Nikolas Mavrogeneiadis
gravitorious
August 24, 2024

Binary Search Complexity

Έστω $T(n)$ ο μέγιστος αριθμός συγκρίσεων που πραγματοποιούνται από την δυαδική αναζήτηση σε έναν πίνακα με μέγεθος n. Η πολυπλοκότητα δίνεται από την επίλυση της αναδρομικής εξίσωσης $T(n) = T(\lfloor \frac{n}{2} \rfloor) + O(1)$, $n \in N$. Μέσω του $Master\ Theorem$, μπορούμε να δούμε ότι $T(n) = O(logn)$.
Ας ελέγξουμε τον αριθμό των συγκρίσεων για μικρά n.

1. n = 1 : (number of comparisons) 1
2. n = 2 : 2
3. n = 3 : 2
4. n = 4 : 3
5. n = 5 : 3
6. n = 6 : 3
7. n = 7 : 3
8. n = 8 : 4
9. n = 9 : 4

Επαγωγικά μπορούμε να δείξουμε ότι για $n \in [2^n, 2^{n+1} - 1)$ ο αριθμός των συγκρίσεων είναι k, και για το εύρος $[2^{n+1}, 2^{n+2} - 1)$ ο αριθμός των συγκρίσεων είναι $k+1$. Ισχύει ότι $k = \lfloor \log_2 n \rfloor + 1$
Αν n δύναμη του 2, είναι εύκολο να αντιληφθούμε γιατί αυτό ισχύει. Ο αριθμός των συγκρίσεων είναι ίσος με το πόσες φορές χρειάζεται το n να διαιρεθεί με το 2 μέχρι να γίνει 1 (αφού σε κάθε 'σπάσιμο' πραγματοποιούμε και μια σύγκριση), $+1$ σύγκριση η οποία είναι η αρχική πριν το σπάσιμο. Άρα $\frac{n}{2^{k_1}} = 1$, δηλαδή $k_1 = log_2 n = \lfloor \log_2 n \rfloor$ και άρα $k = k_1 + 1$.
Αν το n δεν είναι δύναμη του 2, τότε όπως παρατηρήσαμε πάνω, το n βρίσκεται πάντα ανάμεσα σε δύο αριθμούς x, y που είναι δύναμη του 2. Επειδή το y θα πραγματοποιήσει μια σύγκριση παραπάνω από το x (το οποίο πραγματοποιεί $O(logx)$ συγκρίσεις όπως είδαμε στην πρώτη παράγραφο), τότε αναγκαστικά $O(logx) = O(logy) = O(logn)$.
Πιο αναλυτικά, σε οποιοδήποτε σημείο του αλγορίθμου, το μέγεθος του υποπίνακα που αναζητούμε το στοιχείο, έχει μέγεθος $d = j - i + 1$ (με αρχική τιμή $i = 1$ και $j = n$). Επαναλαμβάνουμε την διαδικασία, μέχρι $i > j$ (αυτό θα σημαίνει ότι ο πίνακας πλέον που αναζητούμε το στοιχείο θα είναι άδειος). Μετά από μια συγκεκριμένη επανάληψη, το μέγεθος του νέου υποπίνακα θα είναι $d^{'} = j - i + 1$ και ανάλογα με το αν το στοιχείο που ψάχνουμε είναι μικρότερο ή μεγαλύτερο από το μεσαίο στοιχείο $m = \lfloor \frac{i+j}{2} \rfloor$, το j γίνεται $m - 1$ ή το i γίνεται $m + 1$ αντίστοιχα (δηλαδή ένα από τα i και j στην επόμενη επανάληψη θα αλλάξει). Αν $i = m + 1$, τότε

$$d^{'} = j - (m + 1) + 1 = j - \left\lfloor \frac{i + j}{2} \right\rfloor \leqslant j - \frac{i + j}{2} + \frac{1}{2} = \frac{j - i + 1}{2} = \frac{d}{2} \tag{1}$$

Παρόμοια και για την άλλη περίπτωση, ισχύει τελικά ότι $d^{'} \leqslant \frac{d}{2}$.
Το παραπάνω μας δείχνει ότι ο πίνακας που θα εκτελέσουμε δυαδική αναζήτηση στην επόμενη

