Αριστοτέλειο
Πανεπιστήμιο
Θεσσαλονίκης

Τεχνητή Νοημοσύνη

Αλγόριθμοι Ευριστικής Αναζήτησης

Ιώαννης Βλαχάβας

Τμήμα Πληροφορικής ΑΠΘ

Θεσσαλονίκη, 2013

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 τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

ανοικτάμαθήματα
opencourses

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.

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

Αλγόριθμοι Ευριστικής Αναζήτησης

Αλγόριθμοι Ευριστικής Αναζήτησης

- Οι αλγόριθμοι τυφλής αναζήτησης προχωρούν σε βάθος ή πλάτος χωρίς ένδειξη για το αν πλησιάζουν σε τερματική κατάσταση.
- Σε πολλά προβλήματα ο χώρος αναζήτησης αυξάνεται ραγδαία (συνδυαστική έκρηξη).
  - **Σκοπός**: Μείωση του αριθμού των καταστάσεων που επισκέπτεται ο αλγόριθμος.
  - **Απαιτείται**: Πληροφορία για αξιολόγηση των καταστάσεων που θα οδηγήσει γρηγορότερα στη λύση ή θα βοηθήσει στο κλάδεμα καταστάσεων που δεν οδηγούν πουθενά.
- Οι αλγόριθμοι που εκμεταλλεύονται τέτοιες πληροφορίες (κάποιον ευριστικό μηχανισμό) ονομάζονται Αλγόριθμοι Ευριστικής Αναζήτησης.

Παράδειγμα

- Παράδειγμα ευριστικής αναζήτησης είναι η συναρμολόγηση ενός puzzle.
  - Με τυφλή αναζήτηση θα έπρεπε να προσπαθήσουμε να το φτιάξουμε, χωρίς να δώσουμε καμία σημασία στο χρώμα ή το σχήμα των κομματιών.
  - Στην πραγματικότητα όμως, ενεργούμε εντελώς διαφορετικά. Χωρίζουμε τα κομμάτια σε κατηγορίες, π.χ. αυτά που ανήκουν στην περιφέρεια (μία άκρη τους είναι ευθεία τομή), σε αυτά που πιθανά ανήκουν στο χρώμα του ουρανού ή στη θάλασσα ή στα δένδρα κ.ο.κ.
  - Οι ενέργειες αυτές αντιστοιχούν στη χρήση ενός ευριστικού μηχανισμού που έχουμε αναπτύξει μέσω της εμπειρίας μας.
  - Ωστόσο, ποτέ δεν είμαστε σίγουροι ότι ο διαχωρισμός των κομματιών είναι και ο σωστός. Για παράδειγμα, υπάρχουν κομμάτια του puzzle που μπορεί να ανήκουν στη θάλασσα ή τον ουρανό.

- Αν δεν υπήρχαν ευριστικοί μηχανισμοί, τότε τα προβλήματα αυτά θα λύνονταν πολύ δύσκολα, γιατί οι συνδυασμοί που πρέπει να γίνουν είναι πάρα πολλοί.
  - Θα ήταν σα να προσπαθεί κάποιος να συναρμολογήσει ένα puzzle από την ανάποδη πλευρά ή σα να ψάχνει την κεντρική πλατεία της πόλης με τα μάτια κλειστά.
- Ο ευριστικός μηχανισμός εξαρτάται από τη γνώση που έχουμε για το πρόβλημα.

Ευριστικός Μηχανισμός

Ευριστικός μηχανισμός (heuristic) είναι μία στρατηγική, βασισμένη στη γνώση για το συγκεκριμένο πρόβλημα, η οποία χρησιμοποιείται σα βοήθημα στη γρήγορη επίλυσή του.

- Ο ευριστικός μηχανισμός υλοποιείται με ευριστική συνάρτηση (heuristic function), που έχει πεδίο ορισμού το σύνολο των καταστάσεων ενός προβλήματος και πεδίο τιμών το σύνολο τιμών που αντιστοιχεί σε αυτές.
  - Σε όμοια προβλήματα χρησιμοποιούμε παρόμοιους ευριστικούς μηχανισμούς.

- Ευριστική τιμή (heuristic value) είναι η τιμή της ευριστικής συνάρτησης και εκφράζει το πόσο κοντά βρίσκεται μία κατάσταση σε μία τελική.
  - Δεν είναι πραγματική αλλά μια εκτίμηση (πολλές φορές λανθασμένη)

- Η ευριστική τιμή δεν είναι η πραγματική τιμή της απόστασης από μία τερματική κατάσταση, αλλά μία εκτίμηση (estimate) που πολλές φορές μπορεί να είναι και λανθασμένη.

Ευριστικές Συναρτήσεις σε Μικρά Προβλήματα (1/3)

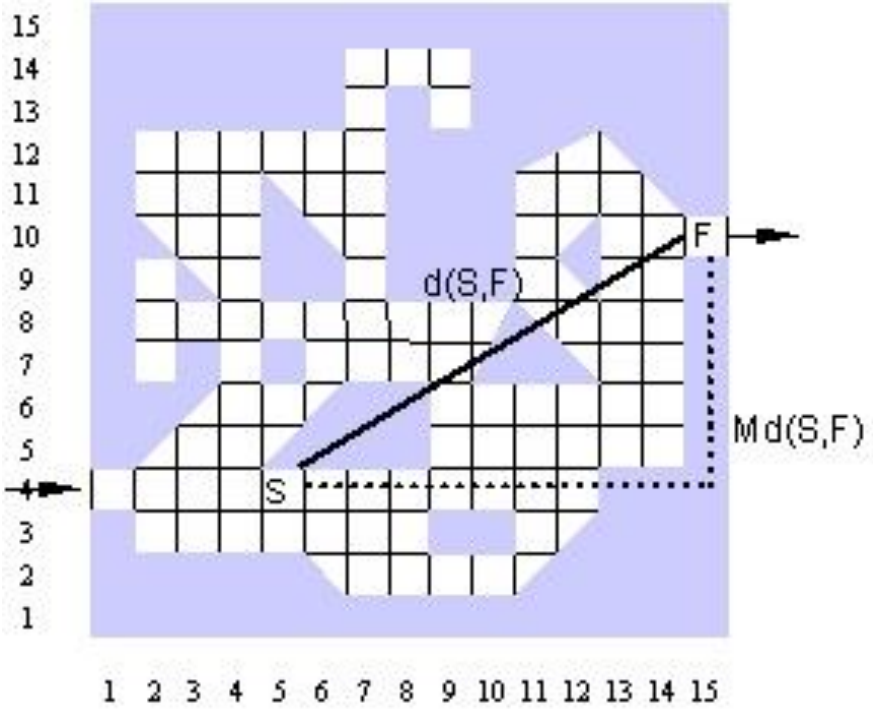
Ευριστικός μηχανισμός και συναρτήσεις σε λαβύρινθο

