Tree Algorithms

Αδάμος Τοφαρή

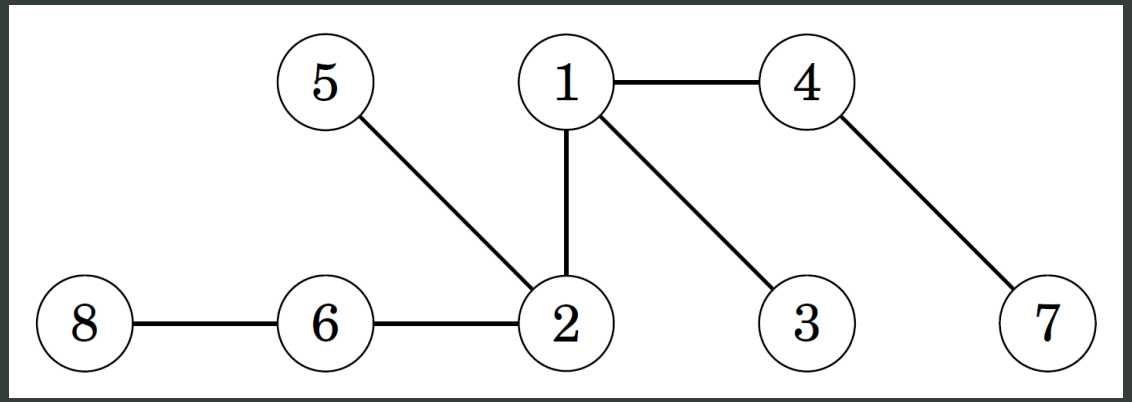
Περιεχόμενα

- Εισαγωγή
- Διάσχιση Δέντρου
- Δυναμικός Προγραμματισμός
- Διάμετρος
  - Αλγόριθμος 1
  - Αλγόριθμος 2
- Όλα τα μακρύτερα μονοπάτια
- Δυαδικά Δέντρα

Εισαγωγή

Ένα δέντρο είναι ένας συνεκτικός, άκυκλος γράφος ο οποίος αποτελείται από Ν κόμβους και Ν-1 ακμές. Διαγράφοντας οποιαδήποτε ακμή από το δέντρο το μοιράζει σε δυο μέρη, και προσθέτοντας οποιαδήποτε ακμή στο δέντρο παράγει κύκλο. Έξαλλου, πάντα υπάρχει μοναδικό μονοπάτι μεταξύ δυο κόμβων του δέντρου.

Για παράδειγμα, το επόμενο δέντρο περιέχει 8 κόμβους και 7 ακμές:

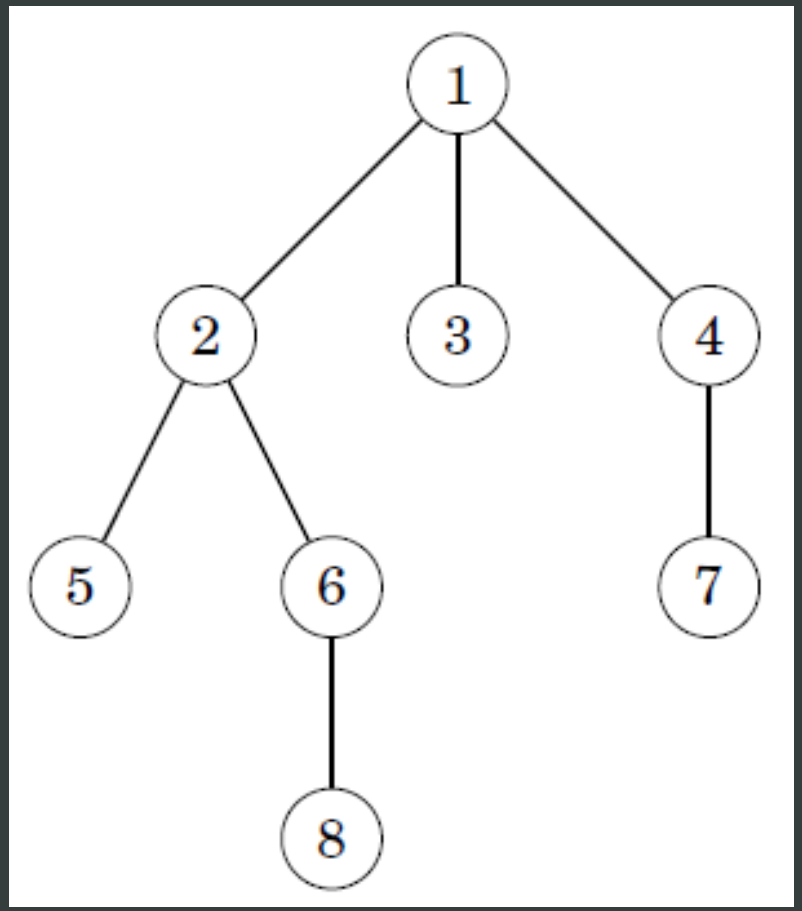


Τα φύλλα (leaves) ενός δέντρου είναι οι κόμβοι με degree 1, δηλ. έχουν μόνο 1 γείτονα.

