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

Θεμελιώδη Θέματα Επιστήμης Υπολογιστών

Τεχνητή Νοημοσύνη

Αλγόριθμοι αναζήτησης λύσης

Γιώργος Στάμου

Κατασκευή μηχανών που αντιμετωπίζουν προβλήματα που προς το παρόν επιλύουν καλύτερα οι άνθρωποι

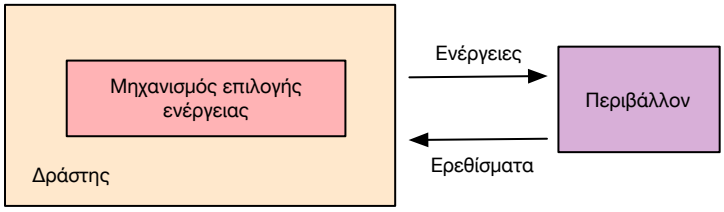
Δοκιμασία Turing

Ο υπολογιστής περνά τη δοκιμασία αν ένας άνθρωπος εξεταστής, αφού αλληλεπιδράσει με το σύστημα, δεν μπορεί να συμπεράνει αν οι αποκρίσεις προέρχονται από άνθρωπο ή από μηχανή

Λειτουργικές απαιτήσεις

- *Αντίληψη* (επεξεργασία σημάτων από αισθητήρες, αλληλεπίδραση με τον άνθρωπο,…)
- *Γνώση, συλλογισμός, διορατικότητα, προσαρμοστικότητα, δημιουργικότητα,…*

ΤΝ και ανάπτυξη ευφυών δραστών



Ορθολογικός δράστης = Αρχιτεκτονική + Πρόγραμμα

Αρχιτεκτονική

- Επιτρέπει τα ερεθίσματα να διοχετεύονται στο Πρόγραμμα
- Εκτελεί το Πρόγραμμα
- Μετατρέπει τα αποτελέσματα σε μηχανισμούς δράσης

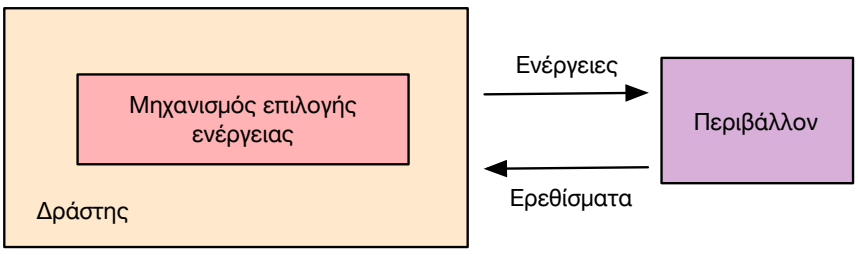
Πρόγραμμα

- Ορίζει την απεικόνιση [*ακολουθία ερεθισμάτων -> ενέργεια*]

Ιδανικός ορθολογικός δράστης

- Για κάθε πιθανή κατάσταση και ερεθίσματα έχει ενεργεί ώστε να *μεγιστοποιήσει την επίδοσή του*

Πλαίσιο δράσης



Περιβάλλον, Αισθητήρες, Δράσεις, Επίδοση

Αφαίρεση

Καθορισμός σημαντικών στοιχείων

Τυπικός ορισμός προβλήματος

Πλαίσιο δράσης - Παράδειγμα

Αυτόματος οδηγός ταξί

[Περιβάλλον, Αισθητήρες, Δράσεις, Επίδοση]

- Περιβάλλον: Δρόμοι, αυτοκίνητα, πεζοί, πελάτες
- Αισθητήρες: Κάμερες, αισθητήρες υπερήχων, κοντέρ, GPS, οδόμετρο, αισθητήρες λειτουργίας αυτοκινήτου, εγκέφαλος αυτοκινήτου
- Δράσεις: Γκάζι, φρένο, τιμόνι, φώτα, κόρνα
- Δείκτες Επίδοσης: Ασφαλής, γρήγορη, άνετη, χωρίς παραβάσεις μεταφορά στον προορισμό