- Ευκλείδεια απόσταση (Euclidian distance):

$$d(S, F) = \sqrt{(X_S - X_F)^2 + (Y_S - Y_F)^2}$$

- Απόσταση Manhattan (Manhattan distance):

$$Md(S,F) = |X_S - X_F| + |Y_S - Y_F|$$



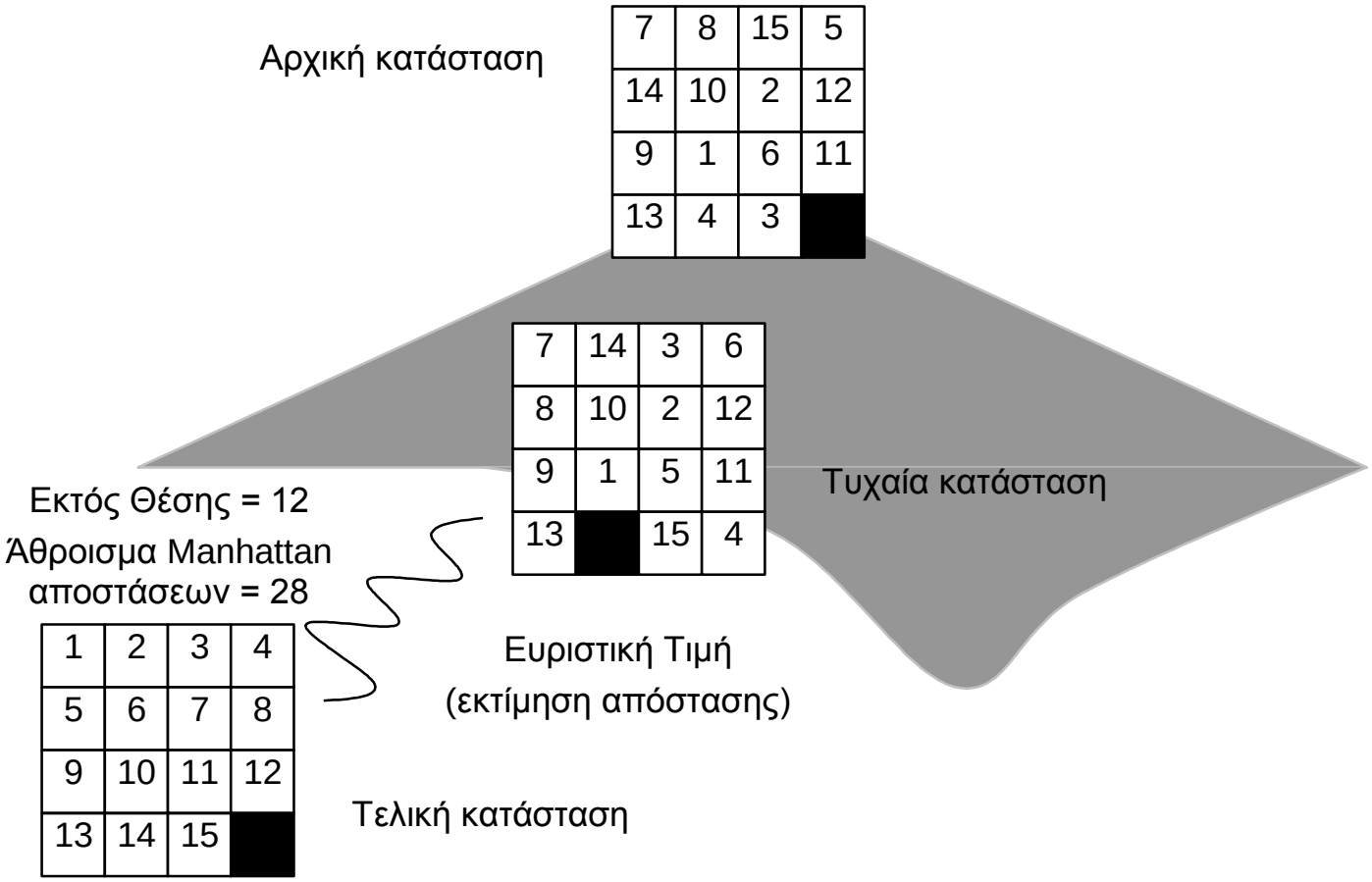
$$d(S, F) = \sqrt{(5-15)^2 + (4-10)^2} = \sqrt{(100+36)} = 11{,}6$$

$Md(S,F)=|5-15| + |4-10|=10+6=16.$

Ευριστικές Συναρτήσεις σε Μικρά Προβλήματα (2/3)

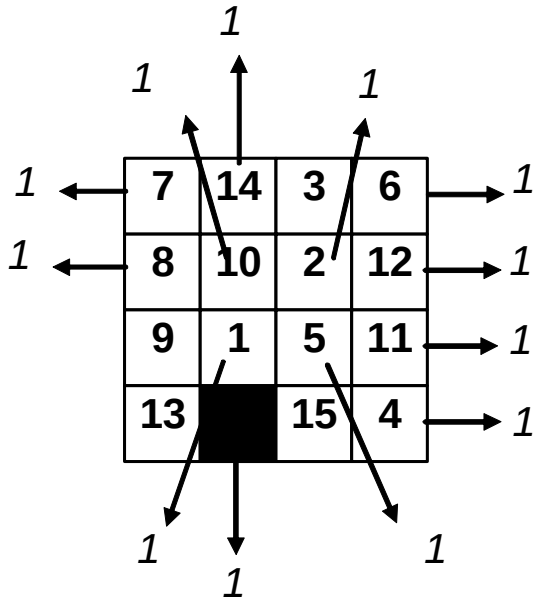
Ευριστικός μηχανισμός και συναρτήσεις στο N-Puzzle

- Πόσα πλακίδια βρίσκονται εκτός θέσης.
- Το άθροισμα των αποστάσεων Manhattan κάθε πλακιδίου από την τελική του θέση.

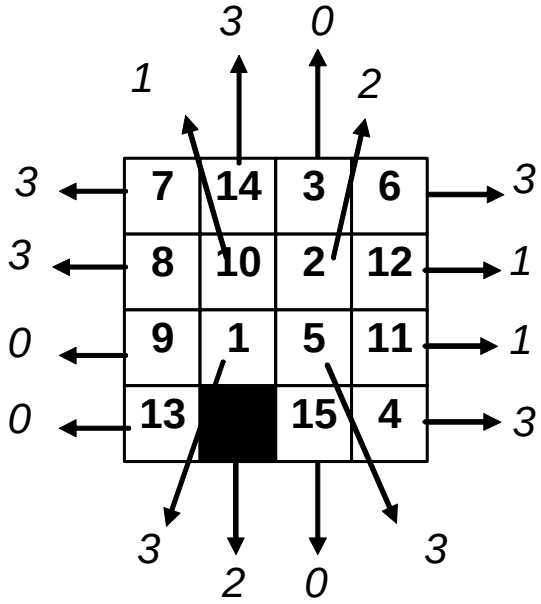


Ευριστικές Συναρτήσεις σε Μικρά Προβλήματα (3/3)

Αναλυτικός υπολογισμός ευριστικής τιμής για μία τυχαία κατάσταση του 15-puzzle.



Εκτός θέσης = 12



Άθροισμα αποστάσεων Manhattan = 28

Ευριστικός μηχανισμός και συναρτήσεις στο TSP

- Η κοντινότερη πόλη έχει περισσότερες πιθανότητες να οδηγήσει σε μία συνολικά καλή λύση.

Αναζήτηση με Αναρρίχηση Λόφων

Η *αναρρίχηση λόφων* (Hill-Climbing Search - HC) είναι ένας αλγόριθμος αναζήτησης που μοιάζει πολύ με τον DFS.