Για παράδειγμα, τα φύλλα του πιο πάνω δέντρου είναι οι κόμβοι 3,5,7 και 8.

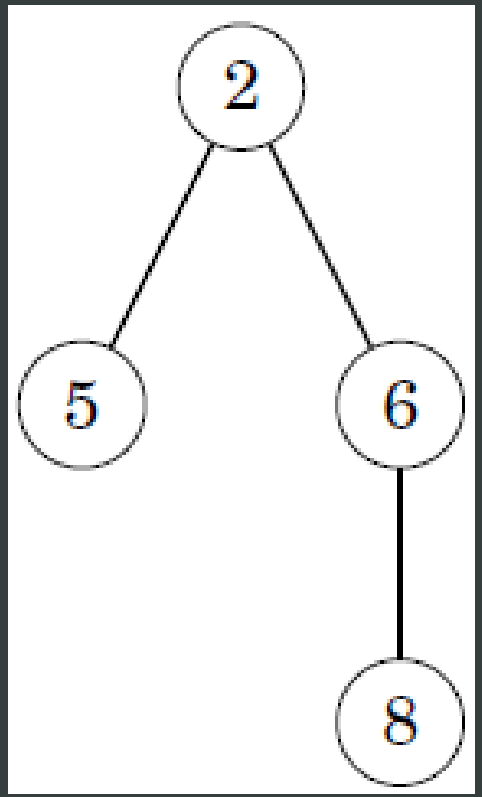
Rooted Tree

Ένα rooted tree, ένας από τους κόμβους καθορίζεται ως η ρίζα του δέντρου, και οι υπόλοιποι κόμβοι τοποθετούνται κάτω από την ρίζα. Για παράδειγμα, το επόμενο δέντρο, ο κόμβος 1 είναι η ρίζα του δέντρου.



Rooted Tree

Σε ένα rooted tree, τα παιδιά ενός κόμβου είναι οι γείτονες από κάτω, και ο γονιός ενός κόμβου είναι ο γείτονας από πάνω. Κάθε κόμβος έχει ακριβώς ένα πατέρα, εκτός της ρίζας που δεν έχει πατέρα, τα παιδιά του κόμβου 2 είναι οι κόμβοι 5 και 6, και ο πατέρας είναι ο κόμβος 1.
Η δομή ενός rooted δέντρου είναι αναδρομική: κάθε κόμβος συμπεριφέρεται ως ρίζα του υπόδεντρου που περιέχει τον κόμβο και τα υπόδεντρα των παιδιών του. Για παράδειγμα, το ποιο πάνω δέντρο, το υπόδεντρο του κόμβου 2 περιέχει τους κόμβους 2, 5, 6 και 8:



Διάσχιση Δέντρου

Οι γενικοί αλγόριθμοι διάσχισης μπορούν να χρησιμοποιηθούν για να διασχίσουμε τους κόμβους των δέντρων. Ωστόσο, η διάσχιση του δέντρου είναι πιο εύκολη να υλοποιηθεί από ένα γενικό γράφο, γιατί δεν υπάρχουν κύκλοι στο δέντρο και είναι αδύνατο να φθάσουμε ένα κόμβο από άλλους κόμβους.

Ο τυπικός τρόπος να διασχίσουμε ένα δέντρο είναι να ξεκινήσουμε DFS σε ένα τυχαίο κόμβο. Η επόμενη αναδρομική συνάρτηση μπορεί να χρησιμοποιηθεί:

```
void dfs(int s, int e) {
    // process node s
    for (auto u : adj[s]) {
        if (u != e) dfs(u, s);
    }
}
```

Διάσχιση Δέντρου

Η συνάρτηση δέχεται δυο παραμέτρους: Ο παρόν κόμβος s και ο προηγούμενος e. Ο σκοπός της παραμέτρου e είναι να βεβαιωθούμε ότι η αναζήτηση κινητέ σε κόμβους που δεν είχαμε επισκεφθεί μέχρι στιγμής.

Το επόμενο κάλεσμα της συνάρτησης ξεκινά την αναζήτηση στον κόμβο x:

```
dfs(x, 0);
```

Στο πρώτο κάλεσμα e=0, γιατί δεν υπάρχει προηγούμενος κόμβος, και επιτρέπεται να προχωρήσουμε σε όποια κατεύθυνση θέλουμε στο δέντρο.

