,Y),
          \+ X==Y,man(X).
sister_of(X,Y):- daughter_of(X,Z), parent_of(Z,Y),
          \+ X==Y.

uncle_of(X,Y):- parent_of(Z,Y), brother_of(X,Z).
aunt_of(X,Y):- sister_of(X,Z),
                (father_of(Z,Y); mother_of(Z,Y)).
```

Άσκηση 1



4. Φτιάξτε μια συνάρτηση Prolog με όνομα next/2 η οποία θα δέχεται το όνομα μιας πόλης και θα επιστρέφει όλες τις πόλεις με τις οποίες η πρώτη έχει απευθείας σύνδεση.

```
?- next(arad,X).
X = zerind ;
X = sibiu ;
X = timisoara ;
false
```

Σχολιάστε την συμμετρία της και την σχέση της με την neighbours/2.

Backtracking (Οπισθοδρόμηση)

```
man(peter).   man(jimmy).   man(george).
woman(helen). woman(jenny). woman(mary).
goal0:- man(X).
goal1:- man(X), write(X), nl.
goal2:- man(X), write(X), nl, fail.
goal3:- man(X), write(X), nl, woman(X).
goal4:- man(X), woman(X), write(X), nl.
```

```
?- man(X).
   X = peter ;
   X = jimmy ;
   X = george
   No

?- goal0.
   Yes
```

```
?- goal1.
   peter
   Yes

?- goal2.
   peter
   jimmy
   george
   No
```

```
?- goal3.
   peter
   jimmy
   george
   No

?- goal4.
   No
```

Matching (Ταίριασμα)

```
now(date(wednesday,16,august,2006),time(23,15,20)).
```

```
?- now(X,time(Y,15,20)).
   X                                =
date(wednesday,16,august,2006)
   Y = 23 ;
   No
?- now(date(X,16,_,2006),_).
   X = wednesday ;
   No
?- now(date(X,16,_,2006),_,_).
   No
?- now(date(X,9,_,_),_).
   No
?- now(gate(X,16,_,_),_).
   No
```

Παραδείγματα

```
sum(5).
```

```
?- sum(5).
   Yes

?- sum(2+3).
   No
```

```
sum(2+3).
```

```
?- sum(5).
   No

?- sum(X).
   X = 2+3 ;
   No

?- sum(2+X).
   X = 3 ;
   No

?- sum(3+X).
   No
```

Είδη Ισότητας

`=`	Κάνει σύγκριση και καταχώρηση με matching. Χειρίζεται τα μέλη ως μη αριθμητικά, γιαυτό και δεν κάνει πράξεις.
`is`	Μόνο για αριθμητικές παραστάσεις. Υπολογίζει το δεξί μέλος και κάνει καταχώρηση ή σύγκριση.
`=:=`	Μόνο για αριθμητικές παραστάσεις. Υπολογίζει και τα δυο μέλη. Κάνει σύκγριση αλλά όχι καταχώρηση.
`==`	Χειρίζεται τα μέλη ως μη αριθμητικά (δεν εκτελεί υπολογισμούς). Δεν κάνει καταχώρηση, μόνο έλεγχο.

Παραδείγματα

```
?- 5=5.
   Yes

?- hello = hello.
   Yes

?- X = hello.
   X = hello ;
   No

?- X=5.
   X = 5 ;
   No

?- 5=X.
   X = 5 ;
   No

?- X=4+1.
   X = 4+1
   No
```

```
?- X is 4+1.
   X = 5 ;
   No

?- 4+1 is X.
   No

?- 5 is 5.
   Yes

?- hello is hello.
   No

?- X is 5, X is 6.
   No

?- X=5, X=6.
   No
```

Παραδείγματα

```
?- 4+5 =:= 3*3.
   Yes

?- X =:= 4+5.
ERROR: =:=/2:
Arguments are
not sufficiently
instantiated

?- hello =:= hello.
   No

?- 5 == 5.
   Yes

?- hello == hello.
   Yes

?- 5 == 4+1.
   No

?- X == 5.
   No
```

```
?- X=5, X == 5.
   X = 5 ;
   No

?- X == X.
   X = _G224 ;
   No

?- X == Y.
   No

?- X = Y.
   X = _G206
   Y = _G206 ;
```

Recursion (Αναδρομικότητα)

```
parent(alex,bill).
parent(bill,charlie).
parent(charlie,don).
    :
    :
    :    % (n γεγονότα)
    :
    :

parent(simon,timothy).

ancestor(X,Y):- parent(X,Y);
         ( parent(X,Z), parent(Z,Y) );
         ( parent(X,Z), parent(Z,W),
parent(W,Y) );
            :
            :
            :  % (n όροι)
```

Recursion (Αναδρομικότητα)