Ο αλγόριθμος HC

1. Η αρχική κατάσταση είναι η τρέχουσα κατάσταση.
2. Αν η κατάσταση είναι μία τελική τότε ανέφερε τη λύση και σταμάτησε.
3. Εφάρμοσε τους τελεστές μετάβασης για να βρεις τις καταστάσεις-παιδιά.
4. Βρες την καλύτερη κατάσταση σύμφωνα με την ευριστική συνάρτηση.
5. Η καλύτερη κατάσταση γίνεται η τρέχουσα κατάσταση.
6. Πήγαινε στο βήμα 2.

- Δεν έχει μέτωπο αναζήτησης

Ο αλγόριθμος HC (Ψευδοκώδικας)

```
algorithm hc(InitialState, FinalState)
  begin
    CurrentState←InitialState;
    while CurrentState≠FinalState do
        Children←Expand(CurrentState);
        if Children=∅ then return failure;
        EvaluatedChildren←Heuristic(Children);
        bestChild←best(EvaluatedChildren);
        if hValue(CurrentState) ≥ hValue(bestChild)
          then return failure;
          else CurrentState←bestChild;
        endif;
    endwhile;
    return success;
  end.
```

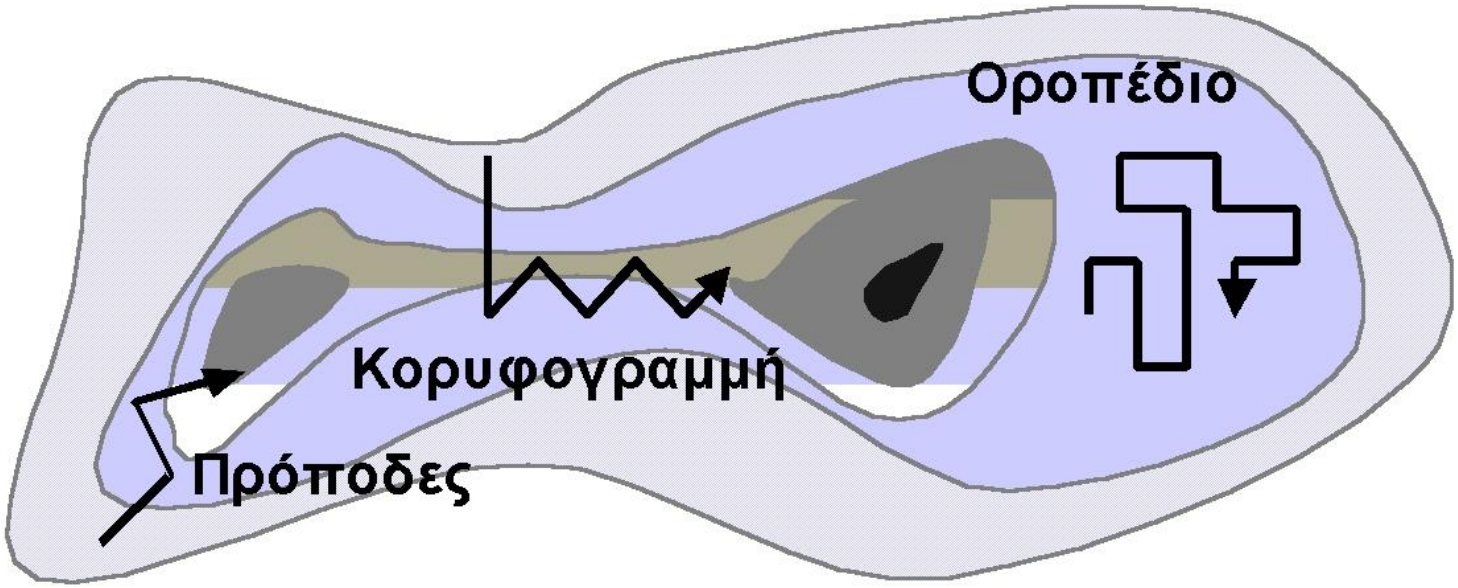
Ο αλγόριθμος HC
Σχόλια (1/2)

- Ο HC χρησιμοποιείται σε προβλήματα όπου πρέπει να βρεθεί μία λύση πολύ γρήγορα, έστω και αν αυτή δεν είναι η καλύτερη, παίρνοντας όμως και το ρίσκο να μη βρεθεί καμία λύση, έστω και αν τέτοια υπάρχει.

- Πλεονεκτήματα:
  - Πολύ αποδοτικός και σε χρόνο και σε μνήμη,
  -
- Μειονεκτήματα:
  - Είναι μη-πλήρης.
  - Βασικά προβλήματα του HC:
    - ✓ Πρόποδες (foothill).
    - ✓ Οροπέδιο (plateau).
    - ✓ Κορυφογραμμή (ridges).

Ο αλγόριθμος HC

Σχόλια (2/2)



- Βελτιώσεις:
  - Εξαναγκασμένη αναρρίχηση λόφου (Enforced Hill-Climbing - EHC)
  - Προσομοιωμένη εξέλιξη (Simulated Annealing - SA)
  - Αναζήτηση με απαγορευμένες καταστάσεις (Tabu Search - TS).

Ακτινωτή Αναζήτηση

Στον *αλγόριθμο ακτινωτής αναζήτησης* (*Beam Search - BS*) δεν κλαδεύονται όλες οι υπόλοιπες καταστάσεις όπως στον HC, αλλά ένας σταθερός αριθμός από τις καλύτερες από αυτές κρατείται στο μέτωπο αναζήτησης.

Αναζήτηση Πρώτα στο Καλύτερο

Ο αλγόριθμος *αναζήτηση* πρώτα στο καλύτερο (Best-First - *BestFS*) κρατά όλες τις καταστάσεις στο μέτωπο αναζήτησης.

Ο αλγόριθμος BestFS

1. Βάλε την αρχική κατάσταση στο μέτωπο αναζήτησης.
2. Αν το μέτωπο αναζήτησης είναι κενό τότε σταμάτησε.
3. Πάρε την πρώτη σε σειρά κατάσταση από το μέτωπο αναζήτησης.
4. Αν η κατάσταση είναι μέλος του κλειστού συνόλου τότε πήγαινε στο 2.
5. Αν η κατάσταση είναι μία τελική τότε ανέφερε τη λύση και σταμάτα.
6. Εφάρμοσε τους τελεστές μεταφοράς για να παράγεις τις καταστάσεις-παιδιά.
7. Εφάρμοσε την ευριστική συνάρτηση σε κάθε παιδί.
8. Βάλε τις καταστάσεις-παιδιά στο μέτωπο αναζήτησης.
9. Αναδιάταξε το μέτωπο αναζήτησης, έτσι ώστε η κατάσταση με την καλύτερη ευριστική τιμή να είναι πρώτη.
10. Βάλε τη κατάσταση-γονέα στο κλειστό σύνολο.
11. Πήγαινε στο βήμα 2.