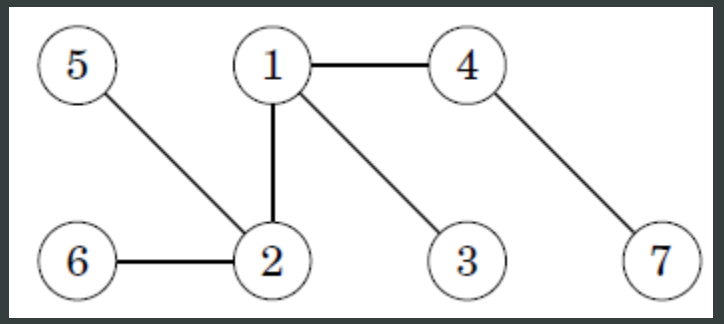
Δυναμικός Προγραμματισμός

Δυναμικός προγραμματισμός μπορεί να χρησιμοποιηθεί για να υπολογίσουμε πληροφορίες κατά την διάρκεια της διάσχισης του δέντρου. Με την χρήση δυναμικού προγραμματισμού, μπορούμε για παράδειγμα, να υπολογίσουμε σε O(n) χρόνο για κάθε κόμβο για κάθε rooted δέντρο το πλήθος των κόμβων στο υπόδεντρο ή το μήκος του μακρύτερου μονοπατιού από κόμβο σε φύλλο.
Για παράδειγμα, ας υπολογίσουμε για κάθε κόμβο s μια τιμή count[s]: το πλήθος των κόμβων ενός υπόδεντρου. Το υπόδεντρο περιέχει τον κόμβο και όλους τους κόμβους στα υπόδεντρα των παιδιών του, έτσι ώστε να υπολογίσουμε το πλήθος των κόμβων αναδρομικά με την χρήση του επόμενου κώδικα.

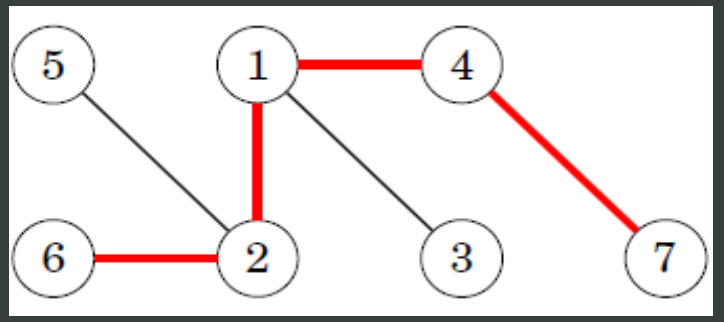
```
void dfs(int s, int e) {
    count[s] = 1;
    for (auto u : adj[s]) {
        if (u == e) continue;
        dfs(u, s);
        count[s] += count[u];
    }
}
```

Διάμετρος

Η διάμετρος ενός δέντρου είναι η μέγιστη απόσταση μεταξύ δυο κόμβων. Για παράδειγμα, σκεφτείτε το επόμενο δέντρο:



Η διάμετρος του δέντρου είναι 4, η οποία αντιστοιχεί στο επόμενο μονοπάτι:



Διάμετρος

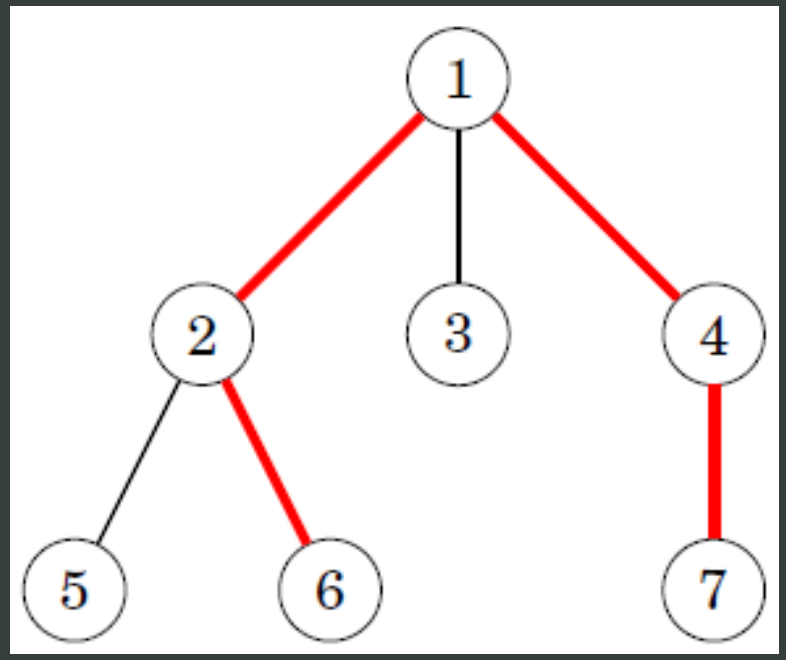
Σημειώστε ότι μπορεί να υπάρχουν πολλαπλά μεγίστου μήκους μονοπάτια. Στο πιο πάνω μονοπάτι, θα μπορούσαμε να αντικαταστήσουμε τον κόμβο 6 με τον κόμβο 5 και να αποκτήσουμε μονοπάτι μήκους 4.

