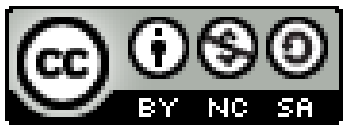
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Υπολογιστική Λογική και Λογικός Προγραμματισμός

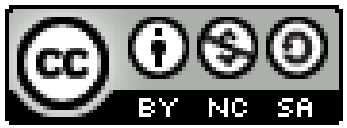
Ενότητα 9: Η Γλώσσα Λογικού Προγραμματισμού Prolog
Ταξινόμηση, Δένδρα, Επίλυση Quiz

Νίκος Βασιλειάδης, Αναπλ. Καθηγητής
Τμήμα Πληροφορικής



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» 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.

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

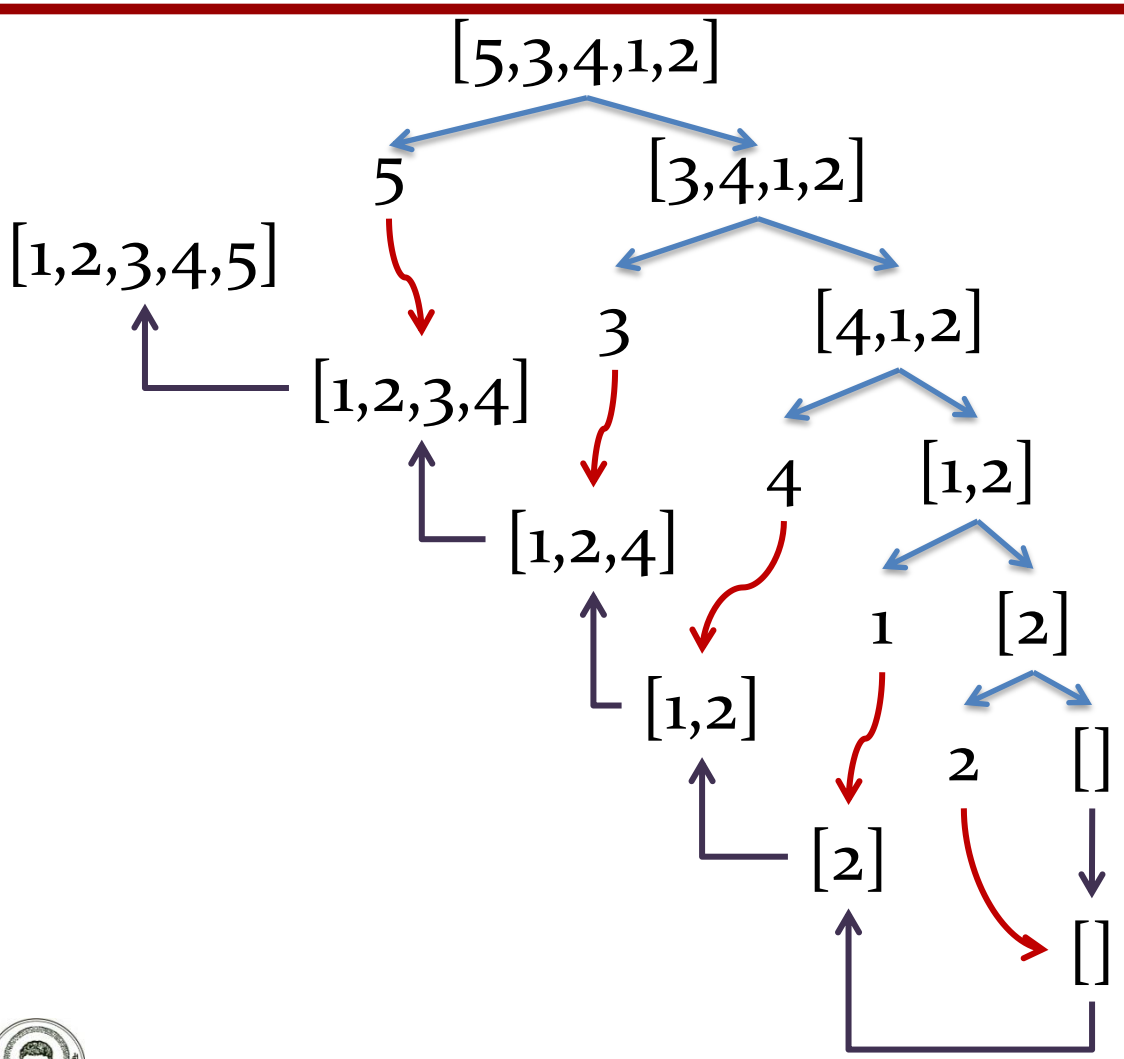
Η Γλώσσα Λογικού Προγραμματισμού Prolog
Ταξινόμηση, Δένδρα, Επίλυση Quiz

