ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

6° Εξάμηνο

- Ενότητα 5 -

Προβλήματα Αναζήτησης

Δημοσθένης Σταμάτης
http:/www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

Περιεχόμενα

2

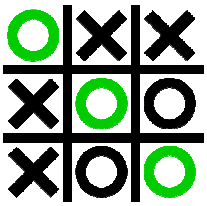
Γενική Περιγραφή Προβλήματος

- Για να γίνει δυνατή η επίλυση ενός **προβλήματος** στην Τεχνητή Νοημοσύνη (ΤΝ) απαιτείται ένας τυποποιημένος και σαφής ορισμός

- Θεωρούμε ότι ένα **πρόβλημα** μπορεί να οριστεί αν:
  - ✓ Υπάρχει μια δεδομένη **αρχική κατάσταση**
  - ✓ Υπάρχει μια επιθυμητή **τελική κατάσταση**
  - ✓ Είναι γνωστές κάποιες **ενέργειες** που πρέπει να γίνουν για να προκύψει η επιθυμητή κατάσταση

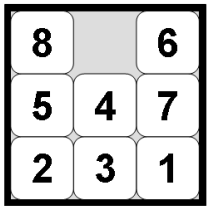
Ενδεικτικά Προβλήματα

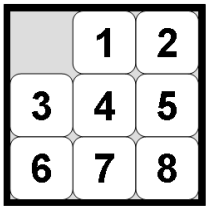
Τριάρα
(tic-tac-toe)



8–puzzle

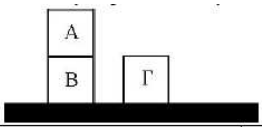
(N-puzzle)





Ενδεικτικά Προβλήματα

Ο κόσμος των κύβων



Αντικείμενα	Ιδιότητες	Σχέσεις
Κύβος Α	Κύβος Α είναι ελεύθερος	Κύβος Α πάνω στον κύβο Β
Κύβος Β	Κύβος Γ είναι ελεύθερος	Κύβος Β πάνω στο Τ
Κύβος Γ	Τ έχει αρκετό ελεύθερο χώρο	Κύβος Γ πάνω στο Τ
Τ είναι Τραπέζι	Κύβος Β δεν είναι ελεύθερος	

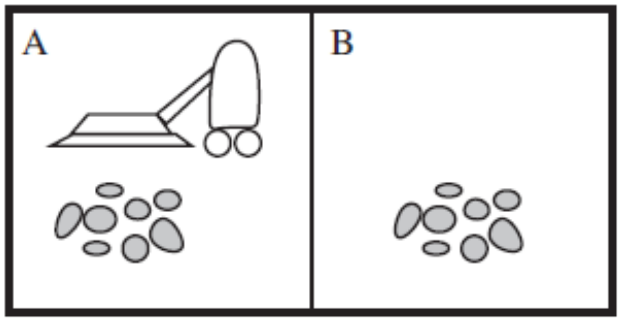
- Κόσμος του προβλήματος: Τρείς κύβοι και ένα τραπέζι.
- Μια Κατάσταση:

Κύβος Α πάνω στον κύβο Β
Κύβος Β πάνω στο Τ
Κύβος Γ πάνω στο Τ
Κύβος Α ελεύθερος
Κύβος Γ ελεύθερος

5

Ενδεικτικά Προβλήματα

Η σκούπα Robot που καθαρίζει τα δύο δωμάτια Α και Β



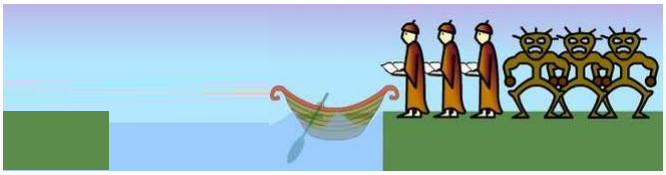
6

Ενδεικτικά Προβλήματα

Ιεραπόστολοι και Κανίβαλοι

Στην όχθη ενός ποταμού βρίσκονται 3 ιεραπόστολοι και 3 κανίβαλοι, οι οποίοι επιθυμούν να το διασχίσουν για να βρεθούν στην άλλη όχθη

- Στη διάθεσή του έχουν 1 βάρκα που χωράει το πολύ 2 άτομα
- Αν σε κάποια όχθη βρεθούν περισσότεροι κανίβαλοι τρώνε τους ιεραπόστολους !