Επόμενος θα συζητήσουμε δυο O(n) αλγορίθμους για τον υπολογισμό της διαμέτρου ενός δέντρου. Ο πρώτος αλγόριθμος είναι βασισμένος σε δυναμικό προγραμματισμό, και ο δεύτερος αλγόριθμος χρησιμοποιεί δυο DFS.

Αλγόριθμος 1

Ένας γενικός τρόπος να προσεγγίσουμε πολλά προβλήματα δέντρων είναι να θέσουμε για τυχαία ρίζα. Μετά από αυτό, μπορούμε να δοκιμάσουμε να λύσουμε το πρόβλημα ξεχωριστά για κάθε υπόδεντρο. Ο πρώτος μας αλγόριθμος για υπολογισμό της διαμέτρου είναι βασισμένος σε αυτή την ιδέα.

Μια σημαντική παρατήρηση είναι ότι κάθε μονοπάτι έχει ένα υψηλότερο σημείο: ο πιο ψήλος κόμβος που ανήκει στο μονοπάτι. Έτσι, εμείς μπορούμε να υπολογίσουμε για κάθε κόμβο το μήκος του μακρύτερου μονοπατιού, οποίου το ψηλότερο σημείο είναι ο κόμβος. Ένα από αυτά τα μονοπάτια αντιστοιχεί στην διάμετρο του δέντρου. Για παράδειγμα, το επόμενο δέντρο, κόμβος 1 είναι το ποιο ψηλό σημείο του μονοπατιού το οποίο αντιστοιχεί στη διάμετρο:



Αλγόριθμος 1

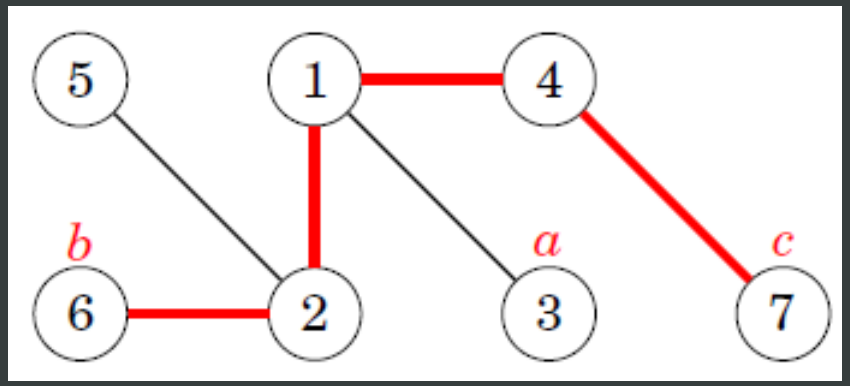
Υπολογίζουμε για κάθε κόμβο x δυο τιμές:

- toLeaf(x): το μέγιστο μήκος μονοπατιού από το x σε οποίο-δηποτε φύλλο
- maxLength(x): το μέγιστο μήκος μονοπατιού όπου το ψηλότερο σημείο είναι το x

Για παράδειγμα, στο πιο πάνω δέντρο, toLeaf(1)=2, γιατί το μονοπάτι 1->2->6, και maxLength(1)=4, γιατί υπάρχει μονοπάτι 6->2->1->4->7. Σε αυτό το παράδειγμα, maxLength(1) είναι ίσο με την διάμετρο. Δυναμικός Προγραμματισμός μπορεί να χρησιμοποιηθεί για να υπολογίσουμε το πιο πάνω σε O(n) χρόνο. Πρώτα, για να υπολογίσουμε το toLeaf(x), διασχίζουμε τα παιδιά του x, διαλέγουμε ένα παιδί c με το μεγαλύτερο toLeaf(c) και προσθέτουμε 1 στην τιμή του. Μετά, για να υπολογίσουμε το maxLength(x), επιλέγουμε 2 διαφορετικά παιδιά a και b έτσι ώστε το άθροισμα toLeaf(a)+toLeaf(b) είναι μέγιστο και προσθέτουμε 1 στο άθροισμα.

