

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ



Τεχνητή Νοημοσύνη

ΑΛΓΟΡΙΘΜΟΙ ΑΝΑΖΗΤΗΣΗΣ ΛΥΣΗΣ

Γιώργος Στάμου

Τυπική διατύπωση προβλήματος

Κόσμος του προβλήματος: Τρεις κύβοι και ένα τραπέζι

Αντικείμενα	Ιδιότητες	Σχέσεις
Κύβος A	Κύβος Α ελεύθερος	Κύβος Α πάνω στον κύβο B
Κύβος B	Κύβος Γ ελεύθερος	Κύβος B πάνω στο T
Κύβος Γ	T έχει αρκετό ελεύθερο χώρο	Κύβος Γ πάνω στο T
Τραπέζι T	Κύβος B δεν είναι ελεύθερος	

Κατάσταση:

Κύβος Α πάνω στον κύβο B
Κύβος B πάνω στο T
Κύβος Γ πάνω στο T
Κύβος Α ελεύθερος
Κύβος Γ ελεύθερος

Τυπική διατύπωση προβλήματος

Σε κάθε πρόβλημα αρχίζουμε από κάπου και καταλήγουμε κάπου

Αρχική και Τελική Κατάσταση στο 8 Puzzle πρόβλημα

8	3	5
4	1	7
2	6	

Τυπικός ορισμός προβλήματος

$P = (I, G, T, S)$

S είναι ο χώρος καταστάσεων
I είναι η αρχική κατάσταση, $I \in S$
G είναι το σύνολο των τελικών καταστάσεων (στόχων), $G \subseteq S$
T είναι το σύνολο των τελεστών μετάβασης, $T: S \rightarrow S$

Λύση (solution) σε ένα πρόβλημα (I, G, T, S) , είναι μία ακολουθία t_1, t_2, \dots, t_n από τελεστές μετάβασης, τέτοια ώστε

$$g = t_n \dots (t_2(t_1(I))) \dots$$

όπου $g \in G$

Τυπική διατύπωση προβλήματος

Κόσμος προβλήματος

- Ένα σύνολο από αντικείμενα, οι ιδιότητές τους και οι σχέσεις που τα συνδέουν (αποτελεί υποσύνολο του πραγματικού κόσμου, αφαίρεση των στοιχείων και των λεπτομερειών που δεν εμπλέκονται στην επίλυση του προβλήματος)

Κατάσταση του κόσμου

- Είναι μία επαρκής, τυπική αναπαράσταση του κόσμου σε μία δεδομένη χρονική στιγμή

Πρόβλημα και επίλυση

- Δεδομένης μίας αρχικής κατάστασης και μίας επιθυμητής τελικής κατάστασης, ζητείται μία ακολουθία από ενέργειες που πρέπει να γίνουν ώστε να φτάσουμε από την αρχική στην τελική κατάσταση

Τυπική διατύπωση προβλήματος

Τελεστής μετάβασης (transition operator) ή ενέργεια (action) είναι μια αντιστοίχιση μίας κατάστασης του κόσμου σε μία άλλη

- Στον κόσμο των κύβων, τελεστές μετάβασης είναι πχ.:
 - Bάλε τον κύβο A πάνω στον κύβο Γ
 - Bάλε τον κύβο A πάνω στον κύβο B
- Επειδή είναι δύσκολο να υπάρχει ένας τελεστής για κάθε περίπτωση, μπορούμε να χρησιμοποιούμε μεταβλητές, πχ.:
 - Bάλε κάποιον κύβο X πάνω σε κάποιον κύβο Y
- Επειδή όλες οι μεταβάσεις δεν είναι δυνατές υπάρχουν προϋποθέσεις εφαρμογής (preconditions) που πρέπει να τηρούνται για να εφαρμοστεί ένας τελεστής.
 - Ο κύβος B δεν μπορεί να πάει επάνω στον κύβο Y

Χώρος αναζήτησης λύσης

Διθέντος ενός προβλήματος (I, G, T, S)

Χώρος αναζήτησης (search space) SP είναι το σύνολο όλων των καταστάσεων που είναι προσβάσιμες από την αρχική κατάσταση

Mία κατάσταση σ ονομάζεται προσβάσιμη (accessible) αν υπάρχει μια ακολουθία τελεστών μετάβασης t_1, t_2, \dots, t_k τέτοια ώστε $s = t_k \dots (t_2(t_1(I))) \dots$

Ο χώρος αναζήτησης είναι υποσύνολο του χώρου καταστάσεων, δηλαδή $SP \subseteq S$

Ο χώρος αναζήτησης μπορεί γενικά να αναπαρασταθεί με γράφο που συνήθως μετατρέπεται σε δένδρο αναζήτησης (OR-δένδρο) (με μονοπάτια πιθανά απείρου μήκους αν ο γράφος έχει κύκλο)

Δένδρα και χαρακτηριστικά προβλήματος



Τμήμα Δένδρου	Αναπαριστά
Κόμβος (Node)	Κατάσταση
Ρίζα (Root)	Αρχική Κατάσταση
Φύλλο (Tip, Leaf) ή Τερματικός κόμβος	Τελική Κατάσταση ή Αθίεξδο (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)	Διαδικασία παραγωγής όλων των καταστάσεων-Παιδιών ενός κόμβου
Παράγοντας Διακλάδωσης (Branching Factor)	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)]