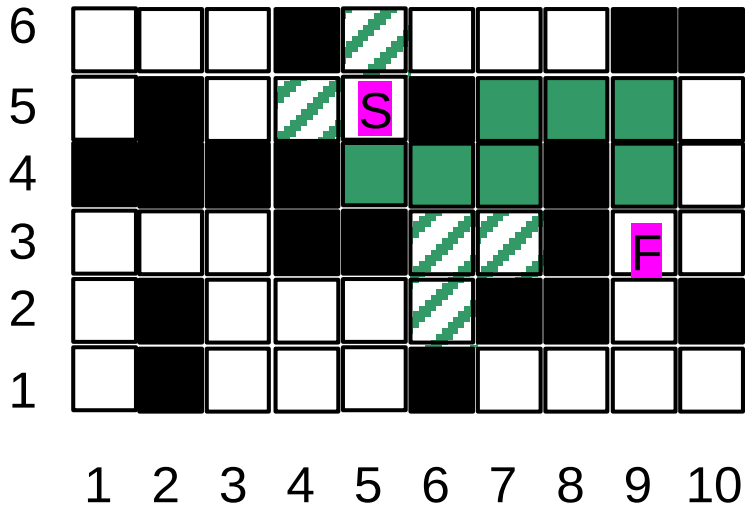
Ο αλγόριθμος BestFS (Ψευδοκώδικας)

```
algorithm bestfs(InitialState, FinalStates)
begin
  Closed←∅;
  EvaluatedInitialState←Heuristic(<InitialState>)
  Frontier←<EvaluatedInitialState>;
  CurrentState←best(Frontier);
  while CurrentState ∉ FinalStates do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet then
    begin
      Children ←Expand(CurrentState);
      EvaluatedChildren←Heuristic(Children);
      Frontier←Frontier ^ EvaluatedChildren;
      Closed←Closed∪{CurrentState};
    end;
    if Frontier= ∅ then return fail;
    CurrentState←best(Frontier);
  endwhile;
  return success;
end.
```

Ο αλγόριθμος BestFS

Σχόλια

- Πλεονεκτήματα:
  - Προσπαθεί να δώσει μια γρήγορη λύση σε κάποιο πρόβλημα. Το αν τα καταφέρει ή όχι εξαρτάται πολύ από τον ευριστικό μηχανισμό.
  - Είναι πλήρης.
- Μειονεκτήματα:
  - Το μέτωπο αναζήτησης μεγαλώνει με υψηλό ρυθμό και μαζί του ο χώρος που χρειάζεται για την αποθήκευσή του, καθώς και ο χρόνος για την επεξεργασία των στοιχείων του.
  - Δεν εγγυάται ότι η λύση που θα βρεθεί είναι η βέλτιστη.



Ο αλγόριθμος BestFS: το πρόβλημα του λαβύρινθου

Μέτωπο Αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά
<5-5>	<>	5-5	5-4^{5},5-6^{7},4-5^{7}
<5-4^{5},5-6^{7},4-5^{7}>	<5-5>	5-4	5-5^{6},6-4^{4}
<6-4^{4},5-5^{6},5-6^{7},4-5^{7}>	<5-5,5-4>	6-4	5-4^{7},6-3^{3},7-4^{3}
<6-3^{3},7-4^{3},5-5^{6},5-6^{7},...>	<5-5,5-4,6-4>	6-3	6-4^{4},6-2^{3},7-3^{2}
<7-3^{2},6-2^{3},7-4^{3},6-4^{4},5-5^{6},...>	<5-5,5-4,...>	7-3	6-3^{3},6-4^{4}
<6-3^{3},6-2^{3},7-4^{3},6-4^{4},5-5^{6},...>	<...,6-3,...>	6-3	Βρόχος
<6-2^{3},7-4^{3},6-4^{4},5-5^{6},5-6^{7},...>	<...>	6-2	5-2^{5},6-3^{3}
<7-4^{3},6-4^{4},5-2^{5},...>	<...>	7-4	7-5^{4},6-4^{4},7-3^{2}
<7-3^{2},7-5^{4},6-4^{4},5-2^{5},...>	<..,7-3,..>	7-3	Βρόχος
<4-5^{4},6-4^{4},5-2^{5},...>	<...>	7-5	7-4^{3},8-5^{3},7-6^{5}
<8-5^{3},7-4^{3},6-4^{4},...>	<...>	8-5	8-6^{4},7-5^{4},9-5^{2}
<9-5^{2},7-4^{3},6-4^{4},8-6^{4},...>	<...>	9-5	8-5^{3},9-4^{1}
<9-4^{1},8-5^{3},7-4^{3},...>	<...>	9-4	9-3^{0},9-5^{2},10-4^{2}
<9-3^{0},9-5^{2},10-4^{2},...>	<...>	9-3	ΤΕΛΙΚΗ ΚΑΤΑΣΤΑΣΗ
		ΤΕΛΟΣ	

- Η λύση στο παραπάνω πρόβλημα είναι η διαδρομή που ορίζεται από τη σειρά των θέσεων: 5-5→5-4→6-4→7-4→7-5→8-5→9-5→9-4→9-3

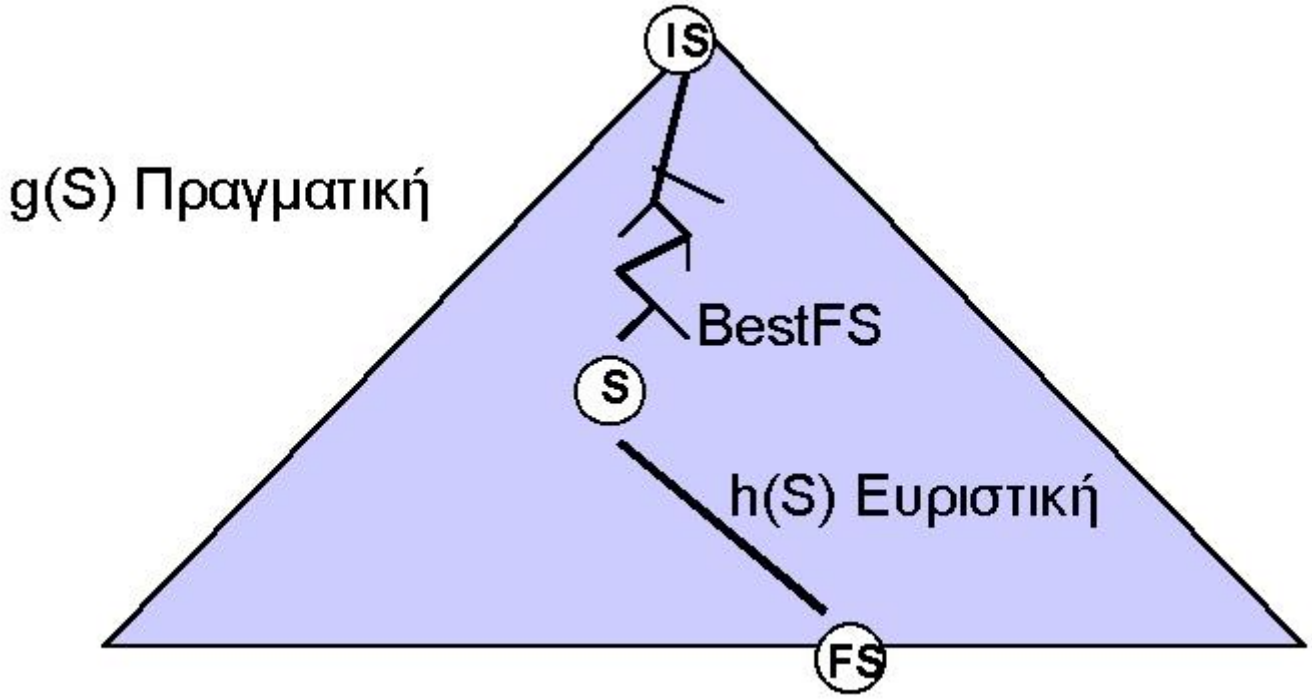
Ο Αλγόριθμος Άλφα-Άστρο (A*)

Ο *αλγόριθμος Α* (*Άλφα Άστρο*) είναι κατά βάσει BestFS, αλλά με ευριστική συνάρτηση:

F(S) = g(S) + h(S)

η g(S) δίνει την απόσταση της S από την αρχική κατάσταση, η οποία είναι πραγματική και γνωστή, και

η h(S) δίνει την εκτίμηση της απόστασης της S από την τελική κατάσταση μέσω μιας ευριστικής συνάρτησης, όπως ακριβώς στον BestFS.



