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

ΕΝΟΤΗΤΑ 1.5
Αλγορίθμων Εύρεσης Βέλτιστης Λύσης σε Χώρο Καταστάσεων

Διδάσκων Ενότητας:
Κώστας Κοντογιάννης

ΜΕΡΟΣ 1: Επίλυση Προβλημάτων στο Χώρο των καταστάσεων

Περίληψη Ενότητας 1.5

Θέματα

- Αλγόριθμοι Εύρεσης Βέλτιστης Λύσης σε χώρο καταστάσεων

Υλικό

- Διαφάνειες παρουσίασης
- Κεφάλαια 3, 4 Βιβλίου

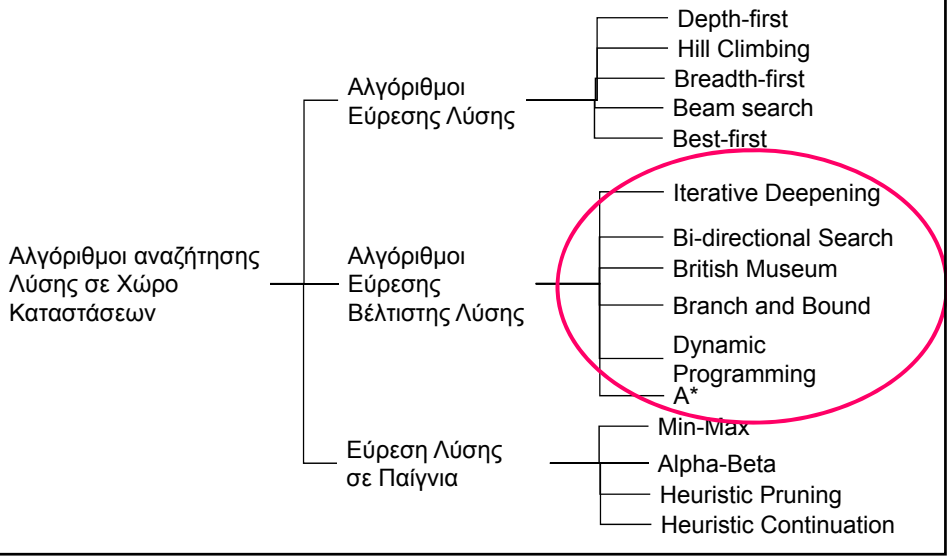
Βιβλιογραφία - Αναφορές

3

- Υλικό για αυτή τη διάλεξη συγκεντρώθηκε
  - από το βιβλίο “Artificial Intelligence” by P.H. Winston, Addison Wesley, 1984.
  - από το βιβλίο «Τεχνητή Νοημοσύνη», Ι. Βλαχάβας κ.α, Εκδόσεις Γκιούρδας