Αλγόριθμοι αναζήτησης λύσης



Ο αλγόριθμος επίλυσης είναι μία αυστηρά καθορισμένη ακολουθία εύρεσης ενός κόμβου στόχου

- ▶ Ονομάζεται εξαντλητικός (exhaustive) όταν το σύνολο των καταστάσεων που εξετάζει ο αλγόριθμος για να βρει τις απαιτούμενες λύσεις είναι ίσο με το χώρο αναζήτησης
- ▶ Ένας αλγόριθμος δεν λύνει πάντα το πρόβλημα, έστω και αν υπάρχει λύση. Ονομάζεται πλήρης (complete) αν εγγυάται ότι θα βρει μία λύση για οποιαδήποτε τελική κατάσταση, αν τέτοια λύση υπάρχει. Σε αντίθετη περίπτωση, ο αλγόριθμος ονομάζεται μη-πλήρης (incomplete).
- ▶ Μία λύση ονομάζεται βέλτιστη (optimal) αν οδηγεί στην καλύτερη, σύμφωνα με τη διάταξη, τελική κατάσταση. Όταν δεν υπάρχει διάταξη, μία λύση ονομάζεται βέλτιστη αν είναι η συντομότερη (shortest).

Αλγόριθμοι αναζήτησης λύσης

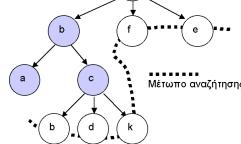


Βασικές διομέδες δεδομένων

- ▶ **Μέτωπο αναζήτησης (search frontier)**
Το διατεταγμένο σύνολο (λίστα) των καταστάσεων που ο αλγόριθμος έχει ήδη επισκεφτεί, αλλά δεν έχουν ακόμη επεκταθεί
- ▶ **Κλειστό σύνολο (closed set)**
Το σύνολο όλων των καταστάσεων που έχουν ήδη επεκταθεί από τον αλγόριθμο

Κάθε επέκταση μιας κατάστασης συνοδεύεται από την εισαγωγή της κατάστασης γονέα στο κλειστό σύνολο και των καταστάσεων παιδιών στο μέτωπο αναζήτησης

Με έναν απλό έλεγχο, αν η κατάσταση προς επέκταση ανήκει ήδη στο κλειστό σύνολο, αποφεύγονται οι βρόχοι (loops)



Αλγόριθμοι αναζήτησης λύσης



10

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), και διέγραψε τις 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.

Αλγόριθμοι αναζήτησης λύσης



Αλγόριθμοι τυφλής αναζήτησης

- ▶ Δεν έχουμε κάποια μέθοδο να εκτιμήσουμε το κόστος των εναλλακτικών για το επόμενο βήμα (σε σχέση με τον τελικό στόχο), δηλαδή ψάχνουμε να λάβουμε υπόψην αυτή την εκτίμηση κατά την αναζήτηση

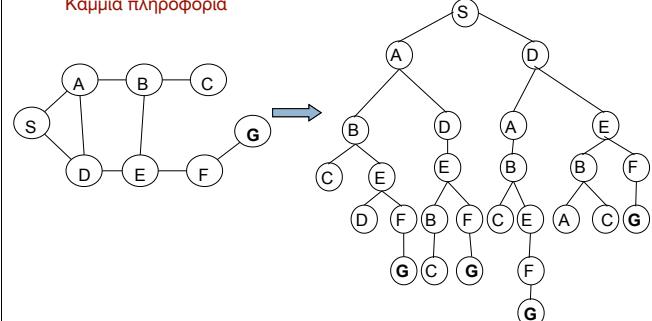
Αλγόριθμοι πληροφορημένης αναζήτησης

- ▶ Μοντελοποιούμε την απόσταση κάθε κόμβου από τον στόχο, με βάση κάποια ευριστική μέθοδο, οπότε μπορούμε να λάβουμε υπόψην αυτή την εκτίμηση κατά την αναζήτηση
- ▶ Η ευριστική τιμή δεν είναι η πραγματική τιμή της απόστασης από μία κατάσταση στόχο, αλλά μία εκτίμηση (που πολλές φορές μπορεί να είναι και λανθασμένη)

