Τεχνολογικο Εκπαιδευτικο Ιδρυμα Πελοποννησου
Σχολη Τεχνολογικων Εφαρμογων
Τμημα Μηχανικων Πληροφορικης Τ.Ε.

Σωροί Fibonacci

Μελέτη και υλοποίηση

Πτυχιακη Εργασια

της

ΕΛΙΣΑΒΕΤ ΑΘΑΝΑΣΙΟΥ

Επιβλέπων: Γρηγόρης Καραγιώργος
Αναπληρωτής καθηγητής

Σπάρτη, Οκτώμβριος 2017

Τεχνολογικο Εκπαιδευτικο Ιδρυμα Πελοποννησου
Σχολη Τεχνολογικων Εφαρμογων
Τμημα Μηχανικων Πληροφορικης Τ.Ε.

Σωροί Fibonacci

Μελέτη και υλοποίηση

Πτυχιακη Εργασια

της

ΕΛΙΣΑΒΕΤ ΑΘΑΝΑΣΙΟΥ

Επιβλέπων: Γρηγόρης Καραγιώργος
Αναπληρωτής καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Οκτώμβριος 2017.

	(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....	.....
Γρηγόρης Καραγιώργος	-	-
Αναπληρωτής καθηγητής	-	-

Σπάρτη, Οκτώμβριος 2017

Τεχνολογικο Εκπαιδευτικο Ιδρυμα Πελοποννησου
Σχολη Τεχνολογικων Εφαρμογων
Τμημα Μηχανικων Πληροφορικης Τ.Ε.

Υπεύθυνη Δήλωση

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Ελισάβετ Αθανασίου

Περίληψη

Η παρούσα πτυχιακή εργασία βασίζεται στους σορούς Fibonacci οι οποίοι παρομιάζουν με τους διωνυμικούς σωρούς. Οι σωροί Fibonacci όπως και οι διωνυμικοί σωροί αποτελούνται από μία συλλογή από δέντρα. Ένα σημαντικό που παρουσιάζουν οι σωροί Fibonacci είναι ότι παρουσιάζουν καλύτερα φράγματα για κάποιες πράξεις σε αντίθεση με τους διωνυμικούς. Οι σωροί Fibonacci για τον χρόνο εκτέλεσης χρησιμοποιούν λογιστικά χρονικά και όχι χρονικά φράγματα χειρότερης περίπτωσης. Έτσι οι πράξεις Εισαγωγή, Ελάχιστο και Συνένωση στους σωρούς Fibonacci έχουν λογιστικό χρόνο εκτέλεσης $O(1)$ ενώ οι πράξεις Διαγραφή, Εξαγωγή ελαχίστου έχουν απαιτούμενο χρόνο εκτέλεσης $O(\log n)$. Ένα πλεονέκτημα για τους σωρούς Fibonacci που παρουσιάζουν είναι ότι η μείωση κλειδιού χρειάζεται λογιστικό χρόνο $O(1)$. Τελειώνοντας οι σοροί Fibonacci συγκρίνονται με άλλες δομές δεδομένων όπως είναι των διωνυμικών σορών και παρουσιάζονται πλεονεκτήματα και μειονεκτήματα.

Λέξεις Κλειδιά

Σωροί Fibonacci, διωνυμικά δέντρα, διωνυμικοί σωροί, πράξεις συγχωνεύσιμου σωρού

Abstract

This thesis is based on the Fibonacci heaps which interfere with the binomial heaps. The Fibonacci heaps as well as the binomial heaps consist of a collection of trees. An important feature of heaps Fibonacci is that they show better bounds for some operations, unlike binomials. The execution time envelopes Fibonacci use accounting times rather than time worst case. Thus, the Introduction, Minimum, and Fuzzy steps in the Fibonacci heaps have a $O(1)$ execution time while the Deleting, Extracted minuses have a required run time of $O(logn)$. An advantage for heaps Fibonacci they show is that the key reduction requires a $O(1)$ accounting time. Finishing the Fibonacci heaps is compared to other data structures such as binomial heaps and presents advantages and disadvantages

Keywords

Fibonacci heaps, binomial trees, binomial heaps, merging heap operations

στους γονείς μου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Γρηγόρη Καραγιώργο για την επίβλεψη αυτής της πτυχιακής εργασίας.

Σπάρτη, Οκτώμβριος 2017

Ελισάβετ Αθανασίου

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Πρόλογος

Η πτυχιακή εργασία εκπονήθηκε στα πλαίσια του προγράμματος σπουδών στο τελευταίο εξάμηνο για την απόκτηση πτυχίου από το τμήμα Μηχανικών Πληροφορικής Τ.Ε. του ΤΕΙ ΠΕΛΟΠΟΝΝΗΣΟΥ (Έδρα: Σπάρτη), Τεχνολογικών Εφαρμογών. Στόχος αυτής της πτυχιακής εργασίας είναι να μελετήσει τους σωρούς Fibonacci και να τους υλοποιήσει με την γλώσσα προγραμματισμού C. Επιπλέον θα γίνει σύγκριση με παρόμοιες δομές δεδομένων όπως των διωνυμικών σωρών και θα παρουσιάσει πλεονεκτήματα και μειονεκτήματα.