Ταξινόμηση Λίστας Insert Sort

1. *Χώρισε την λίστα σε κεφαλή και ουρά.*
2. *Ταξινόμησε την ουρά.*
3. *Βάλε την κεφαλή στην ταξινομημένη ουρά στην «σωστή» θέση.*

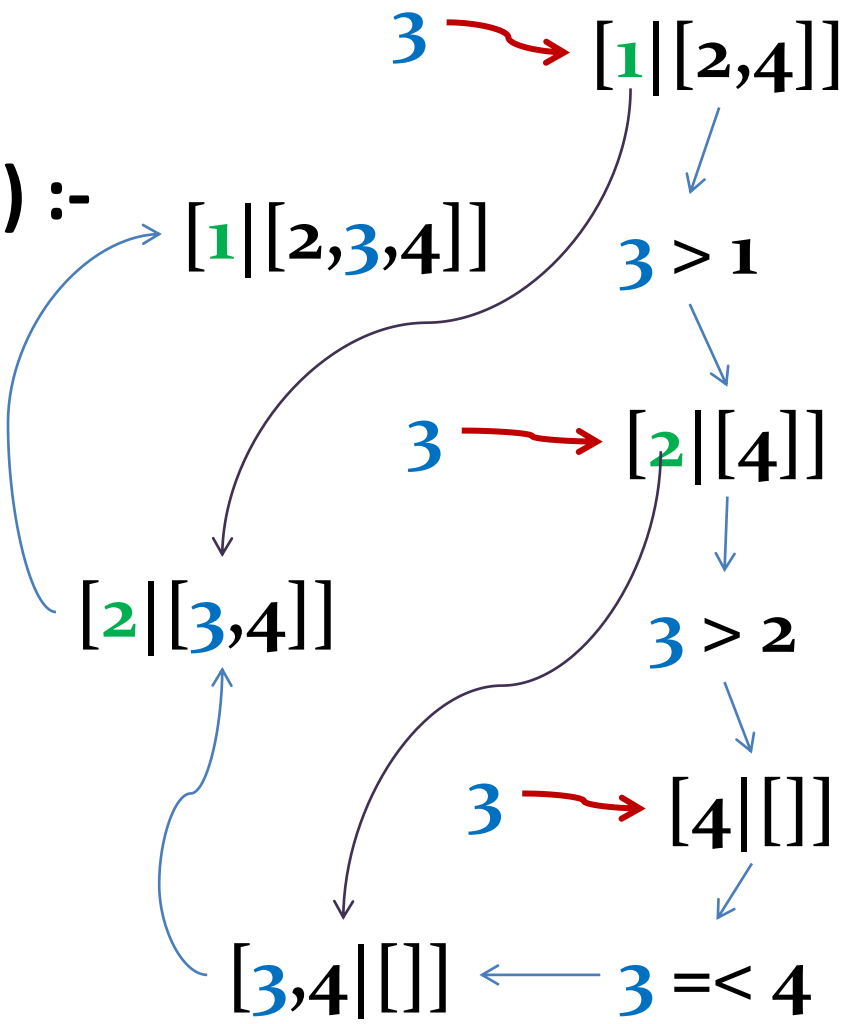
```
isort([],[]).
isort([Head|Tail],Result) :-
        isort(Tail,SortedTail),
        insert(Head,SortedTail,Result).

insert(X,[],[X]).
insert(X,[Y|Tail],[X,Y|Tail]) :- X =< Y.
insert(X,[Y|Tail],[Y|Z]) :- X > Y,
        insert(X,Tail,Z).
```

