Κάτω φράγματα

Δοθέντος ενός προβλήματος P και ενός αλγορίθμου A που λύνει το P, επιθυμούμε να γνωρίζουμε: *"Εάν είναι ο αλγόριθμος A ο καλύτερος δυνατός αλγόριθμος για το πρόβλημα P; "*

Πιθανές βελτιώσεις του A μπορεί να είναι:

- Μείωση του απαιτούμενου χρόνου
- Μείωση του απαιτούμενου χώρου (μνήμη)

Επιθυμούμε ένα θεώρημα της μορφής:

"Όλοι οι αλγόριθμοι για το πρόβλημα P έχουν πολυπλοκότητα: $T(n) \geq f(n)$. "

- Μία συνάρτηση $f(n)$ που εμφανίζεται σε ένα τέτοιο θεώρημα ονομάζεται ένα ***κάτω φράγμα*** *(lower bound)* της πολυπλοκότητας του προβλήματος P.
- Εάν υπάρχει ένας αλγόριθμος για το P πολυπλοκότητας $f(n)$, τότε ο αλγόριθμος αυτός είναι ***βέλτιστος*** *(optimal)*.

Το μοντέλο υπολογισμού πρέπει να είναι καλώς ορισμένο.

Πρέπει να είναι ξεκάθαρο ποιο είναι το μοντέλο υπολογισμού που θεωρούμε όταν καταλήγουμε σε ένα κάτω φράγμα (lower bound).

- Ο τρόπος προσπέλασης των δεδομένων πρέπει να είναι καθορισμένος με ακρίβεια (τυχαία προσπέλαση - σειριακή προσπέλαση).
- Οι επιτρεπτές πράξεις στα δεδομένα πρέπει να είναι καθορισμένες με σαφήνεια (το σύνολο των εντολών που χρησιμοποιεί ο αλγόριθμος).

Παράδειγμα

Ας θεωρήσουμε μία ταξινομημένη λίστα στοιχείων σε μία ταινία, που μπορούμε να διασχίσουμε σειριακά. Η κεφαλή της ταινίας αρχικά βρίσκεται στην αρχή της ταινίας και οι στοιχειώδεις πράξεις είναι:

1. $move\_head(direction)$ $(direction \mapsto Left \text{ ή } Right)$
2. $compare(key)$ (Σύγκριση του key με το στοιχείο που βρίσκεται κάτω από την κεφαλή)
3. **if** "condition "**then** "jump "

Υποθέτουμε ότι οι στοιχειώδεις πράξεις απαιτούν $O(1)$ χρόνο.

- Ένα τετριμμένο κάτω φράγμα για την "αναζήτηση "με αυτό το μοντέλο είναι $\Omega(n)$.

  Εάν το στοιχείο key είναι το τελευταίο στοιχείο της λίστας, τότε θα πρέπει να εκτελέσουμε τουλάχιστον $n-1$ $move\_head(right)$ εντολές.

Ερώτημα Εάν $\Omega(n)$ είναι ένα κάτω φράγμα για το πρόβλημα της αναζήτησης με το μοντέλο που δώθηκε παραπάνω, πώς είναι δυνατόν να έχουμε έναν $O(\log n)$ αλγόριθμο (δυαδική αναζήτηση);

Απάντηση Επειδή ο αλγόριθμος της δυαδικής αναζήτησης είναι σχεδιασμένος για μία τυχαίας προσπέλασης μηχανή (random access machine - RAM). Ένα διαφορετικό μοντέλο υπολογισμού!

Όταν εκφράζουμε ένα κάτω φράγμα για κάποια προβλήματα, θα πρέπει επίσης να περιγράφουμε το μοντέλο υπολογισμού (model of computation) με βάση το οποίο καταλήξαμε σε αυτό το κάτω φράγμα.