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

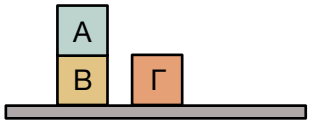
- Είναι μία *επαρκής*, *τυπική* αναπαράσταση του κόσμου σε μία δεδομένη χρονική στιγμή

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

Κόσμος του προβλήματος: Τρεις κύβοι και ένα τραπέζι



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

Κατάσταση:

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

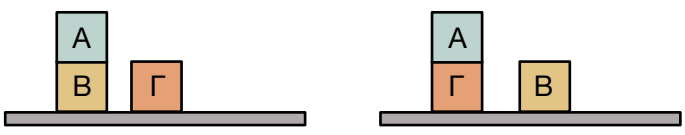
Τυπική διατύπωση προβλήματος

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.:
  - *Βάλε τον κύβο Α πάνω στον κύβο Γ*
  - *Βάλε τον κύβο Α πάνω στον κύβο Β*

Σε κάθε πρόβλημα αρχίζουμε από μία αρχική κατάσταση *I* και καταλήγουμε σε μία τελική κατάσταση (στόχο) *G*

Αρχική και Τελική Κατάσταση στο πρόβλημα με τους κύβους



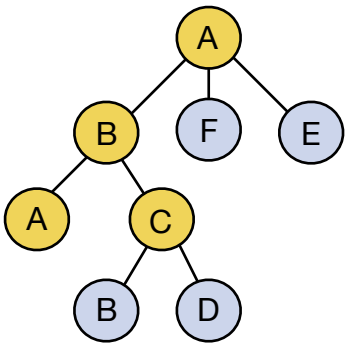
Λύση (solution) σε ένα πρόβλημα, είναι μία ακολουθία $t_1, t_2, \ldots, t_n$ από τελεστές μετάβασης, τέτοια ώστε $G = t_n(\ldots(t_2(t_1(I)))\ldots)$

Αλγόριθμοι αναζήτησης λύσης

Βασικές δομές 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

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 διέγραψέ 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 - Ταξινόμηση

Αλγόριθμοι τυφλής αναζήτησης

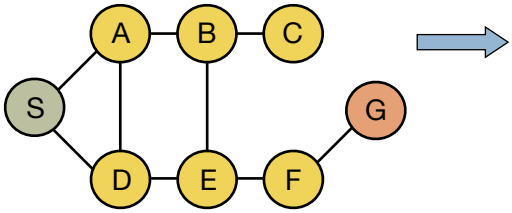
- Δεν έχουμε κάποια μέθοδο να εκτιμήσουμε το κόστος των εναλλακτικών για το επόμενο βήμα (σε σχέση με τον τελικό στόχο), δηλαδή ψάχνουμε τη λύση χωρίς ένδειξη για το αν πλησιάζουμε σε κατάσταση στόχο

Αλγόριθμοι πληροφορημένης αναζήτησης

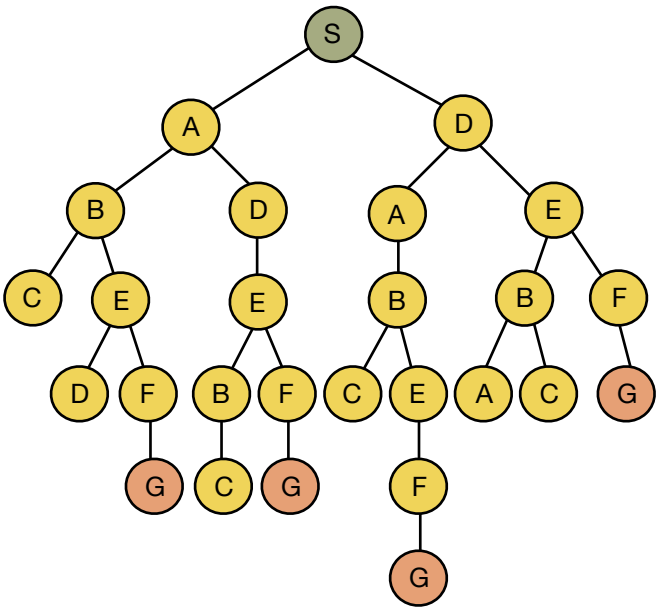
- Μοντελοποιούμε την απόσταση κάθε κόμβου από τον στόχο, με βάση κάποια ευριστική μέθοδο, οπότε μπορούμε να λάβουμε υπόψην αυτή την εκτίμηση κατά την αναζήτηση
- Η ευριστική τιμή δεν είναι η πραγματική τιμή της απόστασης από μία κατάσταση στόχο, αλλά μία εκτίμηση (που πολλές φορές μπορεί να είναι και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