Ο Αλγόριθμος Άλφα-Άστρο (A*)

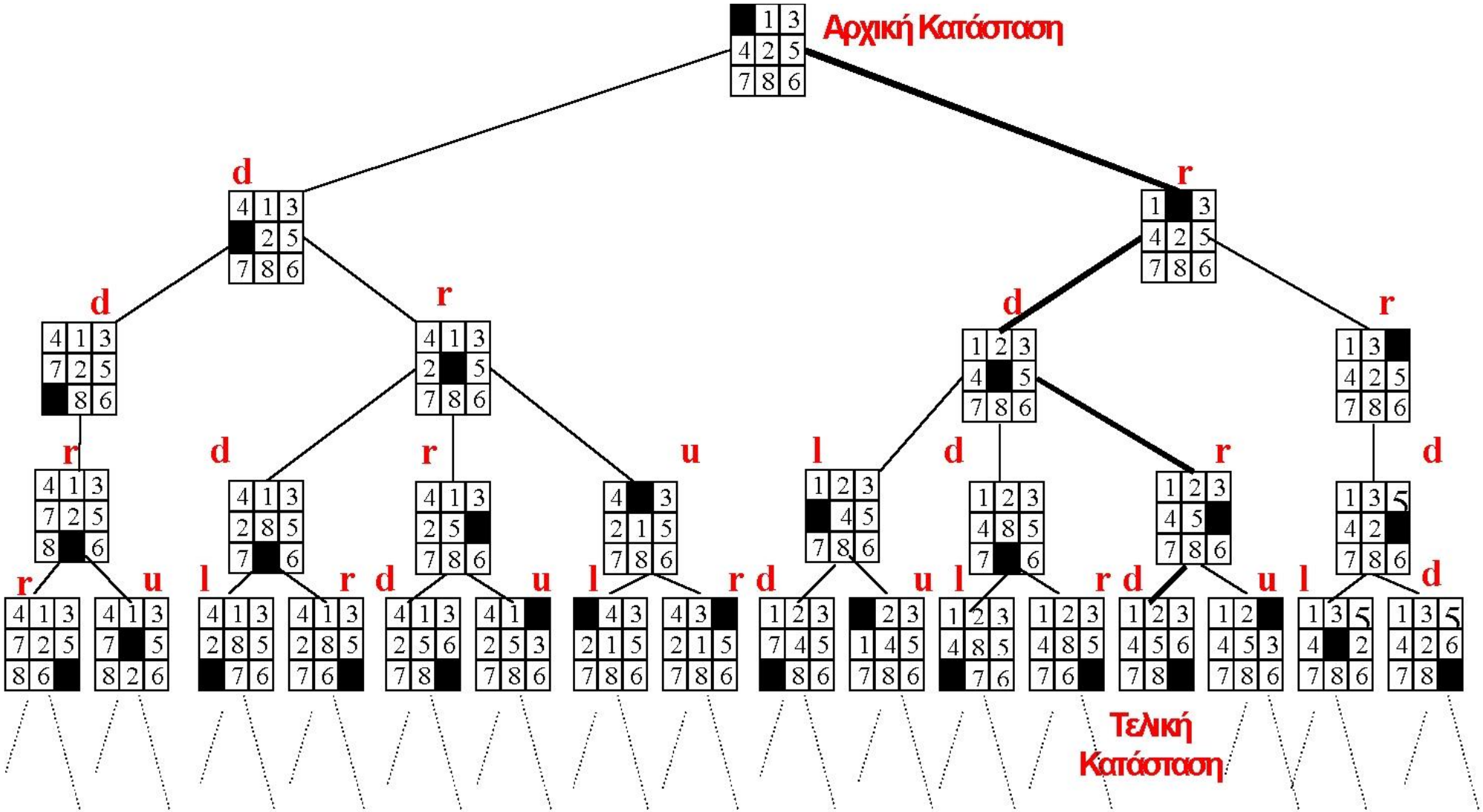
Σχόλια

- Αν για κάθε κατάσταση η τιμή h(S) είναι μικρότερη ή το πολύ ίση με την πραγματική απόσταση της S από την τελική κατάσταση, τότε ο A* βρίσκει πάντα τη βέλτιστη λύση. Στην περίπτωση αυτή, ο ευριστικός μηχανισμός ονομάζεται αποδεκτός (admissible) και ικανοποιεί το κριτήριο αποδοχής (admissibility criterion).

