Μάθημα 6

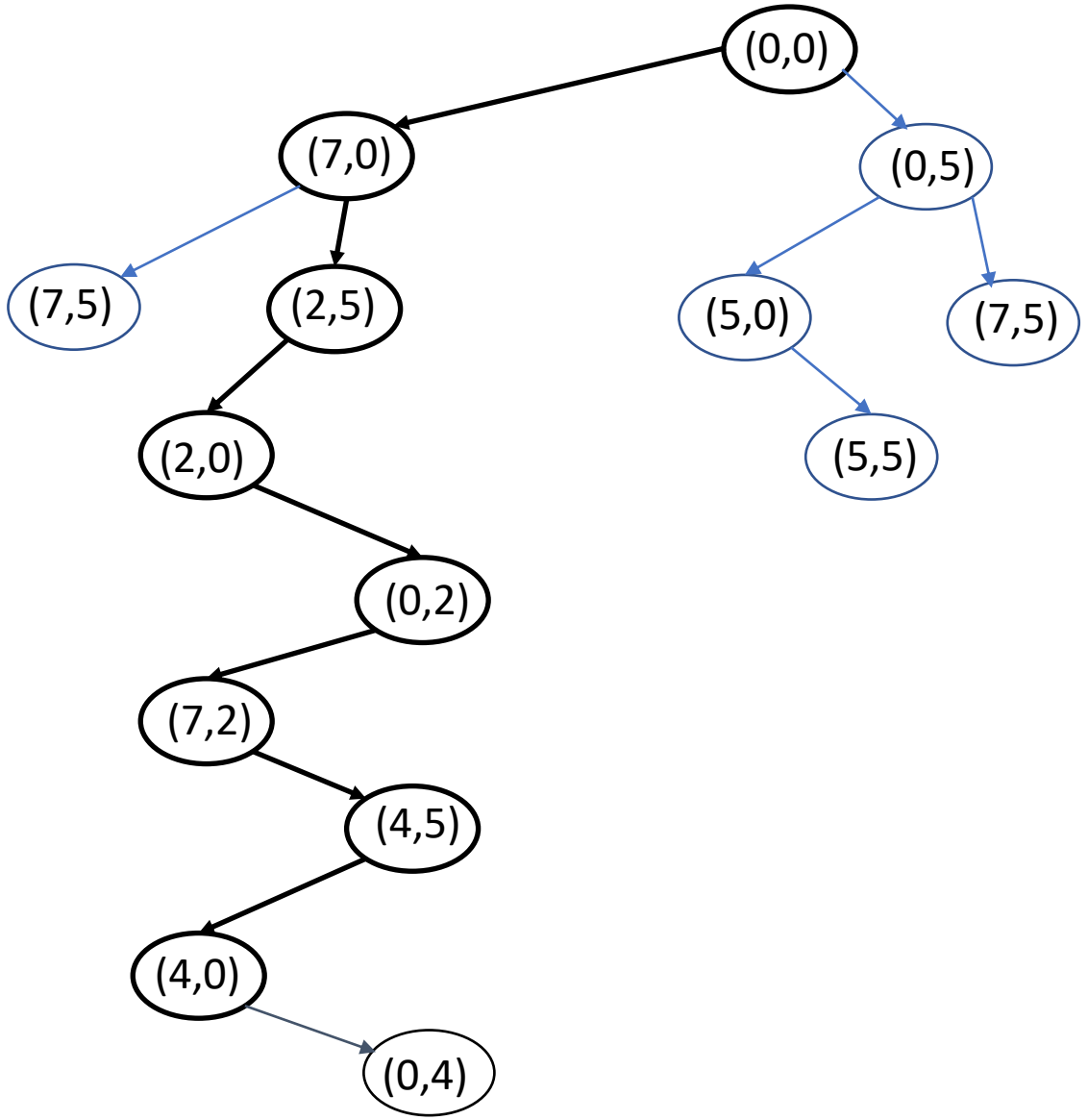
ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Αλγόριθμοι Αναζήτησης Λύσης (Συνέχεια)

