Πανεπιστήμιο Αιγαίου
Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Διδάσκων: Σταματάτος Ευστάθιος

Αλγόριθμοι Τυφλής και Ευρετικής Αναζήτησης

Εργαστηριακοί Συνεργάτες: Σταματάτος Ευστάθιος

Χουβαρδάς Αντώνης - icsd17217

Σάμος 30/04/2021

Εκφώνηση

Θεωρούμε το πρόβλημα όπου n άνθρωποι βρίσκονται στη μία όχθη ενός ποταμού και θέλουν να περάσουν όλοι στην απέναντι όχθη μέσω μιας στενής γέφυρας.
Η γέφυρα μπορεί να αντέξει μέχρι 2 άτομα κάθε φορά.
Επίσης, είναι νύχτα και έχουν μόνο ένα φακό που πρέπει να κρατάει κάποιος από αυτούς που περνούν την γέφυρα.
Ο κάθε άνθρωπος έχει την ικανότητα να περάσει τη γέφυρα με καθορισμένη ταχύτητα.
Όταν 2 άνθρωποι διασχίζουν την γέφυρα μαζί, ο καθένας πρέπει να προσαρμόζει την ταχύτητά του σε αυτή του πιο αργού ανθρώπου.
Όταν 2 άνθρωποι διασχίζουν την γέφυρα μαζί δεν έχει σημασία ποιος κρατάει το φακό.
Ας υποθέσουμε ότι έχουμε 4 ανθρώπους: Α1, Α2, Α3, και Α4.

- Ο Α1 μπορεί να περάσει τη γέφυρα σε 1 λεπτό,
- ο Α2 σε 2 λεπτά,
- ο Α3 σε 5 λεπτά
- ο Α4 σε 10 λεπτά.

Αν ο Α1 και ο Α4 διασχίζουν την γέφυρα μαζί, θα χρειαστούν 10 λεπτά. Για το παράδειγμα αυτό, μία λύση είναι η εξής:

1. Ο Α1 και ο Α4 διασχίζουν την γέφυρα σε 10 λεπτά.
2. Ο Α1 γυρίζει πίσω σε 1 λεπτό.
3. Ο Α1 και ο Α3 διασχίζουν την γέφυρα σε 5 λεπτά.
4. Ο Α1 γυρίζει πίσω σε 1 λεπτό.
5. Ο Α1 και ο Α2 διασχίζουν την γέφυρα σε 2 λεπτά.

Ο συνολικός χρόνος αυτής της λύσης είναι 19 λεπτά. Προσέξτε ότι δεν είναι η καλύτερη δυνατή λύση αφού υπάρχει και τρόπος να περάσουν όλοι απέναντι σε 17 λεπτά.

Περιγραφή Προβλήματος

Ακολουθεί μία σύντομη περιγραφή του προβλήματος. Θα υποθέσουμε ότι θέλουμε να περάσουμε τα άτομα από τη δεξιά μεριά μιας γέφυρας στην αριστερή μεριά της γέφυρας. Είναι παρατηρήσει που βοήθησαν στη λύση του προβλήματος.

1. Δε γίνεται να περάσουν τη γέφυρα πάνω από 2 άτομα.
2. Μέχρι να μείνει ένας στην αριστερή μεριά, πρέπει κάθε φορά που περνούν το πολύ 2 άτομα από τα αριστερά στα δεξιά, να γυρνάει πίσω τουλάχιστον 1 άτομο και το πολύ 2 άτομα(Για να γυρίσει πίσω ο φακός ώστε να γίνει και άλλη μεταφορά).
3. Όταν περνάει ένα άτομο τη γέφυρα, το κάνει στο χρόνο που του αντιστοιχεί.
4. Όταν περνούν 2 άτομα τη γέφυρα, το κάνουν στο μεγαλύτερο χρόνο μεταξύ αυτών των 2 ατόμων.
5. Κάθε φορά που γίνεται μία μεταφορά από τα αριστερά στα δεξιά ή αντίθετα ο χρόνος θα προστίθεται στο άθροισμα των μεταφορών.
6. Αρχική κατάσταση είναι όταν η αριστερή μεριά είναι “γεμάτη”
7. Κατάσταση στόχου είναι όταν η αριστερή μεριά είναι “άδεια”.

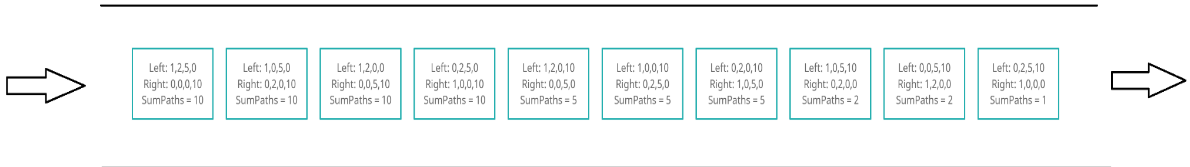
Για τις ανάγκες της αναφοράς όλα τα παραδείγματα θα είναι στην ίδια περίπτωση με το παράδειγμα που έχουμε 4 άτομα και σε αυτά τα άτομα αντιστοιχούν οι χρόνοι 1,2,5 και 10 αντίστοιχα.