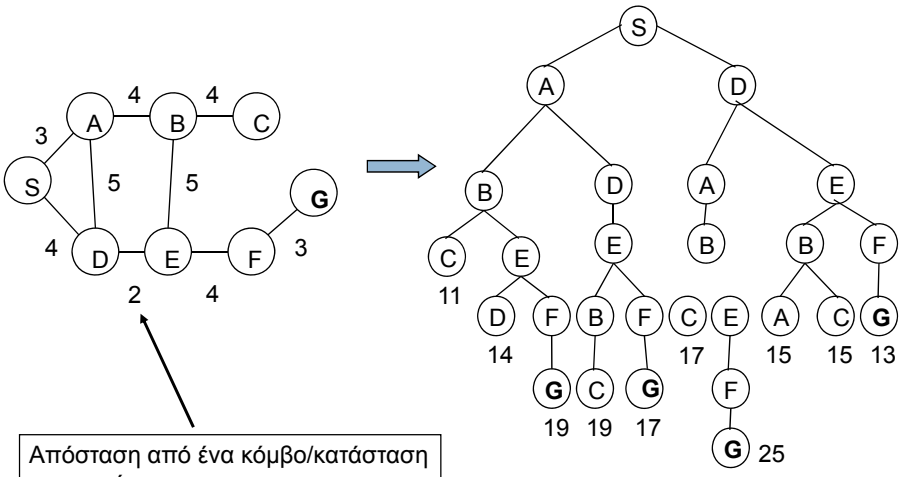
Ταξινόμηση Αλγόριθμων Αναζήτησης

4



Παράδειγμα Δένδρου Αναζήτησης

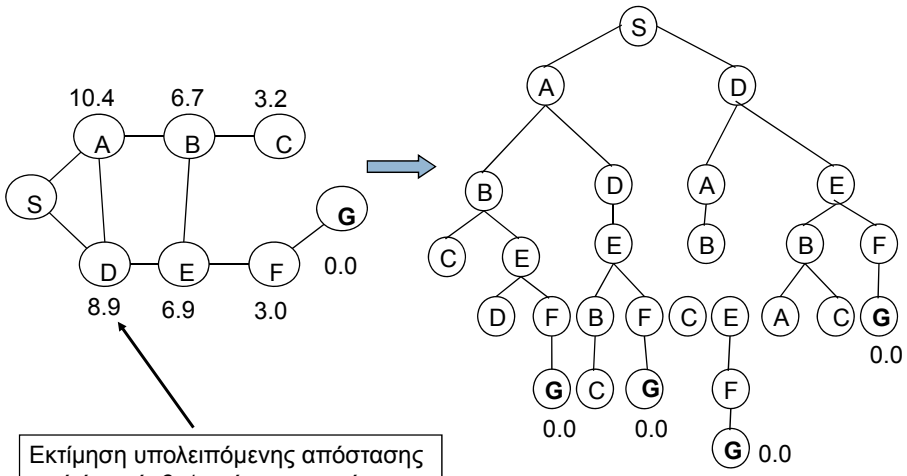
5



Απόσταση από ένα κόμβο/κατάσταση στην επόμενη.

Παράδειγμα Δένδρου Αναζήτησης

6



Εκτίμηση υπολειπόμενης απόστασης από ένα κόμβο/κατάσταση σε ένα κόμβο «στόχο».

Αλγόριθμος Iterative Deepening Επαναληπτικής Εκβάθυνσης (ΕΕ)

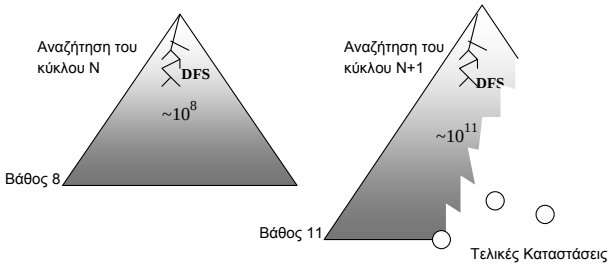
- **Ο αλγόριθμος επαναληπτικής εκβάθυνσης (Iterative Deepening - ID) συνδυάζει με τον καλύτερο τρόπο τους DFS και BFS.**
- **<u>Ο αλγόριθμος ID:</u>**
  1. **Όρισε το αρχικό βάθος αναζήτησης (συνήθως 1).**
  2. **Εφάρμοσε τον αλγόριθμο DFS μέχρι αυτό το βάθος αναζήτησης.**
  3. **Αν έχεις βρει λύση σταμάτησε.**
  4. **Αύξησε το βάθος αναζήτησης (συνήθως κατά 1).**
  5. **Πήγαινε στο βήμα 2.**

Ο αλγόριθμος ID (Ψευδοκώδικας)

```
algorithm id(InitialState, FinalStates)
begin
    depth1
    while solution is not found do
            bounded_dfs(InitialState,FinalStates,depth);
            depthdepth+1
    endwhile;
end.
```

Σχόλια για τον Αλγόριθμο ΕΕ (i)

- Μειονεκτήματα:
  - *Όταν αρχίζει ο DFS με διαφορετικό βάθος δε θυμάται τίποτα από την προηγούμενη αναζήτηση.*
- Πλεονεκτήματα:
  - *Είναι πλήρης.*
  - *Αν το βάθος αυξάνεται κατά 1 σε κάθε κύκλο και ο ID βρει λύση, τότε αυτή η λύση θα είναι η καλύτερη, γιατί αν υπήρχε άλλη, καλύτερη λύση, αυτή θα βρισκόταν σε προηγούμενο κύκλο αναζήτησης.*



Σχόλια για τον Αλγόριθμο ΕΕ (ii)