Ταξινόμηση Λίστας Insert Sort



Εξήγηση insert/3

```
insert(X,[],[X]).
insert(X,[Y|Tail],[X,Y|Tail]) :-
        X =< Y.
insert(X,[Y|Tail],[Y|Z]) :-
        X > Y,
        insert(X,Tail,Z).
```



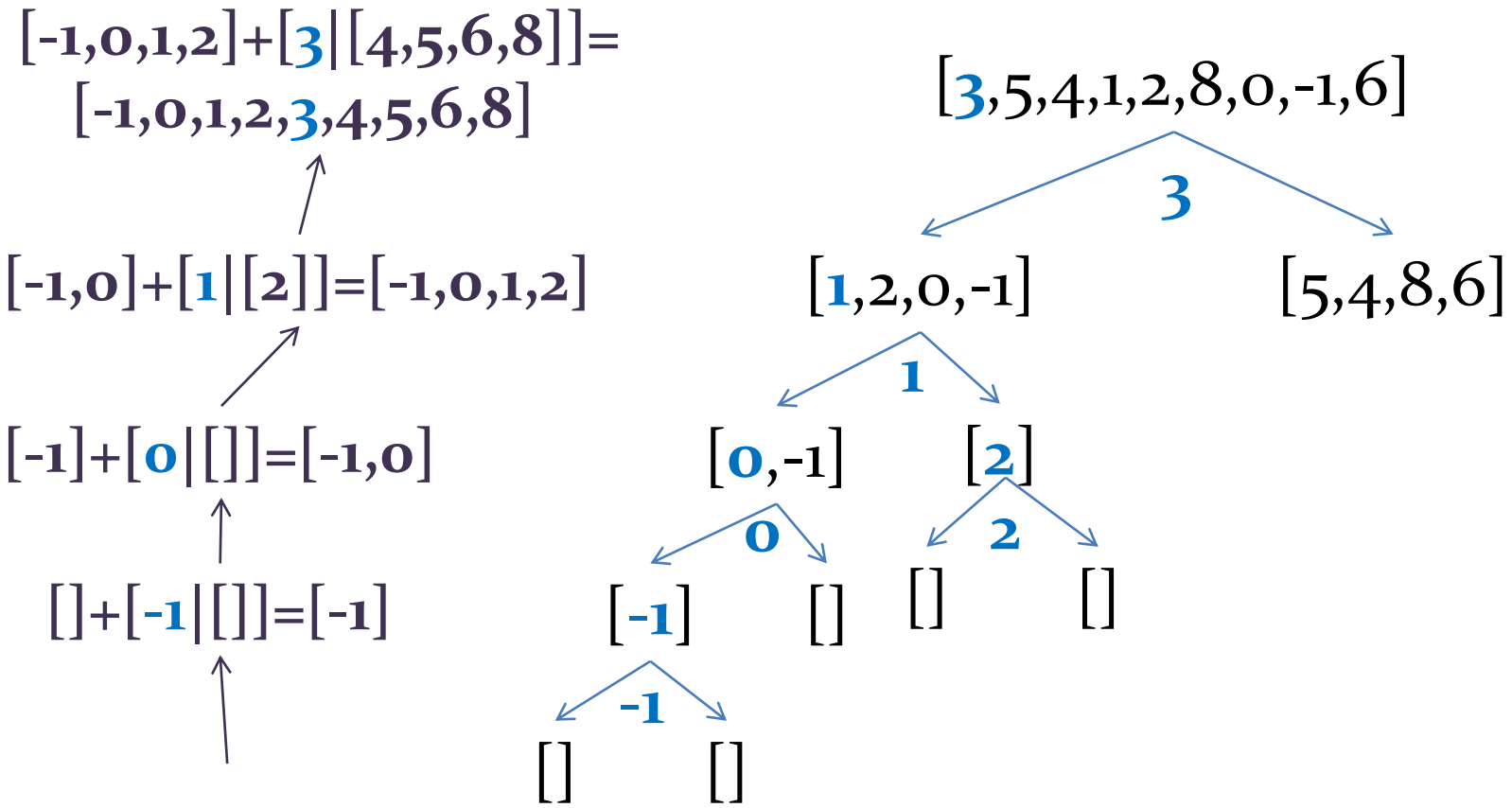
Ταξινόμηση Λίστας - Quick Sort (1/2)

```
qsort([],[]).
qsort([X|L],Sorted) :-
        split(X,L,Small,Big),
        qsort(Small,SortedSmall),
        qsort(Big,SortedBig),
        append(SortedSmall,[X|Sorted Big],Sorted).
split(X,[],[],[]).
split(X,[Y|T],[Y|Small],Big) :- X > Y,
        split(X,T,Small,Big).
split(X,[Y|T],Small,[Y|Big]) :- X=<Y,
        split(X,T,Small,Big).
```