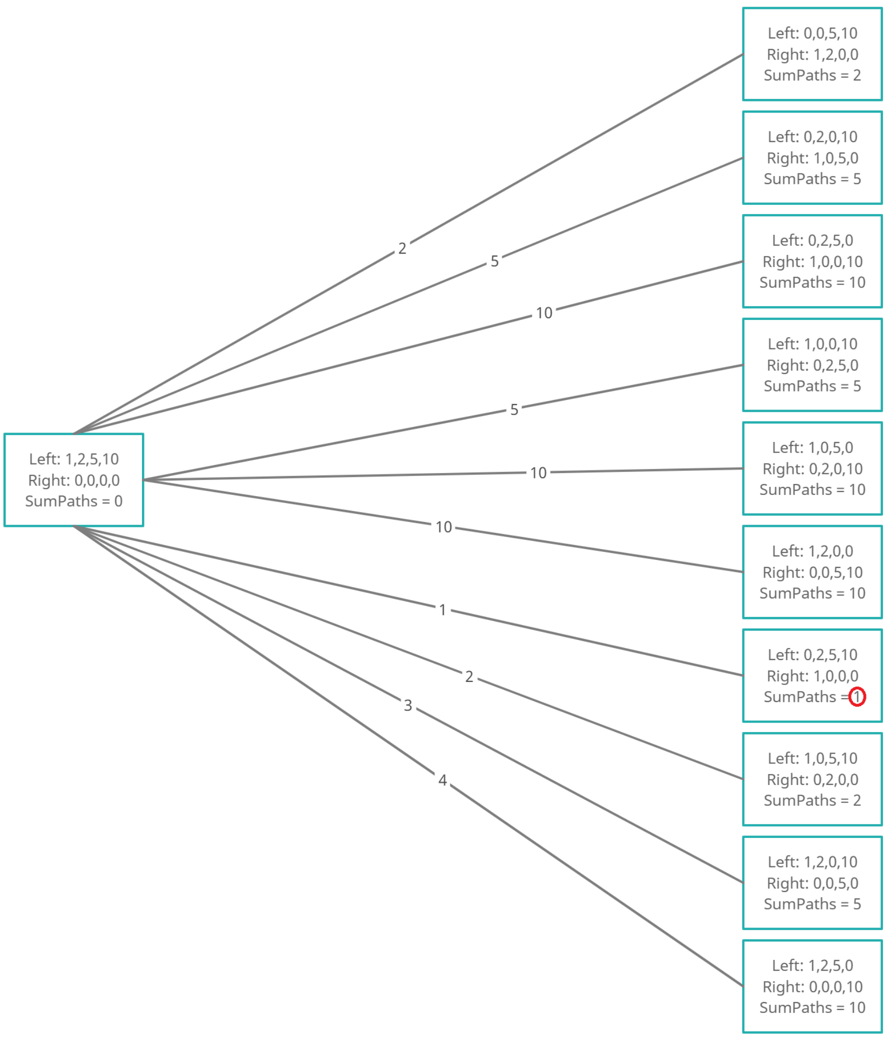
Αλγόριθμοι Τυφλής Αναζήτησης

Αναζήτηση Ενιαίου Κόστους

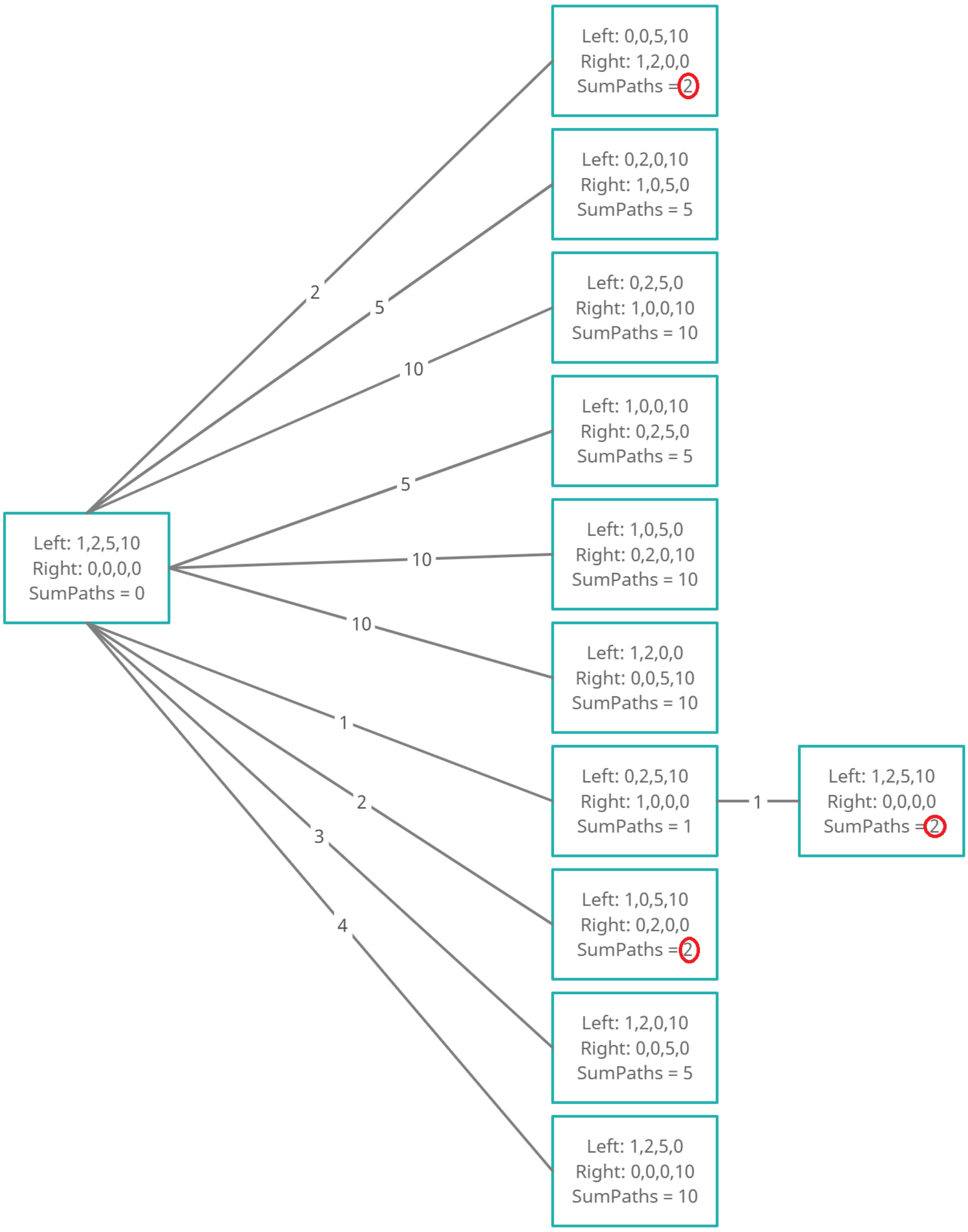
Στη συγκεκριμένη περίπτωση αναφερόμαστε σε μία παραλλαγή της αναζήτησης κατά πλάτος. Η παραλλαγή είναι ότι θα λαμβάνουμε υπόψη, κάθε φορά πριν επεκτείνουμε ένα κόμβο, το συνολικό κόστος που έχει παραχθεί προσθέτοντας τους χρόνους κάθε φορά που κάνουμε μία μεταφορά.