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της εργασίας

Οι αλγόριθμοι είναι ένας τομέα της πληροφορικής όπου επιλύουν υπολογιστικά προβλήματα. Μερικά από αυτά είναι: το πρόβλημα του μονοπατιού σε ένα γράφημα, το πρόβλημα της αναζήτησης, το πρόβλημα της ταξινόμησης, το πρόβλημα του *SAT*, οι σωροί Fibonacci και άλλα πολλά. Κάποια από αυτά έχουν επιλυθεί από έναν αλγόριθμο δηλαδή υπάρχει κάποιος αλγόριθμος που να μπορεί να τα επιλύει ενώ σε κάποια άλλα προβλήματα δεν έχει βρεθεί αλγόριθμος που να μπορεί να τα επιλύει προς το παρών. Έτσι λοιπόν στα προβλήματα που υπάρχει αλγόριθμος για να λυθούν ορίζονται επιλύσιμα προβλήματα ενώ τα προβλήματα που δεν έχει βρεθεί αλγόριθμος ορίζονται μη επιλύσιμα.

Ένα άλλο βασικό θέμα που παίζει σπουδαίο ρόλο στους αλγόριθμους είναι η πολυπλοκότητα των αλγορίθμων. Η πολυπλοκότητα των αλγορίθμων ασχολείται με την ανάλυση του αλγορίθμου. Έτσι η πολυπλοκότητα ασχολείται με τον υπολογισμό των υπολογιστικών πόρων που χρειάζεται ένας αλγόριθμος. Για τους υπολογιστικούς πόρους οι πόροι είναι ο χρόνος και ο χώρος. Δηλαδή υπολογίζει τον απαιτούμενο ή αλλιώς συνολικό χρόνο που απαιτείται ένας αλγόριθμος για να τρέξει καθώς επίσης και τον συνολικό χώρο που απαιτείται ένας αλγόριθμος για την μνήμη. Ανάλογα με τον χρόνο και τον χώρο που διαθέτει ένας αλγόριθμος αναλύεται σε τρεις περιπτώσεις. Αυτές οι περιπτώσεις είναι η καλύτερη περίπτωση, η μέση περίπτωση και η χειρότερη περίπτωση. Επιπλέον η πολυπλοκότητα χωρίζει τα προβλήματα σε κλάσεις πολυπλοκότητας. Δύο από τις σημαντικές κλάσεις της πολυπλοκότητας είναι η κλάση P και η NP.

Στην παρούσα πτυχιακή εργασία θα μελετήσουμε τους σωρούς Fibonacci οι οποίοι είναι δομές δεδομένων και εκτελούν κάποιες πράξεις σε δυναμικά σύνολα. Οι σωροί Fibonacci παρομοιάζουν με τους διωνυμικούς σωρούς ή αλλιώς συγχωνεύσιμων σορών και επίσης μπορούμε να τους δούμε με την μορφή δυωνυμικών δέντρων.

Οι βασικές πράξεις που εκτελούνται όσο στους σωρούς Fibonacci τόσο και στους δυωνυμικούς σωρούς είναι οι πράξεις ΕΙΣΑΓΩΓΗ, ΕΛΑΧΙΣΤΟ, ΕΙΣΑΓΩΓΗ ΕΛΑΧΙΣΤΟΥ, και ΣΥΝΕΝΩΣΗ. Η πράξη συνένωση είναι η πράξη η οποία συγχωνεύει δύο σωρούς. Επιπλέον κάποιες άλλες πράξεις που εκτελούν οι συγκεκριμένες δομές δεδομένων (σωροί Fibonacci, δυωνυμικοί σωροι) είναι η πράξη ΜΕΙΩΣΗ ΚΛΕΙΔΙΟΥ και η πράξη ΔΙΑΓΡΑΦΗ.

Γενικά οι σωροί Fibonnacci παρέχουν μία βελτιωμένη εκδοχή των διωνυμικών σωρών. Οι σωροί Fibonacci χρησιμοποιούν λογιστικά χρονικά φράγματα για τον χρόνο εκτέλεσης.

Έτσι ο λογιστικός χρόνος όσον αφορά για τις πράξεις εισαγωγή, ελάχιστο, συνένωση είναι $O(1)$ ενώ οι πράξεις διαγραφή και μείωση κλειδιού έχουν λογιστικό χρόνο $O(logn)$.

Έτσι, στόχος αυτής της πτυχιακής εργασίας είναι να μελετήσει τους σωρούς Fibonacci και να τους υλοποιήσει με την γλώσσα προγραμματισμού C. Επιπλέον θα γίνει σύγκριση με παρόμοιες δομές δεδομένων όπως των δυωνυμικών σωρών και θα παρουσιάσει πλεονεκτήματα και μειονεκτήματα.

1.2 Διάρθωση της εργασίας

Η πτυχιακή εργασία αποτελείται από εξής κεφάλα:

Στο **Κεφάλαιο 2** θα δούμε κάποια θέματα που βασίζονται στους αλγόριθμους. Επίσης θα μελετήσουμε την ανάλυση του αλγορίθμου που βασίζεται στην πολυπλοκότητα καθώς επίσης θα αναφερθεί η έννοια της δομής δεδομένων.

Στο **Κεφάλαιο 3** θα γίνει μία μελέτη στους σωρούς Fibonacci οι οποίοι είναι το βασικό θέμα της πτυχιακής εργασίας.

Στο **Κεφάλαιο 4** θα δούμε την υλοποίηση των σορών Fibonacci με την γλώσσα προγραμματισμού C.

Στο **Κεφάλαιο 5** γίνεται μία σύγκριση με παρόμοιες δομές δεδομένων όπου παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματά τους.

Στο **Κεφάλαιο 6** εμφανίζονται τα συμπεράσματα από την πτυχιακή εργασία.