Αλγόριθμοι αναζήτησης - Πληροφορίες



Καμμία πληροφορία

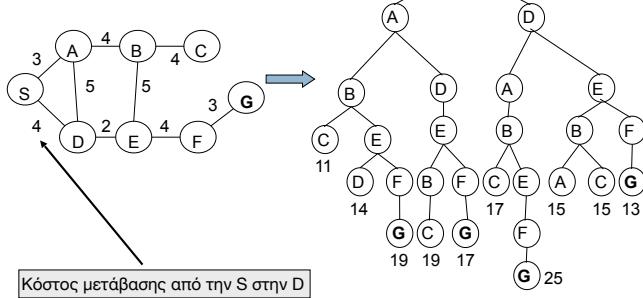


Αλγόριθμοι αναζήτησης - Πληροφορίες



13

Εκτίμηση κόστους μετάβασης από κατάσταση σε κατάσταση

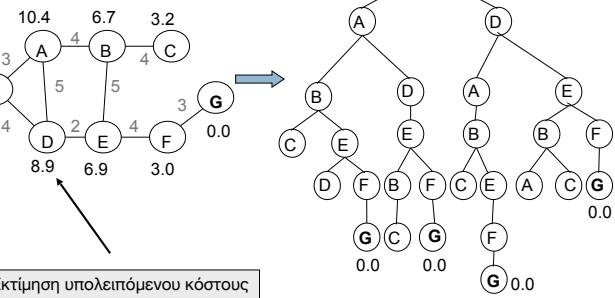


Αλγόριθμοι αναζήτησης - Πληροφορίες



14

Εκτίμηση κόστους μετάβασης από μία κατάσταση στο στόχο

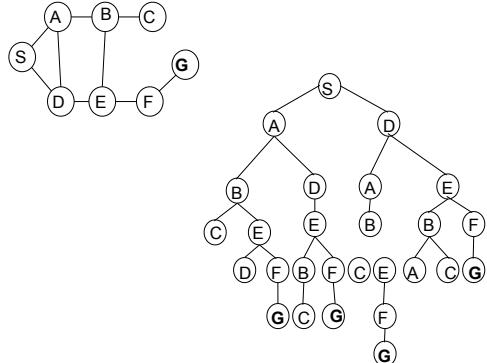


Αλγόριθμοι τυφλής αναζήτησης

Αλγόριθμος αναζήτησης κατά βάθος (DFS)



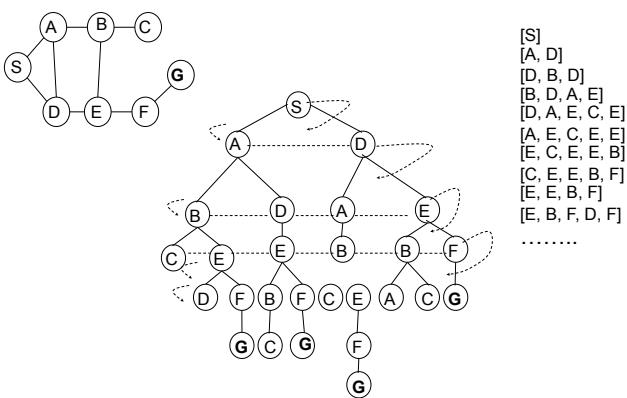
16



Αλγόριθμος αναζήτησης κατά πλάτος (BFS)



17



Αλγόριθμος επαναληπτικής εκβάθυνσης



18

Βήματα Αλγόριθμου (Iterative Deepening - ID)

Βήμα 1. Όριστε το αρχικό βάθος αναζήτησης (συνήθως 1).

Βήμα 2. Εφάρμοσε τον αλγόριθμο Κατά-Βάθος μέχρι αυτό το βάθος αναζήτησης.

Βήμα 3. Αν έχεις βρει λύση σταμάτησε.

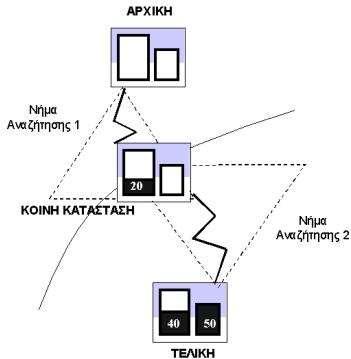
Βήμα 4. Αύξησε το βάθος αναζήτησης (συνήθως κατά 1).

Βήμα 5. Πήγαινε στο Βήμα 2.

Αποδεικνύεται ότι ο ID έχει την ίδια πολυπλοκότητα σε χώρο και χρόνο με τους DFS και BFS, όταν έχουμε μεγάλους χώρους αναζήτησης, παρόλο που επαναλαμβάνει άσκοπα το κτίσμα του χώρου αναζήτησης.

Αλγόριθμος διπλής κατεύθυνσης

19



- Αρχίζουμε την αναζήτηση από την αρχική και τελική κατάσταση ταυτόχρονα
- Αν κάποια κατάσταση που επεκτείνεται είναι κοινή και από τις 2 πλευρές, τότε βρέθηκε λύση
- Λύση είναι η ένωση των μονοπατιών από την κοινή κατάσταση ως την αρχική και ως την τελική κατάσταση

Αλγόριθμοι πληροφορημένης αναζήτησης (αναζήτησης μίας οποιασδήποτε λύσης)

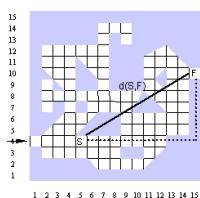
Ευριστικοί μηχανισμοί

21

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):
 $Md(S, F) = |XS - XF| + |YS - YF|$



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

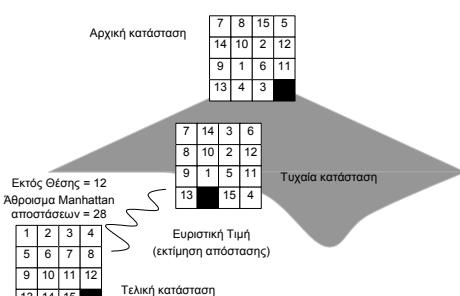
$$Md(S, F) = |5-15| + |4-10| = 10 + 6 = 16$$

Ευριστικοί μηχανισμοί

22

Ευριστικός μηχανισμός και συναρτήσεις στο N-Puzzle

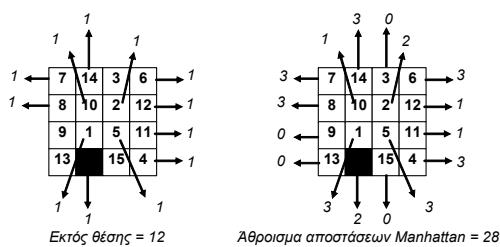
- Πόσα πλακίδια βρίσκονται εκτός θέσης
- Το άθροισμα των αποστάσεων Manhattan κάθε πλακιδίου από την τελική του θέση