Το Πρόβλημα του ταξιδεύοντος εμπόρου

Οδικός Χάρτης Β. Ελλάδας

Το Πρόβλημα του ταξιδεύοντος εμπόρου

Ένας έμπορος πρέπει να επισκεφτεί όλες τις πόλεις της Β. Ελλάδας κάνοντας τα λιγότερα δυνατά χιλιόμετρα

Γράφημα Πόλεων Αποστάσεις από το www.apostaseis.gr

9

Κόσμος ενός προβλήματος

- Ο **Κόσμος** ενός **προβλήματος** (***problem world***) ορίζεται από τα αντικείμενα (ή τις οντότητες) που τον αποτελούν, τις ιδιότητες των αντικειμένων και τις σχέσεις που τα συνδέουν.

- Ο **Κόσμος** ενός προβλήματος χαρακτηρίζεται **ως κλειστός** (***closed world***) όταν κανένα νέο αντικείμενο, ιδιότητα ή σχέση δεν μπορεί να προστεθεί ή να αφαιρεθεί. [είναι στατικός]

- Σε αντίθετη περίπτωση χαρακτηρίζεται **ως ανοιχτός** (***open world***).
  [Η περιγραφή του προβλήματος μπορεί να αλλάζει δυναμικά]

10

Ορισμός Προβλήματος με Χώρο Καταστάσεων (1)

- **Κατάσταση ενός κόσμου προβλήματος** είναι ένα **στιγμιότυπο (*instance*)** που παράγεται σε μία χρονική στιγμή κατά την εξέλιξη του κόσμου.
- Οι καταστάσεις ενός κόσμου **συνδέονται μεταξύ τους** με την έννοια ότι από μία κατάσταση, κατά την επόμενη χρονική στιγμή, μπορούν να προκύψουν μία ή περισσότερες καταστάσεις
- Θεωρούμε ότι μία κατάσταση προκύπτει από μία άλλη με την εφαρμογή ενός **τελεστή μετάβασης (*transition operator*)**

11

Ορισμός Προβλήματος με Χώρο Καταστάσεων (2)

Ένα πρόβλημα **P** ορίζεται ως μία τετράδα **P = { S, I, T, G}**, όπου:

- **S** είναι ο **χώρος καταστάσεων** (το σύνολο όλων των καταστάσεων)
- **I** είναι μία **αρχική κατάσταση** (ανήκει στο **S**)
- **T** είναι το σύνολο των **τελεστών μετάβασης**
- **G** είναι το σύνολο των **τελικών καταστάσεων** (υποσύνολο του **S**)

Λύση ενός προβλήματος **P = { S, I, T, G }** είναι μία ακολουθία από τελεστές μετάβασης

<t1, t2, t3, . . . ,tn>

ώστε να ισχύει:

g = tn(. . . t2(t1(I)) . . .)