Πηγές

1. S. Russell, P. Norvig, “Τεχνητή Νοημοσύνη: Μια σύγχρονη προσέγγιση”, Εκδόσεις Κλειδάριθμος, 2005 (Επιμέλεια Ι. Ρεφανίδης)
2. Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου. Τεχνητή Νοημοσύνη - Γ' Έκδοση, Εκδόσεις Πανεπιστημίου Μακεδονίας, 2011

Αλγόριθμος Αναζήτησης Πρώτα σε Βάθος-DFS: Το Πρόβλημα των Δοχείων



Αρχικοποίηση: S_0=(0,0), S_{F1}=(4,0), S_{F2}=(0,4)
Θέτουμε S_C=S_0 και E=min{E_1, E_2}, Δ=0.01
Υπολογίζουμε το σφάλμα: E_1=|0-4|+|0-0|=4 και E_2=|0-0|+|0-4|=4
και άρα: E=min{4,4}=4. Τέλος θέτουμε E_{Best}=4 και S_{Best}=(0,0)

S_C=(7,0), E_1=|7-4|+|0-0|=3 και E_2=|7-0|+|0-4|=11, E=min{3,11}=3
Άρα E_{Best}=3 και S_{Best}=(7,0)

S_C=(7,5), E_1=|7-4|+|5-0|=8 και E_2=|7-0|+|5-4|=8, E=min{8,8}=8
Άρα E_{Best}=3 και S_{Best}=(7,0)

S_C=(2,5), E_1=|2-4|+|5-0|=7 και E_2=|2-0|+|5-4|=3, E=min{7,3}=3
Άρα E_{Best}=3 και S_{Best}=(2,5)

S_C=(2,0), E_1=|2-4|+|0-0|=2 και E_2=|2-0|+|0-4|=6, E=min{2,6}=2
Άρα E_{Best}=2 και S_{Best}=(2,0)

S_C=(0,2), E_1=|0-4|+|2-0|=6 και E_2=|0-0|+|2-4|=2, E=min{2,6}=2
Άρα E_{Best}=2 και S_{Best}=(0,2)

S_C=(7,2), E_1=|7-4|+|2-0|=5 και E_2=|7-0|+|2-4|=9, E=min{5,9}=5
Άρα E_{Best}=2 και S_{Best}=(0,2)

S_C=(4,5), E_1=|4-4|+|5-0|=5 και E_2=|4-0|+|5-4|=5, E=min{5,5}=5
Άρα E_{Best}=2 και S_{Best}=(0,2)

S_C=(4,0), E_1=|4-4|+|0-0|=0 και E_2=|4-0|+|0-4|=8, E=min{0,8}=0
Άρα E_{Best}=0 και S_{Best}=(4,0)

Αλγόριθμος Αναζήτησης Πρώτα σε Βάθος-DFS

Αλγοριθμική Μεθοδολογία Εύρεσης Λύσης στον DFS

Ορίζουμε εμείς, ως σχεδιαστές/προγραμματιστές, μία κατάλληλη **<u>συνάρτηση σφάλματος</u>**, η οποία είναι το **<u>σφάλμα</u>** μεταξύ μιας κατάστασης και της κατάστασης στόχο (δηλ. τελικής κατάστασης) που θέσαμε ευθύς εξαρχής στον ορισμό του προβλήματος
Το σφάλμα αυτό μας δείχνει **<u>πόσο μακριά</u>** είναι οι δύο αυτές καταστάσεις μεταξύ τους. Άρα, είναι επιθυμητό να είναι το σφάλμα να είναι όσο πιο μικρό γίνεται με ιδανική κατάσταση να είναι 0. Η συνάρτηση αυτή δίνεται ως εξής:

$$E = \left| S_C - S_F \right|$$

E :Συνάρτηση σφάλματος, S_C :Τρέχουσα κατάσταση, S_F :Τελική κατάσταση

Παρατήρηση: Προσέξτε ότι όταν η τρέχουσα κατάσταση είναι ίδια με την τελική τότε $S_C = S_F$ και άρα $E = \left| S_F - S_F \right| = 0$
Όταν μία κατάσταση είναι κοντά στην τελική τότε το σφάλμα θα είναι πολύ μικρό.