Κεφάλαιο 2

Εισαγωγή στους αλγορίθμους

Στο κεφάλαιο αυτό αρχικά θα γίνει μία περιγραφή σχετικά με τους αλγόριθμους καθώς επίσης θα γνωρίσουμε τι είναι ένας αλγόριθμος. Συνεχίζοντας θα μιλήσουμε για την πολυπλοκότητα στους αλγόριθμους η οποία είναι βασική για την ανάλυσή τους καθώς στο τέλος του κεφάλαιού θα αναφερθούμε στις δομές δεδομένων.

2.1 Αλγόριθμοι

Γνωρίζουμε πως στην επιστήμη των υπολογιστών υπάρχουν υπολογιστικά προβλήματα που αντιμετωπίζει η πληροφορική όπως το πρόβλημα της ταξινόμησης, το πρόβλημα της αναζήτησης, το πρόβλημα της βελτιστοποίησης κ.ά. Επίσης υπάρχουν υπολογιστικά προβλήματα που δεν έχουν επιλυθεί μέχρι σήμερα και κάποια άλλα που έχει βρεθεί κάποια λύση δηλαδή έχει βρεθεί κάποιος αλγόριθμος. Έτσι λοιπόν η χρήση των αλγόριθμων είναι η ιδανική ιδέα για την επίλυση των προβλημάτων που αντιμετωπίζει η πληροφορική.

Ένας ορισμός για τον αλγόριθμο είναι ο ακόλουθος.

Ορισμός 2.1. *Αλγόριθμος είναι μία πεπερασμένη σειρά από ενέργειες, αυστηρά καθορισμένες και εκτελέσιμες σε πεπερασμένο χρόνο, οι οποίες στοχεύουν στην επίλυση ενός υπολογιστικού προβλήματος.*

Επιπλέον κάθε αλγόριθμος που εφαρμόζεται για την επίλυση των προβλημάτων θα πρέπει να ακολουθεί κάποια κριτήρια:

- **Είσοδος**: περιλαμβάνει όλα τα δεδομένα που χρειάζεται ένας αλγόριθμος έτσι ώστε να γίνει η επεξεργασία των δεδομένων καθώς επίσης και για την εκτέλεση.
- **Έξοδος**: Ένας αλγόριθμος θα πρέπει να περιλαμβάνει μία έξοδο για τα αποτελέσματα που χρειάζεται ένας αλγόριθμος να εξάγει όταν εκτελεστεί.
- **Πεπερασμένος**: Ένας αλγόριθμος θα πρέπει να είναι πεπερασμένος δηλαδή ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του.
- **Αποτελεσματικός**: Παρόλα αυτά θα πρέπει να είναι αποτελεσματικός ο αλγόριθμος. Όταν λέμε αποτελεσματικός αλγόριθμος εννοούμε όταν υπάρχει λύση η οποία είναι αποτελεσματική.

- **Ορθός**: Ένα χαρακτηριστικό ενός αλγόριθμου είναι η ορθότητα. Ένας αλγόριθμος θα είναι ορθός εάν για κάθε είσοδο μπορεί να τερματίσει δίνοντας στην έξοδο την ορθή λύση. Αντίθετα ένας μη ορθός αλγόριθμος μπορεί να μην τερματίζει ή μπορεί να τερματίζει δίνοντας όμως διαφορετικό αποτέλεσμα από αυτό που ζητάει. Έτσι ένας αλγόριθμος θα μπορεί να επιλύει ένα πρόβλημα όταν ένα πρόβλημα θα μπορεί να εκτελεστεί μετά από πεπερασμένα βήματα με αποτέλεσμα να δίνει την σωστή έξοδο.

Για την εκτίμηση ενός αλγόριθμου θα πρέπει να εξετάζονται κάποια από τα χαρακτηριστικά του όπως εάν είναι ορθός ένας αλγόριθμος, εάν βρίσκει καλές λύσεις, εάν είναι αποδοτικός καθώς επίσης να υπολογίζεται ο συνολικός χρόνος εκτέλεσης που απαιτείται ένας αλγόριθμος για να λυθεί (δηλαδή η χρονική πολυπλοκότητα) και να υπολογίζεται ο συνολικός χώρος που διαθέτει ένας αλγόριθμος (δηλαδή η χωρική πολυπλοκότητα). Έτσι η ανάλυση ενός αλγόριθμου είναι σημαντικό για την εκτίμηση ενός αλγόριθμου όπως θα δούμε στην επόμενη ενότητα και η οποία ασχολείται με την πολυπλοκότητα ενός αλγόριθμου.

Επιπλέον υπάρχουν κάποιες κατηγορίες αλγόριθμων που μπορούμε να συναντήσουμε όπως:

- Αλγόριθμοι διαίρει και βασίλευε: είναι οι αλγόριθμοι όπου διαιρούν το πρόβλημα σε διάφορα υποπροβλήματα και καθένα από αυτά συνδυάζονται για να πάρουν την λύση του προβλήματος. Αλγόριθμοι διαίρει και βασίλευε είναι η γρήγορη ταξινόμηση (quick sort) και η ταξινόμηση με συγχώνευση (merge sort).

- Άπληστοι αλγόριθμοι: είναι οι αλγόριθμοι όπου λύνουν προβλήματα επιλέγοντας κάθε φόρα την βέλτιστη λύση. Ένα παράδειγμα άπληστου αλγόριθμου είναι η εύρεση ελάχιστου δέντρου κάλυψης γράφου.

- Βέλτιστοι αλγόριθμοι ή άριστοι: είναι οι αλγόριθμοι οι οποίοι αναζητούν την βέλτιστη λύση σε ένα πρόβλημα.