```
ancestor(X,Y):- parent(X,Y).
ancestor(X,Y):- parent(X,Z), parent(Z,Y).
ancestor(X,Y):- parent(X,Z), parent(Z,W), parent(W,Y).
          :
          :
          :  % (n κανόνες)
```

τερματική σχέση

```
ancestor(X,Y):- parent(X,Y).
ancestor(X,Y):- parent(X,Z), ancestor(Z,Y).
```

αναδρομική σχέση

Παραδείγματα

```
%  Υπολογισμός Παραγοντικού
   paragontiko(1,1).
   paragontiko(X,Y):-  X2 is X-1,

paragontiko(X2,Y2),
                    Y is Y2*X.

?- paragontiko(4,X).
   X = 24 □
   Yes
```

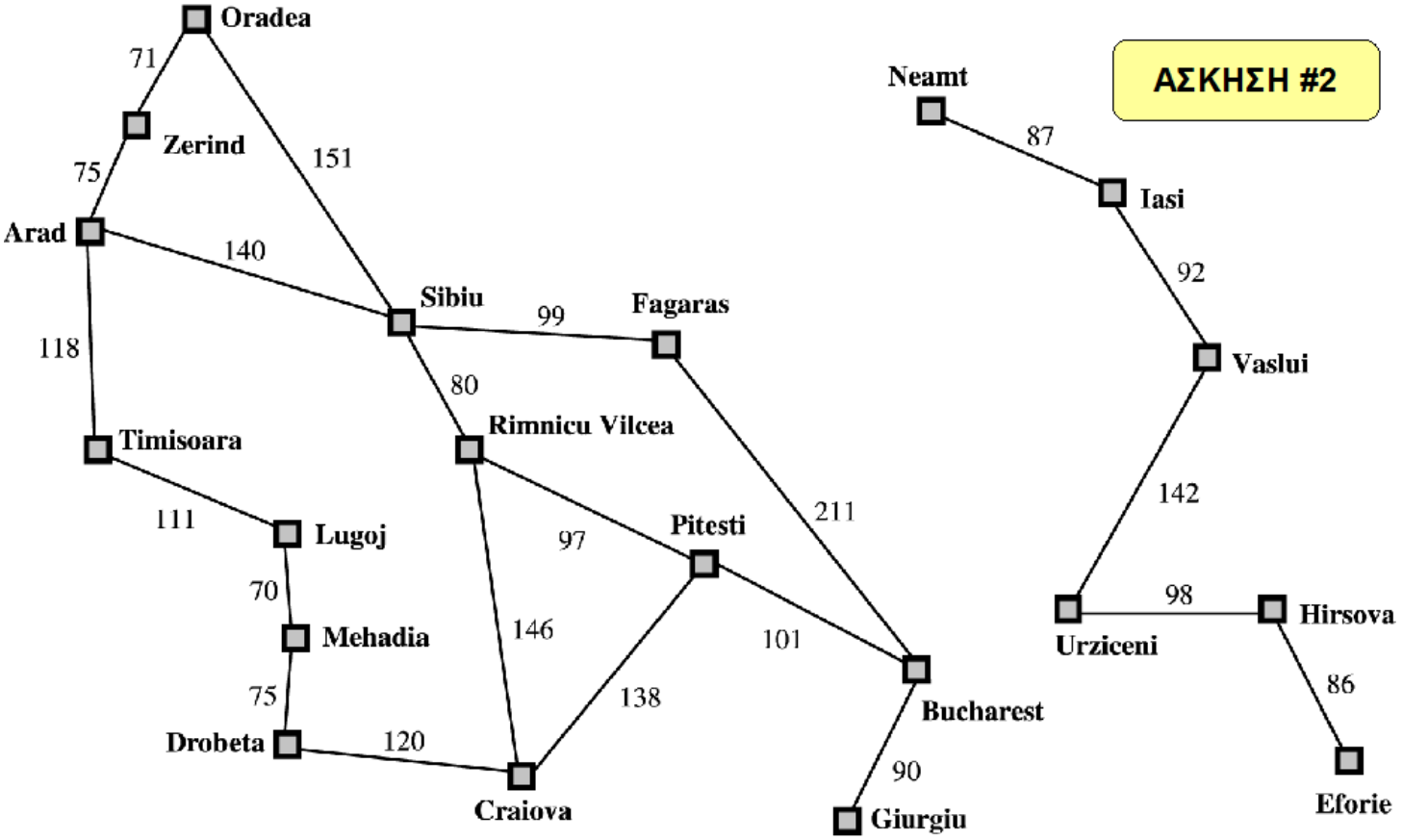
```
%  Ένα ατέρμονο loop στην Prolog
   loop:- loop.
```

Μετρητές

```
%  Ενας μερτητής
   count(X):- write(X), nl,
              NewX is X+1,
             count(NewX).