όπου **g** μία τελική κατάσταση του συνόλου **G**

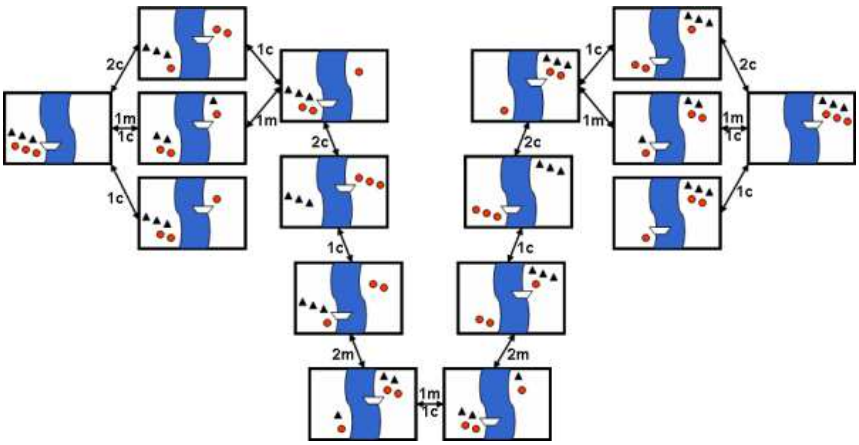
12

Χώρος Αναζήτησης ενός Προβλήματος

- Δοθέντος ενός προβλήματος **P = { S, I, T, G},**
  Χώρος Αναζήτησης (*Search Space*) SP είναι το σύνολο όλων των καταστάσεων που είναι **προσβάσιμες** από την αρχική κατάσταση **I**.

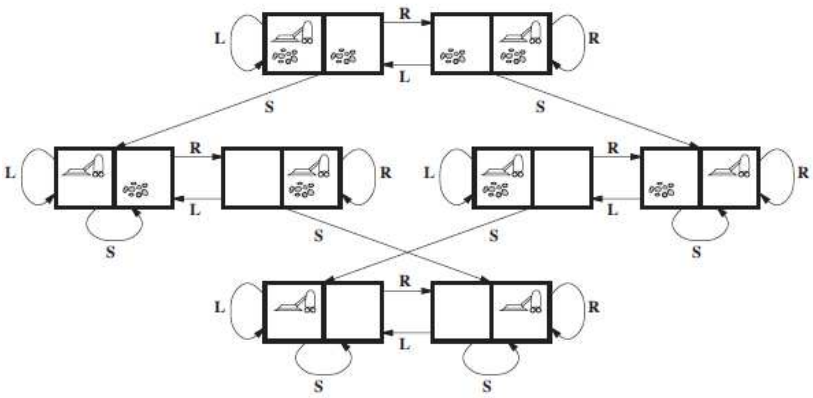
- Ο χώρος αναζήτησης είναι **υποσύνολο** του χώρου καταστάσεων, καθώς το σύνολο **SP** εξαρτάται από την αρχική κατάσταση **I** ενώ το **S** όχι.

Χώρος Καταστάσεων (1)



Ο Χώρος Καταστάσεων του προβλήματος «ιεραπόστολοι & κανίβαλοι

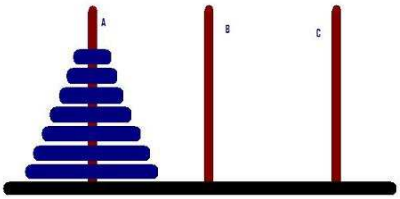
Χώρος Καταστάσεων (2)



Ο Χώρος Καταστάσεων του προβλήματος «της σκούπας Robot»

Περιγραφή προβλήματος με αναγωγή

- Μία ακολουθία από τελεστές **ανάγουν** συνεχώς τα προβλήματα σε άλλα «απλούστερα» έως ότου τα υπο-προβλήματα που προκύπτουν τελικά να είναι **άμεσα επιλύσιμα**.
- Παράδειγμα οι **Πύργοι του Hanoi**: Ένας αριθμός δίσκων σε φθίνουσα διάταξη βρίσκονται σε ένα στύλο και πρέπει να μεταφερθούν σε έναν άλλο με την ίδια διάταξη με τους εξής περιορισμούς:
  - ✓ Επιτρέπεται να μετακινείται ένας δίσκος τη φορά
  - ✓ Δεν επιτρέπεται η τοποθέτηση μεγαλύτερου δίσκου σε μικρότερο
  - ✓ Ένα τρίτος στύλος μπορεί να χρησιμοποιηθεί ως βοηθητικός

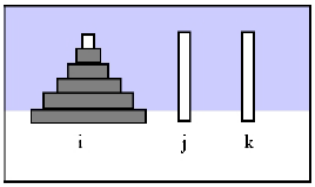


Περιγραφή προβλήματος με αναγωγή

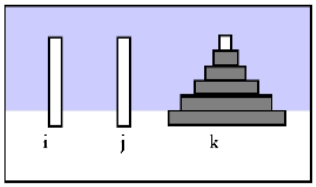
Η λύση του προβλήματος «Πύργοι του Hanoi»