1. *Βάσει του πρώτου στοιχείου X της λίστας χώρισε την σε 2 (μικρότερα, μεγαλύτερα)*
2. *Ταξινόμησε τις 2 λίστες ανεξάρτητα*
3. *Ένωσε τις 2 ταξινομημένες λίστες με το X στη μέση*

Χωρίζει μια λίστα σε δύο μικρότερες λίστες. Η μία έχει τα μικρότερα στοιχεία από το X και η άλλη τα μεγαλύτερα

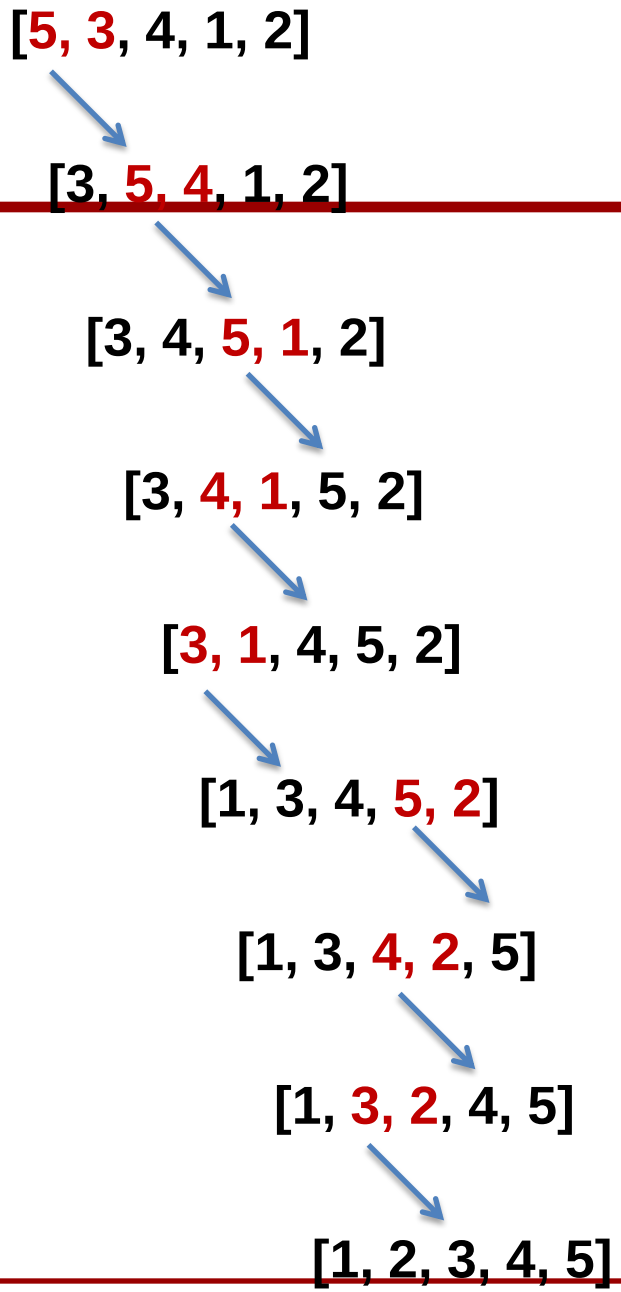
Ταξινόμηση Λίστας - Quick Sort (2/2)