%  Ενας "κακός" μερτητής
   badcount1(X):- write(X), nl,
          NewX is X+1,
          badcount1(NewX),
          nl.

%  Αλλος ένας "κακός" μερτητής
   badcount2(X):- write(X), nl,
          NewX is X+1,
          badcount2(NewX).
   badcount2(X):- X<0,
          write('X is
negative'),
          nl.
```

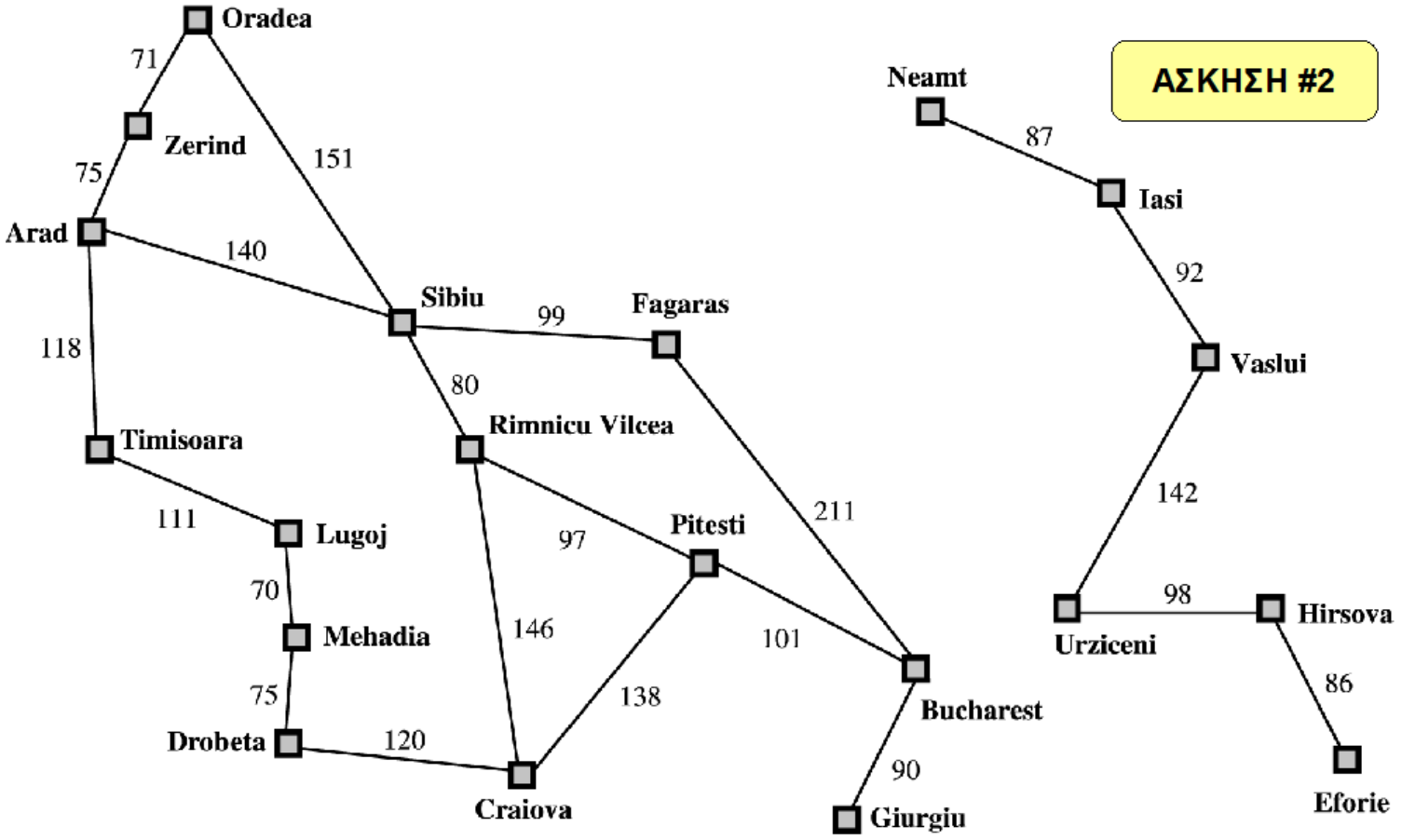
```
%  Ενας εξίσου "κακός" μερτητής
   badcount3(X):- write(X), nl,
          NewX is X+1,
          check(NewX),
          badcount3(NewX).
   check(Z):- Z>0.
   check(Z):- Z<0,
         write('X is negative'),
         nl.
```

Άσκηση 2



5. Ορίστε μια συνάρτηση Prolog **reach(?City1, ?City2, +Link_Distance)** που θα δέχεται δυο πόλεις και ένα όριο απόστασης (μετρημένο σε πλήθος links) και θα ικανοποιείται αν μπορούμε να ταξιδέψουμε από τη μία πόλη στην άλλη με ακριβώς Link_Distance μεταβάσεις (ή Link_Distance-2 σταθμούς).

Άσκηση 2



6. Ορίστε μια συνάρτηση Prolog **reach2(?City1, ?City2, ?Link_Distance)** που θα λειτουργεί όπως η reach/3, με τη διαφορά ότι θα μπορεί να δίνει και ως έξοδο το Link_Distance.
Υπόδειξη: Θεωρήστε ότι το Link_Distance δεν θα ξεπερνά το 9.

Cut (!)