Για να μεταφερθούν n>1 δίσκοι από τον στύλο i στο στύλο k, πρέπει:

- *να μεταφερθούν n-1 δίσκοι από το i στο j,*
- *να μεταφερθεί 1 δίσκος από το i στο k,*
- *να μεταφερθούν n-1 δίσκοι από το j στο k.*



Αρχική Περιγραφή



Τελική Περιγραφή

17

Αλγόριθμοι Αναζήτησης

- Υπάρχουν πολλά προβλήματα της ΤΝ για τα οποία δεν μπορεί να μας δοθεί ένας ειδικός αλγόριθμος για τη λύση τους, αλλά μόνο μία περιγραφή της λύσης. Στην περίπτωση αυτή πρέπει να ψάξουμε να βρούμε τη λύση!
- Πολλά προβλήματα της ΤΝ μπορούν να αναπαρασταθούν αφαιρετικά με τη βοήθεια ενός γραφήματος. Στην περίπτωση αυτή η επίλυση του προβλήματος ταυτίζεται με την αναζήτηση ενός μονοπατιού του γραφήματος
- Στις περισσότερες περιπτώσεις ξεκινούμε από έναν αρχικό κόμβο του γραφήματος και αναζητούμε το μονοπάτι που θα μας οδηγήσει σε έναν κόμβο στόχο

18

Αλγόριθμοι Αναζήτησης

Η γενική ιδέα (1):

- *Δοθέντος ενός γράφου, ενός αρχικού κόμβου και ενός κόμβου στόχου* :
  - Εξερευνούμε επαυξανόμενα μονοπάτια από τον αρχικό κόμβο
  - Διατηρούμε ένα σύνολο-μέτωπο (frontier set) από κόμβους που πρόκειται να εξερευνήσουμε άμεσα.
  - Καθώς η αναζήτηση προχωρά το σύνολο-μέτωπο επεκτείνεται μέχρις ότου βρεθούμε στον κόμβο στόχο.
- **Ο τρόπος που επεκτείνουμε και επεξεργαζόμαστε το σύνολο-μέτωπο καθορίζει τη στρατηγική της αναζήτησης**

19

Δέντρο Αναζήτησης (OR Δέντρο)

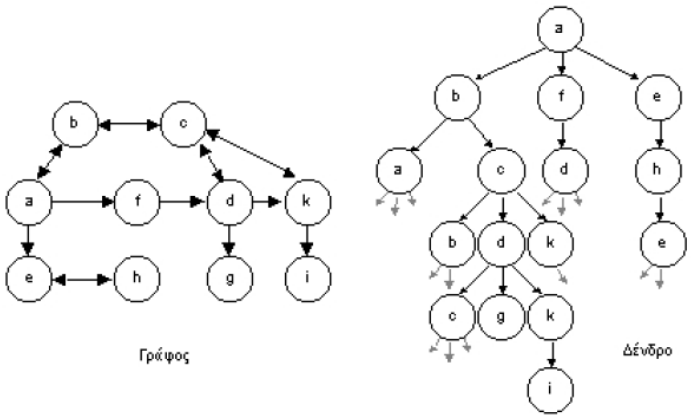
- *Για τις ανάγκες των αλγορίθμων ο χώρος αναζήτησης μπορεί να αναπαρασταθεί με τη βοήθεια ενός* ***OR Δέντρου***