Αλγόριθμοι αναζήτησης - Πληροφορίες

Καμμία πληροφορία

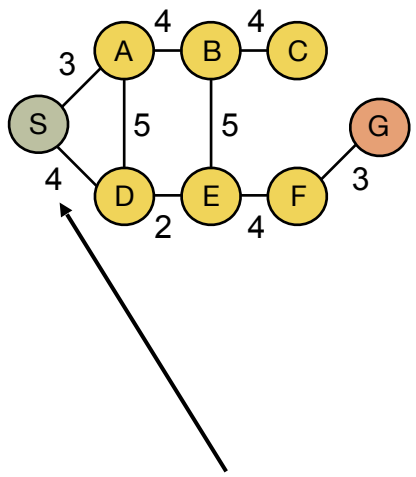


Απόσπασμα ενός δένδρου εκτέλεσης



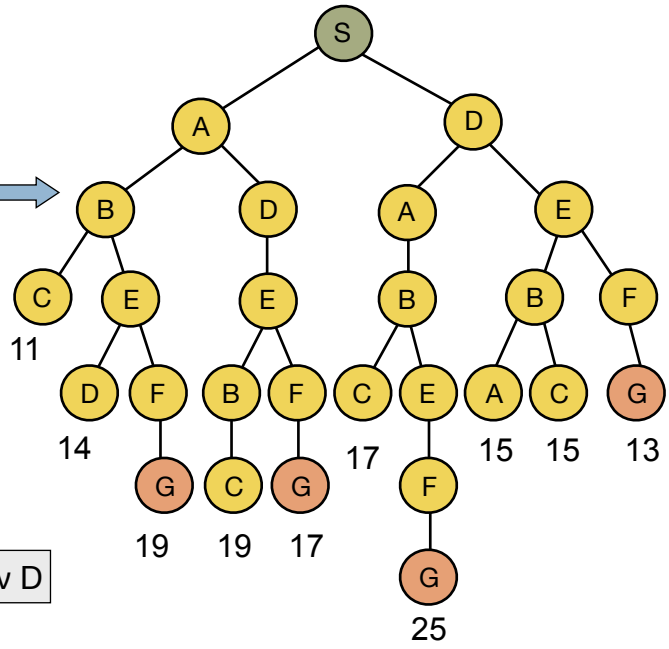
Αλγόριθμοι αναζήτησης - Πληροφορίες

Εκτίμηση κόστους μετάβασης από κατάσταση σε κατάσταση

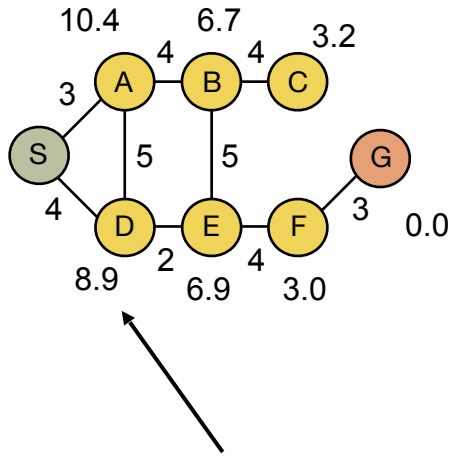


Κόστος μετάβασης από την S στην D

Απόσπασμα ενός δένδρου εκτέλεσης

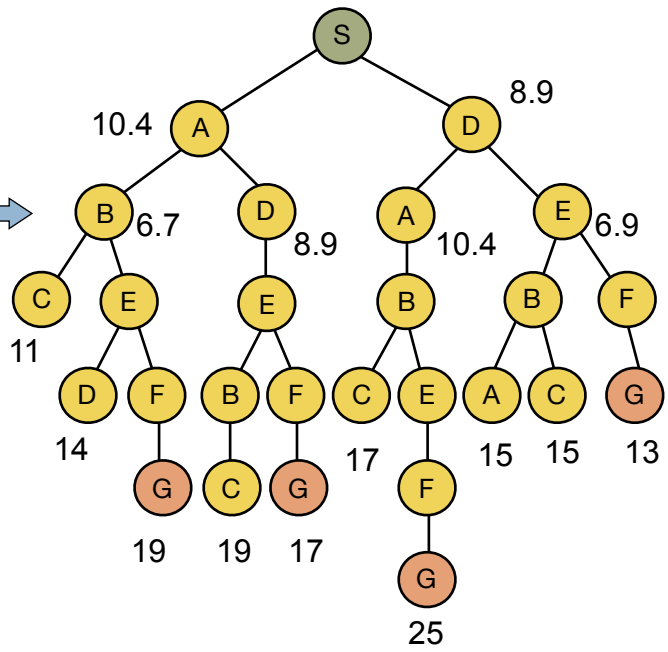


Εκτίμηση κόστους μετάβασης από μία κατάσταση στο στόχο



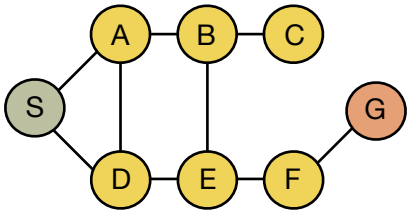
Εκτίμηση υπολειπόμενου κόστους από την D ως τη G

