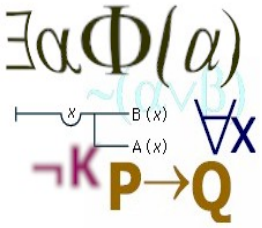
Πανεπιστήμιο Μακεδονίας
Τμ. Εφαρμοσμένης Πληροφορικής

Αναζήτηση σε Γράφους



Δομή

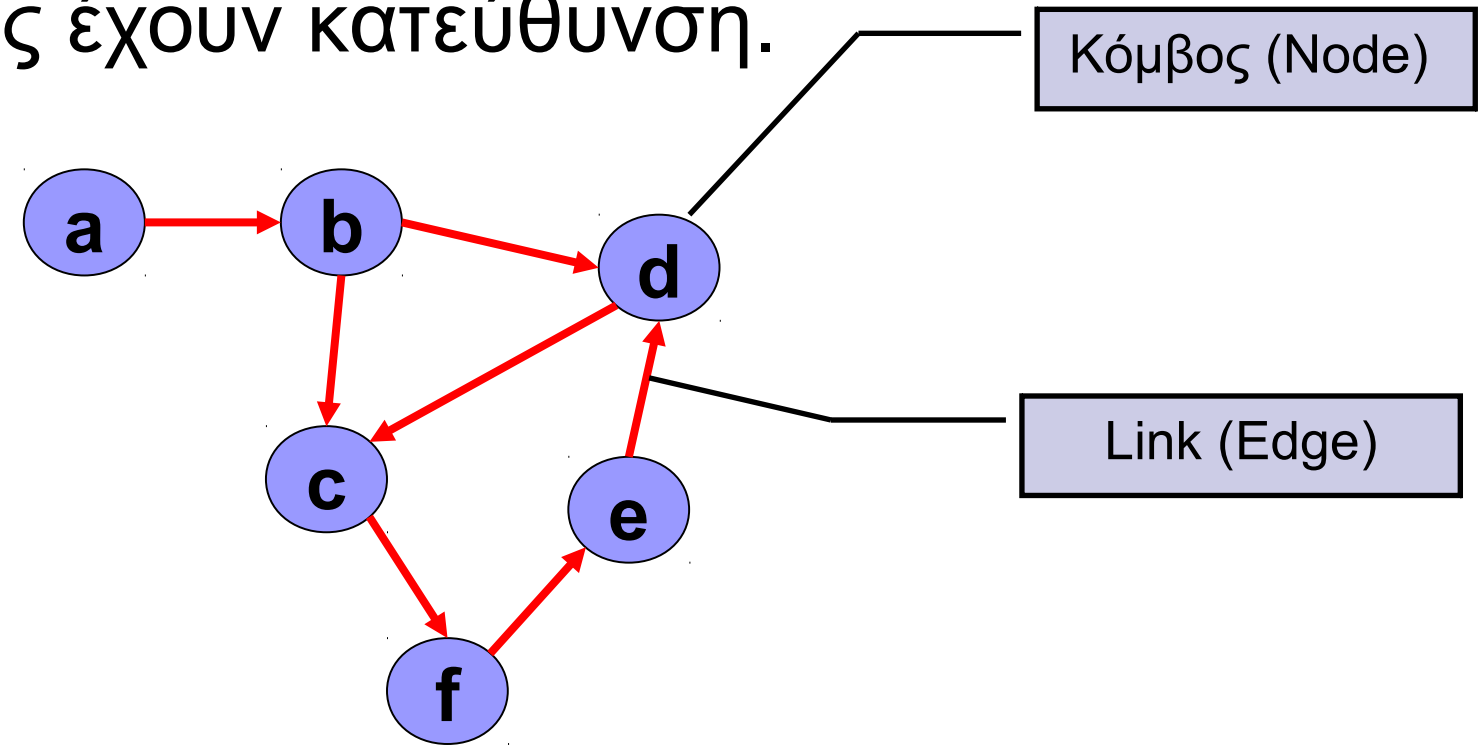
- Γράφοι - Αναπαράσταση σε Prolog
- Εύρεση Διαδρομής σε Γράφους
- Αναζήτηση με περιορισμούς στους κόμβους
- Αναζήτηση σε γράφους με βάρη.