Τμήμα Δένδρου	Αναπαράσταση
Κόμβος (Node)	Κατάσταση
Ρίζα (Root)	Αρχική Κατάσταση
Φύλλο (Tip, Leaf) ή Τερματικός κόμβος	Τελική Κατάσταση ή Αδιέξοδο (Dead Node), δηλαδή κατάσταση στην οποία δεν μπορεί να εφαρμοστεί κανένας τελεστής μετάβασης.
Κλαδί (Branch)	Τελεστής Μετάβασης που μετατρέπει μια κατάσταση-Γονέα (Parent State) σε μία άλλη κατάσταση-Παιδί (Child State).
Λύση (Solution)	Μονοπάτι (Path) που ενώνει την αρχική με μία τελική κατάσταση
Επέκταση (Expansion)	Η διαδικασία παραγωγής όλων των καταστάσεων-παιδιών ενός κόμβου.
Παράγοντας Διακλάδωσης (Branching Factor)	Ο αριθμός των καταστάσεων-Παιδιών που προκύπτουν από την επέκταση μιας κατάστασης. Επειδή δεν είναι σταθερός αριθμός, αναφέρεται και ως Μέσος Παράγοντας Διακλάδωσης (Average Branching Factor).

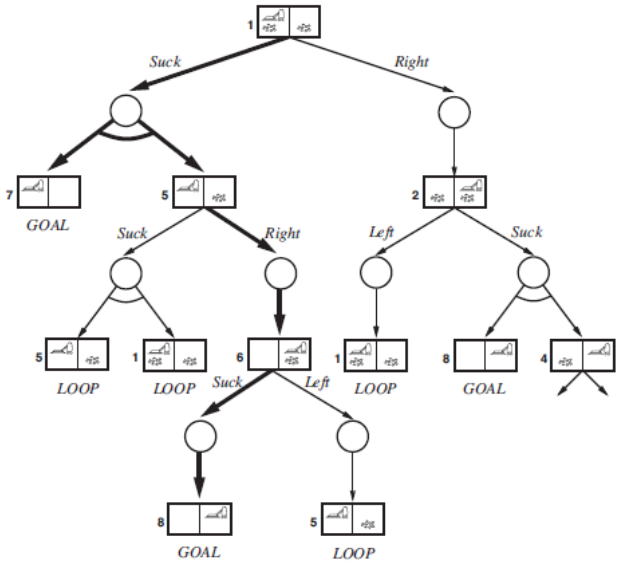
20

Δέντρο Αναζήτησης (OR Δέντρο)

- *Η μετατροπή ενός γράφου σε δέντρο αναζήτησης είναι πάντα εφικτή, εμπεριέχει όμως τον κίνδυνο το δέντρο να αποκτήσει κλαδιά με άπειρο μήκος* ***(συνδυαστική έκρηξη!)***

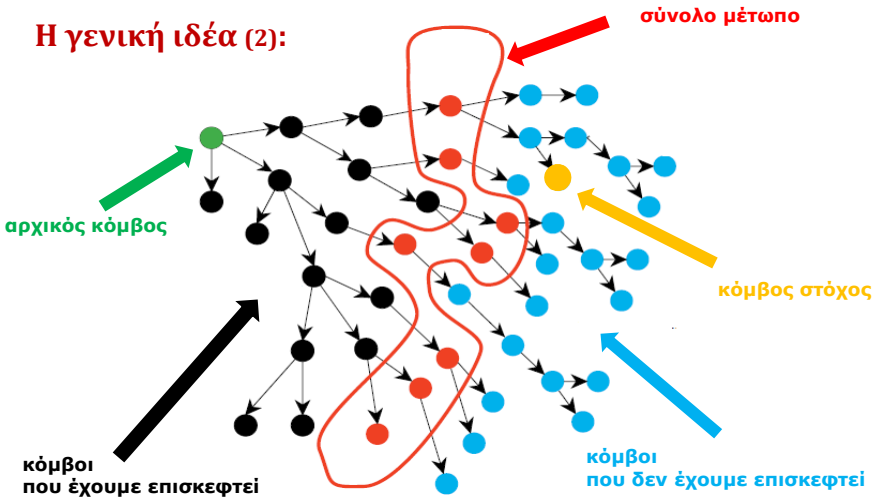


OR Δέντρο της Σκούπας Robot



Αλγόριθμοι Αναζήτησης

Η γενική ιδέα (2):