Αλγόριθμος 2

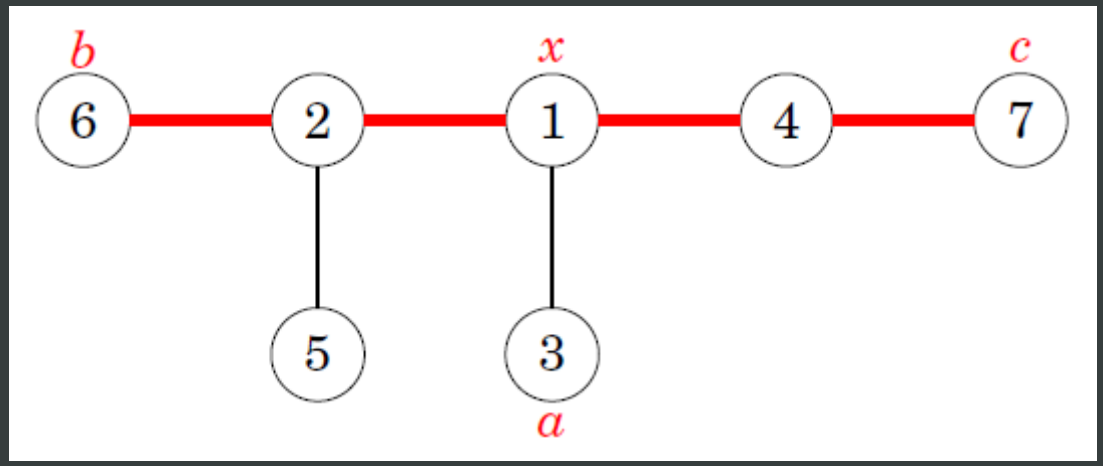
Άλλος αποτελεσματικός τρόπος να υπολογίσουμε την διάμετρο. ενός δέντρου είναι βασισμένος σε 2 DFS. Πρώτα, επιλέγουμε ένα τυχαίο κόμβο a στο δέντρο και ψάχνουμε τον πιο μακρινό κόμβο b από το a. Τότε, ψάχνουμε τον πιο μακρινό κόμβο c από τον b. Η διάμετρος του δέντρου είναι η απόσταση μεταξύ b και c. Στον επόμενο γράφο, τα a,b και c μπορούν να είναι:



Είναι ένας κομψός τρόπος, αλλά γιατί δουλεύει;

Αλγόριθμος 2

Βοηθά στο να ζωγραφίσει το δέντρο διαφορετικά έτσι ώστε το μονοπάτι που αντιστοιχεί στην διάμετρο είναι οριζόντιο, και όλοι οι κόμβοι κρέμονται από αυτόν:

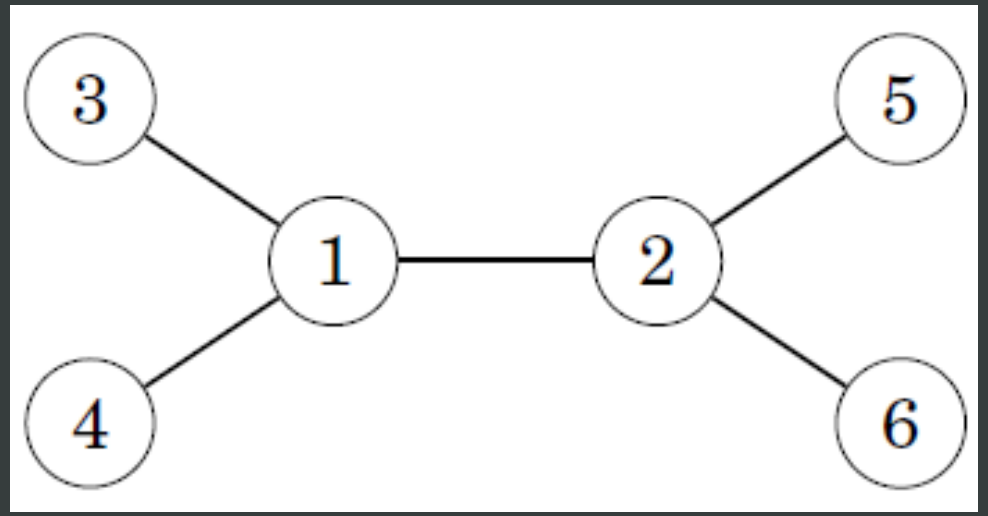


Ο κόμβος x δείχνει το σημείο που το μονοπάτι από το a ενώνεται με το μονοπάτι που αντιστοιχεί στην διάμετρο. Ο μακρύτερος κόμβος από τον κόμβο a είναι ο κόμβος b, κόμβος c η οποίος-διποτε άλλος κόμβος που είναι τουλάχιστον μακριά από τον κόμβο χ. Έτσι, αυτός ο κόμβος είναι πάντα έγκυρη επιλογή για τέλος μονοπατιού που αντιστοιχεί στην διάμετρο.

Όλα τα μακρύτερα μονοπάτια

Το επόμενο πρόβλημα είναι να υπολογίσουμε για κάθε κόμβο το μέγιστο μήκος μονοπατιού που ξεκινά από τον κόμβο. Μπορεί να θεωρηθεί ως γενίκευση της διαμέτρου, γιατί το μακρύτερο από αυτά τα μήκη είναι η διάμετρος του δέντρου. Επίσης το πρόβλημα μπορεί να λυθεί σε O(n) χρόνο.