Γράφοι

- Μοντελοποίηση Προβλημάτων.
  - Πολύ συνηθισμένη μέθοδος στη ΤΝ και σε άλλες περιοχές
  - Μοντελοποίηση πλήθους προβλημάτων
- Γράφος
  - Κόμβοι
  - Σύνολο από ακμές που ενώνουν κόμβους.

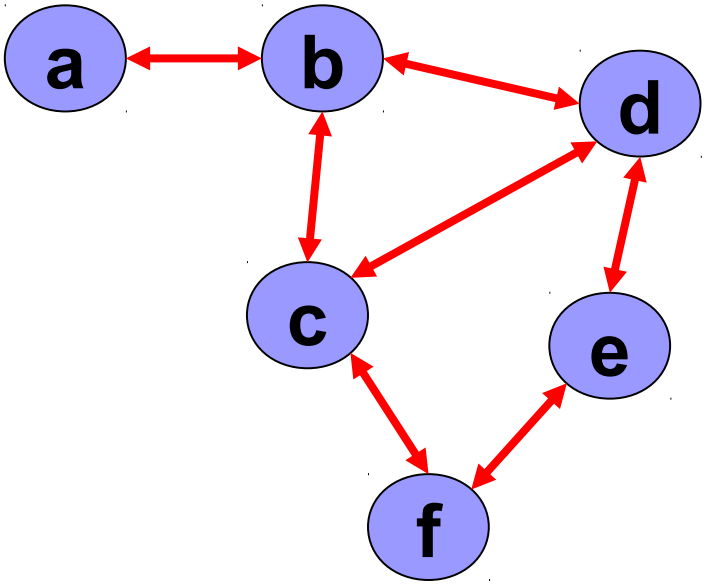
Κατευθυνόμενος Γράφος

Ακμές έχουν κατεύθυνση.



Μη-Κατευθυνόμενος Γράφος

Ακμές **δεν** έχουν κατεύθυνση.



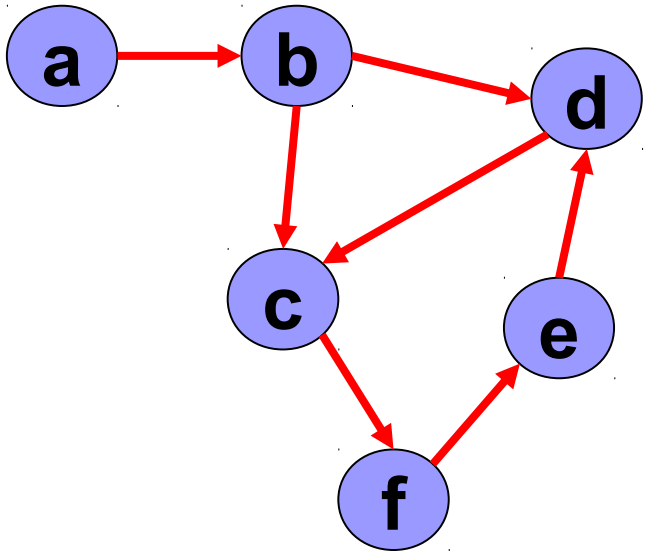
Εύρεση διαδρομής

- Εύρεση διαδρομής από ένα κόμβο σε ένα άλλο.
- Συνηθισμένος πρόβλημα στους γράφους.
- Η εύρεση διαδρομής μπορεί να έχει διάφορες απαιτήσεις:
  - Συντομότερη διαδρομή
  - Περιορισμούς, κλπ.
- Δένδρα είναι επίσης ακυκλικοί γράφοι.