Ευριστικοί μηχανισμοί

23

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



Αλγόριθμος Hill climbing

24

Βήμα 1: Όρισε τον τρέχοντα κόμβο ως τη ρίζα του δένδρου

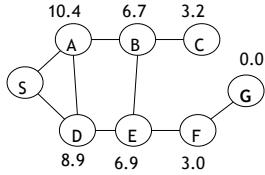
Βήμα 2: Μέχρι που ο τρέχων κόμβος δεν είναι κόμβος στόχος, εκτέλεσε:

- **Βήμα 2.a:** Βρες τα παιδιά του τρέχοντος κόμβου, και στη συνέχεια βρες αυτό με την ελάχιστη υπολογιζόμενη υπόλοιπη απόσταση από το στόχο
- **Βήμα 2.b:** Εάν ο τρέχων κόμβος δεν έχει παιδιά ή το παιδί που βρέθηκε στο Βήμα 2.a έχει μικρότερη τιμή ευριστικής συνάρτησης πηγαίνει στο Βήμα 3.
- **Βήμα 2.y:** Όρισε τον κόμβο που βρέθηκε στο Βήμα 2.a ως τρέχων κόμβο.

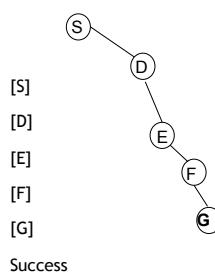
Βήμα 3: Εάν βρήκαμε ένα κόμβο στόχο τότε ανακοινώνουμε επιτυχία αλλιώς ανακοινώνουμε αποτυχία

Αλγόριθμος Hill climbing

25



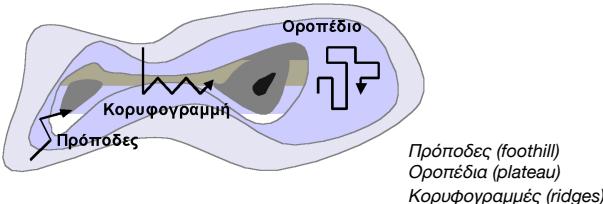
Υποθέτουμε εδώ ότι η υπολογιζόμενη υπολειπόμενη απόσταση από τον κόμβο A στον Στόχο G είναι: 10.4 από τον κόμβο D στον Στόχο G είναι: 8.9 από τον κόμβο E στον Στόχο G είναι: 6.9 από τον κόμβο F στον Στόχο G είναι: 3.0



Αλγόριθμος Hill Climbing

26

Προβλήματα



Βελτιώσεις

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

Αλγόριθμος Hill Climbing

26

Διαισθητικά

- ▶ Προσπαθούμε να βρούμε κάποιο σημείο όπου όλα τα σημεία που μπορούμε να πάμε από αυτό το σημείο με ένα βήμα έχουν χειρότερα χαρακτηριστικά από αυτό. Π.χ.:
 - ▶ Όταν μπαίνουμε σε ένα δωμάτιο και αρχίζουμε να ρυθμίζουμε το θερμοστάτη θέρμανσης/ψύξης μέχρι να βρούμε κάποιο σημείο που αισθανόμαστε άνετα.
 - ▶ Δεν υπάρχει σημείο «στόχος», αλλά σταματάμε σε κάποιο σημείο (θέση του θερμοστάτη) όπου όλες οι άλλες γειτονικές ρυθμίσεις δεν μας δίνουν καλύτερα αποτελέσματα.
 - ▶ Δηλαδή, προσπαθούμε να βρούμε μια λύση (σημείο στο χώρο καταστάσεων) όπου έχει βελτιστοποιηθεί η τιμή μιας παραμέτρου του προβλήματος (π.χ. πόσο άνετα αισθανόμαστε) ή ένας συνδυασμός τιμών παραμέτρων του προβλήματος.

Αλγόριθμος Beam Search

29

Βήμα 1: Κατασκεύασε μια λίστα ουρά που περιέχει τη ρίζα του δένδρου (αρχική κατάσταση)

Βήμα 2: Μέχρι που η λίστα να αδειάσει ή να βρεθεί ένας τελικός κόμβος στόχος

- **Βήμα 2.α:** Εάν ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος μη κάνει τίποτε
- **Βήμα 2.β:** Εάν ο πρώτος κόμβος στη λίστα δεν είναι κόμβος στόχος, τότε βγάλε το πρώτο κόμβο από τη λίστα, και βάλε στο τέλος της λίστας τα w καλύτερα παιδιά του κόμβου, όσον αφορά την τιμή της ευριστικής απόστασης του κόμβου από ένα κόμβο «στόχο»
- **Βήμα 2.γ:** Εάν το πρώτος κόμβος δεν έχει παιδιά, αφαιρέσε τον από τη λίστα και πήγαινε στο βήμα 2

Βήμα 3: Εάν βρήκαμε ένα κόμβο στόχο τότε ανακοινώνουμε επιτυχία, αλλιώς ανακοινώνουμε αποτυχία