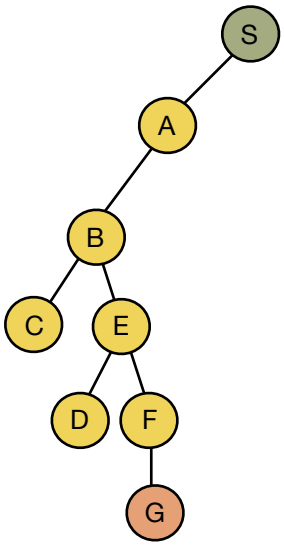
Απόσπασμα ενός δένδρου εκτέλεσης



Αλγόριθμοι τυφλής αναζήτησης

Αλγόριθμος αναζήτησης κατά βάθος (DFS)





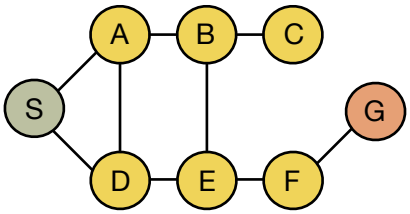
Μέτωπο αναζήτησης

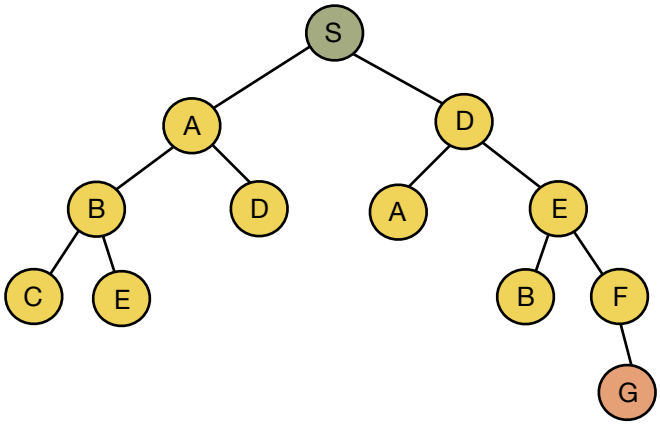
[S]
[A,D]
[B,D,D]
[C,E,D,D]
[E,D,D]
[D,F,D,D]
[F,D,D]
[G,D,D]

Κλειστό σύνολο

[S]
[A,S]
[A,B,S]
[A,B,C,S]
[A,B,C,E,S]
[A,B,C,D,E,S]
[A,B,C,D,E,F,S]

Αλγόριθμος αναζήτησης κατά πλάτος (BFS)





Μέτωπο αναζήτησης

[S]
[A,D]
[D,B,D]
[B,D,A,E]
[D,A,E,C,E]
[A,E,C,E]
[E,C,E]
[C,E,B,F]
[E,B,F]
[B,F]
[F]
[G]