Είσοδος - Έξοδος δεδομένων

- Όταν η είσοδος ενός προβλήματος έχει μέγεθος n, χρειαζόμαστε τουλάχιστον n βήματα για να διαβάσουμε την είσοδο.

  Παρατήρηση: Σε ορισμένες περιπτώσεις θεωρούμε ότι τα δεδομένα βρίσκονται ήδη αποθηκευμένα στην μνήμη.

- Όταν η έξοδος ενός προβλήματος αποτελείται από $f(n)$ στοιχειώδη στοιχεία, τότε χρειαζόμαστε τουλάχιστον $\Omega(f(n))$ χρόνο για να τα αναφέρουμε.

Παραδείγματα

Ταξινόμηση	$\Omega(n)$ κάτω φράγμα.
Πολλαπλασιασμός πινάκων	$\Omega(n^2)$ κάτω φράγμα.
Ελάχιστα μονοπάτια από κοινή αφετηρία	$\Omega(\|V\|)$ κάτω φράγμα.
Ελάχιστα μονοπάτια για κάθε ζεύγος κόμβων	$\Omega(\|V\|^2)$ κάτω φράγμα.
Ελάχιστο διασυνδετικό δένδρο	$\Omega(\|V\|)$ κάτω φράγμα.
κ.ο.κ. ...	

Εύρεση κάτω φράγματος για την μέθοδο του "αντιπάλου"
(Adversary lower bound)

Πρόβλημα Εύρεση του αθροίσματος n αριθμών.

Θεώρημα Η εύρεση του αθροίσματος n αριθμών απαιτεί $\Omega(n)$ χρόνο.

Απόδειξη (Διαίσθηση: ο αλγόριθμος θα πρέπει να εξετάσει όλα τα στοιχεία της εισόδου για να υπολογίσει το άθροισμα τους).

- Ας υποθέσουμε ότι υπάρχει ένας αλγόριθμος που δεν εξετάζει όλα τα στοιχεία.
- Κάποιος αντίπαλος (adversary) παρακολουθεί και όταν ο αλγόριθμος αποφασίσει, παρεμβαίνει και αλλάζει ένα στοιχείο το οποίο δεν εξετάστηκε.
- Εάν εκτελέσουμε ξανά τον αλγόριθμο, τότε θα λάβουμε ξανά το ίδιο αποτέλεσμα, καθώς ο αλγόριθμος δεν θα αντιληφθεί την αλλαγή. Στην πραγματικότητα όμως το αποτέλεσμα άλλαξε και έτσι ο αλγόριθμος είναι λαθεμένος!

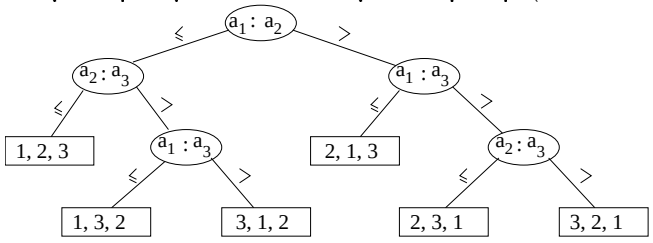
Άρα, κάθε σωστός αλγόριθμος για τον υπολογισμό του αθροίσματος n αριθμών εξετάζει κάθε στοιχείο και έτσι είναι $\Omega(n)$.

Κάτω φράγμα για το πρόβλημα της ταξινόμησης

Μοντέλο υπολογισμού

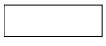
- Μηχανή τυχαίας προσπέλασης (RAM).
- Επιτρέπονται μόνο συγκρίσεις μεταξύ στοιχείων.

- Αλγόριθμοι για το παραπάνω μοντέλο ονομάζονται αλγόριθμοι ταξινόμησης μέσω συγκρίσεων *(comparison sorts)*.
- Μπορούμε να τους παραστήσουμε και ως δένδρα απόφασης *(decision trees)*:



Σημείωση

Εσωτερικός κόμβος. Το a συγκρίνεται με το b.