Γενικός Αλγόριθμος Αναζήτησης

1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης.
2. Αν το μέτωπο αναζήτησης είναι άδειο τότε σταμάτησε.
3. Πάρε την πρώτη σε σειρά κατάσταση του μετώπου της αναζήτησης.
4. Αν είναι η κατάσταση αυτή μέρος του κλειστού συνόλου τότε πήγαινε στο βήμα 2.
5. Αν είναι η κατάσταση αυτή τελική κατάσταση τότε τύπωσε τη λύση και πήγαινε στο βήμα 2.
6. Εφάρμοσε τους τελεστές μετάβασης για να παράγεις τις καταστάσεις-παιδιά.
7. Βάλε τις νέες καταστάσεις-παιδιά στο μέτωπο της αναζήτησης.
8. Κλάδεψε τις καταστάσεις που δε χρειάζονται (σύμφωνα με κάποιο κριτήριο), βγάζοντάς τες από το μέτωπο της αναζήτησης.
9. Κάνε αναδιάταξη στο μέτωπο της αναζήτησης (σύμφωνα με κάποιο κριτήριο).
10. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
11. Πήγαινε στο βήμα 2.

Γενικός Αλγόριθμος Αναζήτησης

```
algorithm general(InitialState, FinalState)
begin
  Closed←Ø;
  Frontier←<InitialState>;
  CurrentState←First(Frontier);
  while CurrentState ∉ FinalState do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet then
      begin
        Next←Expand(CurrentState);
        Frontier←insert(Next,Frontier);
        Frontier←prune(Frontier);
        Frontier←reorder(Frontier);
        Closed←Closed∪{CurrentState};
      end;
    if Frontier=Ø then return failure;
    CurrentState←First(Frontier);
  endwhile;
end.
```

25

Αλγόριθμοι Τυφλής Αναζήτησης

- *Οι αλγόριθμοι τυφλής αναζήτησης εφαρμόζονται όταν **δεν έχουμε επαρκείς πληροφορίες** που να μας επιτρέπουν την «**έξυπνη**» εκτίμηση των επόμενων καταστάσεων ενός προβλήματος.*

Όνομα Αλγορίθμου	Συντομογραφία	Ελληνική Ορολογία
Depth-First Search	DFS	Αναζήτηση Πρώτα σε Βάθος
Breadth-First Search	BFS	Αναζήτηση Πρώτα σε Πλάτος
Iterative Deepening	ID	Επαναληπτική Εκβάθυνση
Bi-directional Search	BiS	Αναζήτηση Διπλής Κατεύθυνσης
Branch and Bound	B&B	Επέκταση και Οριοθέτηση
Beam Search	BS	Ακτινωτή Αναζήτηση

26

Αλγόριθμος Πρώτα σε βάθος (dfs)

- Ο **αλγόριθμος πρώτα-σε-βάθος** επιλέγει να επεκτείνει το σύνολο-μέτωπο με την κατάσταση που βρίσκεται πιο βαθιά στο δέντρο αναζήτησης.

1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης.
2. Αν το μέτωπο της αναζήτησης είναι κενό τότε σταμάτησε.
3. Βγάλε την πρώτη κατάσταση από το μέτωπο της αναζήτησης.
4. Αν η κατάσταση ανήκει στο κλειστό σύνολο τότε πήγαινε στο βήμα 2.
5. Αν η κατάσταση είναι μία από τις τελικές, τότε ανέφερε τη λύση.
6. Αν θέλεις και άλλες λύσεις πήγαινε στο βήμα 2. Αλλιώς σταμάτησε.
7. Εφάρμοσε τους τελεστές μετάβασης για να βρεις τις καταστάσεις-παιδιά.
8. Βάλε τις καταστάσεις-παιδιά στην αρχή του μετώπου της αναζήτησης.
9. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
10. Πήγαινε στο βήμα 2.