Αλγόριθμος Beam Search

28

- ▶ Ο αλγόριθμος ακτινωτής αναζήτησης (Beam Search - BS) είναι παρόμοιος με τον αλγόριθμο αναζήτησης Κατά-Πλάτος
- ▶ Προχωράμε στην αναζήτηση ανά επίπεδο του δένδρου αναζήτησης με τη διαφορά ότι δεν κοιτάμε όλους τους κόμβους του επόμενου επιπέδου, αλλά μόνο τους w καλύτερους κόμβους όσον αφορά στη βαθμολόγησή τους σε σχέση με την υπολειπόμενη απόσταση από κάποιο κόμβο «στόχο»
- ▶ Δηλαδή σε κάθε βήμα κρατάμε τους w καλύτερους κόμβους στο μέτωπο αναζήτησης

Αλγόριθμος Beam Search

29

Βήμα 1: Κατασκεύασε μια λίστα ουρά που περιέχει τη ρίζα του δένδρου (αρχική κατάσταση)

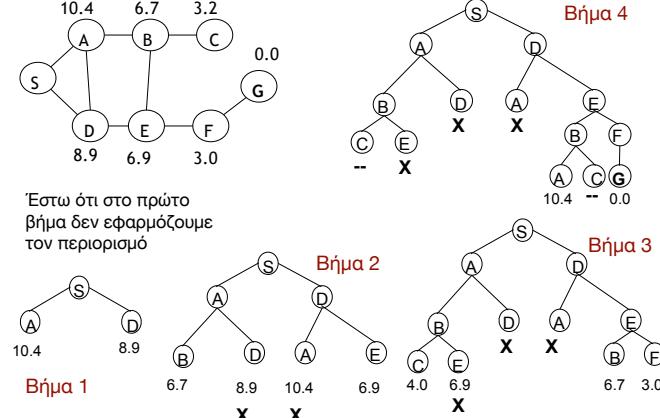
Βήμα 2: Μέχρι που η λίστα να αδειάσει ή να βρεθεί ένας τελικός κόμβος στόχος

- **Βήμα 2.α:** Εάν ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος μη κάνει τίποτε
- **Βήμα 2.β:** Εάν ο πρώτος κόμβος στη λίστα δεν είναι κόμβος στόχος, τότε βγάλε το πρώτο κόμβο από τη λίστα, και βάλε στο τέλος της λίστας τα w καλύτερα παιδιά του κόμβου, όσον αφορά την τιμή της ευριστικής απόστασης του κόμβου από ένα κόμβο «στόχο»
- **Βήμα 2.γ:** Εάν το πρώτος κόμβος δεν έχει παιδιά, αφαιρέσε τον από τη λίστα και πήγαινε στο βήμα 2

Βήμα 3: Εάν βρήκαμε ένα κόμβο στόχο τότε ανακοινώνουμε επιτυχία, αλλιώς ανακοινώνουμε αποτυχία

Αλγόριθμος Beam Search

30



Αλγόριθμος Best First



31

- ▶ Ο Best First μοιάζει με τον Hill Climbing, μόνο που εδώ επεκτείνουμε όχι τον καλύτερο κόμβο από τα παιδιά του κόμβου που είμαστε, αλλά τον καλύτερο κόμβο από όλους τους κόμβους που βρίσκονται στο μετώπιο αναζήτησης του δένδρου.
- ▶ Ο αλγόριθμος αναζήτησης Best First Search είναι πιθανότερο να παράγει μικρότερα μονοπάτια από την αρχική κατάσταση σε κάποιο κόμβο «στόχο».
- ▶ Η βασική διαφορά αυτού του αλγόριθμου από τον Hill Climbing είναι ότι αντί να βρίσκουμε το καλύτερο από τα παιδιά του κόμβου που επεκτείνουμε, εισάγουμε τα παιδιά του κόμβου που επεκτείνουμε στη λίστα, και μετά ταξινομούμε όλη τη λίστα. Οπότε:
 - ▶ Ο κόμβος με τη συνολικά μικρότερη τιμή ευριστικής, ανεβαίνει στην κορυφή της λίστας για επέκταση στο επόμενο βήμα.

Αλγόριθμος Best First



32

Βήμα 1: Κατασκεύασε μια λίστα ουρά που περιέχει τη ρίζα του δένδρου (αρχική κατάσταση)

Βήμα 2: Μέχρι που η λίστα να αδειάσει ή να βρεθεί ένας τελικός κόμβος στόχος, εξέτασε εάν ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος τότε πήγαινε στο Βήμα 3

- **Βήμα 2.a:** Εάν ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος τότε πήγαινε στο Βήμα 3

- **Βήμα 2.b:** Εάν ο πρώτος κόμβος στη λίστα δεν είναι κόμβος στόχος, τότε βγάλε το πρώτο κόμβο από τη λίστα, βρες στο παιδιά αυτού του κόμβου, και μετά **ταξινόμισε σε αυξουσα σειρά ολόκληρη τη λίστα** σε σχέση με την ευριστική τιμή για την απόσταση του κάθε κόμβου στη λίστα από ένα κόμβο «στόχο»