Αναπαράσταση Γράφων στη Prolog

- Οι ακμές αποτελούν σχέσεις μεταξύ των κόμβων.
  - Η κατεύθυνση αναπαριστάται από τη θέση των ορισμάτων.
- Οι γράφοι μπορούν να αναπαρασταθούν σαν γεγονότα:

link(a,b).
link(b,c).
link(b,d).
link(d,c).
...



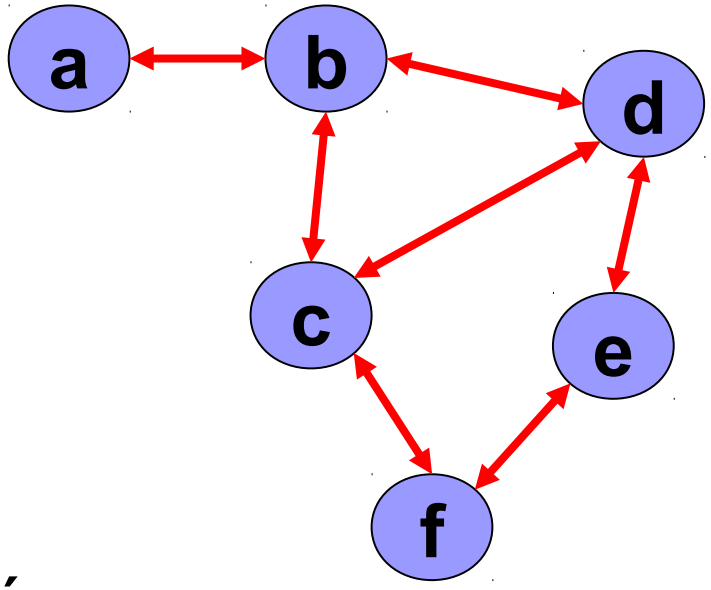
Αναπαράσταση μη Κατευθυνόμενων Γράφων

- Ρητή αναπαράσταση όλων των σχέσεων.

  link(a,b).

  link(b,a).

  ...



- Ή καλύτερα:

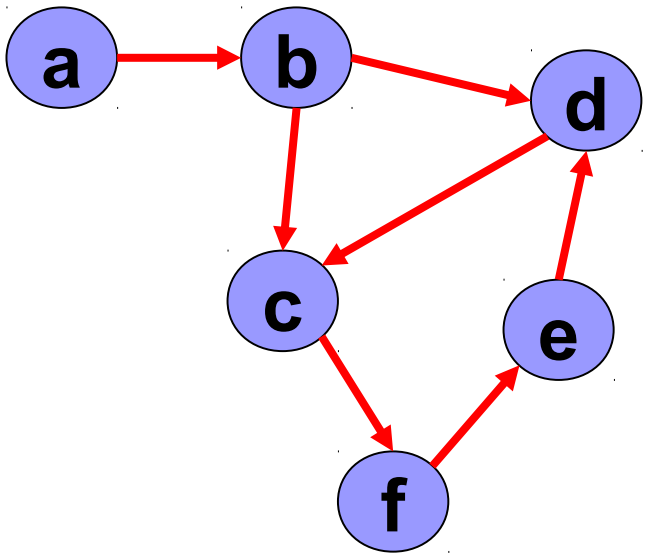
  next(X,Y):- link(X,Y).

  next(X,Y):- link(Y,X).

- Μπορεί να χρησιμοποιηθεί το
  link(X,Y):- link(Y,X). ?