9

- Έχει αποδειχθεί ότι ο ID έχει την ίδια πολυπλοκότητα σε χώρο και χρόνο με τους DFS και BFS, όταν έχουμε μεγάλους χώρους αναζήτησης, παρ' όλο που επαναλαμβάνει άσκοπα το κτίσιμο του χώρου αναζήτησης,
- Για παράδειγμα:
  - *Έστω ότι το δένδρο αναζήτησης έχει σταθερό παράγοντα διακλάδωσης 10. Εφαρμόζουμε τον ID σε βάθος 5.*
  - *Οι καταστάσεις που θα επεκταθούν είναι:*
  - 100+101+102+103+104+105 = 111.111
  - *Αν αυξηθεί το βάθος κατά 2 (συνολικό βάθος 7), οι καταστάσεις που θα επεκταθούν συνολικά είναι:*
  - 107+106+105+ …+ 100 = 11.111.111
  - *άρα το χάσιμο ήταν 11.111/11.111.111 = 0,01%*
- Ο ID δεν κινδυνεύει να χαθεί σε κάποιο κλαδί απείρου μήκους. Αν βρει λύση θα είναι η καλύτερη, αν το βάθος αυξάνεται κατά 1 σε κάθε κύκλο.

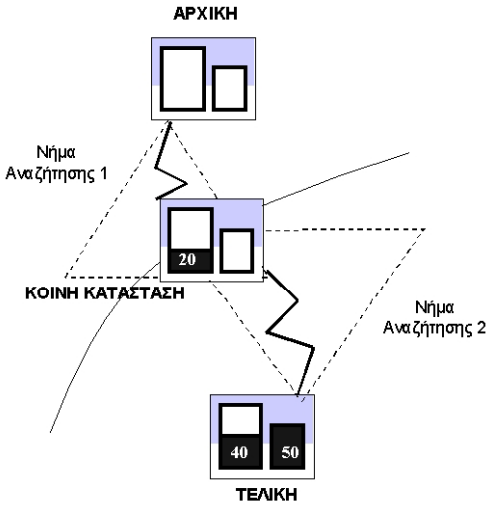
Αλγόριθμος Bi-Directional Search - Διπλής Κατεύθυνσης

10

- **Η ιδέα της *αναζήτησης διπλής κατεύθυνσης* (Bidirectional Search - *BiS*) πηγάζει από τη δυνατότητα του παραλληλισμού (parallelism) στα υπολογιστικά συστήματα.**
- Προϋποθέσεις κάτω από τις οποίες μπορεί να εφαρμοστεί:
  - *Οι τελεστές μετάβασης είναι αντιστρέψιμοι (reversible), και*
  - *Είναι πλήρως γνωστή η τελική κατάσταση.*
- Τότε
  - *Μπορούμε να αρχίσουμε 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 εως την αρχική και εως την τελική κατάσταση.*
- Μειονεκτήματα:
  - *Υπάρχει επιπλέον κόστος που οφείλεται στην επικοινωνία μεταξύ των δύο αναζητήσεων.*

Σχηματική Παρουσίαση Bi-directional Search

11



Αλγόριθμος Αναζήτησης British Museum

12

- Είναι αλγόριθμος brute force.
- Η ιδέα είναι ότι για να βρούμε τη καλύτερη λύση 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 κατα-βάθος ή αναζήτηση κατά-πλάτος. Μόνο που εδώ δεν σταματάμε όταν βρούμε κάποια 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 – Επέκτασης και Οριοθέτησης