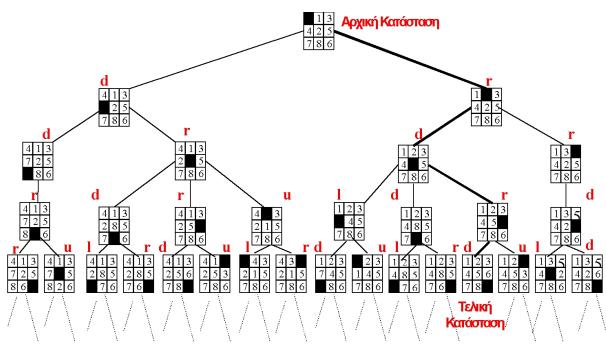
- **Βήμα 2.y:** Εάν ο πρώτος κόμβος δεν έχει παιδιά απλά αφαίρεσέ τον από τη λίστα και πήγαινε στο βήμα 2

Βήμα 3: Εάν βρήκαμε ένα κόμβο στόχο τότε ανακοινώνουμε επιτυχία αλλιώς ανακοινώνουμε αποτυχία

Αλγόριθμος Best First



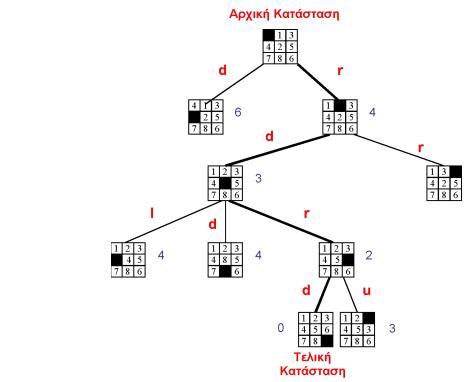
33



Αλγόριθμος Best First



34



Αλγόριθμοι πληροφορημένης αναζήτησης (αναζήτησης βέλτιστης λύσης)

Στρατηγική αλγορίθμων βελτιστοποίησης



35

- ▶ Αποθηκεύουμε μονοπάτια από την αρχική κατάσταση σε όλες τις καταστάσεις «στόχους»
- ▶ Επιλέγουμε ένα από τα μονοπάτια για να επεκτείνουμε με μετάβαση σε μία προσβάσιμη κατάσταση
- ▶ Οργανώνουμε την επιλογή και επέκταση των μονοπατιών έτσι ώστε ή να ελέγχουμε τελικά όλα τα μονοπάτια ή όσα αποκλείσουμε, **διασφαλισμένα** να μην οδηγούν σε βέλτιστη λύση

Αλγόριθμος British Museum



37

- ▶ Βρίσκουμε όλα τα μονοπάτια από την αρχική κατάσταση σε όλες τις καταστάσεις «στόχους» και επιλέγουμε το καλύτερο μονοπάτι.
- ▶ Μπορούμε να βρούμε όλα τα μονοπάτια με αναζήτηση κατά-βάθος ή αναζήτηση κατά-πλάτος. Μόνο που εδώ **δεν** σταματάμε όταν βρούμε λύση. Συνεχίζουμε μέχρι να βρούμε όλες τις λύσεις για να επιλέξουμε τη καλύτερη.
- ▶ Ο αλγόριθμος αυτός δεν είναι πρακτικός στις περισσότερες περιπτώσεις. Για παράδειγμα με παράγοντα διακλάδωσης $b=10$ και βάθος δένδρου $d=10$ έχουμε 10 δισεκατομμύρια μονοπάτια

Αλγόριθμος Branch and Bound



38

Βήμα 1: Κατασκεύασε μια λίστα από μονοπάτια (που αρχικά είναι κενή)

Βήμα 2: Μέχρι που η λίστα να είναι άδεια ή το πρώτο μονοπάτι της λίστας να οδηγεί σε «στόχο» και όλα τα άλλα μονοπάτια που δεν έχουν ακόμη οδηγήσει σε στόχο έχουν μεγαλύτερο κόστος

- **Βήμα 2.a:** Εάν το πρώτο μονοπάτι οδηγεί σε στόχο, κράτησέ το σαν πιθανή λύση. Εάν είναι καλύτερο από κάποια προηγούμενη λύση, κράτησέ το σαν την καλύτερη πιθανή λύση και Προχώρησε στο Βήμα 2.β

Αλγόριθμος Branch and Bound



39

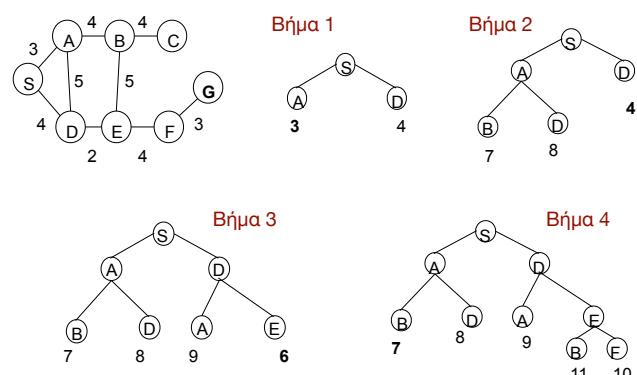
- **Βήμα 2.β:** Εάν το πρώτο μονοπάτι δεν οδηγεί σε στόχο, ή υπάρχουν άλλα μονοπάτια δεν έχουν ακόμη οδηγήσει σε στόχο και έχουν μικρότερο κόστος από το μονοπάτι που ήδη βρήκαμε
 - Βήμα 2.β.1. Βγάλε το πρώτο μονοπάτι από τη λίστα
 - Βήμα 2.β.2. Φτιάξε μονοπάτια που μπορούν να φτιαχτούν από το μονοπάτι που ήδη βρήκαμε επεκτείνοντας το κατά ένα άντρα (branch)
 - Βήμα 2.β.3. Βάλε τα νέα μονοπάτια στη λίστα
 - Βήμα 2.β.4. Ταξινόμησε σε αύξουσα σειρά όλα τα μονοπάτια στη λίστα σύμφωνα με το κόστος του κάθε μονοπατιού (από την αρχική κατάσταση στο τελευταίο κόμβο του μονοπατιού)
 - Βήμα 2.β.5 Κλαδεύει τα μονοπάτια που έχουν κόστος μεγαλύτερο από ένα **όριο** που διαφαίλισμένα δεν μπορεί να οδηγήσει σε βέλτιστη λύση (bound) (π.χ. το κόστος της καλύτερης λύσης που έχουμε βρει μέχρι τότε)
 - Βήμα 2.β.6 Εάν βρήκαμε ένα μονοπάτι που οδηγεί σε κόμβο στόχο τότε ανακοινώνουμε μερική επιτυχία αλλιώς ανακοινώνουμε αποτυχία