Εναλλακτική Αναπαράσταση

- Μια πρόταση για κάθε κόμβο



link(a,[b]).

link(b,[c,d]).

next(X,Y):-

link(X, ListOfNodes),

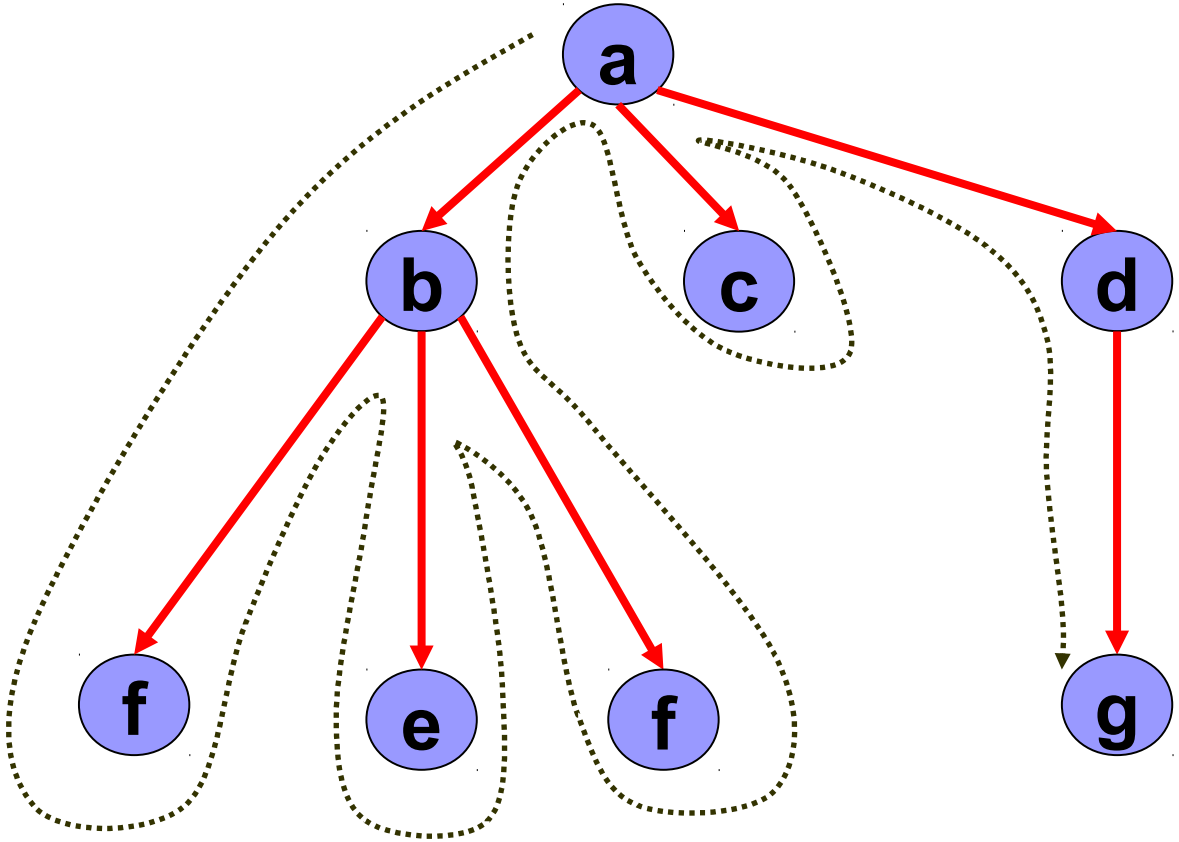
member(Y,ListOfNodes).

- Περισσότερο συμπαγής αναπαράσταση.
- Απαιτεί περισσότερή υπολογισιτκή εργασία κατά την εκτέλεση.

Πρόβλημα Εύρεσης Διαδρομής

- Κοινό πρόβλημα σε όλες τις εφαρμογές που χρησιμοποιούν γράφους.
  - Υπάρχει πλήθος αλγορίθμων που αναζητούν διαδρομή σε γράφους.
- Αναζήτηση **πρώτα σε βάθος.**
  - Αναζήτηση λύσης σε κόμβους που βρίσκονται βαθύτερα στο δένδρο πρώτα.
- Πολύ εύκολα υλοποιήσιμος στην Prolog.
  - Αντιστοιχεί στην εκτέλεση της Prolog.