Κλειστό σύνολο

[S]
[A,S]
[A,D,S]
[A,B,D,S]
[A,B,D,S]
[A,B,D,S]
[A,B,D,E,S]
[A,B,C,D,E,S]
[A,B,C,D,E,S]
[A,B,C,D,E,S]
[A,B,C,D,E,F,S]

Αλγόριθμος επαναληπτικής εκβάθυνσης

Βήματα Αλγόριθμου (Iterative Deepening - ID)

Βήμα 1. Όρισε 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 κτίσιμο του χώρου αναζήτησης

Αλγόριθμοι πληροφορημένης αναζήτησης (αναζήτησης μίας οποιασδήποτε λύσης)

Ευριστικοί μηχανισμοί

Ευριστικές σε Ευκλείδειους χώρους

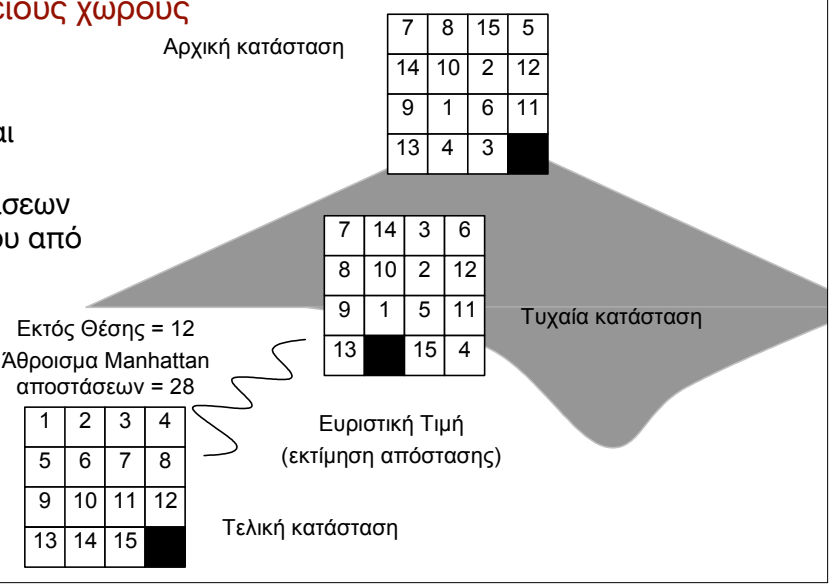
Ευκλείδεια απόσταση (Euclidian distance) $d(s,f) = \sqrt{|x_s - x_f|^2 + |y_s - y_f|^2}$

Απόσταση Manhattan (Manhattan distance): $d(s,f) = |x_s - x_f| + |y_s - y_f|$

Ευριστικές σε μη Ευκλείδειους χώρους

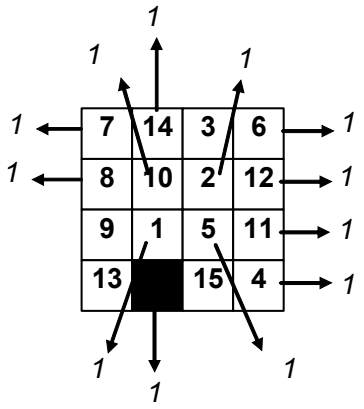
Ευριστικός μηχανισμός και συναρτήσεις στο N-Puzzle

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

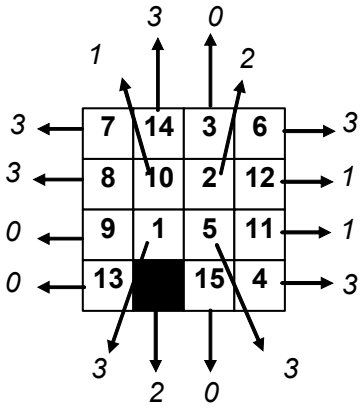


Ευριστικοί μηχανισμοί

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



Εκτός θέσης = 12



Άθροισμα αποστάσεων Manhattan = 28

Αλγόριθμος Hill climbing

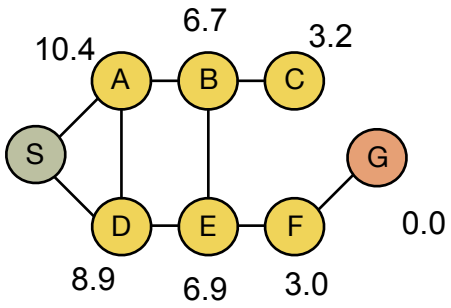
Βήμα 1: Όρισε τον τρέχοντα κόμβο ως τη ρίζα του δένδρου