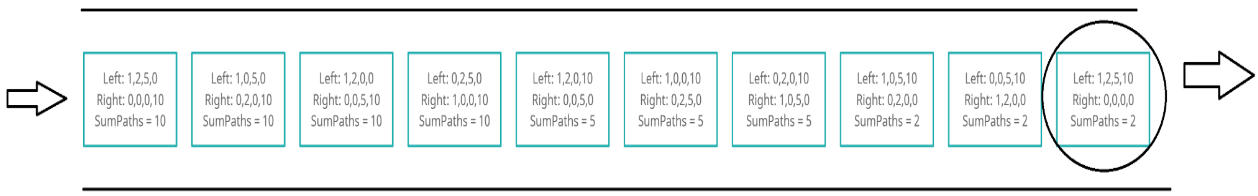
1. Αρχικά ξεκινάει ο αλγόριθμος και επεκτείνει το root node και δημιουργεί όλα τα πιθανά παιδιά και τα περνάει αυτομάτως σε ένα priority queue.
2. Στη συνέχεια παίρνουμε το πρώτο στοιχείο του priority queue και κάνουμε την ίδια διεργασία. Με κόκκινο στην πρώτη φωτογραφία.





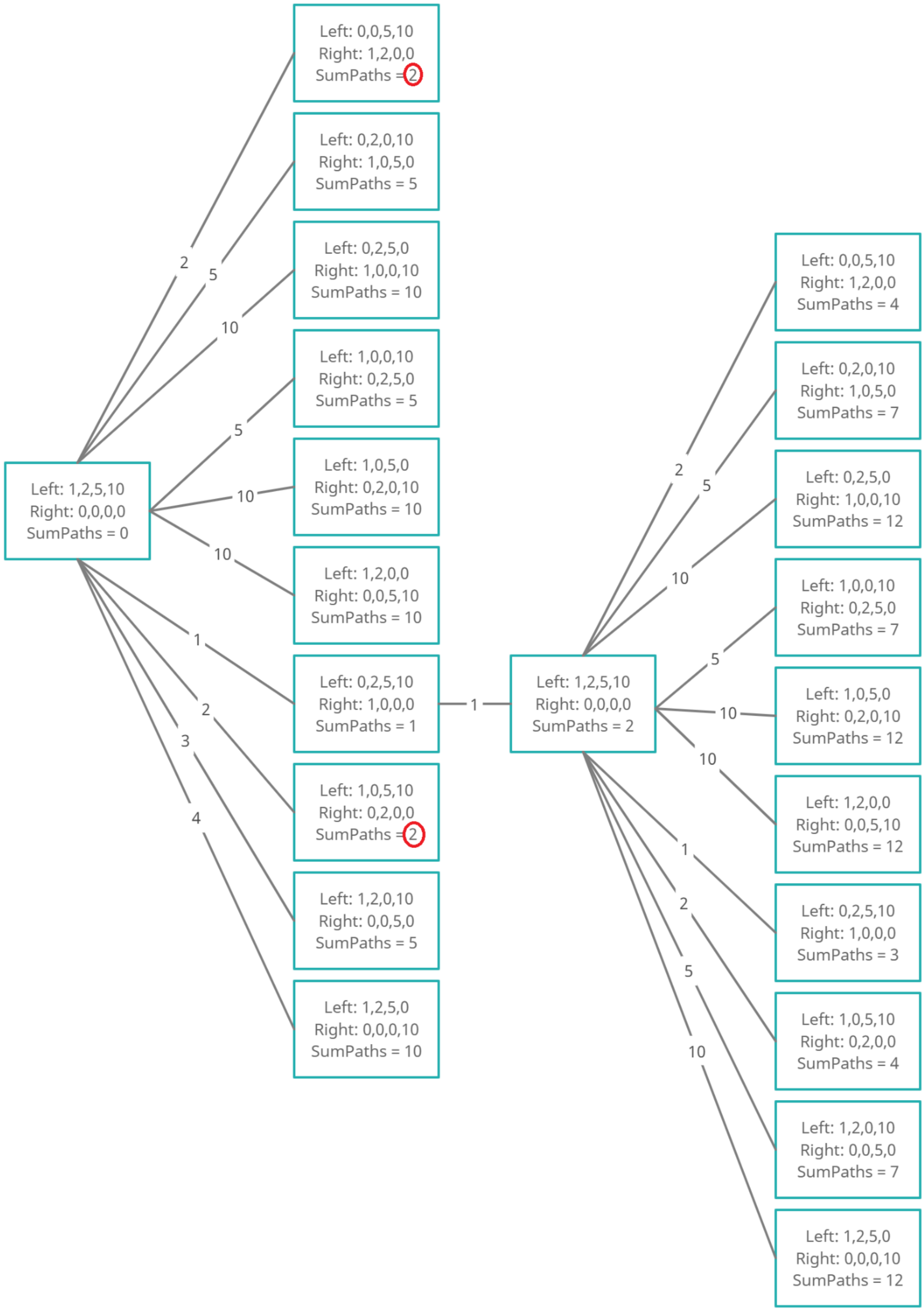
Στη συνέχεια έχουμε 3 Nodes με το ίδιο κόστος(Με κοκκινο στην από πάνω φωτογραφία). Βγάζουμε το πρώτο απο το priority queue μας που έχει κοστος 2.



