Μέρος 2-Ασκήσεις με ουρές προτεραιότητας

Χρυσάνθη Μαυράκη 3130128

Χατζηαράπης Ελευθέριος 3130255

1. Έχουμε ενα πίνακα Α με ν αριθμούς και μία παράμετρο κ όπου 1<= κ <= ν. Αναζητάμε το κ-οστό μεγαλύτερο στοιχείο στον πίνακα. Πρώτο βήμα είναι η ταξινόμηση του πίνακα. Αναζητάμε μια μέθοδο για την ταξινόμηση αυτή. Χρησιμοποιώντας την Quicksort ταξινομούμε τον πίνακα και μετά εύκολα μπορούμε να πάρουμε το στοιχείο που αναζητάμε απο τον πίνακα με μηδαμινό κόστος(O(1)). Το συνολικό κόστος για όλη αυτή τη διαδικασία θα είναι O(nlogn) σε μια μέση κατάσταση(average case). Σε αντίθετη περίπτωση, δηλαδή στη χειρότερη κατάσταση(worst case) θα είναι $O(n^2)$. Φυσικά η διαδικασία της ταξινόμησης μπορεί να γίνει και με άλλους τρόπους ταξινόμησης! Μια άλλη περίπτωση είναι όταν έχουμε μια ουρά προταιρεότητας υλοποιημένη με σωρό. Επειδή έχουμε ουρά με σωρό, για να βρούμε το κ-οστό μεγαλύτερο, χρησιμοποιούμε τη getmax() k φορές ώστε να πάρουμε αυτό το στοιχείο (κόστος του getmax() είναι O(logn). Συνολικά το κόστος θα είναι Κόστος = kO(logn). Επειδή από την υπόθεση k << n, το κόστος είναι O(logn).
2. Μας δίνεται ένα d-ιαδικό δέντρο, δηλαδή κάθε πατέρας θα έχει d παιδιά. Το ύψος του δέντρου θα

είναι $\log_d N$. Τώρα για την εισαγωγή ενός στοιχείου στη σωρό(insert) παρατηρείται μια βελτίωση. Αυτό συμβαίνει διότι μέσα από τη μέθοδο swim θα γίνουν λιγότερες συγκρίσεις και συγκεκριμένα $\log_d N$ άρα το κόστος θα γίνει $O(\log_d N)$.Η βελτίωση έγγυται στο ότι το $\log_d N < \log N$. Τώρα για τη getmax() θα υπάρξει μια επιβάρυνση και συγκεκριμένα γιατι μέσω της μεθόδου sink θα γίνουν d συγκρίσεις, οπότε το κόστος θα γίνει $O(d*\log_d N)$. Οπότε το sink θα είναι χειρότερο για d μεγαλύτερο του 2 γιατί σε αντίθεση με το swim το οποίο κάνει 1 συγκρισή με το πατέρα του κόμβου το sink θα κάνει μια σύγκριση με κάθε παιδί του κόμβου (d παιδιά). Οι περισσότερες συγκρίσεις για δυαδικό δέντρο είναι 2^logn ενώ για ένα d δέντρο είναι $d^{\log_d n}$.

3. Έχουμε k ταξινομοιμένους πίνακες με n στοιχεία ο καθένας.Θέλουμε να συγχωνεύσουμε τους πίνακες σε έναν ταξινομημένο πίνακα μεγέθους nk.Πρώτη δουλειά η μεταφορα των στοιχείων στον καινουριο πίνακα.Αυτό θα γίνει με k*n επαναλήψεις που αφορουν την μεταφορά των n στοιχείων απο κάθε πίνακα.Άρα έχουμε μέχρι στιγμής $O(k*n)=O(n)$ κόστος.Η επόμενη δουλειά μας είναι η ταξινόμηση.Χρησιμοποιώντας τη μέθοδο MergeSort ταξινομούμε τον πίνακα με κόστος $O(k*n\log(k*n))=O(n\log(n))$.Συνεπώς το συνολικό κόστος θα είναι το εξής: Κόστος = $O(n) + O(n\log(n)) = O(n\log(n))$. (το k απορροφήθηκε επειδή είναι μικρότερης τάξης μεγέθους απο το n).

Με τη χρήση σωρού θα μπορούσαμε να πάρουμε το μέγιστο στοιχείο κάθε πίνακα, σε χρόνο O(1). Να τα εισάγουμε στο σωρό με k inserts και μετά επαναληπτικά θα κάνουμε getMax(). Το στοιχείο που μας επιστρέφεται θα το βάλουμε στην επόμενη θέση του συνολικά ταξινομημένου πίνακα. Έπειτα απο τον υποπίνακα στον οποίο άνηκε το προηγούμενο max κάνουμε insert το επόμενο στοιχείο. Αυτή η ιδέα υλοποιείται σε χρόνο: n inserts, τα οποία υλοποιούνται σε O(nlogk), n getMax() σε χρόνο O(nlogk) οπότε συνολικά σε O(2nlogk) = O(nlogk).Το log ειναι logk και όχι logn αφού στο δέντρο θα εισάγουμε το πολύ ένα στοιχείο απο κάθε υποπίνακα πλήθους k οπότε το ύψος θα είναι σταθερά logk. Αφού έχουμε θεωρήσει οτι το k είναι μικρότερης τάξης μεγέθους απο το n ,η πολυπλοκότητα τελικά θα είναι O(n).Αυτό είναι καλύτερη υλοποίηση απο τη Mergesort.