Αναζήτηση κατά βάθος - Depth First Search



Κώδικας Prolog Code

- Το κατηγόρημα path/3 βρίσκει τη συντομότερη διαδρομή μεταξύ των δύο κόμβων Node και FinalNode.
- Αναδρομή.

```
path(Node,FinalNode,[Node,FinalNode]):-
    next(Node,FinalNode).
path(Node,FinalNode,[Node|RestPath]):-
    next(Node,NextNode),
    path(NextNode, FinalNode, RestPath).
```

Πρόβλημα

- Σε ένα γράφο που περιέχει βρόγχους (κυκλικός) ο κώδικας μπορεί να πέσει σε ατέρμονες βρόχους.
- Πρέπει να γίνεται έλεγχος αν ο αλγόριθμος έχει επισκεφτεί τον κόμβο ξανά.
  - Περίληψη μιας ακόμη μεταβλητής που αποθηκεύει κόμβους.

Ασφαλέστερη Έκδοση

- Η παράμετρος *Visited* αποθηκεύει τους κόμβους που έχει επισκεφτεί ο αλγόριθμος.

```
path_safe(InitialNode, FNode, Route):-
    path(InitialNode, FNode, [InitialNode], Route).

path(Node, FNode, _, [Node, FNode]):-
    next(Node, FNode).

path(Node, FNode, Visited, [Node | RestRoute]):-
    next(Node, NextNode),
    not(member(NextNode, Visited)),
    path(NextNode, FNode, [NextNode|Visited], RestRoute).
```

Εναλλακτική Έκδοση

- Εκμετάλλευση του γεγονότος ότι η λίστα Visited περιέχει ήδη τη διαδρομή.

```
path_safe_alt(InitialNode, FNode, FinRoute):-
    path_alt(InitialNode, FNode, [InitialNode], Route),
    reverse(Route,FinRoute).

path_alt(Node, FNode, Route, [FNode|Route]):-
    next(Node, FNode).

path_alt(Node, FNode, Visited, Route):-
     next(Node, NextNode),
     not(member(NextNode, Visited)),
     path_alt(NextNode, FNode, [NextNode|Visited], Route).
```

Αναζήτηση με Κριτήρια

- Υπάρχουν περιπτώσεις όπου η παραγόμενη διαδρομή πρέπει να ικανοποιεί κάποια κριτήρια.
- Για παράδειγμα:
  - Η διαδρομή πρέπει να περάσει από συγκεκριμένο κόμβο (positive constraint).
  - Η διαδρομή ΔΕΝ πρέπει να περάσει από κάποιο κόμβο (negative constraint).
- Έκφραση των κριτηρίων σαν κατηγορήματα.