Εκτός από αυτές τις κατηγορίες υπάρχουν και άλλες πολλές όπως τυχαιοκρατικοί αλγόριθμοι, προσεγγιστικοί αλγόριθμοι, αλγόριθμοι τυφλής αναζήτησης κ.ο.κ. Επιπλέον ένας αλγόριθμος γράφεται με την μορφή της ψευδογλώσσας το οποίο βοηθά κάθε προγραμματιστή να κατανοεί πιο εύκολα τον αλγόριθμο. Ένα άλλο σημαντικό είναι ότι με την σχεδίαση των αλγόριθμων, πολλοί προγραμματιστές κατάφεραν να τους υλοποιήσουν σε διάφορες γλώσσες προγραμματισμού όπως για παράδειγμα Java, C++, C, Python κ.ά.

2.2 Πολυπλοκότητα αλγορίθμων

Η πολυπλοκότητα των αλγορίθμων είναι ένας τομέας της θεωρίας της πολυπλοκότητας όπου ασχολείται με την ανάλυση των αλγόριθμων και την επίλυση των υπολογιστικών προβλημάτων. Πιο συγκεκριμένα ασχολείται με τον υπολογισμό των υπολογιστικών πόρων που χρειάζονται για την αλγοριθμική επίλυση ενός υπολογιστικού προβλήματος. Οι υπολογιστικοί πόροι είναι η χρονική πολυπλοκότητα δηλαδή το συνολικό χρόνο που χρειάζεται να λυθεί ένας αλγόριθμος ή αλλιώς τα συνολικά βήματα που χρειάζεται ένας αλγόριθμος μέχρι να εκτελεστεί, και η χωρική πολυπλοκότητα που είναι ο συνολικός χώρος μνήμης που χρειάζεται ένας αλγόριθμος. Έτσι τα βασικά μέτρα για τον υπολογισμός της πολυπλοκότητας

ενός αλγορίθμου είναι ο χώρος και ο χρόνος. Έτσι λοιπόν σε ένα αλγόριθμο θα πρέπει να υπολογίζεται η πολυπλοκότητα του αλγόριθμου.

Επίσης η πολυπλοκότητα χωρίζει τα προβλήματα σε εύκολα και σε δύσκολα προβλήματα. Ένα πρόβλημα θα λέγεται εύκολο πρόβλημα όταν ο χρόνος για να εκτελεστεί ένας αλγόριθμος ή ένα υπολογιστικό πρόβλημα για να εκτελεστεί είναι σχετικά μικρός καθώς επίσης υπάρχει λύση. Ένας αλγόριθμος ή ένα πρόβλημα έχει λύση τότε θα λέμε ότι είναι επιλύσιμα ενώ όταν δεν υπάρχει ακόμη λύση θα λέμε ότι είναι μη επιλύσιμο. Αντίστοιχα ένα πρόβλημα θα λέγεται δύσκολο όταν ο χρόνος για να λυθεί είναι αρκετά μεγάλος. Επίσης όταν δεν έχει βρεθεί κάποια λύση για τον αλγόριθμο θα λέμε είναι μη επίλυσιμος.

Επειδή υπάρχουν εύκολα και δύσκολα προβλήματα, η πολυπλοκότητα χώρισε τα υπολογιστικά προβλήματα σε κλάσεις πολυπλοκότητας. Οι δύο σημαντικές κλάσεις πολυπλοκότητας που υπάρχουν είναι η κλάση P και η κλάση NP. Παρακάτω βλέπουμε τον ορισμό τους.

Ορισμός 2.2. *Κλάση P (Deterministic Polynomial Time): είναι η κλάση των προβλημάτων για τα οποία μπορούν να επιλυθούν σε πολυωνυμικό χρόνο από μία υτετερμινιστική μηχανή Turing.*

Στη κλάση P ανήκουν τα προβλήματα απόφασης για τα οποία υπάρχει ένας αλγόριθμος τα οποία επιλύονται σε πολυωνυμικό χρόνο. Έτσι τα προβλήματα που ανήκουν στην κλάση αυτή είναι επιλύσιμα. Μερικά από τα προβλήματα που ανήκουν στην κλάση P είναι το πρόβλημα του μονοπατιού σε ένα κατευθυνόμενο γράφημα, το πρόβλημα του αθροίσματος και άλλα.

Ορισμός 2.3. *Κλάση NP (Non Deterministic Polynomial Time): είναι η κλάση των προβλημάτων για τα οποία μπορούν να επιλυθούν και να επαληθευτούν σε πολυωνυμικό χρόνο από μία μη υτετερμινιστική μηχανή Turing.*

Στη κλάση NP ανήκουν τα προβλήματα απόφασης τα οποία δεν έχει βρεθεί κάποιος αλγόριθμος που να τα επιλύει. Έτσι τα προβλήματα που ανήκουν στην κλάση αυτή είναι μη επιλύσιμα και θεωρούνται δύσκολα προβλήματα λόγο ότι χρειάζονται αρκετό χρόνο για την επίλυση τους. Μερικά από τα προβλήματα που ανήκουν στην κλάση P είναι το πρόβλημα του SAT, το πρόβλημα του πλανόδιου πωλητή κ.ά.

Επιπλέον ανάλογα με τον πόσο χρόνο χρειάζεται ένας αλγόριθμος για να λυθεί, οι αλγόριθμοι χωρίζονται σε κάποιες κατηγορίες ή κλάσεις πολυπλοκότητας οι οποίες εμφανίζονται στην συνέχεια.