Αλγόριθμος

Βήμα 1. Ο αλγόριθμος, κάθε φορά που επισκέπτεται έναν κόμβο, υπολογίζει το παραπάνω **<u>σφάλμα</u>** μεταξύ της τρέχουσας κατάστασης, την οποία ελέγχει αυτή τη στιγμή, και της τελικής κατάστασης

Βήμα 2. Συγκρίνει την παραπάνω τιμή του σφάλματος με το μικρότερο σφάλμα που έχει βρει μέχρι αυτή την στιγμή και από τα δύο αυτά σφάλματα επιλέγει το πιο μικρό και τον αντίστοιχο κόμβο/κατάσταση

Βήμα 3. Τα Βήματα 1 και 2 επαναλαμβάνονται μέχρι ο αλγόριθμος να διατρέξει όλο το δέντρο ή μέχρι να βρεθεί μία τιμή σφάλματος που είναι μικρότερη από έναν μικρό αριθμό που θέτουμε εμείς ως σχεδιαστές, ο οποίος στο πλαίσιο του μαθήματος συμβολίζεται ως Δ

Αλγόριθμος Αναζήτησης Πρώτα σε Βάθος-DFS

Το πρόβλημα των δοχείων

Αρχική Κατάσταση: $S_0=(S_{01}, S_{02})= (0, 0)$ Τελικές Καταστάσεις: $S_{F1} =(S_{F11}, S_{F12})=(4, 0)$ ή $S_{F2} =(S_{F21}, S_{F22})=(4, 0)$

Ορισμός Συνάρτησης Σφάλματος

- Κάθε κατάσταση που επισκέπτεται ο αλγόριθμος έχει δύο συνιστώσες μία για κάθε δοχείο και την συμβολίζουμε ως: $S_C =(S_{C1}, S_{C2})$
- Έχουμε δύο πιθανές τελικές καταστάσεις S_{F1} ή S_{F2}, αλλά δεν ξέρουμε σε ποια θα φτάσει πρώτα ο αλγόριθμος αναζήτησης λύσης. Οπότε, για την τρέχουσα κατάσταση (δηλ. κόμβο) που εξετάζει αυτή τη στιγμή ο αλγόριθμος υπολογίζουμε την συνάρτηση σφάλματος για κάθε μία από τις τελικές καταστάσεις:

$$E_1 = |S_C - S_{F1}| = |S_{C1} - S_{F11}| + |S_{C2} - S_{F12}| \qquad E_2 = |S_C - S_{F2}| = |S_{C1} - S_{F21}| + |S_{C2} - S_{F22}|$$

Αν υποθέσουμε ότι ο αλγόριθμος φτάνει κάποια στιγμή στην κατάσταση $S_C = (4,0)$ η οποία είναι η S_{F1} τότε

$$E_1 = |S_C - S_{F1}| = |4-4| + |0-0| = 0 \qquad E_2 = |S_C - S_{F2}| = |4-0| + |0-4| = 8$$

Παρατηρήστε ότι το πρώτο σφάλμα είναι 0 αλλά το δεύτερο είναι πολύ μεγάλο και ίσο με 8. Όμως το πρόβλημα λύθηκε γιατί φτάσαμε στην πρώτη κατάσταση. Συνεπώς η σωστή επιλογή είναι να διαλέγουμε κάθε φορά τη=ο μικρότερο σφάλμα από τα δύο ως εξής:

$$E = \min\{E_1, E_2\}$$

Αυτή η συνάρτηση αναφέρεται μόνο στο συγκεκριμένο πρόβλημα. Σε άλλο πρόβλημα θα πάρουμε άλλη συνάρτηση σφάλματος

Αλγόριθμος Αναζήτησης Πρώτα σε Βάθος-DFS

Αλγόριθμος DFS: Γενική Περιγραφή

Αρχικοποίηση

- Ορίζουμε την αρχική κατάσταση S_0 και την τελική κατάσταση S_F
- Ορίζουμε την συνάρτηση σφάλματος, και μία μικρή τιμή Δ
- Θέτουμε ως $S_C=S_0$ και υπολογίζουμε το σφάλμα $E=|S_C- S_F|$
- Επειδή αρχίζουμε τώρα, θέτουμε ως το καλύτερο (δηλ. πιο μικρό) σφάλμα που έχουμε ήδη βρει: $E_{Best}=E$ και ως καλύτερο κόμβο τον κόμβο της ρίζας $S_{Best}=S_0$