```
man(peter).    man(jimmy).    man(george).
a:- man(X), write(X), nl, fail.
b:- man(X), !, write(X), nl, fail.
c:- man(X), write(X), nl, !, fail.
d:- !, man(X), write(X), nl,  fail.
e:- man(X), write(X), nl, fail, !.
```

```
?- a.
   peter
   jimmy
   george
   No

?- b.
   peter
   No
```

```
?- c.
   peter
   No

?- d.
   peter
   jimmy
   george
   No
```

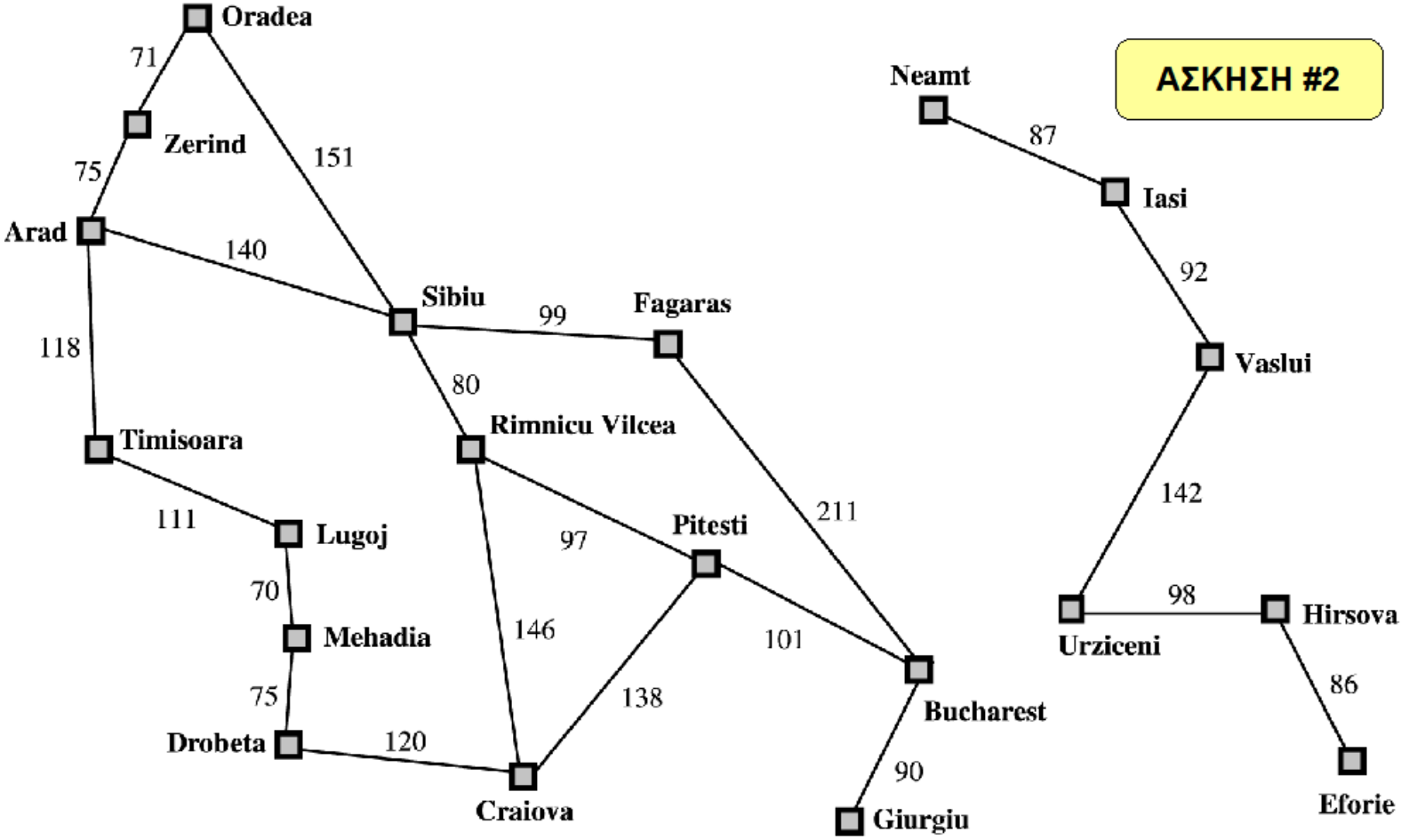
```
?- e.
   peter
   jimmy
   george
   No
```

Το Cut στο body ενός κανόνα...

- απαγορεύει το backtracking προς τα αριστερά του
- ακυρώνει τους επόμενους εναλλακτικούς κανόνες με το ίδιο head