επανάληψη έχει το πολύ το μισό μέγεθος του αρχικού. Επομένως μετά από k επαναλήψεις ο πίνακας θα έχει το πολύ $\frac{n}{2^k}$ στοιχεία. Για $k > log_2 n$ επαναλήψεις, ο πίνακας πλέον δεν έχει στοιχεία (αφού μετά από $log_2 n$ επαναλήψεις, στην επόμενη επανάληψη θα ισχύει $i > j$) και άρα ο αλγόριθμος θα πρέπει να τερματήσει. Επειδή σε κάθε επανάληψη κάνουμε σταθερό αριθμό εντολών (3 συγκρίσεις), συνολικά έχουμε $O(logn)$.
Ας το δούμε τώρα και λίγο διαφορετικά.

Πρόταση 1. *Αν το n είναι θετικός ακέραιος, τότε $k = \lfloor \log_2 n \rfloor + 1$ ή αλλιώς $k = \lceil \log_2 (n+1) \rceil$*

Απόδειξη. Αφού n θετικός ακέραιος μεγαλύτερος του 0, θα ισχύει ότι

$$2^{k_1} \leqslant n < 2^{k_1+1} \tag{2}$$
$$k_1 \leqslant log_2 n < k_1 + 1 \tag{3}$$
$$\lfloor \log_2 n \rfloor = k_1 \tag{4}$$
$$\lfloor \log_2 n \rfloor + 1 = k_1 + 1 = k \tag{5}$$

Επιπλέον είναι

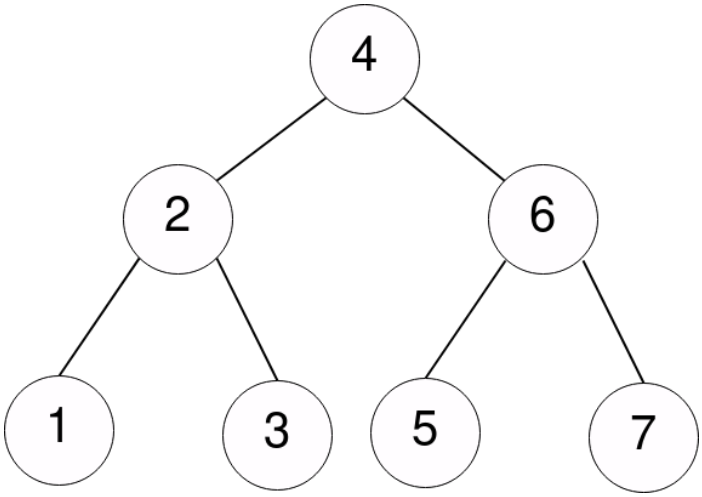
$$2^{k_1} \leqslant n \leqslant 2^{k_1+1} - 1 \tag{6}$$
$$2^{k_1} + 1 \leqslant n + 1 \leqslant 2^{k_1+1} \tag{7}$$
$$2^{k_1} < n + 1 \leqslant 2^{k_1+1} \tag{8}$$
$$k_1 < \log_2 (n+1) \leqslant k_1 + 1 \tag{9}$$
$$\lceil \log_2 (n+1) \rceil = k_1 + 1 = k \tag{10}$$

Από $5, 10$ έπεται το ζητούμενο. □

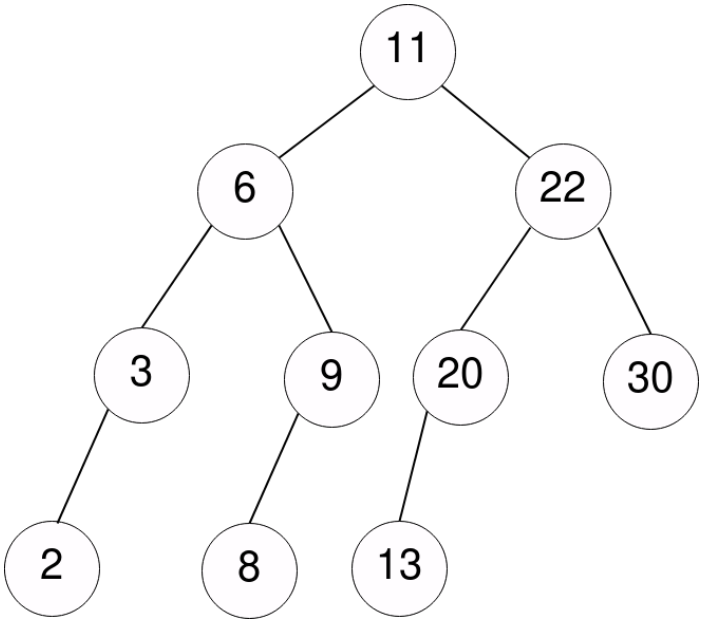
Στην ουσία το k_1 μας δίνει τον αριθμό των διαιρέσεων που χρειάζεται να κάνουμε μέχρι να καταλήξουμε σε ένα στοιχείο.
Για να κατανοηθούν καλύτερα τα παραπάνω, δίνουμε έναν επιπλέον ορισμό.