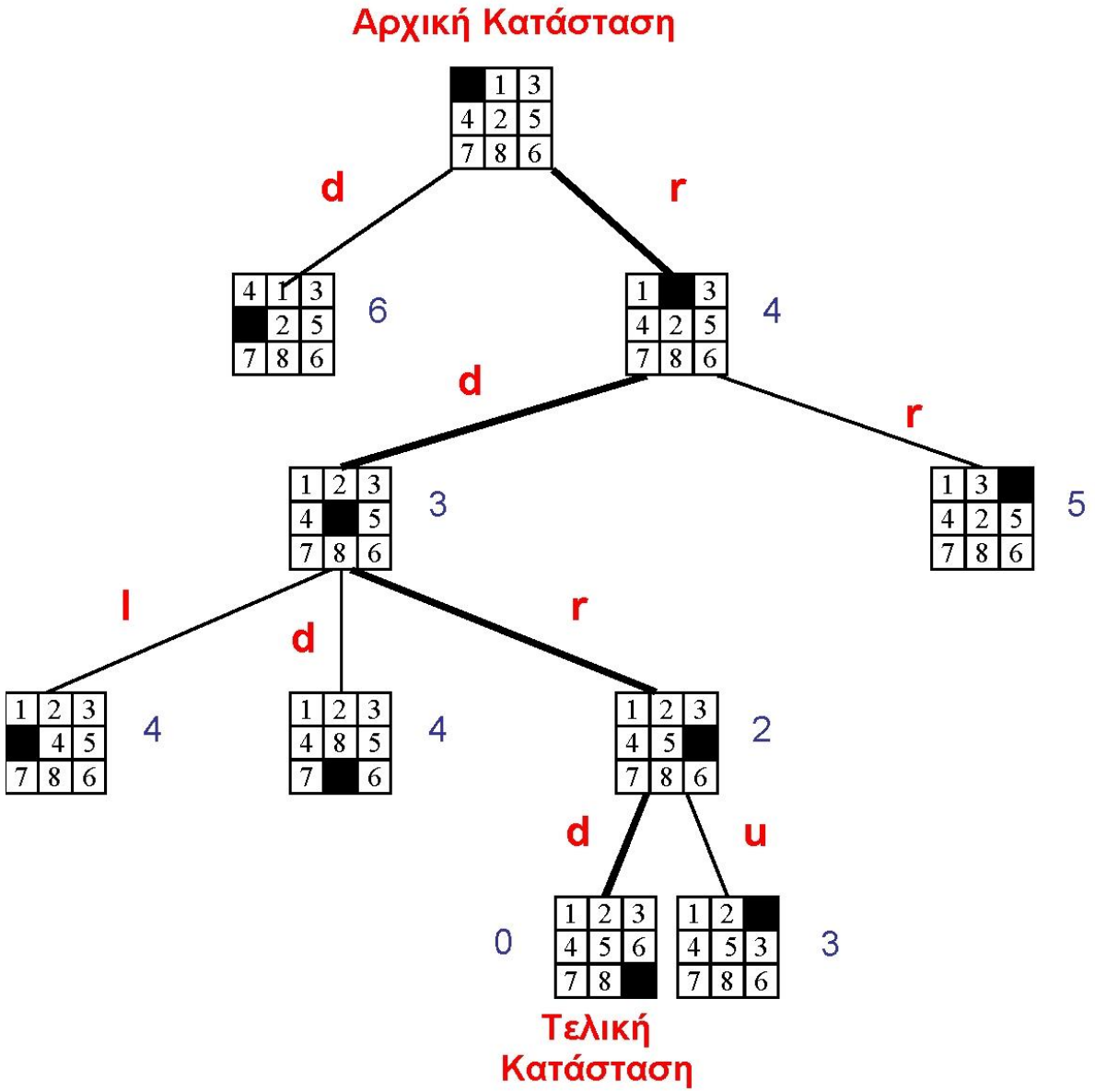
- Βελτιώσεις:
  - A* με επαναληπτική εκβάθυνση (Iterative Deepening A* - IDA)

Εφαρμογή των Αλγορίθμων Ευριστικής Αναζήτησης

Χώρος Αναζήτησης στο 8-puzzle

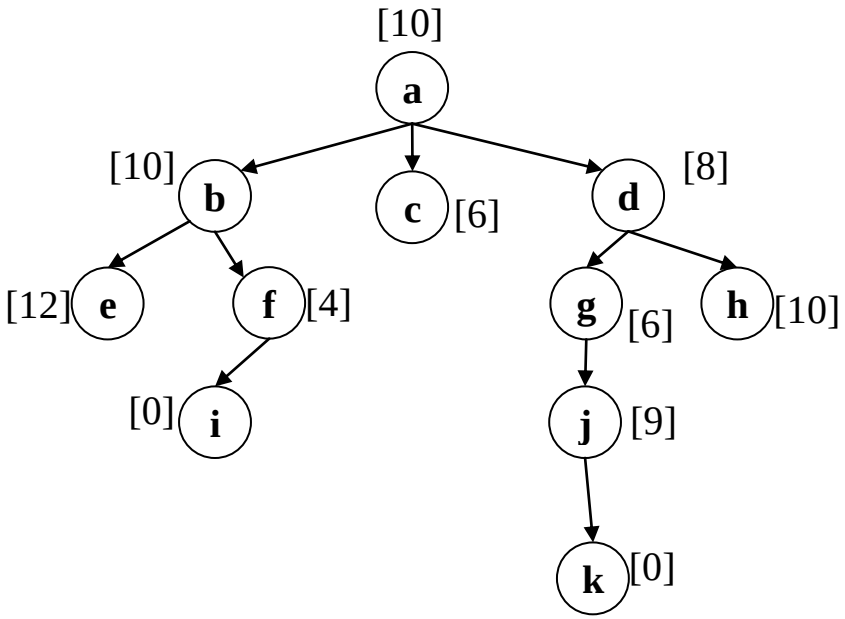


Εφαρμογή αλγορίθμου BestFS στο 8-puzzle



Άσκηση 4.1

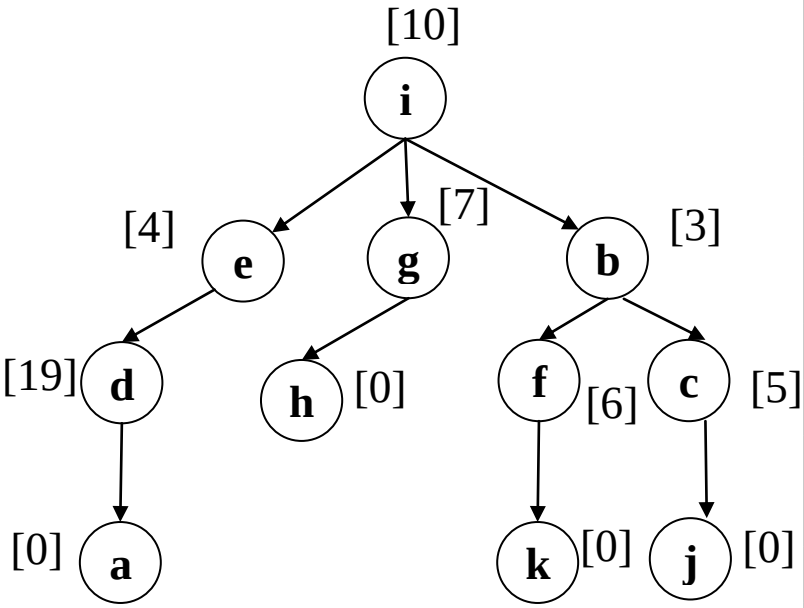
- Στο διπλανό σχήμα παρουσιάζεται ένα δένδρο αναζήτησης, όπου ο αριθμός μέσα σε αγκύλες δίπλα σε κάθε κόμβο αντιστοιχεί στην τιμή μιας ευριστικής συνάρτησης για αυτόν. Οι κόμβοι που έχουν τιμή 0, είναι οι τερματικοί κόμβοι της αναζήτησης.
- Γράψτε την σειρά με την οποία θα εξεταστούν οι κόμβοι του δέντρου (π.χ. a,b,c,…) μέχρι να βρεθεί τερματική κατάσταση από τους ακόλουθους αλγορίθμους:



- HC:……
- Best FS: ……
- A* : ……...

Άσκηση 4.2

- Στο διπλανό σχήμα παρουσιάζεται ένα δένδρο αναζήτησης, όπου ο αριθμός μέσα σε αγκύλες δίπλα σε κάθε κόμβο αντιστοιχεί στην τιμή μιας ευριστικής συνάρτησης για αυτόν. Οι κόμβοι που έχουν τιμή 0, είναι οι τερματικοί κόμβοι της αναζήτησης.
- Γράψτε την σειρά με την οποία θα εξεταστούν οι κόμβοι του δέντρου (π.χ. a,b,c,…) μέχρι να βρεθεί τερματική κατάσταση από τους ακόλουθους αλγορίθμους:



- HC:……....
- Best FS: ……
- A* : ……...

Άσκηση 4.3

- Να προτείνετε έναν ευριστικό μηχανισμό στο παρακάτω πρόβλημα puzzle και να εφαρμόσετε τον αλγόριθμο Best First Search για την εύρεση των επόμενων 2 βημάτων.