```
logicnot(X):- X, !, fail.
logicnot(_).

?- logicnot(1==2).
Yes

?- logicnot1(2==2).
No
```

Άσκηση 2



7. Ορίστε μια συνάρτηση Prolog **min_link_dist(?City1, ?City2, ?Link_Distance)** που θα λειτουργεί όπως η reach2/3, με τη διαφορά ότι θα επιστρέφει το ελάχιστο Link_Distance μεταξύ δύο πόλεων.
   Υπόδειξη: Μετρήστε πρώτα την διάμετρο του χώρου καταστάσεων.

Άσκηση 2

ΑΣΚΗΣΗ #2

8. Έστω η συνάρτηση **find_link_dist/3** που ορίζεται ως εξής:

```
find_link_dist(X,X,0):- !.
find_link_dist(X,Y,1):- next(X,Y), !.
find_link_dist(X,Y,CD):- next(X,Z), find_link_dist(Z,Y,CDn), CD is CDn+1.
```

Ελέγξτε αν μπορεί να αντικαταστήσει επαρκώς την min_link_dist/3. Τί προβλήματα μπορεί να προκαλέσει; Κάτω από ποιές συνθήκες;

Άσκηση 2

ΑΣΚΗΣΗ #2

9. Έστω η συνάρτηση **find_link_dist2/3** που ορίζεται ως εξής:

```
find_link_dist2(X,X,0,_):- !.
find_link_dist2(X,Y,1,_):- next(X,Y), !.
find_link_dist2(X,Y,CD,Pass_prev_X):- next(X,Z), \+(Z==Pass_prev_X),
                                      find_link_dist2(Z,Y,CDn,X),
                                      CD is CDn+1.
```

Ελέγξτε πώς λειτουργεί και εξηγήστε τί κάνει. Μπορεί να προκαλέσει προβλήματα; Κάτω από ποιές συνθήκες;

Λίστες

```
[1,2,3]

[abc,'JOHN',4711,date(10,june,2006)]

[4,[yet,f(2,a(1,4))],[4,3,f,2,[0]],end]

[bill]  ≠  bill

[]
```

Chop

```
chop(LIST,HEAD,TAIL):- LIST = [HEAD|TAIL].

?- chop([1,2,3],H,T).
   H = 1
   T = [2,3] ;
   No

?- chop([2,3],H,T).
   H = 2
   T = [3] ;
   No

?- chop([3],H,T).
   H = 3
   T = [] ;
   No

?- chop([],H,T).
   No
```

Συμμετρικότητα

```
?- chop([[1,2,3],4,5],H,T).
   H = [1,2,3]
   T = [4,5] ;
   No

?- chop(L,5,[10,20]).
   L = [5,10,20] ;
   No
```

ισοδύναμοι ορισμοί

```
chop(LIST,HEAD,TAIL):- LIST = [HEAD|TAIL].

chop([H|T],H,T).
```

Συνάρτηση member/2

```
member(X,[X|_]).
member(X,[_|T]):- member(X,T).

?- member(2,[1,2,3,4]).
   Yes

?- member(5,[1,2,3,4]).
   No

?- member(X,[1,2,3,4]).
   X = 1 ;
   X = 2 ;
   X = 3 ;
   X = 4 ;
   No
```

Συνάρτηση append/3

```
append([],L,L).
append([H|T],L1,[H|L2]):-
append(T,L1,L2).