- Σταθερού χρόνου $O(1)$: Στην κατηγορία αυτή ανήκουν αλγόριθμοι ή υπολογιστικά προβλήματα όπου ο χρόνος εκτέλεσης που χρειάζεται για να εκτελεστεί είναι σταθερός και δεν μεταβάλλεται.

- Λογαριθμικού χρόνου $O(logn)$: Στην κατηγορία αυτή ανήκουν αλγόριθμοι ή υπολογιστικά προβλήματα όπου ο χρόνος εκτέλεσης που χρειάζεται για να εκτελεστεί είναι λογαριθμικός με βάση το 2. Οι αλγόριθμοι που ανήκουν σε αυτή την κατηγορία είναι γρήγοροι και μπορούν να λύσουν προβλήματα σε πολύ γρήγορο χρόνο και ελάχιστο καθώς και όταν το μέγεθος n αυξάνεται πολύ.

- Γραμμικού χρόνου $O(n)$: Στην κατηγορία αυτή ανήκουν αλγόριθμοι ή υπολογιστικά προβλήματα όταν ο χρόνος εκτέλεσης που χρειάζεται για να εκτελεστεί είναι γραμμικός και αυτοί αλγόριθμοι θεωρούνται γρήγοροι.

- Γραμολογαριθμικού χρόνου $O(nlogn)$: Στην κατηγορία αυτή ανήκουν αλγόριθμοι ή υπολογιστικά προβλήματα όπου ο χρόνος εκτέλεσης που χρειάζεται για να εκτελεστεί είναι $logn$ και επίσης αυτοί οι αλγόριθμοι διαχωρίζουν τα προβλήματα σε μικρότερα υποπροβλήματα.

- Πολυωνυμικού χρόνου $O(n^k)$, $k > 0$: Στην κατηγορία αυτή ανήκουν αλγόριθμοι ή υπολογιστικά προβλήματα όπου ο χρόνος εκτέλεσης που χρειάζεται για να επιλυθεί είναι πολυωνυμικός.

- Εκθετικού χρόνου 2^n: Στην κατηγορία αυτή ανήκουν αλγόριθμοι ή υπολογιστικά προβλήματα όπου ο χρόνος εκτέλεσης είναι εκθετικός καθώς επίσης οι αλγόριθμοι θεωρούνται δύσκολοι διότι απαιτούνται περισσότερο χρόνο για να επιλυθούν.

2.3 Δομές δεδομένων

Στην επιστήμη της πληροφορικής οι δομές δεδομένων είναι ένας τρόπος όπου αποθηκεύονται και οργανώνονται τα δεδομένα. Γενικά τα δεδομένα είναι κάποια στοιχεία τα οποία επεξεργάζονται και καταχωρίζονται στις θέσεις της μνήμης. Τα δεδομένα μπορούν να αποθηκευτούν σε διάφορες μορφές δομών δεδομένων. Κάποιες από αυτές τις μορφές δομών δεδομένων είναι ο σωρός, ο πίνακας, η στοίβα, η συνδεδεμένη λίστα, η ουρά, τα δέντρα κ.ά. για την καλύτερη διαχείριση και επεξεργασία. Επίσης κάποιες εφαρμογές που χρησιμοποιούν δομές δεδομένων είναι τα δυαδικά δέντρα τα οποία χρησιμοποιούνται στα συστήματα βάσεις δεδομένων. Παρακάτω θα ορισθεί ο ορισμός της δομής δεδομένων.

Ορισμός 2.4. *Ένα σύνολο από στοιχεία δεδομένων αποτελεί δομή όταν υπάρχουν καθορισμένες σχέσεις μεταξύ των στοιχείων. Μια δομή δεδομένων ορίζεται ως η διαδικασία εισαγωγής και απομάκρυνσης στοιχείων με τρόπο ώστε όλη η δομή να μην αλλοιώνεται. Κάθε δομή δεδομένων έχει ως αφηρημένη έννοια συγκεκριμένο ορισμό, δηλαδή διαδικασία εισαγωγής/απομάκρυνσης στοιχείων, αλλά μπορεί να υλοποιείται σε έναν Η/Υ με διαφορετικούς τρόπους*

Οι βασικές πράξεις που μπορούν να εφαρμοστούν σε μία δομή δεδομένων είναι ως εξής:

- Εισαγωγή (insertion): είναι μία λειτουργία η οποία προσθέτει ένα νέο κόμβο σε μία υπάρχουσα δομή.

- Αναζήτηση (searching): είναι μία λειτουργία η οποία αναζητά και εντοπίζει ένα ή περισσότερα στοιχεία μίας δομής με βάση κάποια κριτήρια.

- Διαγραφή (deletion): είναι μία λειτουργία η οποία αφαιρεί ένα κόμβο από την δομή χωρίς να επηρεάζεται η οργάνωση και οι σχέσεις των στοιχείων της δομής.

- Ταξινόμηση (sorting): είναι μία λειτουργία όπου ταξινομεί τους κόμβους μίας δομής σε αύξουσα ή σε φθίνουσα σειρά.

- Προσπέλαση (access): είναι μία λειτουργία η οποία αναζητά ένα κόμβο μίας δομής για να εξεταστεί ή να εμφανιστεί ή να τροποποιηθεί το περιεχόμενό του.
- Εμφάνιση (εκτύπωση): είναι μία λειτουργία η οποία εμφανίζει τα δεδομένα μίας δομής.
- Συγχώνευση (mergin): είναι μία λειτουργία η οποία συγχωνεύει δύο ή περισσότερες δομές σε μία δομή του ίδιου τύπου.