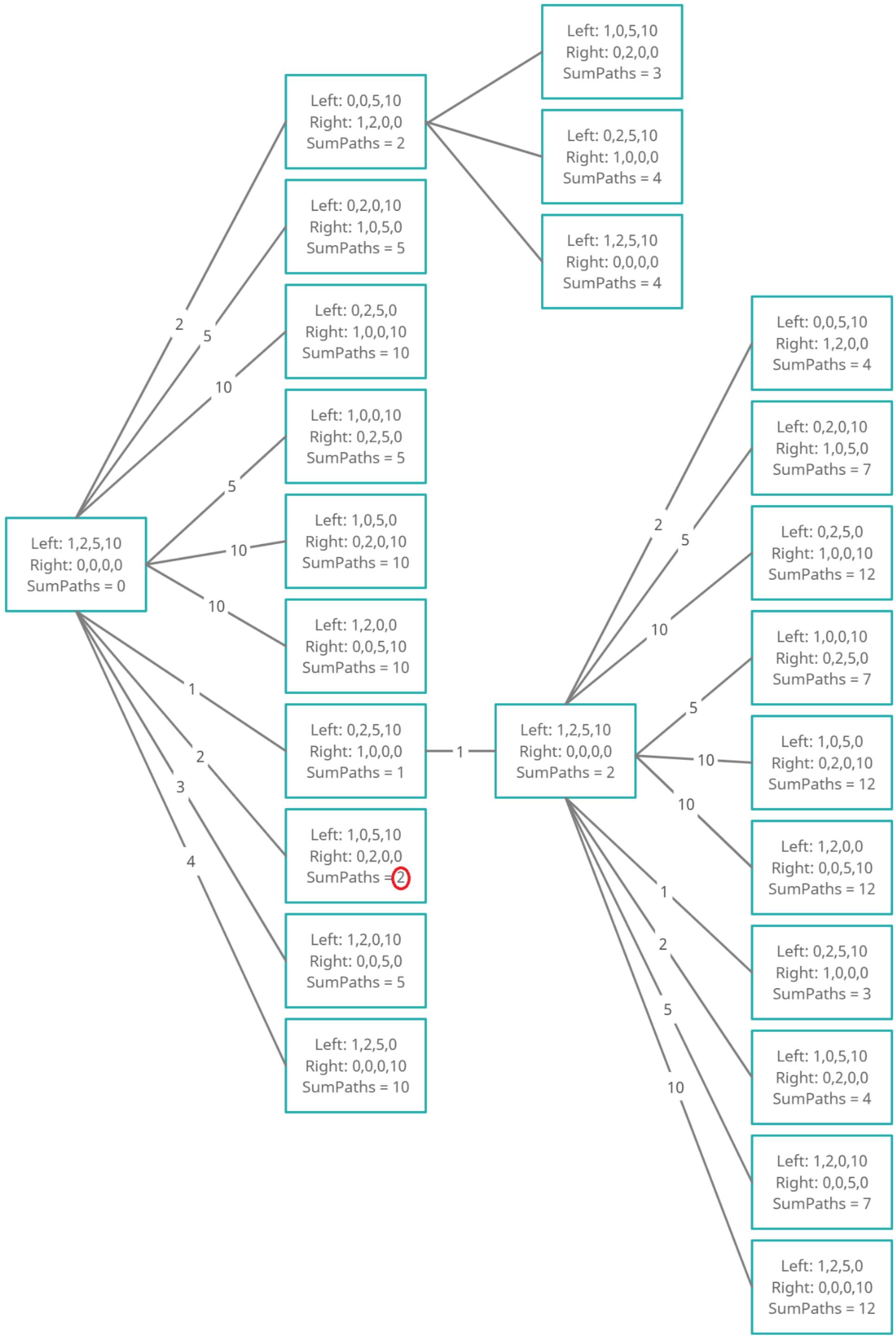


Παρατηρούμε ότι το καινούριο Node που είναι και πρώτο στη priority queue,άρα είναι και αυτό που μας επιστρέφει ο αλγόριθμός μας για επέκταση, έχει τη μορφή του root node.

Οπότε έχουμε το παρακάτω αποτέλεσμα. Τα Nodes με κόκκινο είναι αυτα που είναι τώρα πρώτα στη priority queue μας οπότε είναι και αυτά από τα οποία θα μας επιστρέψει ο αλγόριθμός μας ώστε να επεκτείνουμε. Εδώ παρατηρούμε ότι η priority queue μας έχει γίνει πολύ μεγάλη ωστε να την αναπαραστησουμε.