IS:

2	5	1
3	6	4
7		8

FS:

1	2	3
4	5	6
7	8	

.....

.....

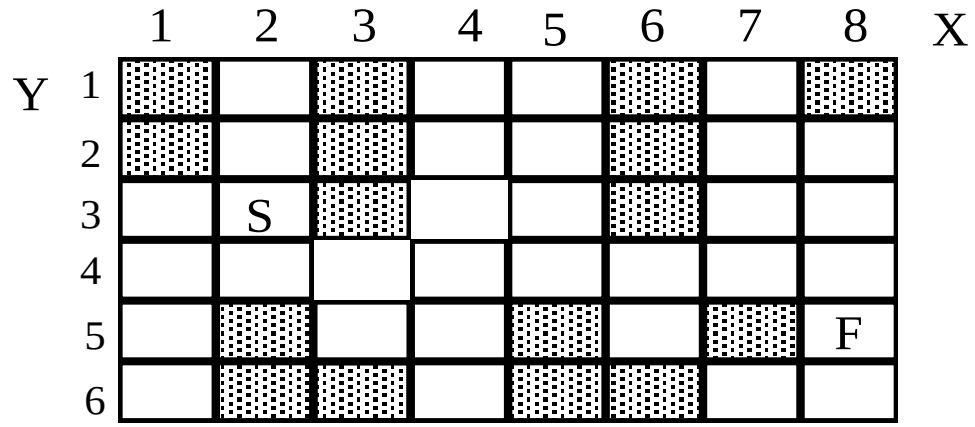
.....

.....

.....

Άσκηση 4.4

- Να εφαρμόσετε έναν ευριστικό μηχανισμό στο ακόλουθο πρόβλημα του λαβυρίνθου και να προτείνετε την επόμενη κίνηση από την αρχική θέση S (σημείωση: διαγώνια κίνηση δεν επιτρέπεται).



.....

.....

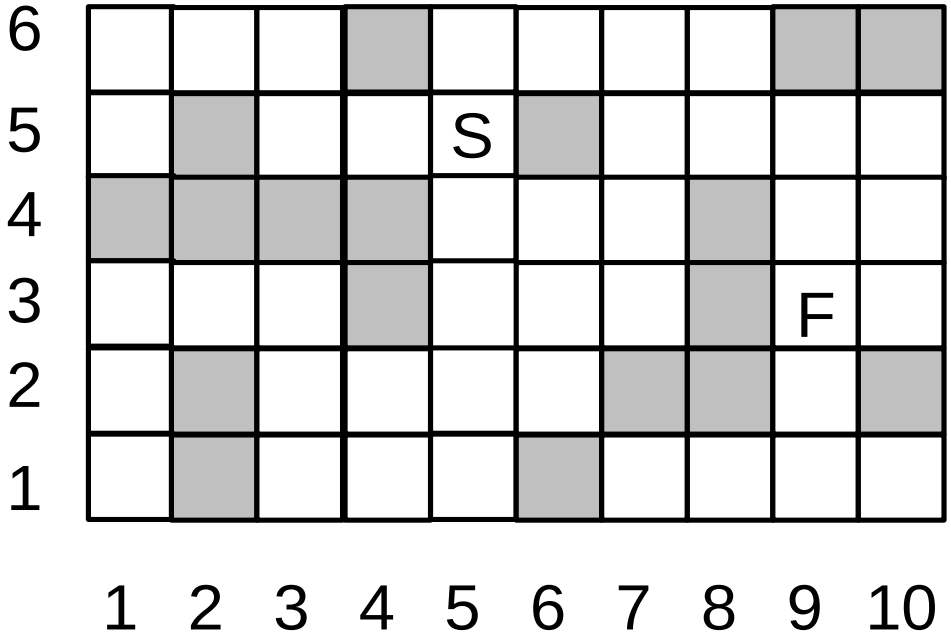
.....

.....

.....

Άσκηση 4.5

- Να εφαρμοστεί ο BestFS στο λαβύρινθο, στον οποίο ζητείται μία διαδρομή από το S στο F, χρησιμοποιώντας ως ευριστική συνάρτηση την απόσταση Manhattan.
- Να βρεθούν 2 βήματα και να συμπληρωθεί ο πίνακας, γράφοντας την τιμή της συνάρτησης ως εκθέτη.



Μέτωπο Αναζήτησης	Κλειστό Σύνολο	Κατάσταση	Παιδιά

Υλοποίηση Αλγορίθμων Ευριστικής Αναζήτησης σε Prolog

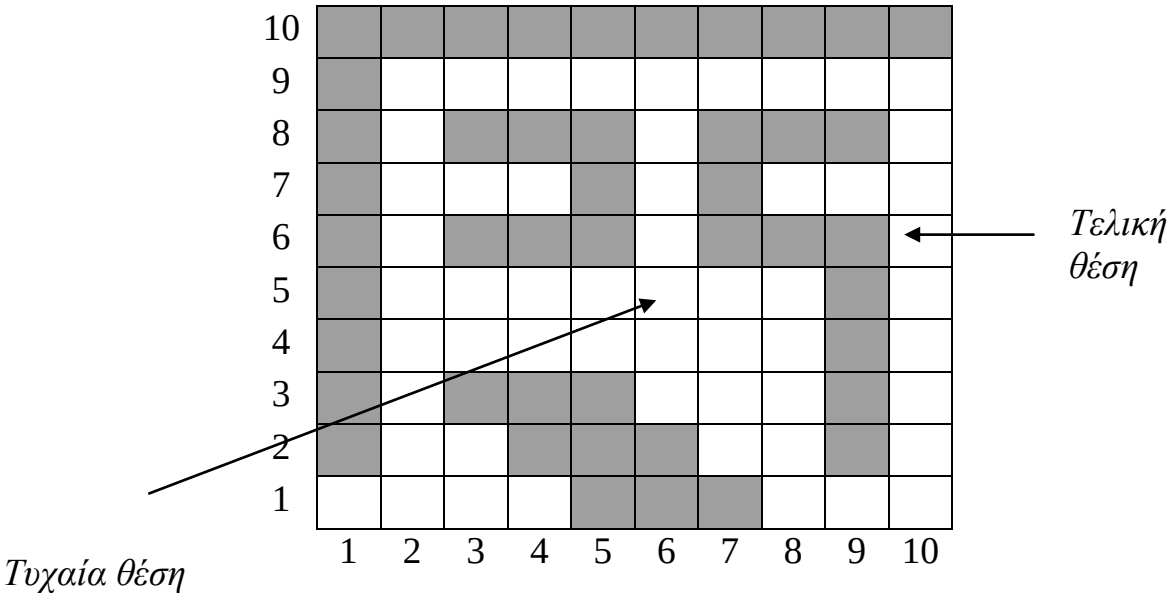
Αναρρίχηση Λόφων (Hill Climbing)

```
godhc(Solution):-
  initial_state(IS),
  goal(FS),
  heuristic(IS,FS,V),
  hc(IS,[V-IS],Solution1,FS),
  reverse(Solution1,Solution).

hc(FS,Solution,Solution,FS) :- !.
hc(State,PathSoFar,Solution,FS) :-
  % Βρες όλες τις καταστάσεις-παιδιά
  next_states(State,Children,FS),
  % Ταξινόμηση παιδιών και επιλογή του καλύτερου
  keysort(Children,[BestChild|_]),
  % Έλεγχος για βρόχο
  not member(BestChild,PathSoFar),
  hc(BestChild,[BestChild|PathSoFar],Solution,FS).

% Βρες όλες τις επόμενες καταστάσεις
% (με την ευριστική τιμή τους)
next_states(V-State,Children,FS):-
  findall(HV - Child,
      (operator(State,Child),
       heuristic(Child,FS,HV)),
      Children).
```

Δεν παράγει εναλλακτικές λύσεις!