Επιπλέον οι δομές είναι πολύ σημαντικές στην πληροφορική. Με την βοήθεια των αλγόριθμων και των δομών αυτοί οι δύο όροι συνδέονται μεταξύ τους. Έτσι υπάρχει περίπτωση αλγόριθμοι οι οποίοι είναι απλοί να μπορούν να δημιουργήσουν δομές που είναι πολύπλοκες καθώς επίσης και αντίστροφα να συμβεί αλγόριθμοι που είναι πολύπλοκοι να χρησιμοποιούν δομές οι οποίες είναι απλές. Στο επόμενο κεφάλαιο θα επικεντρωθούμε σε δομές δεδομένων σορού.

Κεφάλαιο 3

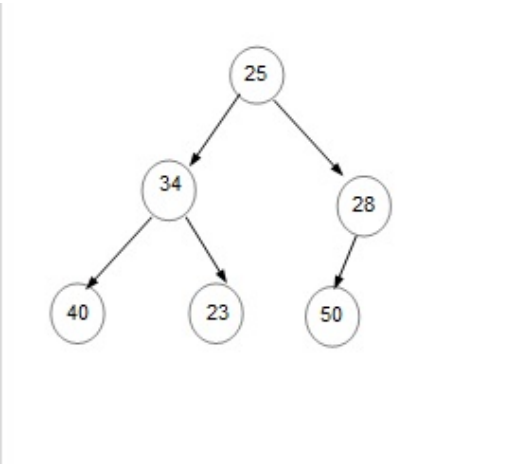
Σωροί Fibonacci

Στο τρέχων κεφάλαιο θα μελετηθούν οι σωροί Fibonacci που το οποίο είναι το βασικό θέμα της πτυχιακής εργασίας. Πιο συγκεκριμένα θα δούμε κάποιες από τις πράξεις που χρησιμοποιούν οι σωροί Fibonacci όπως τις πράξεις ενός συγχωνεύσιμου σωρού (κατασκευή σωρού Fibonacci, εισαγωγή κόμβου, εύρεση του ελάχιστου κόμβου, συνένωση δύων σορών Fibonacci, εξαγωγή του ελάχιστου κόμβου, μείωση κλειδιού και διαγραφή κόμβου)

3.1 Δομή δεδομένων σωρού

Ένας σωρός ή αλλιώς (heap) είναι ένα δυαδικό δέντρο όπου αποτελείται από κόμβους. Στη κορυφή ενός δέντρου υπάρχει η ρίζα όπου η τιμή κάθε κόμβου είναι μεγαλύτερη από τις τιμές των παιδιών του. Επιπλέον ο αριθμός που παίρνει κάθε κόμβος στο δέντρο δηλώνει την τιμή που είναι αποθηκευμένη μέσα στον κόμβο. Επίσης οι σωροί αποτελούν μία ιδανική λύση δομής δεδομένων και χρησιμοποιούνται σε πολλές εφαρμογές όπως σε συστήματα προσομοίωσης, σε χρονοπρογραμματισμό κ.ά. επειδή θεωρούνται μία καλή δομή για την αναζήτηση καθώς επίσης χρησιμοποιούνται σε αλγορίθμους και στην επίλυση προβλημάτων με γράφους.

Παρακάτω βλέπουμε ένα σωρό (βλπ. 3.1).



Εικόνα 3.1: *Σωρός*

Ένα δέντρο είναι μία δομή δεδομένων το οποίο χρησιμοποιείται για την γρήγορη αποθήκευση των δεδομένων. Γενικά είναι ένας γράφος που αποτελείται από κόμβους και ακμές.

Ο κόμβος σε κάθε κορυφή είναι η ρίζα δηλαδή ο γονέας και οι υπόλοιποι κόμβοι είναι τα παιδιά της ρίζας. Οι ακμές είναι οι συνδέσεις που συνδέουν τον κόμβο της ρίζας με τους υπόλοιπους κόμβους. Επίσης κάθε δέντρο έχει ένα μονοπάτι το οποίο ξεκινάει πάντα από την ρίζα του δέντρου. Έτσι ένα δέντρο είναι ένα δυαδικό δέντρο. Σε κάθε δυαδικό δέντρο ισχύει ότι τα παιδιά της ρίζας που είναι μικρότερα από την ρίζα βρίσκονται αριστερά ενώ τα παιδιά της ρίζας που είναι μεγαλύτερα από την ρίζα βρίσκονται δεξιά. Η ρίζα βρίσκεται πάντα στην κορυφή του δέντρου

Επίσης υπάρχουν δύο είδη σωρών όπως ο σωρός μεγίστου και ο σωρός ελαχίστου. Στο σωρό μεγίστου σε ένα δυαδικό δέντρο, εξάγεται το στοιχείο που έχει την μεγαλύτερη τιμή και τότε η ρίζα θα έχει το στοιχείο αυτό με την μεγαλύτερη τιμή, ενώ στο σωρό ελαχίστου εξάγεται το στοιχείο που έχει την μικρότερη τιμή και τότε η ρίζα θα έχει το στοιχείο αυτό με την μικρότερη τιμή. Επίσης κάποια άλλα είδη σωρών είναι οι δυωνυμικοί σωροί καθώς επίσης και οι σωροί Fibonacci όπου η πτυχιακή εργασία θα επικεντρωθεί σε αυτό το είδος σωρού.