Φύλλο. Περιέχει την έξοδο του αλγορίθμου.

- Υπάρχουν $n!$ πιθανές μεταθέσεις της εισόδου. Άρα, το δένδρο απόφασης θα πρέπει να έχει τουλάχιστον $n!$ φύλλα.

  Το μήκος του μεγαλύτερου μονοπατιού από την ρίζα του δένδρου απόφασης πρός τα φύλλα παριστάνει το αριθμό των συγκρίσεων που εκτελεί ο αλγόριθμος ταξινόμησης στην χειρότερη περίπτωση.

Προτάσεις

1. Ένα δυαδικό δένδρο ύψους h έχει το πολύ 2^h φύλλα.

2. $n! > \left(\frac{n}{e}\right)^n$

   $e = 2.71828...$

   Ο παραπάνω τύπος απορρέει από την προσέγγιση του Stirling:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

Θεώρημα Κάθε δένδρο απόφασης που ταξινομεί n στοιχεία έχει ύψος $\Omega(n \log n)$.

Απόδειξη

Έστω ότι ένα τέτοιο δένδρο έχει ύψος h. Τότε,

$$n! \leq 2^h \qquad \text{(Πρόταση 1)}$$

$$\Longrightarrow \quad h \geq \log(n!)$$

$$\Longrightarrow \quad h \geq \log\left(\frac{n}{e}\right)^n \qquad \text{(Πρόταση 2)}$$

$$\Longrightarrow \quad h \geq \log\left(\frac{n^n}{e^n}\right) = n \log n - n \log e$$

$$\Longrightarrow \quad h = \Omega(n \log n)$$

□

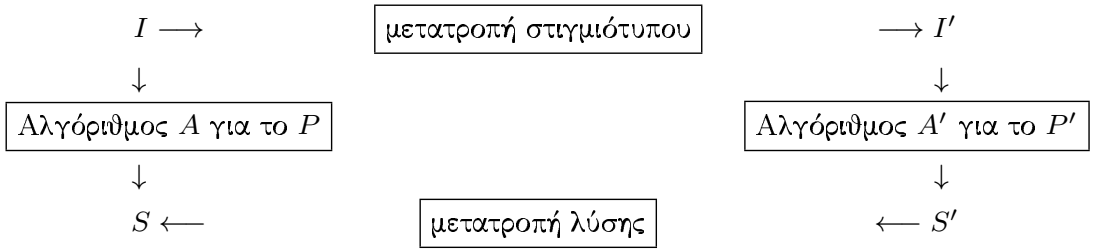
Θεώρημα Ο αλγόριθμος *MergeSort* είναι ασυμπτωτικά βέλτιστος.

Απόδειξη Ο αλγόριθμος *MergeSort* χρησιμοποιεί το ίδιο μοντέλο υπολογισμού με αυτό που χρησιμοποιήσαμε παραπάνω για να λάβουμε το $\Omega(n \log n)$ κάτω φράγμα. Επιπλέον, η πολυπλοκότητα χρόνου του είναι $O(n \log n)$.

□

Αναγωγές (Reductions)

- Όταν δύο προβλήματα σχετίζονται, είναι δυνατόν να χρησιμοποιήσουμε έναν αλγόριθμο του ενός για να επιλύσουμε το άλλο.
- Έστω $P,\ P'$ δύο προβλήματα και I ένα αυθαίρετο στιγμιότυπο του P. Επίσης, υποθέτουμε ότι γνωρίζουμε έναν αλγόριθμο A' που επιλύει το P'. Τότε, πιθανόν να μπορούμε να επιλύσουμε το P όπως φαίνεται παρακάτω:



- Εάν υπάρχει ένας αλγόριθμος μετατροπής του I στο I' και της S' στο S, τότε λέμε ότι το P ανάγεται (reduces) στο P' ή, ότι το P μετασχηματίζεται (transforms) στο P'.
- Συμβολισμός: $P \propto P'$.