Έστω ότι πρώτο στην priority queue μας είναι το Node με κόστος 2 και left μεριά 0,0,5,10. Ο agent μας επεκτείνε το Node και παράγει τα παιδιά του. Και έχουμε το παρακάτω αποτέλεσμα. Τώρα το Node με το μικρότερο κόστος που θα επεκταθεί στη συνέχεια είναι αυτο με το κόκκινο κύκλο στο παρακάτω σχήμα.

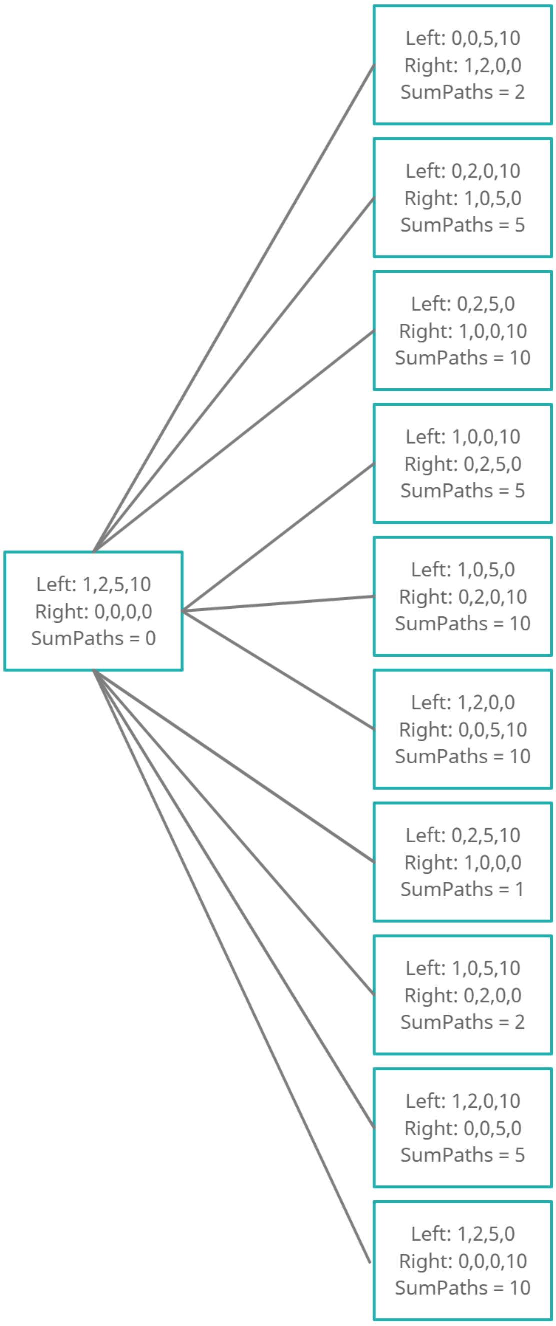


Έτσι θα συνεχίσει το πρόγραμμα μας ώστε να βρει τελικά την βέλτιστη διαδρομή.Παρατηρούμε, έχοντας υπόψη οτι η βέλτιστη διαδρομή για αυτά τα νούμερα είναι 17, ότι ο αλγόριθμός μας θα επεκτείνεις όλα αυτά τα Nodes.