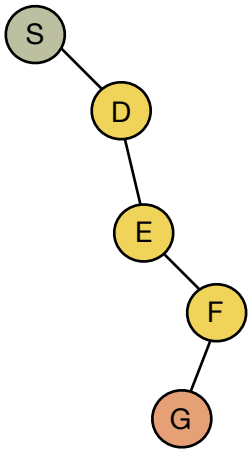
Βήμα 2: Μέχρι που ο τρέχων κόμβος δεν είναι κόμβος στόχος, εκτέλεσε:

- **Βήμα 2.α**: Βρες τα παιδιά του τρέχοντος κόμβου, και στη συνέχεια βρες αυτό με την ελάχιστη υπολογιζόμενη υπόλοιπη απόσταση από το στόχο
- **Βήμα 2.β**: Εάν ο τρέχων κόμβος δεν έχει παιδιά ή το παιδί που βρέθηκε στο βήμα 2.α έχει μεγαλύτερη τιμή ευριστικής από αυτόν πήγαινε στο Βήμα 3.
- **Βήμα 2.γ**: Όρισε τον κόμβο που βρέθηκε στο Βήμα 2.α ως τρέχων κόμβο.

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

Αλγόριθμος Hill climbing





[S]

[D]

[E]

[F]

[G]

[S]

[D,S]

[D,E,S]

[D,E,F,S]

Αλγόριθμος Beam Search

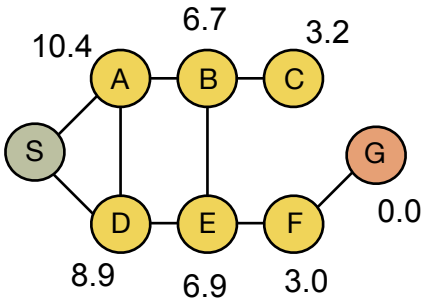
Βήμα 1: Κατασκεύασε μια λίστα ουρά που περιέχει τη ρίζα του δένδρου (αρχική κατάσταση)

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

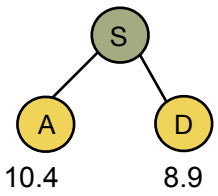
- **Βήμα 2.α**: Εάν ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος μην κάνεις τίποτε
- **Βήμα 2.β**: Εάν ο πρώτος κόμβος στη λίστα δεν είναι κόμβος στόχος, τότε βγάλε το πρώτο κόμβο από τη λίστα, και βάλε στο **τέλος** της λίστας τα w καλύτερα παιδιά του κόμβου, όσον αφορά την τιμή της ευριστικής απόστασης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