While ($E_{best}>\Delta$) **Or** (οι κόμβοι του δέντρου δεν έχουν ελεγχθεί όλοι) **Do**

Διέτρεξε τους κόμβους του δέντρου σύμφωνα με την στρατηγική το DFS (την δείξαμε σε προηγούμενο μάθημα)

Κάθε φορά που επισκέπτεσαι έναν κόμβο S_C υπολόγισε το σφάλμα $E=|S_C- S_F|$

If E< = E_{best} **Then**
$E_{best}=E$
$S_{Best}=S$
EndIf

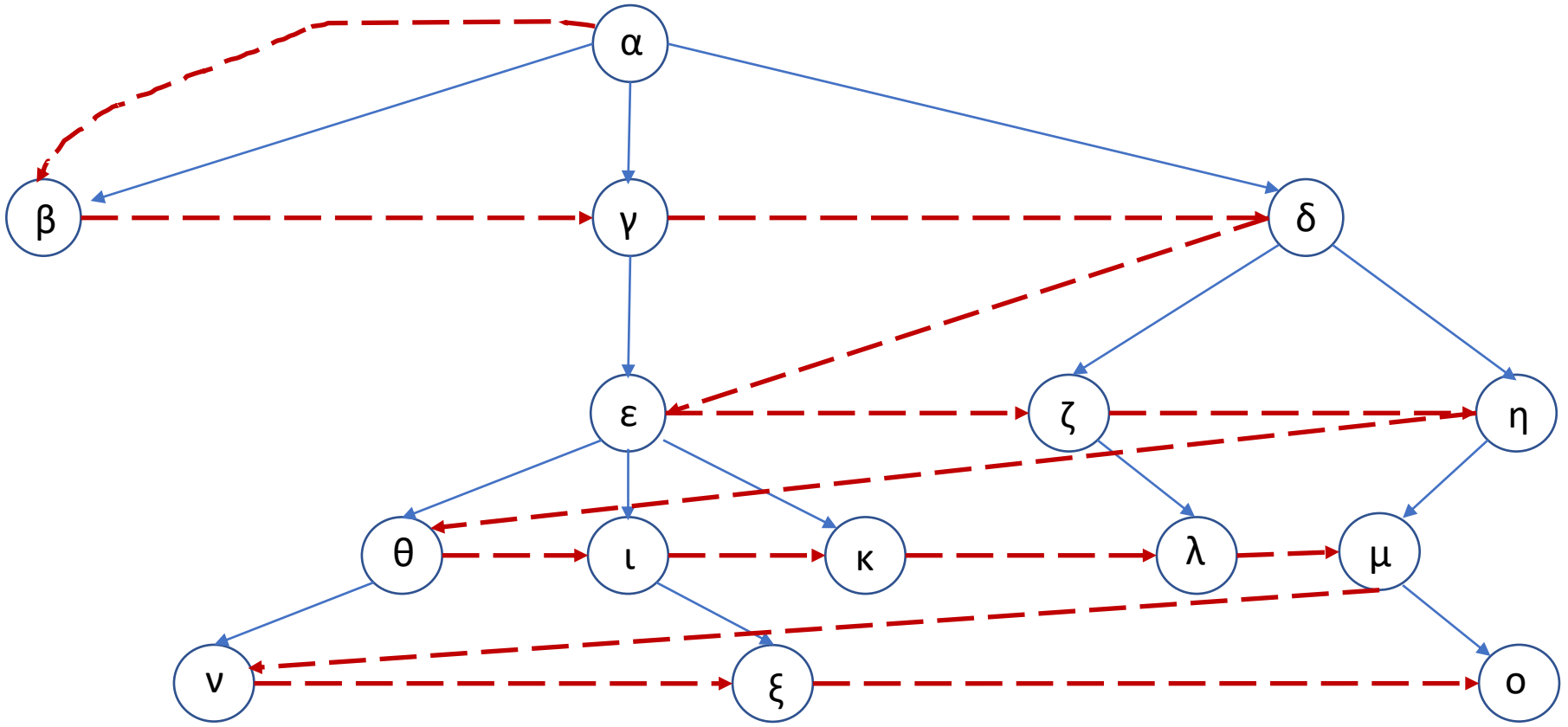
EndWhile

Αλγόριθμος Αναζήτησης Πρώτα σε Πλάτος (Breadth First Search Algorithm-BFS)

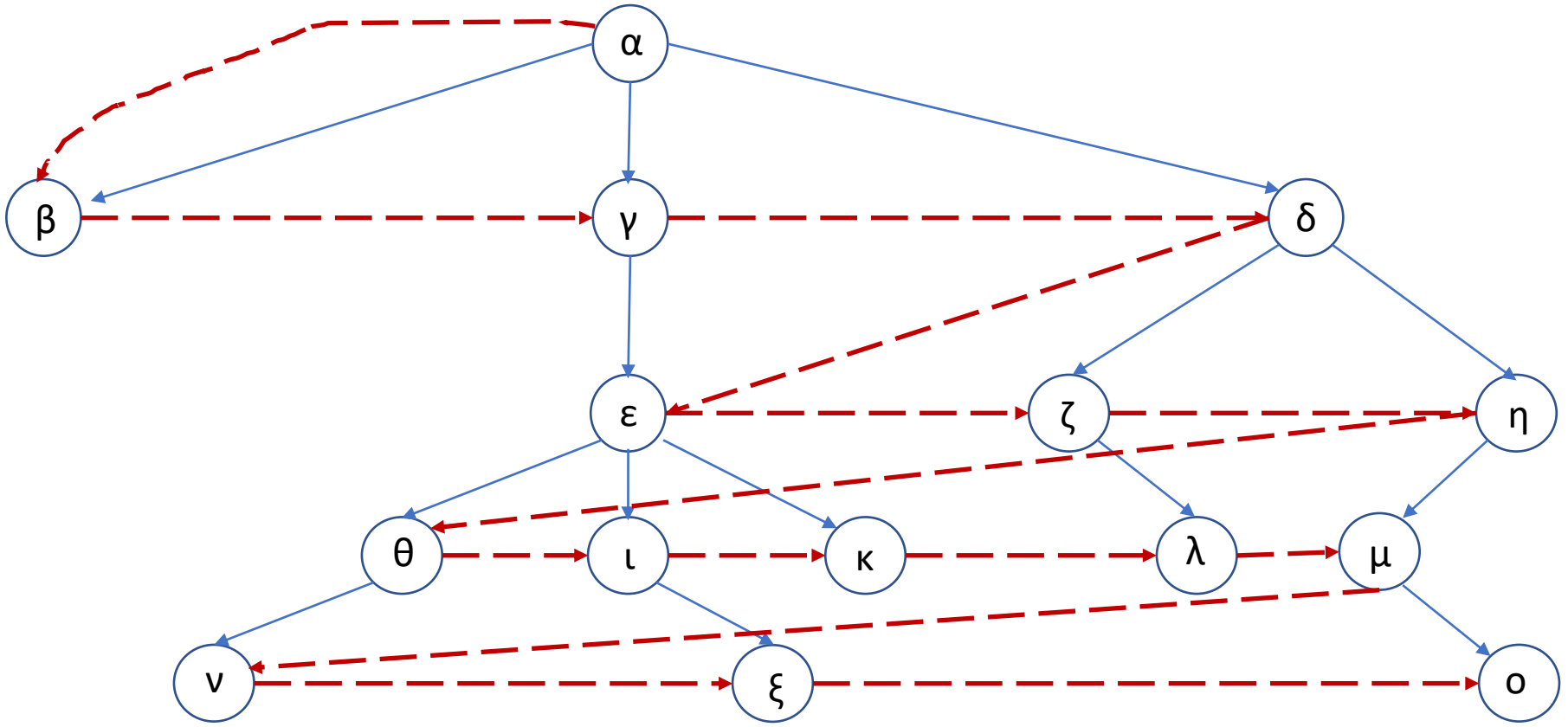
Ο αλγόριθμος BFS διατρέχει το δέντρο αναζήτησης λύσης του προβλήματος ως εξής:

- Αρχίζει από την ρίζα και επιλέγει το αριστερότερο κόμβο-παιδί της ρίζας
- Διατρέχει όλο το επίπεδο που αντιστοιχεί στον κόμβο αυτόν και φτάνει στον δεξιότερο κόμβο (που ταυτόχρονα και ο δεξιότερος κόμβος-παιδί της ρίζας)
- Στην συνέχεια πηγαίνει στον αριστερότερο κόμβο του επόμενου επιπέδου
- Διατρέχει όλο το επίπεδο μέχρι τον δεξιότερο κόμβο
- Επαναλαμβάνει την παραπάνω διαδικασία μέχρι να επισκεφτεί όλα τα επίπεδα και όλους τους κόμβους του δέντρου
- Για κάθε κόμβο που επισκέπτεται υπολογίζει το σφάλμα και το συγκρίνει με το καλύτερο που έχει βρει μέχρι στιγμής. Αν είναι πιο μικρό τότε επιλέγεται αυτός ο κόμβος ως η καλύτερη λύση
- Ο αλγόριθμος σταματάει αν βρεθεί η λύση που μας ικανοποιεί ή αν έχει διατρέξει όλο το δέντρο

Αλγόριθμος Αναζήτησης Πρώτα σε Πλάτος
(Breadth First Search Algorithm-BFS)



Αλγόριθμος Αναζήτησης Πρώτα σε Πλάτος
(Breadth First Search Algorithm-BFS)



Αλγόριθμος Αναζήτησης Πρώτα σε Πλάτος (Breadth First Search Algorithm-BFS)

Αρχικοποίηση

- Ορίζουμε την αρχική κατάσταση S_0 και την τελική κατάσταση S_F
- Ορίζουμε την συνάρτηση σφάλματος, και μία μικρή τιμή Δ
- Θέτουμε ως $S_C=S_0$ και υπολογίζουμε το σφάλμα $E=|S_C- S_F|$
- Επειδή αρχίζουμε τώρα, θέτουμε ως το καλύτερο (δηλ. πιο μικρό) σφάλμα που έχουμε ήδη βρει: $E_{Best}=E$ και ως καλύτερο κόμβο τον κόμβο της ρίζας $S_{Best}=S_0$

While ($E_{best}>\Delta$) **Or** (οι κόμβοι του δέντρου δεν έχουν ελεγχθεί όλοι) **Do**

Διέτρεξε τους κόμβους του δέντρου σύμφωνα με την στρατηγική το BFS

Κάθε φορά που επισκέπτεσαι έναν κόμβο S_C υπολόγισε το σφάλμα $E=|S_C- S_F|$

If E< = E_{best} **Then**
$E_{best}=E$
$S_{Best}=S$
EndIf

EndWhile

Αλγόριθμος Αναζήτησης Πρώτα σε Πλάτος
(Breadth First Search Algorithm-BFS)

Η κύρια διαφορά μεταξύ του BFS και του DFS είναι στο μέτωπο αναζήτησης.
Σε γενικές γραμμές ο BFS έχει μικρότερο μέτωπο αναζήτησης από τον BFS και αυτό τον κάνει ταχύτερο

Πλεονεκτήματα του BFS

- Βρίσκει πάντα την πιο καλή λύση εφόσον αυτή υπάρχει
- Είναι πολύ καλός στο να λύνει προβλήματα μικρά σε μέγεθος ή μέτρια

Μειονεκτήματα του DFS

- Μεγάλο υπολογιστικό κόστος
- Ανάγκη για μεγάλο μέγεθος μνήμης

Σε γενικές γραμμές για μικρού μεγέθους δέντρα προτιμάται ο DFS ενώ για μεγάλου μεγέθους δέντρα προτιμάται ο BFS

Καλό Απόγευμα