Αλγόριθμος Επαναληπτική Εκβάθυνσης(IDS)

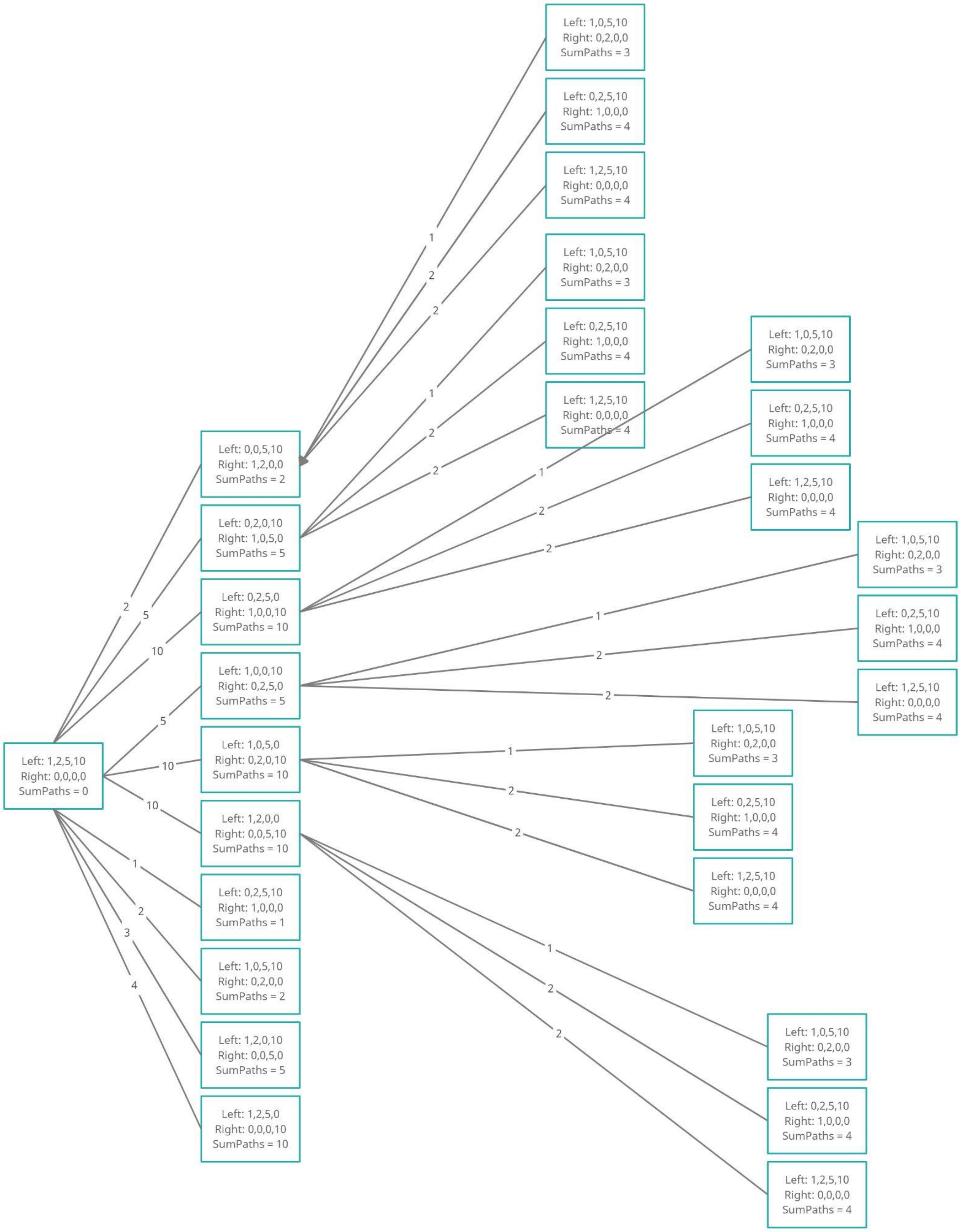
Ο συγκεκριμένος αλγόριθμος μας βρίσκει λύση αλλά όχι τη βέλτιστη. Είναι ένας συνδυασμός του αλγόριθμου κατα βάθος και του αλγόριθμου κατά πλάτος. Ο αλγόριθμος αυτός καλεί τον αλγόριθμο κατα βάθος για αυξανόμενα βάθη ξεκινώντας απο το 0 που είναι το βάθος της ρίζας. Αν δε βρούμε στόχο επεκτείνουμε όλα τα nodes στο βάθος που βρισκόμαστε. Συνεχίζουμε έτσι μέχρις ότου βρεθούμε σε βάθος που βρίσκεται λύση. Παρακάτω παρουσιάζεται με φωτογραφίες ο τρόπος που βρήκε ο αλγόριθμος λύση στο πρόβλημα μας.

1. Ξεκινάμε και επεκτείνουμε τη ρίζα εφόσον η ρίζα δεν είναι η λύση.



2. Τσεκάρουμε όλα τα καινούρια Nodes του καινούριου βάθους και αν δεν βρίσκεται εδώ η λύση επεκτείνουμε όλα τα nodes στο επόμενο βάθος.Εδώ το συνολικό κόστος δεν

παίζει κανένα ρόλο



3. Αν πάλι δεν υπάρχει κόμβος στόχου σε αυτό το βάθος επεκτείνουμε και τα καινούρια nodes του νέου μας βάθους και ελέγχουμε αν είμαστε σε βάθος που έχει κόμβο στόχου.