Έκφραση Κριτηρίων

- Αρνητικοί περιορισμοί

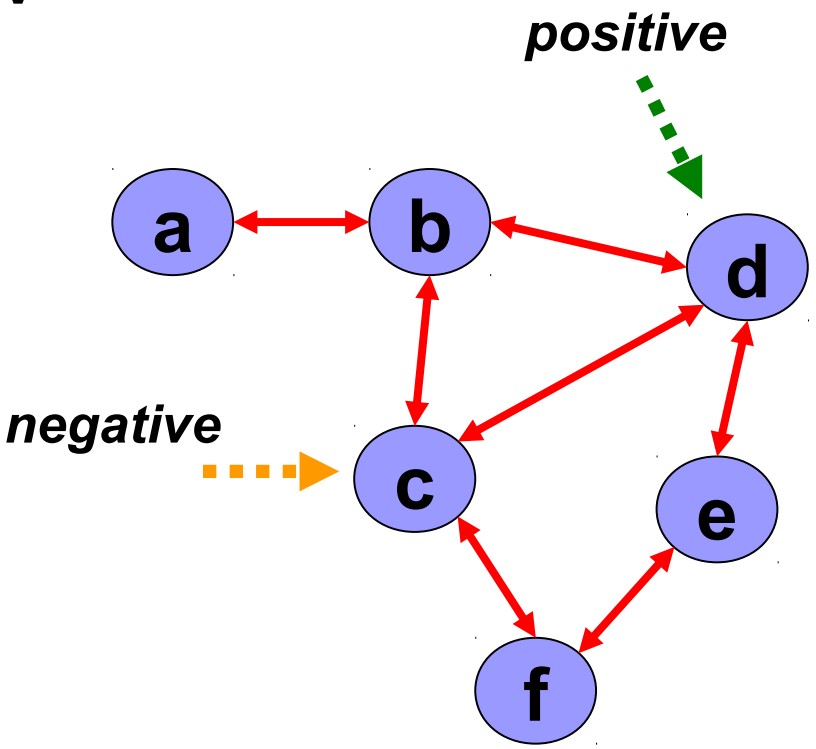
  negative_constraint(Path):-

  member(c, Path).

- Θετικοί Περιορισμοί.

  positive_constraint(Path):-

  member(d, Path).



Αναζήτηση με Κριτήρια

- Έκδοση που λαμβάνει υπόψη κριτήρια.

```
path_constraints(InitialNode, FNode, Route):-
    path(InitialNode, FNode, [InitialNode], Route).

path(Node, FNode, Path, [Node, FNode]):-
   next(Node, FNode),
   positive([FNode|Path]).

path(Node, FNode, Visited, [Node | RestRoute]):-
   next(Node, NextNode),
   not(member(NextNode, Visited)),
   not(negative([NextNode|Visited])),
   path(NextNode, FNode, [NextNode|Visited], RestRoute).
```

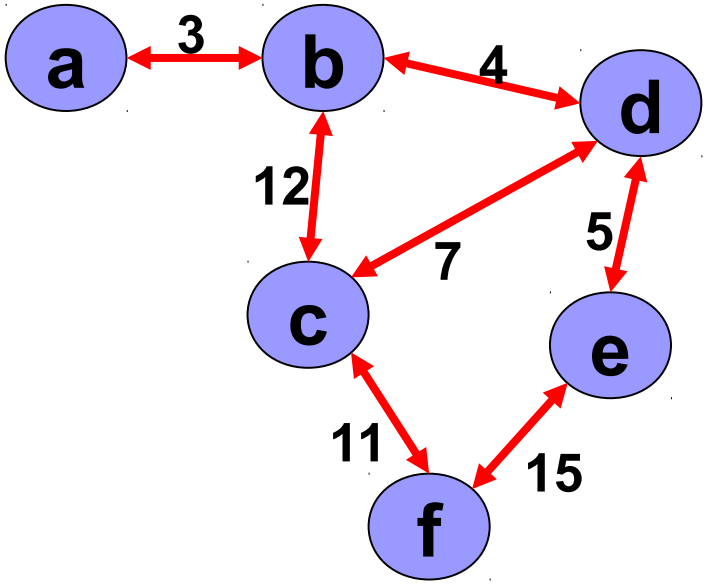
Γράφοι με βάρη

- Κάθε ακμή έχει ένα βάρος.
- Η διαδικασία αναζήτησης θα πρέπει να βρίσκει και το συνολικό κόστος της διαδρομής.
- Αναπαράσταση

  link(a,b,3).

  link(b,c,12).

  ....



Εύρεση διαδρομής με Κόστος

- Το όρισμα κόστος αποθηκεύει το κόστος της διαδρομής

```
path_cost(InitialNode, FNode, Route, Cost):-
    path(InitialNode, FNode, [InitialNode], Route, Cost).

path(Node, FNode, _, [Node, FNode], Cost):-
    next(Node, FNode,Cost).

path(Node, FNode, Visited, [Node | Route],Cost):-
    next(Node, Next,NextCost),
    not(member(Next, Visited)),
    path(Next, FNode, [Next|Visited], Route,RestCost),
    Cost is NextCost + RestCost.
```