3.2 Σωροί Fibonacci

Πρίν μιλήσουμε για σωρούς Fibonacci θα δούμε αναφορικά για την ακολουθία Fibonacci από που προήλθε και που χρησιμοποιείται. Η ακολουθία Fibonacci είναι μία ακολουθία που αποτελείται από αριθμούς όπως βλέπουμε στην συνέχεια:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377...$$

όπου αρχικά προσθέτοντας τους δύο πρώτους όρους δηλαδή το 0 και το 1 και στην συνέχεια με τον ίδιο τρόπο παίρνουμε το άθροισμα που προέκυψε από τους δύο πρώτους όρους δηλαδή τους προηγούμενους όρους και το αθροίζουμε με το επόμενο όρο που είναι το 2. Με την ίδια διαδικασία γίνεται και για τους άλλους όρους. Άρα η ακολουθία Fibonacci ορίζεται από τον ακόλουθο τύπο:

$$Fib(n) = Fib(n-1) + Fib(n-2)$$

όπου $Fib(0) = 0$ και $Fib(1) = 1$ είναι οι δύο πρώτοι όροι. Η ακολουθία Fibonacci πήρε το όνομα από τον Fibonacci αφού αυτός ήταν που την ανακάλυψε. Έτσι λοιπόν η ακολουθία Fibonacci είναι πολύ σημαντική στην πληροφορική, στην πολυπλοκότητα των αλγορίθμων, στις δομές δεδομένων όπως θα δούμε στους σωρούς Fibonacci κ.ά.

Στην πληροφορική ο σωρός Fibonacci είναι μία δομή δεδομένων που παρομοιάζει με τον δυωνυμικό σωρό και είναι μία συλλογή από δυαδικά δέντρα.

Ορισμός 3.5. *Ένας δυωνυμικός σωρός είναι μία συλλογή από διωνυμικά δέντρα.*

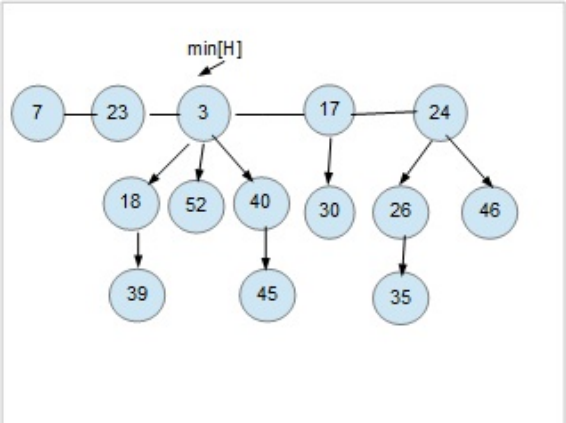
Ορισμός 3.6. *Οι σωροί Fibonacci μοιάζουν με τους διωνυμικούς σωρούς και αποτελούν μία συλλογή από δέντρα δομημένα κατά σωρού ελαχίστου. Ωστόσο χρησιμοποιούν τις γνωστές πράξεις όπως της εισαγωγής, της διαγραφής και της συγχώνευσης.*

Όσον αφορά για την ανάλυσή τους οι σωροί Fibonacci παρουσιάζουν μία καλή επίδοση ασυμπτωτικά από ότι οι δυωνυμικοί σωροί καθώς επίσης παρουσιάζουν καλύτερα φράγμα-

τα για τον χρόνο. Επιπλέον τα δέντρα στους σωρούς αυτούς θεωρούνται μη διατεταγμένα δυωνυμικά δέντρα.

Ένα μη διατεταγμένο διωνυμικό δέντρο φαίνεται να είναι παρόμοιο με ένα διωνυμικό δέντρο. Ένας κόμβος που αποτελεί ρίζα σε ένα μη διατεταγμένο δυωνυμικό δέντρο έστω U_k έχει βαθμό k ο οποίος βαθμός είναι μεγαλύτερο από ένα άλλο βαθμό σε οποιοδήποτε δυωνυμικό δέντρο. Έτσι ένας σωρός Fibonacci ο οποίος αποτελείται από ένα σύνολο κόμβων n αποτελείται από συλλογές μη διατεταγμενών διωνυμικών δέντρων τότε ισχύει ότι $D(n) = logn$.

Παρακάτω στην εικόνα 3.2 βλέπουμε ένα παράδειγμα σωρού Fibonacci ο οποίος αποτελείται από 5 δέντρα τα οποία είναι δομημένα κατά σωρό ελαχίστου. Επίσης το επάνω επίπεδο στα δέντρα είναι το ριζικό επίπεδο. Δηλαδή αποτελείται από τις ρίζες ή αλλιώς γονείς οι οποίες βρίσκονται σε κάθε κορυφή στα δέντρα. Στα δέντρα στο σωρό Fibonacci στο συγκεκριμένο παράδειγμα οι κόμβοι με τα κλειδιά 7,23,3,17 και 24 αποτελούν ρίζες των δέντρων. Επίσης παρατηρούμε πως ο κόμβος που αποτελεί το κλειδί 3 είναι ο ελάχιστος κόμβος του σωρού Fibonacci μιας και έχει το μικρότερο κλειδί από τα υπόλοιπα καθώς επίσης είναι η ρίζα του δέντρου και βρίσκεται στο ριζικό επίπεδο. Έτσι λοιπόν ο ελάχιστος κόμβος που είναι η ρίζα του δέντρου περιέχει το ελάχιστο κλειδί και θα συμβολίζεται $min[H]$ και θα είναι ο κόμβος ελάχιστου κόμβου του σωρού Fibonacci. Οπότε το $min[H]$ δηλώνει τον κόμβο της ρίζας στο δέντρο όπου το κλειδί ή αλλιώς η τιμή που παίρνει είναι το ελάχιστο. Επίσης ο κόμβος με το ελάχιστο κλειδί βρίσκεται στο ριζικό επίπεδο.