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

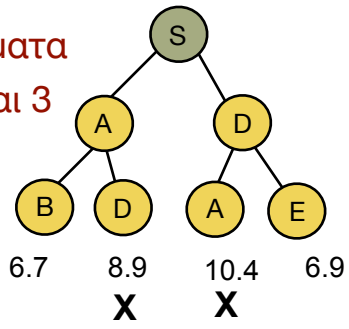
Αλγόριθμος Beam Search



Βήμα 1

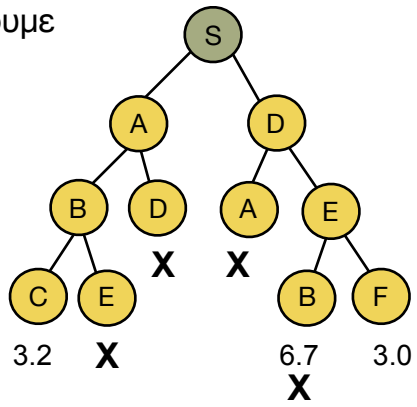


Βήματα 2 και 3

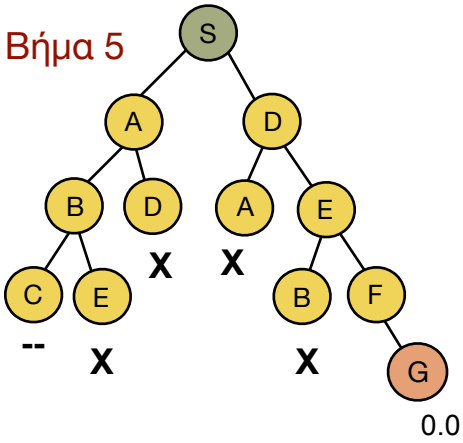


Έστω ότι στο πρώτο βήμα δεν εφαρμόζουμε τον περιορισμό

Βήματα 3 και 4



Βήμα 5



Αλγόριθμος Best First

- 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 μέτωπο 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

Βήμα 1: Κατασκεύασε μια λίστα 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

- **Βήμα 2.α**: Εάν ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος τότε πήγαινε στο Βήμα 3
- **Βήμα 2.β**: Εάν ο πρώτος κόμβος στη λίστα δεν είναι κόμβος στόχος, τότε βγάλε το πρώτο κόμβο από τη λίστα, βρες στο παιδιά αυτού του κόμβου, βάλε στη λίστα τα παιδιά αυτού του κόμβου, και μετά **ταξινόμησε σε αύξουσα σειρά ολόκληρη τη λίστα** σε σχέση με την ευριστική τιμή για την απόσταση του κάθε κόμβου στη λίστα από ένα κόμβο «στόχο»
- **Βήμα 2.γ**: 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

Αλγόριθμοι πληροφορημένης αναζήτησης (αναζήτησης βέλτιστης λύσης)

Στρατηγική αλγορίθμων βελτιστοποίησης

- 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 ελέγξουμε τελικά όλα τα μονοπάτια ή όσα αποκλείσουμε, διασφαλισμένα να μην οδηγούν σε βέλτιστη λύση

Αλγόριθμος British Museum

- 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

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

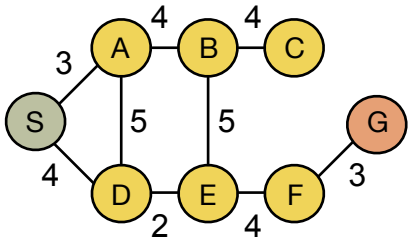
- **Βήμα 2.α**: Εάν το πρώτο μονοπάτι οδηγεί σε στόχο, κράτησέ το σαν πιθανή λύση. Εάν είναι καλύτερο από κάποια προηγούμενη λύση, κράτησέ το σαν την καλύτερη πιθανή λύση και Προχώρησε στο Βήμα 2.β

Αλγόριθμος Branch and Bound

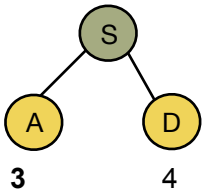
- **Βήμα 2.β**: Εάν το πρώτο μονοπάτι δεν οδηγεί σε στόχο, ή υπάρχουν άλλα μονοπάτια δεν έχουν ακόμη οδηγήσει σε στόχο και έχουν μικρότερο κόστος από το μονοπάτι που ήδη βρήκαμε
  - Βήμα 2.β.1. Βγάλε το πρώτο μονοπάτι από τη λίστα
  - Βήμα 2.β.2. Φτιάξε μονοπάτια που μπορούν να φτιαχτούν από το μονοπάτι που βγάλαμε επεκτείνοντας το κατά ένα βήμα (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. Κλάδεψε τα μονοπάτια που έχουν κόστος μεγαλύτερο από ένα *όριο* που διασφαλισμένα 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 επιτυχία αλλιώς ανακοινώνουμε αποτυχία