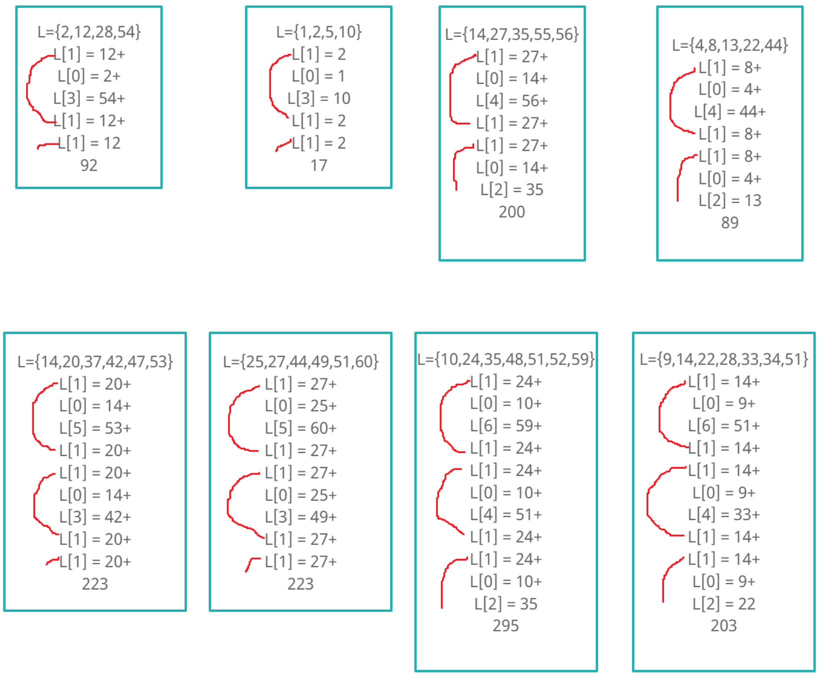
Αλγόριθμοι Ευρετικής και Τοπικής Αναζήτησης

Οι αλγόριθμοι αυτοί χρησιμοποιούν μία ευρετική συνάρτηση ώστε να υπολογίσουν έτσι το κόστος που θα χρειαστεί μέχρι να φτάσουν στο στόχο. Παρακάτω θα αναλυθεί η σκέψη που οδήγησε στην ευρετική συνάρτηση που χρησιμοποιήθηκε στην συγκεκριμένη άσκηση.

Ευρετική Συνάρτηση

Για να υλοποιηθεί η ευρετική συνάρτηση βοήθησε η ιστοσελίδα (https://learninglover.com/examples.php?id=81).
Βρίσκοντας τη βελτιστη λύση χρησιμοποιόντας την ιστοσελιδα αυτή παρατηρήθηκα 3 μοτίβα που αναλογα με το πλήθος των αριθμών και τι αριθμοι ήταν αυτοί επαναλαμβάνονται.Τα μοτίβα αυτά παρουσιάζονται παρακάτω.

1. Αρχικά στο πρώτο μοτίβο παρατηρούμε ότι ο μικρότερος αριθμός και ο δεύτερος μικρότερος χρησιμοποιούνται για να μεταφερθούν απο την άλλη μεριά της γέφυρας οι μεγαλύτεροι αριθμοι αν δυο ώστε να μη μετρηθουν και οι 2 στο τελικό άθροισμα.



2. Στο δεύτερο μοτίβο παρατηρείται ότι χρησιμοποιείται ο μικρότερος αριθμός ώστε να μεταφερθουν οι μεγαλύτεροι αριθμοι ενας ενας στην άλλη μεριά.

L={1,24,45,52}
L[3] = 52
L[0] = 1
L[2] = 45
L[0] = 1
L[1] = 24
123

L={12,36,43,54}
L[3] = 54
L[0] = 12
L[2] = 43
L[0] = 12
L[1] = 36
157

L={13,34,35,43,54}
L[4] = 54+
L[0] = 13+
L[3] = 43+
L[0] = 13+
L[2] = 35+
L[0] = 13+
L[1] = 34
205

L={10,38,39,43,48}
L[4] = 48+
L[0] = 10+
L[3] = 43+
L[0] = 10+
L[2] = 39+
L[0] = 10+
L[1] = 38
198

L={17,39,45,47,50,57}
L[5] = 57+
L[0] = 17+
L[4] = 50+
L[0] = 17+
L[3] = 47+
L[0] = 17+
L[2] = 45+
L[0] = 17+
L[1] = 39
306

L={17,33,34,39,44,47}
L[5] = 47+
L[0] = 17+
L[4] = 44+
L[0] = 17+
L[3] = 39+
L[0] = 17+
L[2] = 34+
L[0] = 17+
L[1] = 33
265

3. Στο 3ο μοτίβο βλέπουμε ότι είναι ένας συνδυασμός των 2.

L={11,19,26,47,59,60}
L[1] = 19+
L[0] = 11+
L[5] = 60+
L[1] = 19+
L[3] = 47+
L[0] = 11+
L[2] = 26+
L[0] = 11+
L[1] = 19
223

L={11,19,26,47,59,60}
L[1] = 19+
L[0] = 11+
L[5] = 60+
L[1] = 19+
L[3] = 47+
L[0] = 11+
L[2] = 26+
L[0] = 11+
L[1] = 19
223

L={21,32,36,41,45,48,50}
L[1] = 32+
L[0] = 21+
L[6] = 50+
L[1] = 32+
L[4] = 45+
L[0] = 21+
L[3] = 41+
L[0] = 21+
L[2] = 36+
L[0] = 21+
L[1] = 32
352