Ορισμός 1. *Ορίζουμε ως δέντρο δυαδικής αναζήτησης, το δέντρο στο οποίο ως ρίζα θέτουμε τον αριθμό που θα ελέγξουμε πρώτα (μεσαίο στοιχείο $a_{\lfloor n/2 \rfloor}$ του (ταξινομημένου) πίνακα a) και κάθε κόμβος a_i έχει το πολύ δύο παιδιά, από τα οποία το αριστερό είναι το μεσαίο στοιχείο του αριστερού υποπίνακα που (μπορεί να) πραγματοποιούμε δυαδική αναζήτηση και το δεξί ανάλογα.*

Το δέντρο δυαδικής αναζήτησης είναι δυαδικό και το αριστερό παιδί (αν έχει) του κόμβου a_i είναι μικρότερο από το a_i και το δεξί μεγαλύτερο από το a_i. Η τάξη (το σύνολο των κόμβων) του δέντρου είναι το μέγεθος του πίνακα, δηλαδή n.
Ένα παράδειγμα για τον πίνακα $[1, 2, 3, 4, 5, 6, 7]$ είναι
Επιπλέον ένα παράδειγμα για τον πίνακα $[2, 3, 6, 8, 9, 11, 13, 20, 22, 30]$ είναι



Σχήμα 1: Παράδειγμα για τον πίνακα $[1, 2, 3, 4, 5, 6, 7]$



Σχήμα 2: Παράδειγμα για τον πίνακα $[2, 3, 6, 8, 9, 11, 13, 20, 22, 30]$

Παρατηρούμε ότι κάθε επίπεδο του δέντρου, αποτελεί και μια πιθανή σύγκριση. Το ύψος αυτού του δέντρου (που το ορίζουμε σαν το μεγαλύτερο μονοπάτι από κάποιο φύλλο ως προς την ρίζα) μας δίνει τον αριθμό των διαιρέσεων και άρα το σύνολο των επιπέδων μας δίνει τον αριθμό των συγκρίσεων (k).

Πρόταση 2. *Το σύνολο των επιπέδων (συγκρίσεων) είναι* $\lfloor \log_2 n \rfloor + 1$.

Απόδειξη. Παρατηρούμε ότι για κάθε επίπεδο $i \leqslant 0$ το δέντρο έχει 2^i κόμβους, εκτός ίσως από το τελευταίο επίπεδο. Επιπλέον ισχύει ότι $2^0 + 2^1 + ... + 2^{k_1-1} = 2^{k_1} - 1$. Αφού ισχύει ότι $2^{k_1} \leqslant n < 2^{k_1+1}$ για θετικό ακέραιο k_1, μπορούμε να δημιουργήσουμε ένα δέντρο που αρχικά θα έχει $2^{k_1} - 1$ κόμβους, με ύψος $k_1 - 1$ και κάθε επίπεδο i θα έχει ακριβώς 2^i κόμβους. Στο επόμενο επίπεδο αυτού του δέντρου, μπορούμε να προσθέσουμε μέχρι 2^{k_1} κόμβους. Άρα το άθροισμα των κόμβων θα είναι το πολύ $2^0 + 2^1 + ... + 2^{k_1-1} + 2^{k_1} = 2^{k_1+1} - 1$ και επειδή $n < 2^{k_1+1}$, δηλαδή $n \leqslant 2^{k_1+1} - 1$, ισχύει $\lceil \log_2 (n+1) \rceil = k_1 + 1 = k$. Άρα το δέντρο αυτό έχει ύψος k_1 και συνολικά $\lceil \log_2 (n+1) \rceil$ επίπεδα (συγκρίσεις). Από την πρόταση 1 έπεται το συμπέρασμα. □