?- append([a,b,c],[d,e],X).
   X = [a,b,c,d,e] ;
   No

?- append(X,[d,e],[a,b,c,d,e]).
   X = [a,b,c] ;
   No

?- append([a,b,c],X,[a,b,c,d,e]).
   X = [d,e] ;
   No
```

Παράδειγμα χρήσης των συμμετριών της append/3

```
take(L,X,L2):- append(B,[X|A],L),
               append(B,A,L2).

?- take([a,b,c,d,e],c,L).
   L = [a,b,d,e] ;
   No
```

Άσκηση 3

ΑΣΚΗΣΗ #3

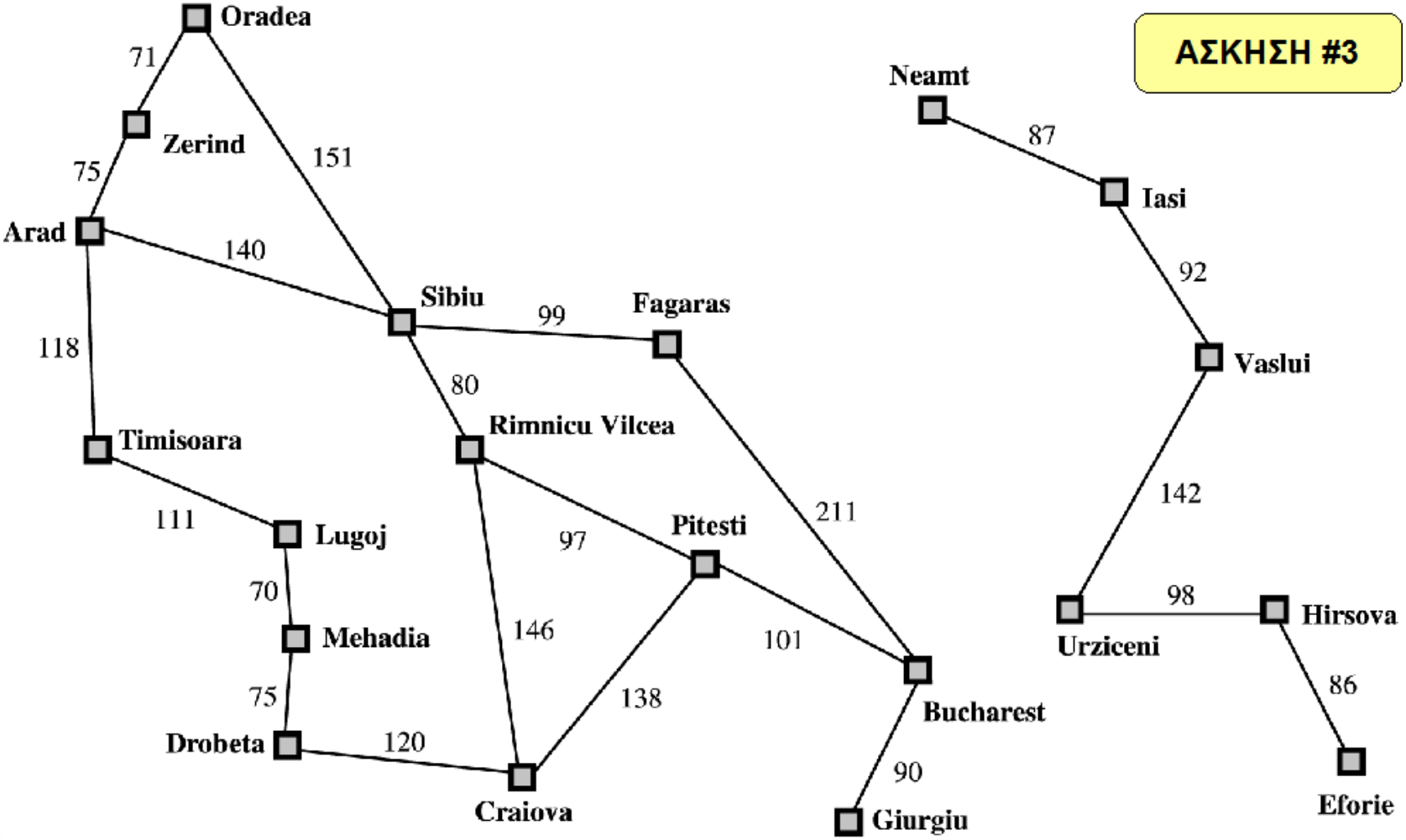
10. Προσθήκη μνήμης:

Ερώτημα 10.a

Ορίστε μια συνάρτηση **smart_find_link_dist(+City1, +City2, ? Link_Distance, +Memory)** που θα λειτουργεί όπως η reach2/3, αλλά θα αποφεύγει τους κύκλους με τη χρήση της λίστας Memory που θα θυμάται τις πόλεις που έχει συναντήσει.

Υπόδειξη: Κατά την κλήση η Memory ξεκινά με αρχική τιμή [].
(Τί χρησιμότητα θα είχε μια διαφορετική αρχική τιμή για τη λίστα Memory;)

Άσκηση 3


ΑΣΚΗΣΗ #3
Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
Urziceni
86
Eforie

findall/3, sort/2, msort/2