Ταξινόμηση - Bubble Sort

```
bsort(List,Sorted) :-
        swap(List,List1), ! ,
        bsort(List1,Sorted).
bsort(Sorted,Sorted).

swap([X,Y|Rest],[Y,X|Rest]) :- X > Y.
swap([Z|Rest],[Z|Rest1]) :- swap(Rest,Rest1).
```

Κλήσεις swap



9 περάσματα

Επιπεδοποίηση λίστας (1/2)

- Γράψτε το κατηγόρημα **flatten(L1,L2)** όπου το δεύτερο όρισμα **L2** είναι η πεπλατυσμένη λίστα **L1**:
  - Αν η **L1** περιέχει άλλες λίστες μέσα της τότε η **L2** περιέχει όλα τα άτομα που περιέχει η **L1** χωρίς όμως να περιέχει άλλες υπολίστες.

?- flatten([a,[[b,c],[[[d,e],f]],g],L2).

L2 = [a,b,c,d,e,f,g]

Επιπεδοποίηση λίστας (2/2)

```
flatten([],[]).
flatten([Head|Tail],FlatList) :-
    is_list(Head), !,
    flatten(Head,FlatHead),
    flatten(Tail,FlatTail),
    append(FlatHead,FlatTail,FlatList).
flatten([Head|Tail],[Head|FlatTail]) :-
    flatten(Tail,FlatTail).
```

```
is_list([]).
is_list([H|T]) :-
        is_list(T).
```

Επιπεδοποίηση λίστας - Εφαρμογή

- Να οριστεί το κατηγόρημα **alter(List1,List2)**, το οποίο να μετατρέπει μία φράση (λίστα με λέξεις) που εισάγει ο χρήστης σε μία αντίστοιχη λίστα λέξεων που να απαντάει στην αρχική φράση.
- Για παράδειγμα στην φράση του χρήστη: ***You are a computer***, να απαντάει: ***I am not a computer***

?- alter([do,you,know,french],A).

A = [no,i,know,german]

?- alter([you,are,a,computer],A).

A = [i,[am,not],a,computer]

Κώδικας

```
alter([],[]).
alter([H|T],[H1|T1]) :-
        change(H,H1), !,
        alter(T,T1).
```

```
change(you,i).
change(are,[am,not]).
change(french,german).
change(do,no).
change(X,X).
```

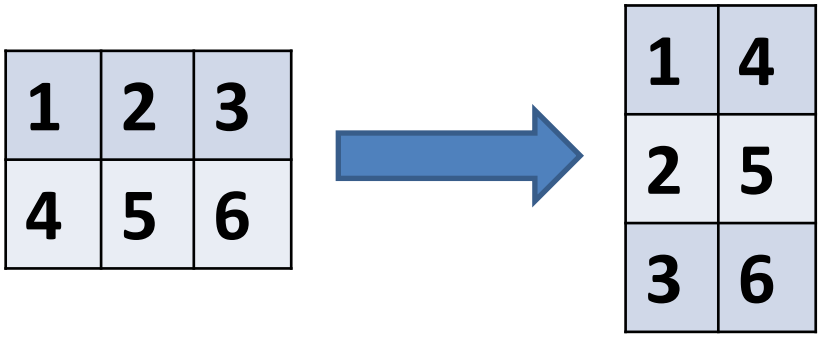
Ολοκλήρωση εφαρμογής

```
alter_phrase(In,Out) :-
        alter(In,Temp),
        flatten(Temp,Out).
```