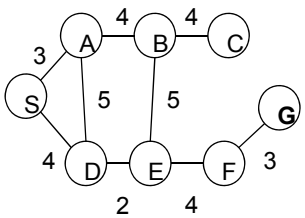
13

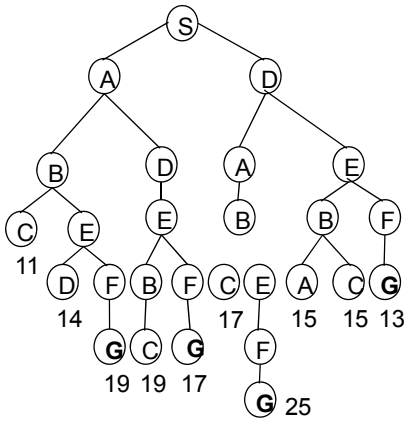
- **Βήμα 1**: Κατασκεύασε μια λίστα από μονοπάτια που αρχικά είναι κενή που περιέχει δηλαδή τα μονοπάτια από τη ρίζα του δένδρου (αρχική κατάσταση) στο πουθενά
- **Βήμα 2**: Μέχρι που η λίστα να αδειάσει ή, το πρώτο μονοπάτι στη λίστα οδηγεί σε «στόχο» και όλα τα άλλα μονοπάτια δεν έχουν ακόμη οδηγήσει 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, Προχώρησε στο Βήμα 2.β
  - **Βήμα 2.β**: Εάν το πρώτο μονοπάτι δεν οδηγεί σε στόχο, ή υπάρχουν άλλα μονοπάτια δεν έχουν ακόμη οδηγήσει σε στόχο 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
    - 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, δηλαδή το κόστος που κάναμε να φτάσουμε από την αρχική κατάσταση στο τελευταίο κόμβο κάθε μονοπατιού
    - Βήμα 2.β.5 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

Παράδειγμα Εφαρμογής Αλγόριθμου Branch and Bound

14

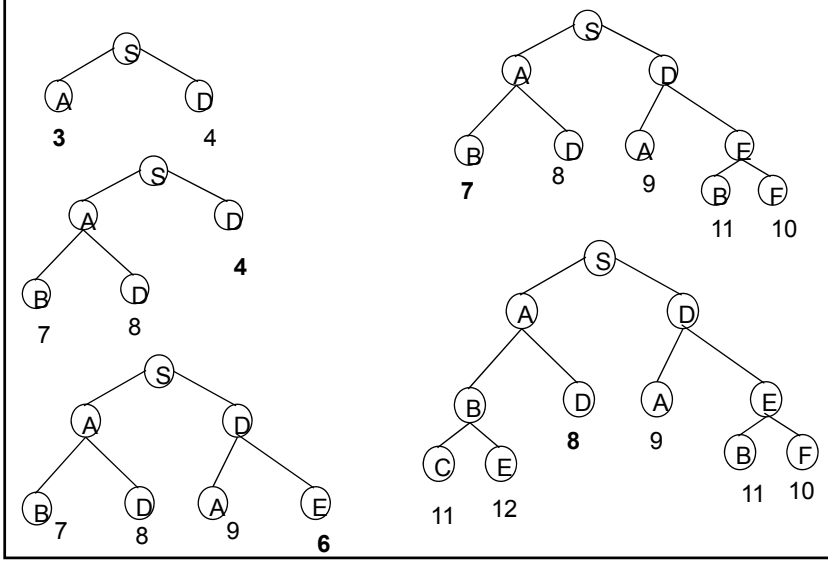
Έστω ότι έχουμε τον παρακάτω χώρο καταστάσεων και το αντίστοιχο δένδρο αναζήτησης





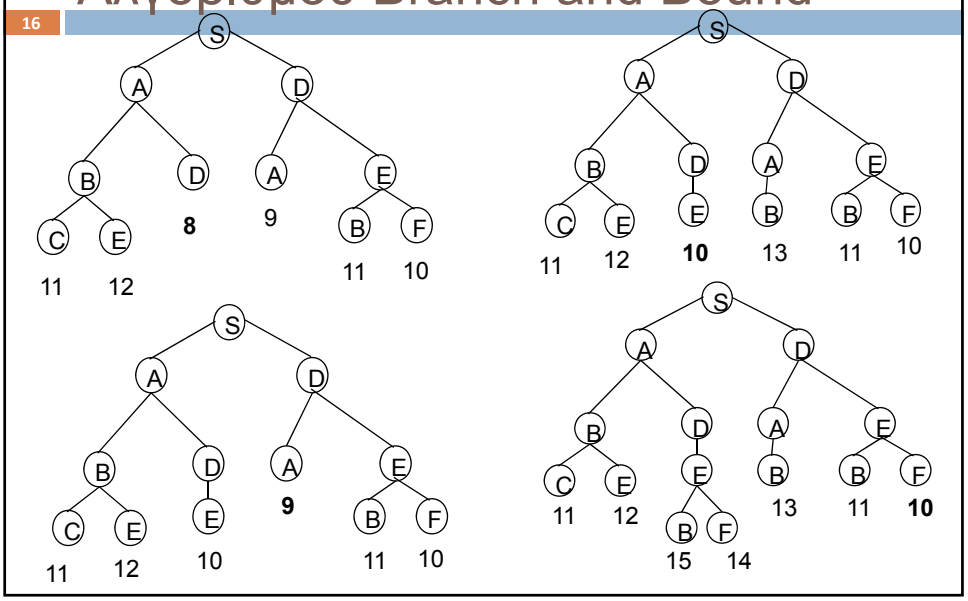
Παράδειγμα Εφαρμογής Αλγόριθμου Branch and Bound