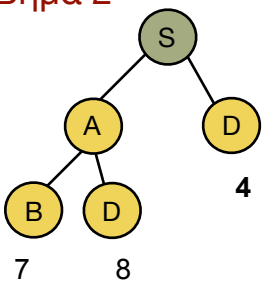
Αλγόριθμος Branch and Bound



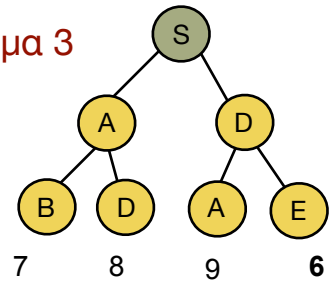
Βήμα 1



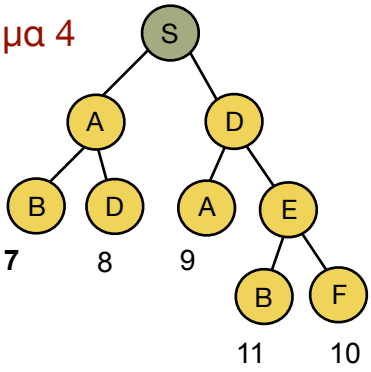
Βήμα 2



Βήμα 3



Βήμα 4

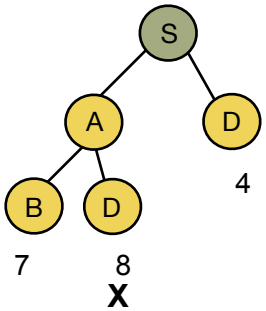


Δυναμικός προγραμματισμός (DP)

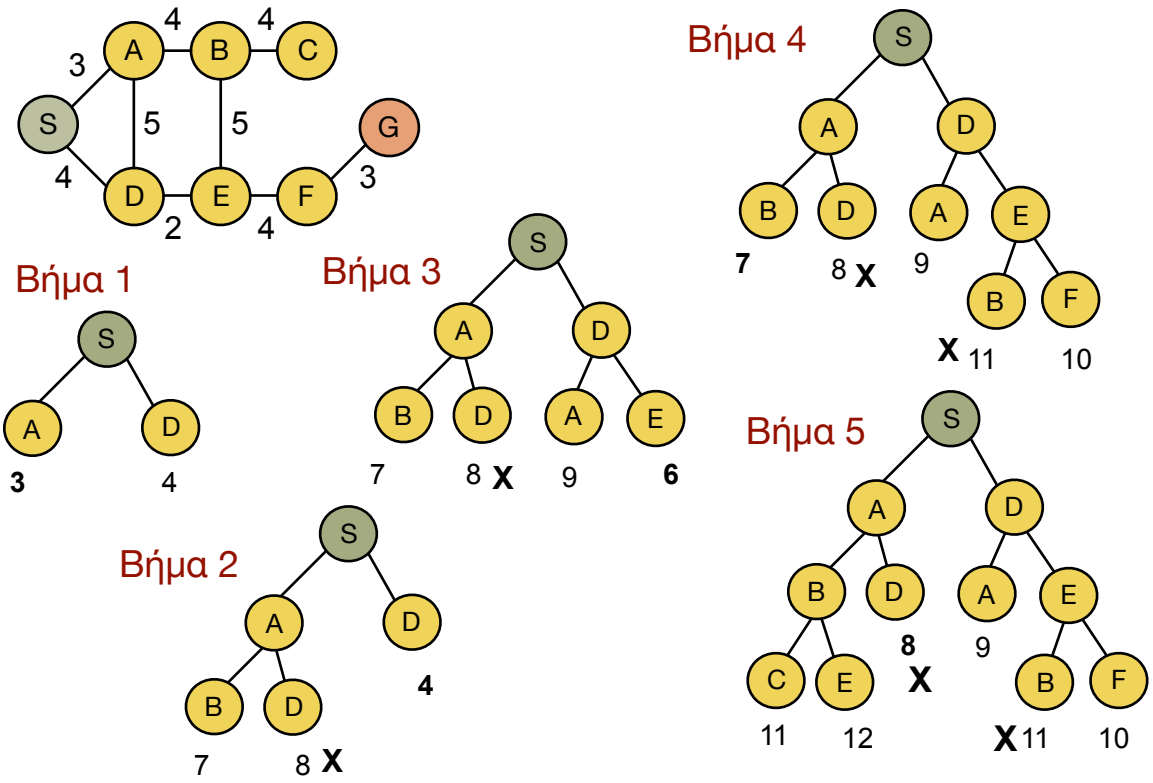
Διαισθητικά

- 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



Αλγόριθμος A*

- 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 $F(k) = g(k) + h(k)$, όπου:

  $g(k)$ 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*

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*