Βήμα 3: Εάν βρήκαμε ένα μονοπάτι που οδηγεί σε κόμβο στόχο τότε ανακοινώνουμε μερική επιτυχία αλλιώς ανακοινώνουμε αποτυχία

Αλγόριθμος Branch and Bound



40



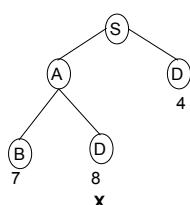
Δυναμικός προγραμματισμός (DP)



41

Διαισθητικά

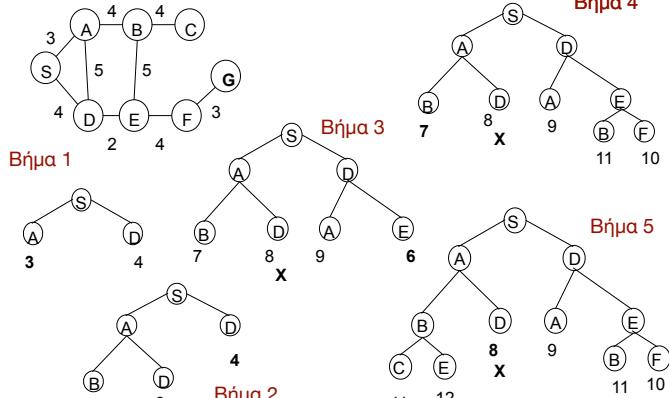
- ▶ Επεκτείνουμε δυναμικά το όριο απόρριψης κάποιου μονοπατιού:
Αν κατασκευάσουμε ένα μονοπάτι που οδηγεί σε κάποιο κόμβο, ενώ υπάρχει κάποιο άλλο μονοπάτι που οδηγεί στον ίδιο κόμβο αλλά με μικρότερο κόστος, τότε «κλαδεύουμε» το μονοπάτι με το μεγαλύτερο κόστος



DP και Branch and Bound



42



Αλγόριθμος A*



43

- ▶ Ακολουθούμε τη λογική του Branch and Bound
- ▶ Επεκτείνουμε το μονοπάτι με τον καλύτερο από όλους τους κόμβους που βρίσκονται στο μέτωπο αναζήτησης του δένδρου (λογική Best First).
- ▶ Για το σκοπό αυτό χρησιμοποιούμε τη σύνθετη ευριστική συνάρτηση $F(k) = g(k) + h(k)$
 - $g(k)$ η απόσταση της k από την αρχική κατάσταση, η οποία είναι πραγματική και γνωστή
 - $h(k)$ μία εκτίμηση της απόστασης της k από το στόχο (μέσω μιας ευριστικής συνάρτησης)
- ▶ Εφαρμόζουμε για οριοθέτηση (bound) δυναμικό προγραμματισμό

Αλγόριθμος A*



44

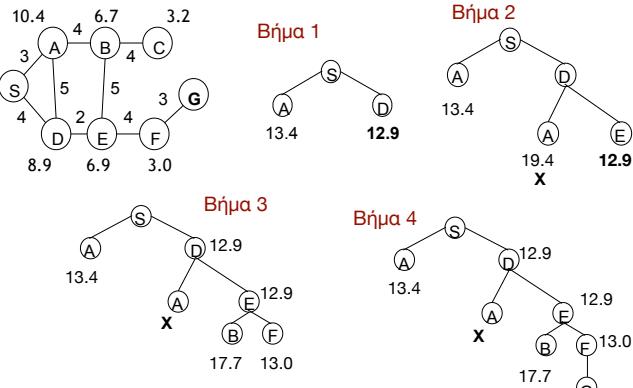
Παρατηρήσεις

- ▶ Αν για κάθε κατάσταση η τιμή $h(k)$ είναι μικρότερη ή ίση με την πραγματική απόσταση της k από την τελική κατάσταση, τότε ο A* βρίσκει πάντα τη βέλτιστη λύση (αποδεικνύεται σχετικά εύκολα)
- ▶ Στην περίπτωση αυτή, ο ευριστικός μηχανισμός ονομάζεται αποδεκτός (*admissible*)
- ▶ Ο A* εκτελείται στη χειρότερη περίπτωση σε πολυωνυμικό χρόνο (σε σχέση με τον αριθμό των μεταβάσεων του βέλτιστου μονοπατιού), αν $|h(x) - h^*(x)| = O(\log(h^*(x)))$
- ▶ Όπου $h^*(x)$ η βέλτιστη ευριστική, δηλαδή η πραγματική απόσταση από το στόχο