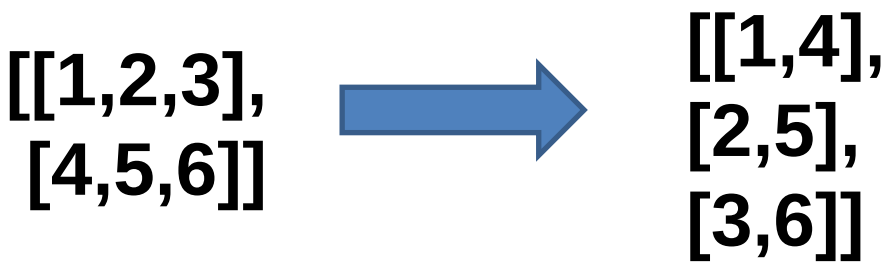
- Παράδειγμα:

```
?- alter_phrase([do,you,know,french],A).
A = [no,i,know,german]
?- alter_phrase([you,are,a,computer],A).
A = [i,am,not,a,computer]
```

Αντιμετάθεση Πίνακα



Αναπαράσταση στην Prolog με λίστες



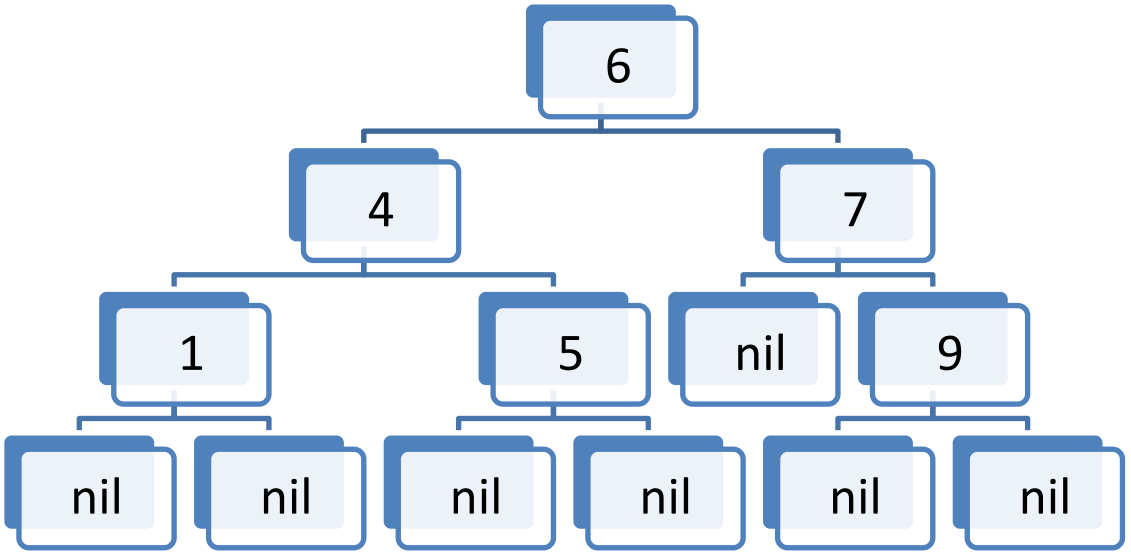
Κατηγόρημα transpose

```
transpose([[]|_],[]).
transpose([[H1|HT]|T],[[H1|T1]|T3]) :-
        transpose_one(T,T1,T2),
        transpose([HT|T2],T3).

transpose_one([],[],[]).
transpose_one([[H1|T1]|T],[H1|T2],[T1|T3]) :-
        transpose_one(T,T2,T3).
```

```
?- transpose([[1,2,3],[4,5,6]],A).
A = [[1, 4], [2, 5], [3, 6]]
```

Δυαδικά Δένδρα

```
tree(6,
     tree(4,
          tree(1,
               empty,
               empty),
          tree(5,
               empty,
               empty)),
     tree(7,
          empty,
          tree(9,
               empty,
               empty)))
```



Δυαδικά Δένδρα

- Σε κάθε κόμβο του δένδρου υπάρχει:
  - Ένα στοιχείο (αριθμός).
  - Το αριστερό υπο-δένδρο.
    - Όλοι οι αριθμοί που περιέχονται σε αυτό είναι μικρότεροι από τον αριθμό του τρέχοντος κόμβου.
  - Το δεξί υπο-δένδρο.
    - Όλοι οι αριθμοί που περιέχονται σε αυτό είναι μεγαλύτεροι από τον αριθμό του τρέχοντος κόμβου.
- Οι τερματικοί κόμβοι (nil) δεν έχουν τίποτε από τα παραπάνω.