15



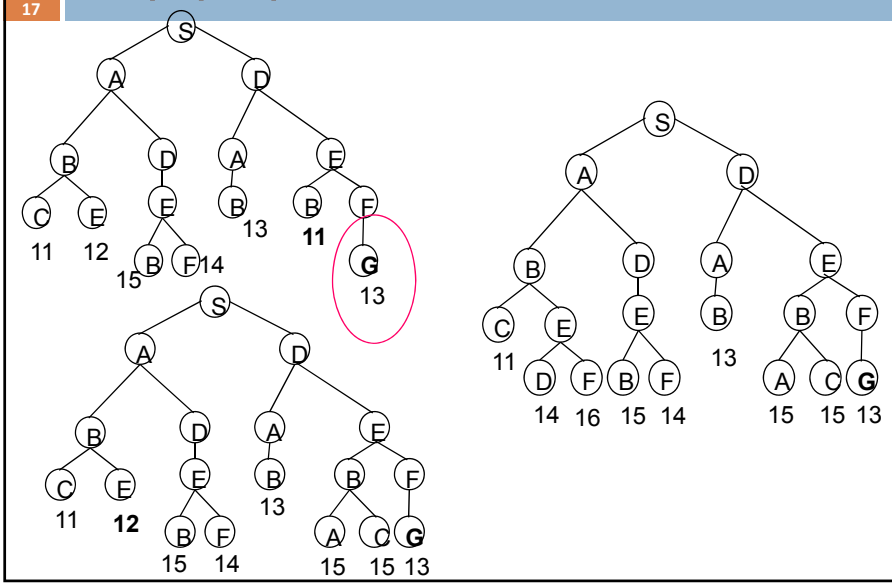
Παράδειγμα Εφαρμογής Αλγόριθμου Branch and Bound

16



Παράδειγμα Εφαρμογής Αλγόριθμου Branch and Bound

17



Branch and Bound και Υπολογιζόμενη Υπολειπόμενη Απόσταση από το Στόχο

18

- Εάν θεωρήσουμε μια επέκταση στον αλγόριθμο Branch and Bound με τη μορφή ότι το κόστος κάθε κόμβου δεν είναι μόνο το κόστος να φτάσουμε σε αυτό το κόμβο από τον αρχικό κόμβο, αλλά δίνεται από το άθροισμα του κόστους να φτάσουμε σε αυτό το κόμβο από τον αρχικό κόμβο συν μια υποεκτίμηση της υπολογιζόμενης υπολειπόμενης απόστασης του κόμβου από τον στόχο, τότε ο αλγόριθμος BB ελέγχει λιγότερα μονοπάτια
- Δηλαδή

  u(total path length) = d(already traveled) + u(distance remaining)

- Η παραπάνω έκφραση είναι μια υποεκτίμηση του πραγματικού κόστους που είναι

  e(total path length) = d(already traveled) + e(distance remaining)

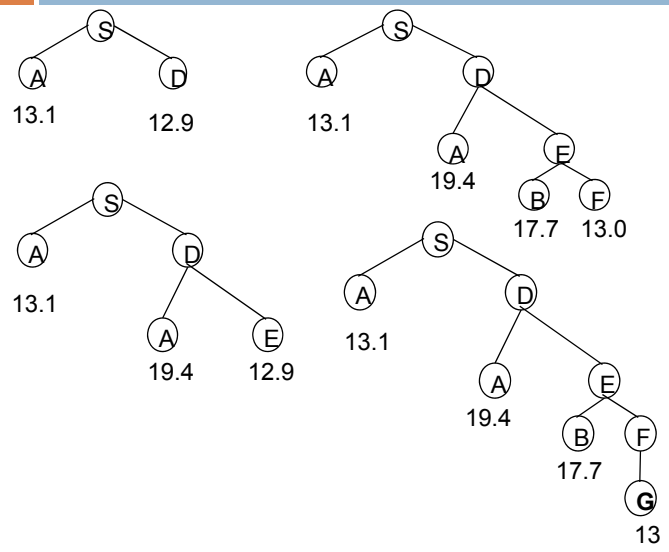
Αλγόριθμος Αναζήτησης B&B με Υποεκτίμηση Απόστασης