Αλγόριθμος Πρώτα σε βάθος (dfs)

```
algorithm dfs(InitialState, FinalStates)
begin
  Closed←∅;
  Frontier←<InitialState>;
  CurrentState←First(Frontier);
  while CurrentState ∉ FinalStates do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet then
    begin
      ChildrenStates ←Expand(CurrentState);
      Frontier←ChildrenStates ^ Frontier;
      Closed←Closed∪{CurrentState};
    end;
    if Frontier=∅ then exit;
    CurrentState←First(Frontier);
  endwhile;
return success;
 end.
```

Αλγόριθμος Πρώτα σε πλάτος (bfs)

- Ο **αλγόριθμος πρώτα σε πλάτος** επιλέγει να επεκτείνει το σύνολο-μέτωπο με όλες τις καταστάσεις που βρίσκονται στο ίδιο βάθος και μετά προχώρα σε καταστάσεις μεγαλύτερου επιπέδου

1. Βάλε την αρχική κατάσταση στο μέτωπο της αναζήτησης.
2. Αν το μέτωπο της αναζήτησης είναι κενό τότε σταμάτησε.
3. Βγάλε την πρώτη κατάσταση από το μέτωπο της αναζήτησης.
4. Αν είναι η κατάσταση ανήκει στο κλειστό σύνολο τότε πήγαινε στο βήμα 2.
5. Αν η κατάσταση είναι μία τελική τότε ανέφερε τη λύση.
6. Αν θέλεις και άλλες λύσεις πήγαινε στο βήμα 2. Αλλιώς σταμάτησε.
7. Εφάρμοσε τους τελεστές μεταφοράς για να βρεις τις καταστάσεις-παιδιά.
8. Βάλε τις καταστάσεις-παιδιά στο τέλος του μετώπου της αναζήτησης.
9. Βάλε την κατάσταση-γονέα στο κλειστό σύνολο.
10. Πήγαινε στο βήμα 2.

Αλγόριθμος Πρώτα σε πλάτος

```
algorithm bfs(InitialState, FinalStates)
begin
Closed←∅;
Frontier←<InitialState>;
CurrentState←First(Frontier);
while CurrentState ∉ FinalStates do
    Frontier←delete(CurrentState,Frontier);
    if CurrentState ∉ ClosedSet
    begin
      ChildrenStates ←Expand(CurrentState);
      Frontier← Frontier ^ ChildrenStates;
      Closed←Closed∪{CurrentState};
    end;
  if Frontier= ∅ then exit;
  CurrentState←First(Frontier);
endwhile;
return success;
end.
```

Αλγόριθμοι Ευρετικής Αναζήτησης

- Μία ευρετική τεχνική βασίζεται σε κριτήρια με βάση τα οποία μπορούμε να επιλέξουμε ανάμεσα σε πολλές επιλογές την πιο αποτελεσματική, η οποία θα μας οδηγήσει στο στόχο μας. Συνήθως η ευρετική τεχνική υλοποιείται με μία ευρετική συνάρτηση.
- Μία ευρετική τεχνική:
  - Βελτιώνει την αποτελεσματικότητα της διαδικασίας αναζήτησης, ενδεχομένως θυσιάζοντας την πληρότητα των λύσεων
  - Δεν διασφαλίζει ότι θα βρούμε τη βέλτιστη λύση αλλά σχεδόν πάντα βρίσκει μία ικανοποιητική λύση

31

Αλγόριθμοι Ευρετικής Αναζήτησης

Hill Climbing	HC	Αναρρίχηση Λόφων
Best-First Search	BestFS	Αναζήτηση Πρώτα στο Καλύτερο
A* (A-star)	A*	A* (Άλφα Άστρο)

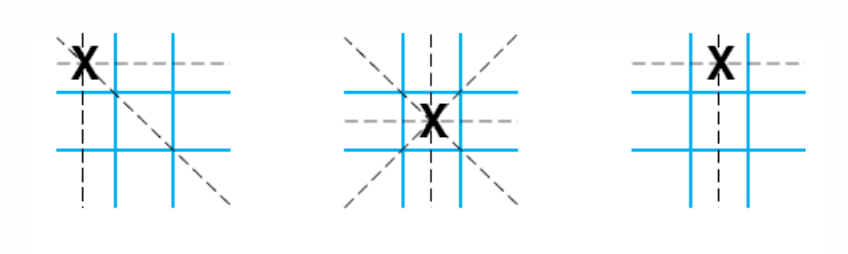
Αλγόριθμοι Αναζήτησης με αντίπαλο (π.χ. παιχνίδια 2 παικτών)