Αλγόριθμος A*



45



Αλγόριθμοι αναζήτησης για παιγνία

Αλγόριθμοι για προβλήματα δύο αντιπάλων



47

- ▶ Σε πολλά προβλήματα της TN, η εξέλιξη των καταστάσεων εξαρτάται από δύο διαφορετικά σύνολα τελεστών μετάβασης που εφαρμόζονται εναλλάξ από δύο δράστες
- ▶ Αυτά τα προβλήματα αναφέρονται και ως ανταγωνιστικά παιγνία ή παιγνία δύο αντιπάλων (adversary ή two-person games)
- ▶ Κατασκευάζουμε το δένδρο αναζήτησης έτσι ώστε οι κινήσεις δύο διαδοχικών επιπέδων να ανήκουν σε διαφορετικό παίκτη, αφού θεωρούμε ότι οι παίκτες παίζουν εναλλάξ
- ▶ Ο παίκτης αξιολογεί τις καταστάσεις που θα προκύψουν από πιθανές διαφορετικές εξελίξεις του παιχνιδιού και αποφασίζει ποια από τις εναλλακτικές κινήσεις θα τον οδηγήσει σταδιακά στην πιο ευνοϊκή για αυτόν εξέλιξη

Αλγόριθμος minimax



48

- ▶ Δεδομένης μίας κατάστασης του παιχνιδιού (κόμβος στο δένδρο), ο αλγόριθμος καλείται να αποφασίσει ποια θα είναι η επόμενη κίνηση (μετάβαση στον επόμενο κόμβο)
- ▶ Ο ένας παίκτης ονομάζεται **max** και ο άλλος ονομάζεται **min**. Ανάλογα διακρίνονται οι κόμβοι του δένδρου διακρίνονται σε **max** ή **min** (ποιος έχει σειρά να παίξει)
- ▶ Το ζητούμενο είναι:
 - Na χριστεί το δένδρο μέχρι κάποιο βάθος και με βάση αυτό Νa βρεθεί η **καλύτερη** κίνηση από την παρούσα κατάσταση με βάση το που σδημούν οι κινήσεις
- ▶ Χρησιμοποιούμε μία ευριστική συνάρτηση που καλείται συνάρτηση αξιολόγησης (evaluation function) και εκφράζει για κάθε κόμβο την υπεροχή έναντι του αντιπάλου

Αλγόριθμος min-max



49

Βήμα 1. Εφάρμοσε τη συνάρτηση αξιολόγησης σε όλους τους κόμβους-φύλλα του δένδρου.

Βήμα 2. Έως ότου η ρίζα του δένδρου αποκτήσει τιμή, επανέλαβε:

Βήμα 3. Αρχίζοντας από τα φύλλα του δένδρου και προχωρώντας προς τη ρίζα, μετέφερε τις τιμές προς τους ενδιάμεσους κόμβους του δένδρου ως εξής:

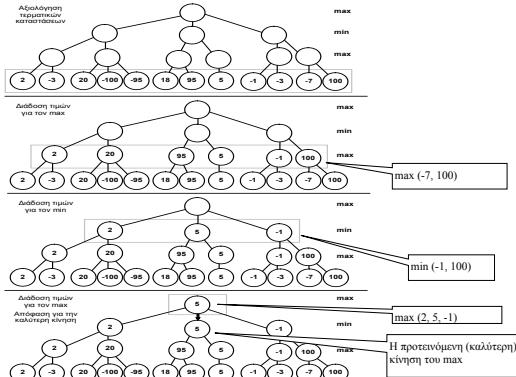
Βήμα 3.1 Η τιμή κάθε κόμβου Max είναι η μέγιστη (maximum) των τιμών των κόμβων-παιδιών του.

Βήμα 3.2 Η τιμή κάθε κόμβου Min είναι η ελάχιστη (minimum) των τιμών των κόμβων-παιδιών του.

Βήμα 4. Καλύτερη κίνηση είναι η κίνηση που οδηγεί στον κόμβο που έδωσε την πιο συμφέρουσα στη ρίζα τιμή (μέγιστη για το Max, ελάχιστη για το Min).

Αλγόριθμος min-max

50



Αλγόριθμος Alpha-Beta



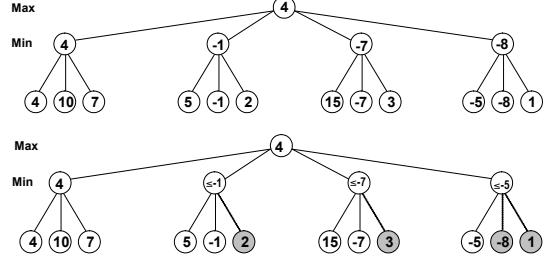
51

- Αποφεύγουμε την αξιολόγηση καταστάσεων που ικανοποιούν ορισμένες συνθήκες



Αλγόριθμος Alpha-Beta

52



- Αξιολογούνται οι πρώτοι αριστεροί κόμβοι (4, 10, 7) και δίνουν την τιμή 4 στον κόμβο πατέρα.
► Η αξιολόγηση προχωρά στο δεύτερο κλαδί και βρίσκει 5 και -1 οπότε και σταματά αφού η τιμή του είναι μικρότερη από τη μεγαλύτερη τιμή του ανωτέρου επιπέδου (δηλ. το 4).