19

- **Βήμα 1**: Κατασκεύασε μια λίστα από μονοπάτια που αρχικά είναι κενή που περιέχει δηλαδή τα μονοπάτια από τη ρίζα του δένδρου (αρχική κατάσταση) στο πουθενά
- **Βήμα 2**: Μέχρι που η λίστα να αδειάσει ή, το πρώτο μονοπάτι στη λίστα οδηγεί σε «στόχο» και όλα τα άλλα μονοπάτια 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, Προχώρησε στο Βήμα 2.β
  - **Βήμα 2.β**: Εάν το πρώτο μονοπάτι δεν οδηγεί σε στόχο, ή υπάρχουν άλλα μονοπάτια δεν έχουν ακόμη οδηγήσει σε στόχο 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
    - 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 συνολικό κόστος του κάθε μονοπατιού, δηλαδή το κόστος που κάναμε να φτάσουμε από την αρχική κατάσταση στο τελευταίο κόμβο κάθε μονοπατιού συν το υπολογιζόμενη (υπο-εκτιμόμενο) απόσταση από το στόχο
    - Βήμα 2.β.5 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

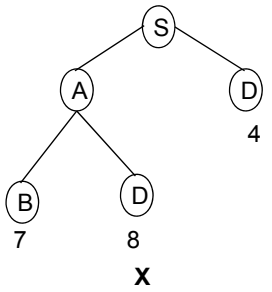
Παράδειγμα Εφαρμογής Αναζήτησης BB με Υποεκτίμηση Κόστους

20



Δυναμικός Προγραμματισμός στους Αλγόριθμους Αναζήτησης

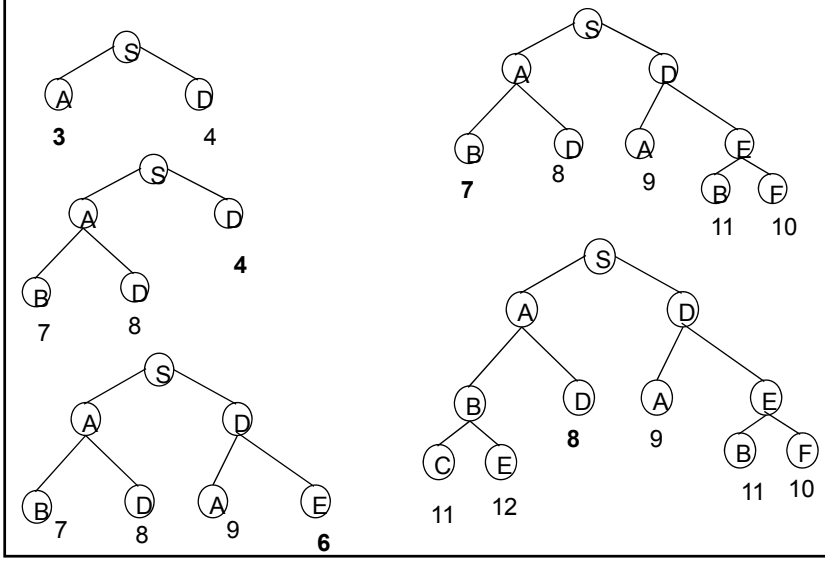
21



Η ιδέα είναι ότι εάν κατασκευάσουμε ένα μονοπάτι που οδηγεί σε κάποιο κόμβο Π.χ. D και υπάρχει κάποιο άλλο μονοπάτι που οδηγεί στον ίδιο κόμβο αλλά με μικρότερο κόστος, τότε «κλαδεύουμε» το μονοπάτι με το μεγαύτερο κόστος, και δεν το επεκτείνουμε πια.