```
woman(helen). woman(jenny). woman(mary).
?- findall(X,woman(X),L).
X = _G320
L = [helen, jenny, mary]


?- X1=[4,1,2,5,4], sort(X1,X2).
X1 = [4, 1, 2, 5, 4]
X2 = [1, 2, 4, 5]


?- X1=[4,1,2,5,4], msort(X1,X2).
X1 = [4, 1, 2, 5, 4]
X2 = [1, 2, 4, 4, 5]
```

Άσκηση 3

ΑΣΚΗΣΗ #3

Ερώτημα 10.b

Ορίστε μια συνάρτηση **min_link_dist2/3** που θα λειτουργεί όπως η min_link_dist/3, χωρίς όμως να λαμβάνει υπ'όψιν τη διάμετρο του χώρου καταστάσεων.

Υπόδειξη: Χρησιμοποιήστε την findall/3, την smart_find_link_dist/4 και την sort/2.

Άσκηση 3

ΑΣΚΗΣΗ #3

11. Προσθήκη διαδρομής:

(a)
Ορίστε μια συνάρτηση **smart_find_link_path/5** που θα λειτουργεί όπως η smart_find_link_dist/4, αλλά θα επιστρέφει και μια λίστα με το μονοπάτι της διαδρομής.

(b)
Ορίστε μια συνάρτηση **min_link_path/4** που θα λειτουργεί όπως η min_link_dist/3, αλλά θα επιστρέφει και μια λίστα με το μονοπάτι της διαδρομής. Αν δύο ή περισσότερες διαδρομές έχουν το ίδιο μήκος, να τις επιστρέφει όλες (μη ντετερμινιστικά).

Άσκηση 3

12. Υπολογισμός πραγματικής απόστασης:

ΑΣΚΗΣΗ #3

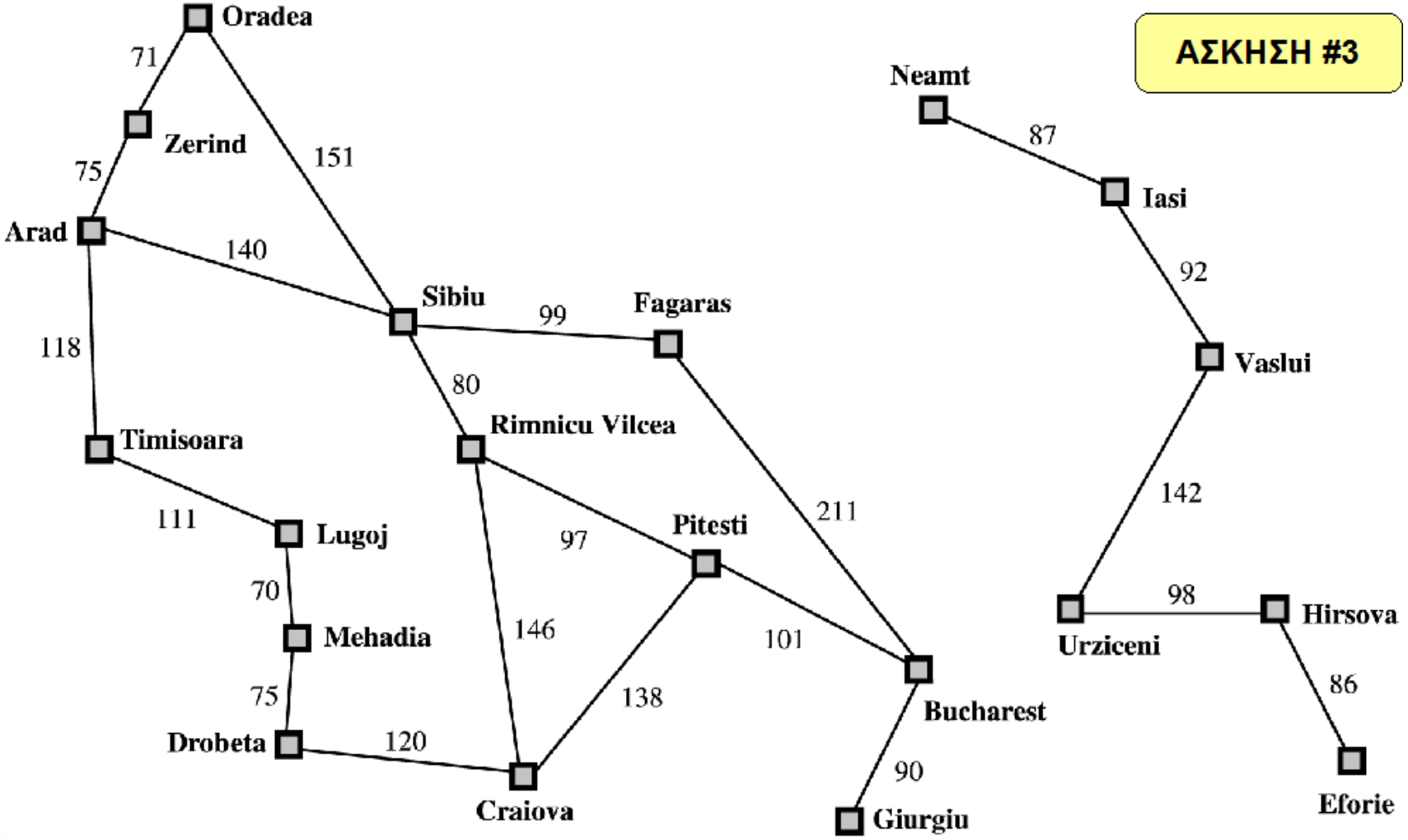
(a)
Ορίστε μια συνάρτηση **smart_find_distance/4** που θα λειτουργεί όπως η smart_find_link_dist/4 αλλά η απόσταση που θα μετράει δεν θα είναι σε πλήθος links αλλά σε χιλιόμετρα.

(b)
Ορίστε μια συνάρτηση **min_distance/3** που θα λειτουργεί όπως η min_link_dist2/3 αλλά η απόσταση που θα μετράει δεν θα είναι σε πλήθος links αλλά σε χιλιόμετρα.

(c)
Ορίστε μια συνάρτηση **smart_find_dist_path/5** που θα λειτουργεί όπως η smart_find_link_path/5, αλλά θα μετράει την απόσταση σε χιλιόμετρα.

(d)
Ορίστε μια συνάρτηση **min_dist_path/4** που θα λειτουργεί όπως η min_link_path/4, αλλά θα μετράει την απόσταση σε χιλιόμετρα.

Άσκηση 3


ΑΣΚΗΣΗ #3
Oradea
71
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
Urziceni
86
Eforie

Auto-executable Goals