Για παράδειγμα υποθέστε το επόμενο δέντρο:

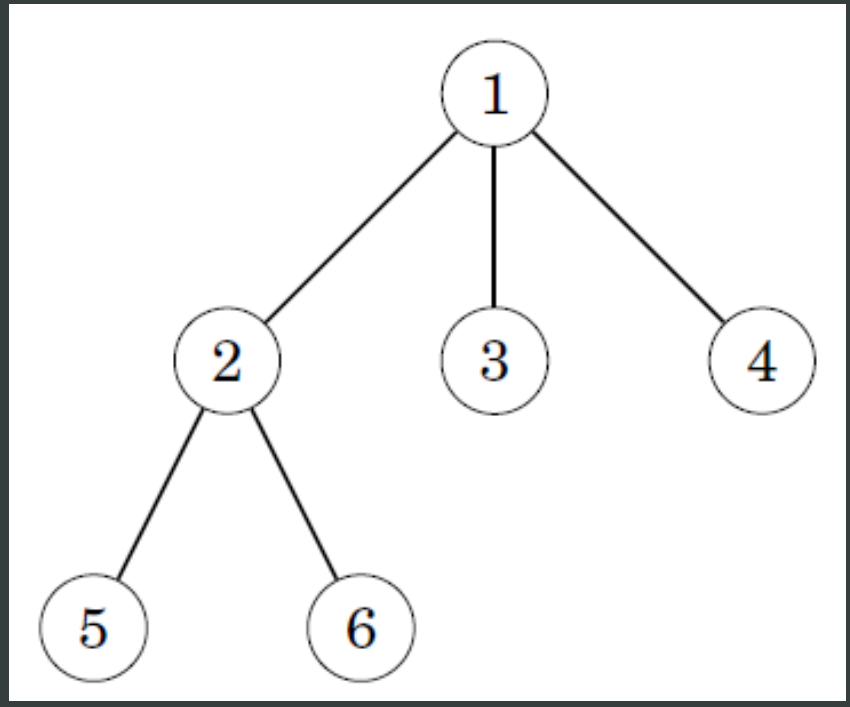


Όλα τα μακρύτερα μονοπάτια

Ας ορίσουμε το maxLength(x) το μέγιστο μήκος μονοπατιού το όποιο ξεκινά στον κόμβο x. Για παράδειγμα, στο ποιο πάνω δέντρο, maxLength(4)=3, γιατί υπάρχει μονοπάτι 4→1 →2→6. Εδώ είναι ένας πλήρης πίνακας τιμών.

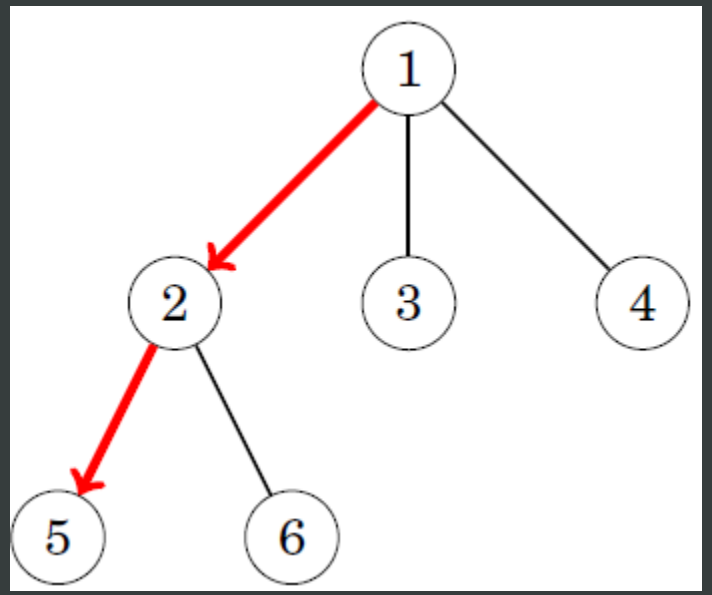
node x	1	2	3	4	5	6
maxLength(x)	2	2	3	3	3	3

Επίσης για αυτό το πρόβλημα, για αρχή θα ήταν καλό για να το λύσουμε είναι να θέσουμε μια τυχαία ρίζα:



Πρώτη φάση

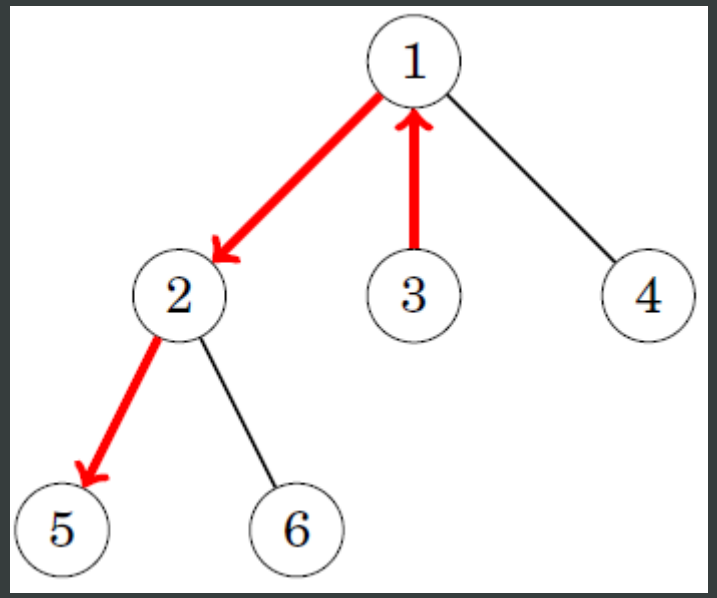
Η πρώτη φάση του προβλήματος είναι να υπολογίσουμε για κάθε κόμβο χ τη μέγιστη απόσταση μονοπατιού που περνά από παιδί του x. Για παράδειγμα, το μακρύτερο μονοπάτι του κόμβου 1 περνά από τον παιδί του 2:



Αυτό το κομμάτι λύνεται εύκολα σε O(n) χρόνο, γιατί μπορούμε να χρησιμοποιήσουμε δυναμικό προγραμματισμό όπως όπως κάναμε πιο πριν.

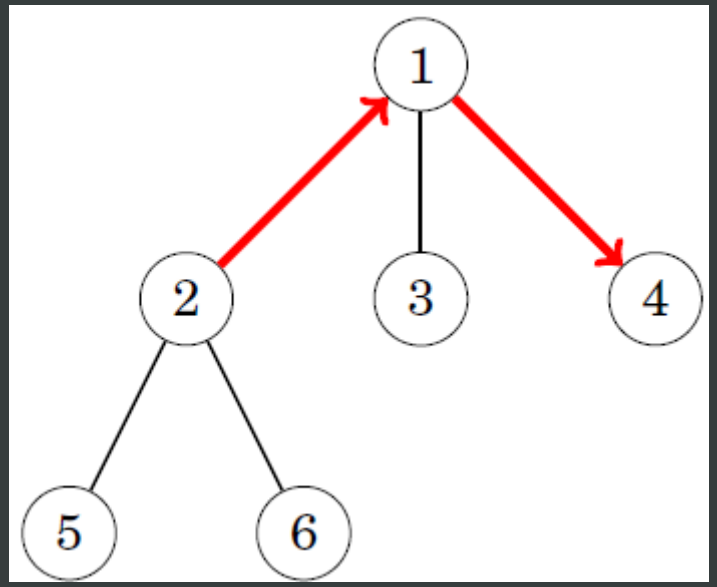
Δεύτερη φάση

Μέτα η δεύτερη φάση του προβλήματος είναι να υπολογίσουμε για κάθε κόμβο x το μέγιστο μήκος μονοπατιού που περνά από τον πάτερα p. Για παράδειγμα, το μακρύτερο μονοπάτι από τον κόμβο 3 περνά από τον πάτερα του 1:



Μικρό Λαθάκι

Με μια μάτια, φαίνεται ότι πρέπει να πάρουμε το μακρύτερο μονοπάτι από τον p. Ωστόσο, αυτό δεν δουλεύει πάντα, επειδή το μακρύτερο μονοπάτι από το p μπορεί να πηγαίνει μέσω x. Εδώ είναι ένα παράδειγμα της περίπτωσης:



Δυο Τιμές

Αλλά πάλι μπορούμε να λύσουμε τη δεύτερη φάση σε O(n) χρόνο με το να αποθηκεύουμε τα 2 μεγαλύτερα μονοπάτια για κάθε κόμβο x:

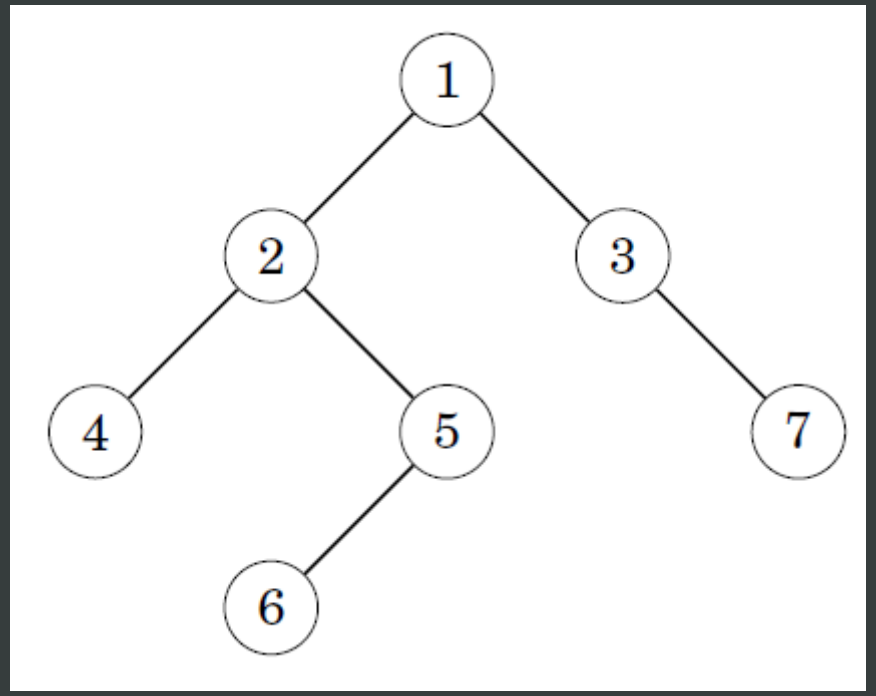
- maxLength_1(x): η μέγιστη απόσταση μονοπατιού από το x
- maxLength_2(x): η μέγιστη απόσταση μονοπατιού από το x αλλά σε άλλη κατεύθυνση από το 1ο μονοπάτι.

Για παράδειγμα, στον πιο πάνω γράφο, maxLength_1(1)=2 χρησιμοποιώντας το μονοπάτι 1->2->5, και maxLength_2(1)=1 χρησιμοποιώντας το μονοπατι 1->4.

Εν τέλη, αν το μονοπάτι που αντιστοιχεί στο maxLength_1(p) που περνά από το x, καταλήγουμε ότι η μεγίστη απόσταση είναι maxLength_2(p)+1, και σε διαφορετική περίπτωση η μεγίστη απόσταση είναι maxLength_1(p)+1.

Δυαδικά Δέντρα

Ένα Δυαδικό Δέντρο είναι ένα rooted δέντρο όπου κάθε κόμβος έχει αριστερό και δεξί υπόδεντρο. Είναι δυνατόν ένα υπόδεντρο. ενός κόμβου να είναι κενό. Έτσι, κάθε κόμβος σε κάθε δυαδικό δέντρο έχει μηδέν, ένα ή δυο υπόδεντρα.
Για παράδειγμα, το επόμενο δέντρο είναι δυαδικό δέντρο:



Orderings

Οι κόμβοι ενός δυαδικού δέντρου έχουν τρεις φυσικές ταξινομήσεις που αντιστοιχούν στους διαφορετικούς τρόπους να διασχίσουν αναδρομικά το δέντρο:

pre-order: πρώτα επεξεργαζόμαστε την ρίζα, μετά διασχίζουμε το αριστερό υπόδεντρο, μετά διασχίζουμε το δεξί υπόδεντρο.

in-order: πρώτα διασχίζουμε το αριστερό υπόδεντρο, μετά επεξεργαζόμαστε την ρίζα, μετά διασχίζουμε το δεξί υπόδεντρο.

post-order: πρώτα διασχίζουμε το αριστερό υπόδεντρο, μετά διασχίζουμε το δεξί υπόδεντρο, μετά επεξεργαζόμαστε την ρίζα.

Για το πιο πάνω δέντρο, οι κόμβοι σε pre-order είναι [1,2,4,5,6,3,7], σε in-order [4,2,6,5,1,3,7] και σε post-order [4,6,5,2,7,3,1].

Ξανακτίζοντας το δέντρο

Αν γνωρίζουμε τις pre-order και in-order ταξινομήσεις ενός δέντρου, μπορούμε να ξανακτίσουμε το δέντρο. Για παράδειγμα, το ποιο πάνω δέντρο είναι το μόνο δυνατό δέντρο με pre-order [1,2,4,5,6,3,7] και in-order [4,2,6,5,1,3,7]. Με παρόμοιο τρόπο, οι post-order και in-order μπορούμε να ξανακτίσουμε το δέντρο.

Ωστόσο, η περίπτωση είναι διαφορετική αν γνωρίζουμε μόνο την pre-order και post-order ενός δέντρου. Σε αυτή τη περίπτωση, μπορεί να υπάρχουν παραπάνω από ένα δέντρα που να μπορούν να κτιστούν από τις ταξινομήσεις. Για παράδειγμα, σε αυτά τα δυο δέντρα

Το pre-order είναι [1,2] και το post-order είναι [2,1], αλλά τα δέντρα είναι διαφορετικά.

“It’s not a bug
It’s a strange feature”

adamos2468@gmail.com