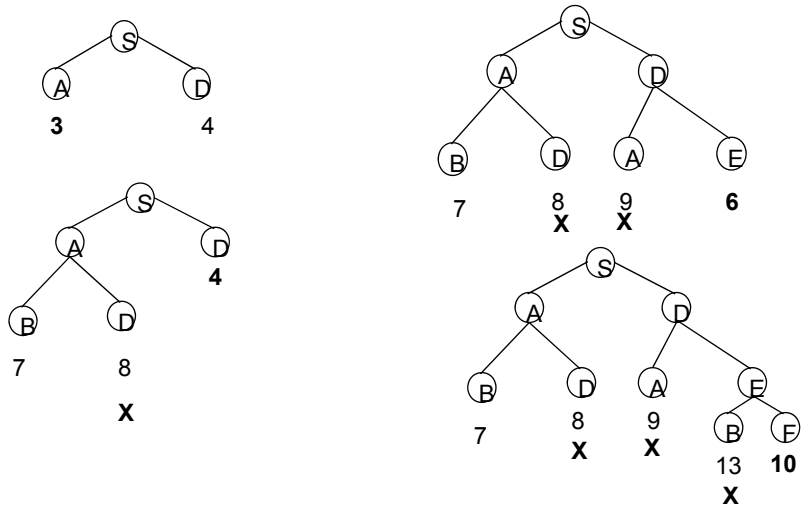
Παράδειγμα Εφαρμογής Αλγόριθμου Branch and Bound

22



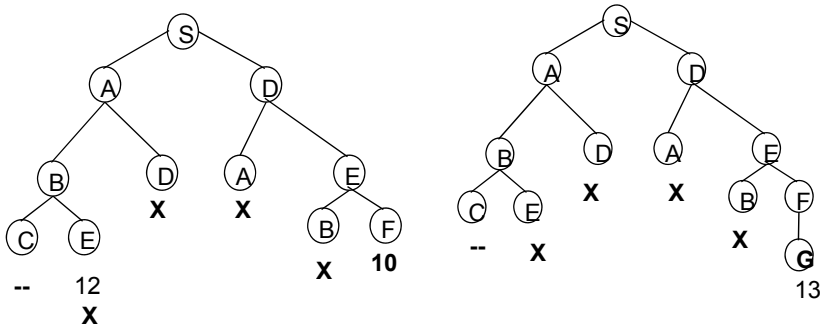
Εφαρμογή B&B με Δυναμικό Προγραμματισμό

23



Εφαρμογή B&B με Δυναμικό Προγραμματισμό

24



Αλγόριθμος A*

25

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

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

η g(S) δίνει την απόσταση της S από την αρχική κατάσταση, η οποία είναι πραγματική και γνωστή, καιη h(S) δίνει την εκτίμηση της απόστασης της S από την τελική κατάσταση μέσω μιας ευριστικής συνάρτησης, όπως ακριβώς στον BestFS

Αλγόριθμος A*

26

- 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)
- Δηλαδή ο A* είναι:

  A* = B&B + Χρήση Υπο-εκτιμητών + Δυναμικός Προγραμματισμός

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

Συνοπτική Σύγκριση Αλγόριθμων Αναζήτησης

Κριτήριο	Breadth-First	Branch and Bound (Uniform-Cost)	Depth-First	Depth-Limited	Iterative-Deepening	Bidirectional
Πλήρης?	Ναι	Ναι	Όχι	Όχι	Ναι	Ναι
Χρονική Πολυπλο-κότητα	$O(b^{d+1})$	$O(b^{1 + floor(C^*/\varepsilon)})$	$O(b^m)$	$O(b^{\prime})$	$O(b^d)$	$O(b^{d/2})$
Πολυπλο-κότητα Πόρων	$O(b^{d+1})$	$O(b^{1 + floor(C^*/\varepsilon)})$	$O(b^m)$	$O(b^{\prime})$	$O(b^d)$	$O(b^{d/2})$
Βέλτιστος?	Ναι [1] (1) Ίδια κόστη	Ναι	Όχι	Όχι	Ναι	Ναι [1,2] (2) Χρήση Κατά-Βάθος

b: Μέγιστος παράγοντας διακλάδωσης δένδρου αναζήτησης (πεπερασμένος αριθμός)
d: Ελάχιστο βάθος που βρίσκεται μια λύση του προβλήματος
m: Βάθος του δένδρου αναζήτησης
$\prime$: Μέγιστο επιτρεπτό βάθος αναζήτησης
C*: Το κόστος της βέλτιστης λύσης
ε: Σταθερά η οποία είναι μικρότερη από το κόστος κάθε κίνησης (π.χ. Ελάχιστη υπολειπόμενη απόσταση)