Εισαγωγή στοιχείου σε δένδρο

- Σύγκριση με το στοιχείο ενός κόμβου.
  - Εισαγωγή στο αριστερό υπο-δένδρο αν είναι μικρότερο, ή στο δεξί αν είναι μεγαλύτερο.
  - Δημιουργία νέου τερματικού κόμβου, αν φθάσουμε σε άδειο δένδρο.

```
insert(New,empty,tree(New,empty,empty)).
insert(New,tree(Element,Left,Right),
                              tree(Element,NewLeft,Right)) :-
        New < Element,
        insert(New,Left,NewLeft).
insert(New,tree(Element,Left,Right),
                              tree(Element,Left,NewRight)) :-
        New >= Element,
        insert(New,Right,NewRight).
```

Ταξινόμηση με δένδρο

- Για να ταξινομήσουμε μια λίστα την μετατρέπουμε σε δυαδικό δέντρο και μετά μετατρέπουμε το δυαδικό δέντρο πίσω σε λίστα.

```
treesort(List,NewList) :-
        list_to_tree(List,Tree),
        tree_to_list(Tree,NewList).
```

```
?- treesort([6,4,7,1,5,9],A).
A = [1, 4, 5, 6, 7, 9] .
```

Εισαγωγή όλων των στοιχείων της λίστας στο δένδρο

```
list_to_tree(List,Tree) :-
        insert_list(List,empty,Tree).

insert_list([],Tree,Tree).
insert_list([Head|Tail],TempTree,NewTree) :-
        insert(Head,TempTree,NextTree),
        insert_list(Tail,NextTree,NewTree).
```

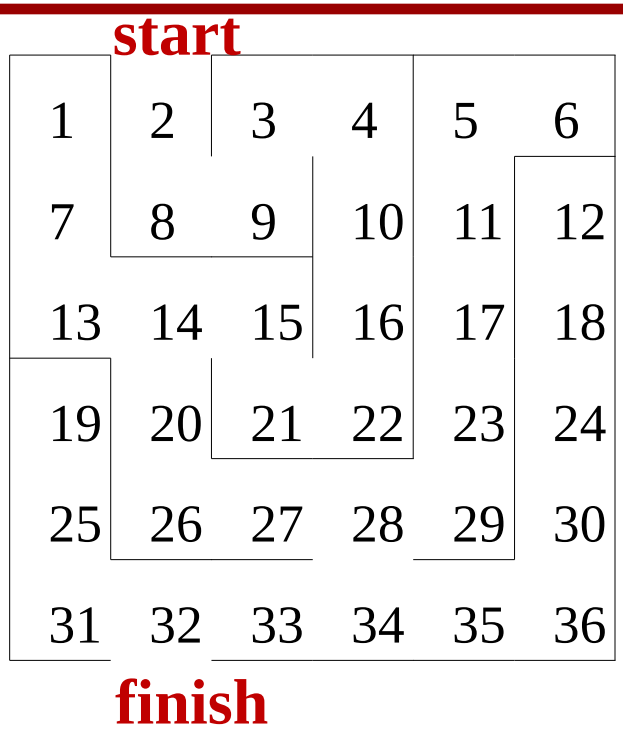
Εξαγωγή των στοιχείων από το δένδρο σε λίστα

- Η διάσχιση του δένδρου γίνεται με **ενδο-διατεταγμένη** (**in-order**) διάσχιση.

```
tree_to_list(Tree,List) :-
        tree_to_list_aux(Tree,[],List).

tree_to_list_aux(empty,List,List).
tree_to_list_aux(tree(Item,Left,Right),
                                    OldList,NewList) :-
        tree_to_list_aux(Right,OldList,List1),
        tree_to_list_aux(Left,[Item|List1],NewList).
```