```
:- write('NOW LOADING PROGRAM'), nl.
      :
      :
      : (Πρόγραμμα: γεγονότα, κανόνες, κλπ.)
      :
      :
:- write('Ready to answer your questions'), nl.
```

Εκτελούνται την ώρα του consult, χωρίς να περιμένουν από τον χρήστη να γράψει κάτι στο prompt.

Μια τέτοια εντολή στο τέλος του αρχείου το κάνει εκτελέσιμο.

assert/retract

```
:- dynamic paragontiko/2.
paragontiko(1,1):- !.
paragontiko(X,Y):- X2 is X-1,
                   paragontiko(X2,Y2),
                   Y is Y2*X,

asserta(paragontiko(X,Y):- !).
```

```
?- paragontiko(4,X).
   X = 24 ;
   No
```

```
?- listing.
   paragontiko(4,24):- !.
   paragontiko(3,6):- !.
   paragontiko(2,2):- !.
   paragontiko(1,1):- !.
   paragontiko(A,B):-
       C is A-1,
       paragontiko(C,D),
       B is D*A,
       asserta((paragontiko(A,B):-!)).
   Yes
```

Παράδειγμα

```
:- dynamic link/3.
link(a,z,75).
link(a,s,140).
link(a,t,118).

:- link(X,Y,Z), \+link(Y,X,Z),
   assert(link(Y,X,Z)), fail.


?- listing.
link(a, z, 75).
link(a, s, 140).
link(a, t, 118).
link(z, a, 75).
link(s, a, 140).
link(t, a, 118).
```

Τελεστές

```
father(alex,bill).
father(bill,charlie).
father(charlie,don).

alex is_father_of bill.
bill          is_father_of
charlie.
charlie is_father_of don.
          :- op(700,xfx,is_father_of).

          X is_father_of Y :- father(X,Y).


          'Ολοι οι τελεστές που είναι ορισμένοι:
          ?- current_op(X,Y,Z).
```

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Έχουν προηγηθεί οι κάτωθι εκδόσεις:

- Έκδοση 1.0 διαθέσιμη εδώ.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

Διαφάνεια 4, 5: Εικόνες από τον editor Swi-Prolog