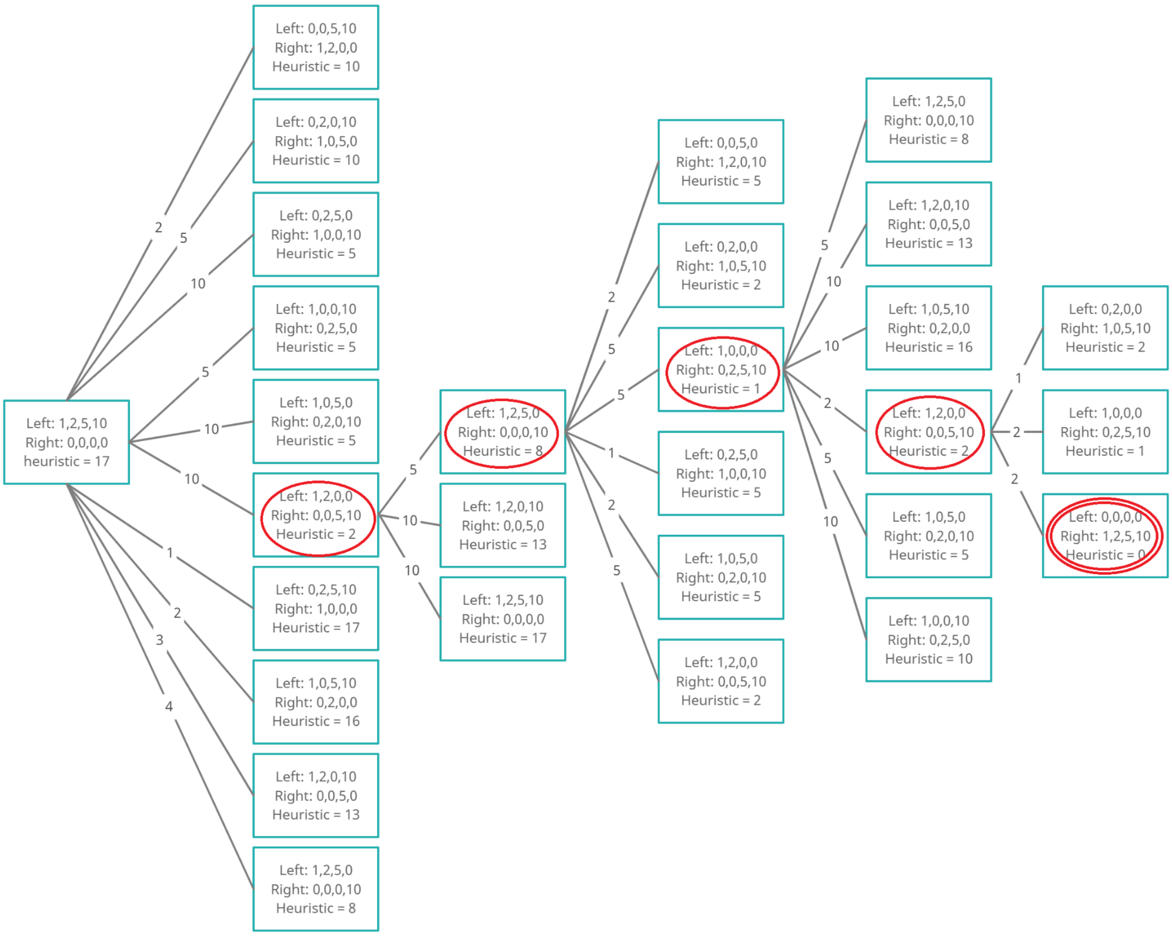
L={15,30,32,39,45,53,55}
L[1] = 30+
L[0] = 15+
L[6] = 55+
L[1] = 30+
L[4] = 45+
L[0] = 15+
L[3] = 39+
L[0] = 15+
L[2] = 32+
L[0] = 15+
L[1] = 30
321

Η συνάρτηση μας δοκιμάζει και τα 3 μοτίβα και όποιο είναι το μικρότερο νούμερο το παρουσιάζει ώς το άθροισμα της ευρετικής.

Κάθε φορά που έχουμε ένα νέο πίνακα στην αριστερή μεριά της γέφυρας, η συνάρτηση παίρνει τα νούμερα ώστε να επιστρέψει το αποτέλεσμα της ευρετικής ώστε να ακολουθήσει ο αλγόριθμος το καλύτερο μονοπάτι.

Αναζήτηση Πρώτα στο Καλύτερο(Best-First Search)

Αυτός ο αλγόριθμος ήταν ο πιό εύκολος αφού απλά επεκτείνει τους κόμβους με το μικρότερο αποτέλεσμα ευρετικής συνάρτησης ώστε να βρει γρήγορα μία λύση αλλά όχι τη βελτιστη.



Αναζήτηση A*

Αυτός ο αλγόριθμος είναι μία παραλαγή του UCS αλλά αντί για το συνολικό κόστος από την πηγή, δουλεύουμε με το άθροισμα του συνολικού κόστους με το αποτέλεσμα της ευρετικής.

Αναφορές

https://learninglover.com/examples.php?id=81
Διαφάνειες μαθήματος
https://www.udemy.com/course/search-algorithms-in-artificial-intelligence-with-java/learn/lecture/12388854?start=1875#overview