Λαβύρινθος

```
solve_maze (Begin,Solution):-
    path([Begin],Solution).

path([finish|Rest],[finish|Rest]).
path([Old|RestPath],Solution) :-
    connected_to(New,Old),
    not(member(New,RestPath)),
    path([New,Old|RestPath],Solution).

connected_to(A,B) :-
    connect(A,B).
connected_to(A,B) :-
    connect(B,A).
```




```
connect(start,2).   connect(1,7).
connect(2,8).       connect(3,4).
…
connect(34,35).     connect(35,36).
connect(32,finish).
```

Εκτέλεση

?- solve_maze(start,A).

A = [finish,32,33,34,28,27,26,20,14,15,21,22,16,10,4,3,9,8,2,start]

- Για να μην επιστρέψει τη λύση ανάποδα:

```
solve_maze(Start,Solution) :-
    path([Start],RSolution),
    reverse(RSolution,Solution).
```

Λαβύρινθος – Βέλτιστη διαδρομή

- Η λύση που δώσαμε για τον λαβύρινθο δίνει μια διαδρομή.
- Με οπισθοδρόμηση μπορούμε να βρούμε όλες τις εναλλακτικές λύσεις.
- Ποια είναι όμως η βέλτιστη;

Εναλλακτικές Λύσεις

- Έστω ότι υπάρχει μεταξύ **17/18** ανοιχτή πόρτα
  - Προσθέτω: **connect(17,18).**
- Το **solve_maze** θα δώσει 2 λύσεις:

?- solve_maze(start,A).

A=[start,2,8,9,3,4,10,16,22,21,15,14,20,26,27,28,

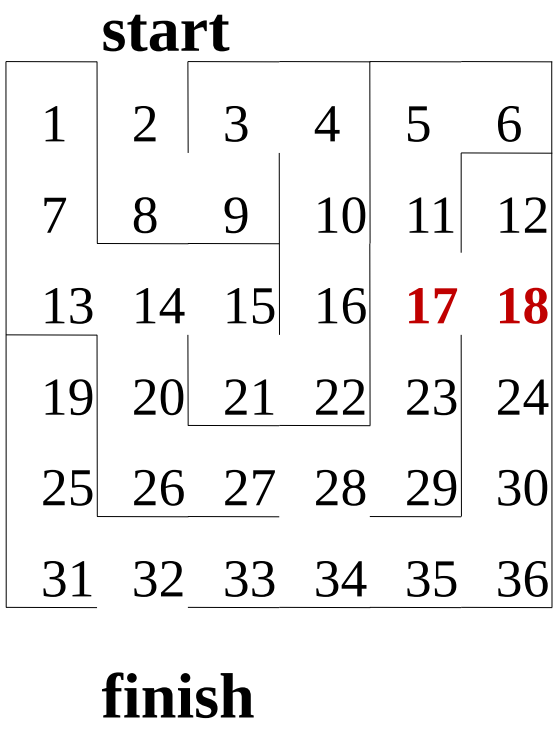
29,23,17,18,24,30,36,35,34,33,32,finish];

A=[start,2,8,9,3,4,10,16,22,21,15,14,20,26,27,28,

34,33,32,finish];

false

- ***Πώς θα πάρω την συντομότερη διαδρομή?***



Συντομότερη Διαδρομή

```
best_solution(Begin,BestS,BestL) :-
        setof(L-P,(path([Begin],P),length(P,L)),List),
        List = [BestL-BestRS|_],
        reverse(BestRS,BestS).
```

Υπολογίζει μια διαδρομή και «μετράει» το μήκος της.

Στη λίστα υπάρχουν ζεύγη μήκος-διαδρομή ταξινομημένα βάσει μήκους.

?- best_solution(start,A,L).

A=[start,2,8,9,3,4,10,16,22,21,15,14,20,26,27,28,34,33,32,finish]

L = 20

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2013-2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

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.