Πιθανά σημεία αποτυχίας

Παράδειγμα ευριστικής συνάρτησης στον λαβύρινθο



- Ευκλείδια απόσταση

```
heuristic((X,Y),(XF,YF),HV) :-
  HV is sqrt((XF-X)^2+(YF-Y)^2)).
```

Αναζήτηση Best-First

```
go_best_first(Solution):-
  initial_state(IS),
  goal(FS),
  heuristic(IS,FS,V),
  best_first([V-[IS]],Solution1,FS),
  reverse(Solution1,Solution).

best_first([Value-[FinalState|Path]|_],[FinalState|Path],
                                          FinalState) :- !.
best_first([BestPath|RestPaths],Solution,FS):-
  next_states(BestPath,NewPaths,FS),
  append(NewPaths,RestPaths,Frontier),
  keysort(Frontier,OrderedFrontier),
  best_first(OrderedFrontier,Solution,FS).

next_states(V-[State|Path],NewPaths,FS):-
  findall(HV -[NewState,State|Path],
      (
      operator(State,NewState),
      heuristic(NewState,FS,HV)
      ),
    NewPaths).
```

3x3 Παζλ (Αναπαράσταση προβλήματος)

```
initial_state(([(1,1,2),(1,2,3),(1,3,6),
          (2,1,1),(2,2,5),(2,3,4),
          (3,1,7),(3,2,e),(3,3,8)]).
```

2	3	6
1	5	4
7		8

```
goal(([(1,1,1),(1,2,2),(1,3,3),
     (2,1,4),(2,2,5),(2,3,6),
     (3,1,7),(3,2,8),(3,3,e)]).
```

1	2	3
4	5	6
7	8	

```
% Τελεστής μετάβασης
operator(P,NP):-
  member((X,Y,e),P),      % Βρες το κενό τετράγωνο
  move(X,Y,X1,Y1),        % Μετακίνησέ το σε νέα θέση
  member((X1,Y1,T),P),    % Ποιο κομμάτι ήταν σε αυτή τη θέση;
  % Αντάλλαξε το κομμάτι με το κενό
  replace((X1,Y1,T),(X,Y,T),P,P1),
  replace((X,Y,e),(X1,Y1,e),P1,NP).

% Δημιουργία νέας θέσης XY του άδειου τεμαχίου
move(X,Y,X,NY):-NY is Y-1,NY>0.
move(X,Y,X,NY):-NY is Y+1,NY<4.
move(X,Y,NX,Y):-NX is X-1,NX>0.
move(X,Y,NX,Y):-NX is X+1,NX<4.

% replace(Old,New,OldList,NewList)
replace(_,_,[],[]).
replace(Old,New,[Old|T],[New|T]) :- !.
replace(Old,New,[H|T],[H|T1]) :-
  replace(Old,New,T,T1).
```

3x3 Παζλ (Ευριστική συνάρτηση)

```
% Η ευριστική συνάρτηση υπολογίζει πόσα πλακίδια
% βρίσκονται "εκτός" τελικής θέσης
heuristic([],_,0).
heuristic([(X,Y,T)|R],FinalState,V):-
  member((X,Y,T), FinalState), !,
  heuristic(R,FinalState,V).
heuristic([_|R],FinalState,V):-
  heuristic(R,FinalState,RV),
  V is RV+1.
```

2	3	6
1	5	4
7		8

Τιμή ευριστικής συνάρτησης = 7
(υπολογίζεται και η κενή θέση)

- Άλλη ευριστική συνάρτηση η οποία αθροίζει τη γεωμετρική απόσταση (**manhattan**) κάθε πλακιδίου από την τελική του θέση

```
heuristic([],_,0).
heuristic([(X,Y,T)|R],FinalState,V):-
  member((XF,YF,T),FinalState),
  Dis is abs(XF-X) + abs(YF-Y),
  heuristic(R,FinalState,RV),
  V is Dis + RV.
```

- Για **Ευκλείδια** απόσταση:

```
Dis is sqrt((XF-X)* (XF-X) + (YF-Y)*(YF-Y))
```

Επίλυση Προβλημάτων στην Prolog

- Χρησιμοποιούμε κάποιο αλγόριθμο αναζήτησης σε Prolog, ανάλογα με το πρόβλημα και το ζητούμενο (Τυφλός, ευριστικός, βέλτιση λύση, κλπ).
- Αναπαριστούμε το πρόβλημα. Ουσιαστικά ορίζουμε 5 κατηγορήματα:
- `initial_state(IS), goal(FS)`
  - ✓ Περιγράφουν την αρχική και τελική κατάσταση του προβλήματος, στη μορφή αναπαράστασης που έχουμε επιλέξει.
  - ✓ Η αρχική κατάσταση είναι πάντα μία συγκεκριμένη.
  - ✓ Η τελική κατάσταση πολλές φορές δεν είναι μία, αλλά πολλές.
    - Τότε ίσως χρειάζεται να την περιγράψουμε με πολλά γεγονότα.
      - Π.χ. ένας λαβύρινθος με πολλές εξόδους.
    - Μπορεί να χρειαστεί να περιγράψουμε την τελική κατάσταση αφηρημένα, με τη χρήση κανόνων οι οποίοι ελέγχουν τις ιδιότητες της τρέχουσας κατάστασης και τις συγκρίνουν με αυτές μιας τελικής κατάστασης.
      - Π.χ. η τερματική κατάσταση "ματ" στο σκάκι.
- `operator(State,Child)`
  - ✓ Περιγράφει τους τελεστές του προβλήματος, δηλαδή τον τρόπο με τον οποίο πάμε από μια κατάσταση σε μια άλλη.
  - ✓ Οι περιορισμοί του προβλήματος εκφράζονται μέσω των τελεστών.
  - ✓ Συνήθως υπάρχουν περισσότεροι του ενός κανόνες-τελεστές.
  - ✓ Για να περιγραφούν οι τελεστές μπορεί να χρειάζονται αρκετά βοηθητικά κατηγορήματα.
- `heuristic(CurrentState,FinalState,HeuristicValue)`

ή

- `heuristic(CurrentState,HeuristicValue)`
  - ✓ Υπολογίζει την τιμή της ευριστικής συνάρτησης για την τρέχουσα κατάσταση είτε σε σχέση με την τελική, ή χρησιμοποιώντας κάποιο άλλο κριτήριο.
  - ✓ Η ευριστική τιμή είναι συνήθως ένας αριθμός (πραγματικός ή ακέραιος).
  - ✓ Πάντα αντιστοιχεί μία μόνο τιμή σε κάθε κατάσταση.
  - ✓ Για να υπολογιστεί μπορεί να χρειάζονται βοηθητικά κατηγορήματα.