Minimax	Minimax	Αναζήτηση Μεγίστου-Ελαχίστου
Alpha-Beta	AB	Άλφα-Βήτα

32

Ευρετικές συναρτήσεις (1)

Παράδειγμα: Τριάρα (tic-tac-toe)



Η καλύτερη θέση είναι στο κέντρο γιατί πιθανολογούνται 4 περιπτώσεις για τη νίκη

Ευρετικές συναρτήσεις (2)

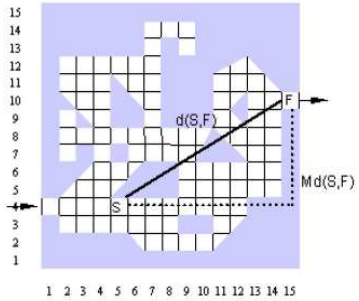
Παράδειγμα: Το πρόβλημα του λαβυρίνθου

- Ευκλείδεια απόσταση (Euclidian distance):

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

- Απόσταση Manhattan (Manhattan distance):

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



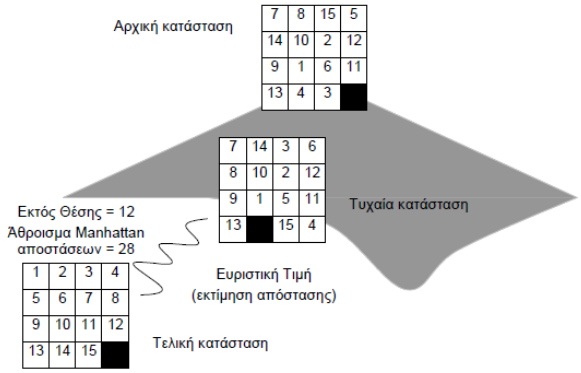
$$d(S,F) = \sqrt{(5-15)^2 + (4-10)^2} = \sqrt{(100+36)} = 11,6$$

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

Ευρετικές συναρτήσεις (3)

Παράδειγμα: Το 15-puzzle

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



Αλγόριθμος Πρώτα στο καλύτερο (bestfs)

- Ο **αλγόριθμος πρώτα στο καλύτερο** επεκτείνει το σύνολο-μέτωπο με βάση τον «καλύτερο» κόμβο, τον κόμβο δηλαδή που ήδη υπάρχει στο συνολο-μέτωπο και η ευρετική συνάρτηση του δίνει την καλύτερη τιμή.

```
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.
```

Αλγόριθμος A*

- Ο **αλγόριθμος A*** είναι ειδική περίπτωση του Bestfs, κατά την οποία η ευρετική συνάρτηση **F(S)** για κάθε κόμβο **S** ορίζεται σε συνδυασμό με την πραγματική απόσταση που έχει διανυθεί και την εκτίμηση της απόστασης μέχρι τον κόμβο στόχο:

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

- **g(S)** δίνει την πραγματική απόσταση του κόμβου **S** από τον αρχικό
- **h(S)** είναι η ευρετική συνάρτηση που δίνει την εκτίμηση της απόστασης του κόμβου **S** από τον κόμβο στόχο.

Web Links

State problems
http://www.plastelina.net/game2.html
http://www.plastelina.net/game1.html

Breadth-First Search (BFS)
Erik Demaine, MIT Open Courses
http://www.youtube.com/watch?v=s-CYnVz-uh4
http://www.youtube.com/watch?v=we2xFCPkH0Y

Depth-First Search (DFS), Topological Sort
Erik Demaine, MIT Open Courses
http://www.youtube.com/watch?v=AfSk24UTFS8

How Google makes improvements to its search algorithm
http://www.youtube.com/watch?v=J5RZOU6vK4Q

Shortest Path using Dijkstra's Algorithm
http://www.youtube.com/watch?v=WN3Rb9wVYDY

Computational Complexity
http://www.youtube.com/watch?v=moPtwq_cVH8