Θεώρημα

Υποθέτουμε ότι $P \propto P'$ και ότι οι πολυπλοκότητες (στην χείριστη περίπτωση) για την μετατροπή του I στο I' και της S' στο S είναι $f_1(n)$ και $f_2(n)$, αντίστοιχα, όπου n είναι το μέγεθος του στιγμιότυπου I. Τότε,

1. Για κάθε αλγόριθμο για το P' με χείριστη πολυπλοκότητα $W'(n)$, υπάρχει ένας αλγόριθμος για το P με πολυπλοκότητα $f_1(n) + W'(n) + f_2(n)$.

2. Εάν $g(n)$ είναι ένα κάτω φράγμα για την πολυπλοκότητα του P, τότε $g(n) - f_1(n) - f_2(n)$ είναι ένα κάτω φράγμα για την πολυπλοκότητα του P'.

Απόδειξη (Βλέπε προηγούμενα σχήματα)

1. Κατασκευάζοντας τον αλγόριθμο
2. Με εις άτοπο επαγωγή.

□

Ένα κάτω φράγμα για το πρόβλημα κατασκευής δένδρων Huffman

Πρόβλημα Δοθέντος n βάρων $w_1,\ w_2, \ldots, w_n$ να βρεθεί ένα δένδρο Huffman γι αυτά.

- Ένα δένδρο Huffman ελαχιστοποιεί το άθροισμα $\sum_{c \text{ is leaf}} w(c) \cdot d_T(c)$.

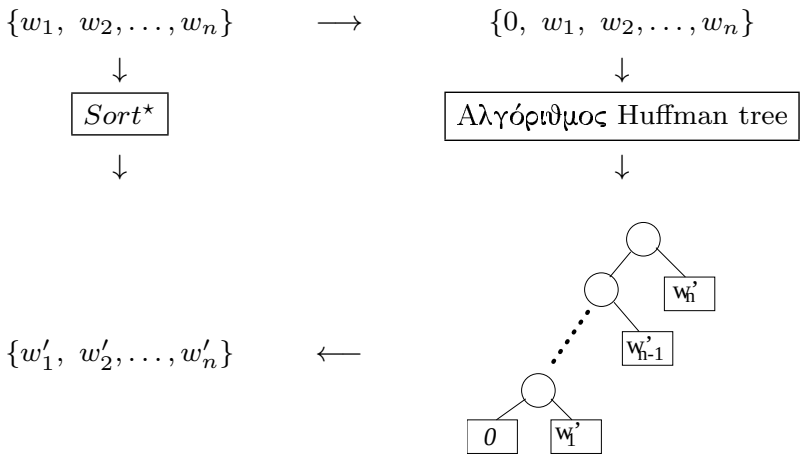
Θεώρημα Η κατασκευή των δένδρων Huffman είναι πολυπλοκότητας $\Theta(n \log n)$.

Απόδειξη Θεωρούμε το παρακάτω πρόβλημα $Sort^\star$:

> Δίνεται ένα σύνολο n θετικών αριθμών $w_1,\ w_2, \ldots, w_n$ για τους οποίους υπάρχει μία μετάθεση $w'_1,\ w'_2, \ldots, w'_n$ έτσι ώστε:
>
> $$w'_i > \sum_{j=1}^{i-1} w'_j$$
>
> για κάθε i. Να ταξινομηθούν οι αριθμοί αυτοί.

- Είναι εύκολο να δούμε ότι $Sort^\star = \Omega(n \log n)$. Κάθε δένδρο απόφασης θα πρέπει να έχει τουλάχιστον $n!$ φύλλα.

$Sort^\star \propto$ Huffman tree problem



Κατά συνέπεια, κάθε αλγόριθμος για την κατασκευή δένδρων Huffman, ο οποίος βασίζεται σε συγκρίσεις έχει πολυπλοκότητα $\Omega(n \log n)$.