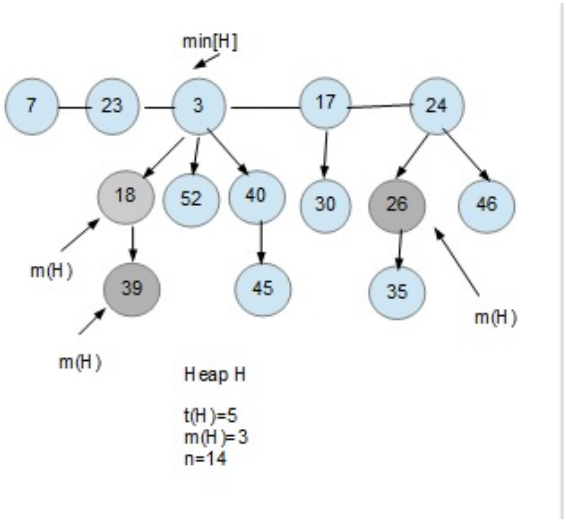


Εικόνα 3.2: *Σωρός Fibonacci*

Επιπλέον για κάποιο σωρό Fibonacci H έχει ένα πεδίο το $n[H]$ όπου αποθηκεύονται τα σύνολα των κόμβων για κάποια συγκεκριμένη στιγμή που έχει ο σωρός H. Οπότε, υποθέτουμε ότι $H(t)$ συμβολίζεται το σύνολο των δέντρων του σωρού Fibonacci H και $m(h)$ συμβολίζεται το σύνολο των επισημασμένων κόμβων του H. Οπότε το δυναμικό για ένα σωρό Fibonacci είναι ως εξής: $\Phi(H) = t(H) + 2m(H)$. Δηλαδή το δυναμικό σε ένα πλήθος από σωρούς Fibonacci είναι το άθροισμα των δυναμικών των σωρών-μελών.

Αν θέλαμε να βρούμε ποιο είναι το δυναμικό του σωρού Fibonacci με βάση την εικόνα που δείχνει παρακάτω (βλπ.3.3) ,τότε το δυναμικό προκύπτει ότι είναι $\Phi(H) = t(H)+2m(H) = \Phi(H) = 5 + 2 \cdot 3 = 11$, μιας και $t(H) = 5$ δηλώνει τον συνολικό αριθμό των δέντρων που ανήκουν στο ριζικό επίπεδο δηλαδή είναι ρίζες των δέντρων, $n = 14$ δηλώνει τον συνολικό

αριθμό των κόμβων του σωρού και $m(H) = 3$ δηλώνει τον συνολικό αριθμό των επισημασμένων κόμβων του σωρού. Στην εικόνα οι επισημασμένοι κόμβοι είναι τρεις ο κόμβος 18,39 και 26.



Εικόνα 3.3: *Παράδειγμα Σωρός Fibonacci*

3.3 Πράξεις συγχωνεύσιμου σωρού

Σε αυτή την ενότητα θα μελετήσουμε και θα αναλύσουμε την εκτέλεσης στις πράξεις ενός συγχωνεύσιμου σωρού στα χρονικά φράγματα ακριβώς σαν και αυτές τις πράξεις που αφορούν τους σωρούς Fibonacci όπως είναι η κατασκευή σωρού Fibonacci, η εισαγωγή κόμβου, η εύρεση του ελάχιστου κόμβου, η συνένωση δύων σορών Fibonacci και η εξαγωγή του ελάχιστου κόμβου. Επίσης για ένα σωρό Fibonacci που περιλαμβάνει μία συλλογή από μη διατεταγμένα δέντρα τότε θα ισχύει ότι $D(n) = logn$. Έτσι ένα μη διατεταγμένο δέντρο ορίζεται παρόμοια με ένα δυωνυμικό δέντρο.

3.3.1 Κατασκευή σωρού Fibonacci

Αρχικά για την κατασκευή του σωρού Fibonacci δημιουργείται ένας κενός σωρός Fibonacci η οποία η δημιουργία γίνεται με την διαδικασία της ΚΑΤΑΣΚΕΥΗΣ του ΣΩΡΟΥ Fibonacci όπου δεσμεύει και επιστέφει ένα αντικείμενο H όπου $n(H) = 0$ και $min(H) =$ κενό ($n(H)$ είναι το σύνολο των κόμβων που αποτελεί ο σωρός ενώ το $min(H)$ είναι το ελάχιστο κλειδί που θα αποτελεί ο σωρός). Στην περίπτωση αυτή δεν υπάρχει δέντρο όποτε θα έχουμε $t(H) = 0$ και επίσης $m = 0$ όπου $t(H)$ δηλώνει το σύνολο των δέντρων που αποτελείται ο σωρός Fibonacci H και m δηλώνει τον αριθμό των επισημασμένων κόμβων. Οπότε το δυναμικό του σωρού Fibonacci όταν ο σωρός είναι κενός είναι 0 δηλαδή $\Phi H) = 0$. Άρα ο χρόνος εκτέλεσης για την Κατασκευή του σωρού Fibonacci είναι $O(1)$ δηλαδή σταθερός ο χρόνος εκτέλεσης.

3.3.2 Εισαγωγή κόμβου

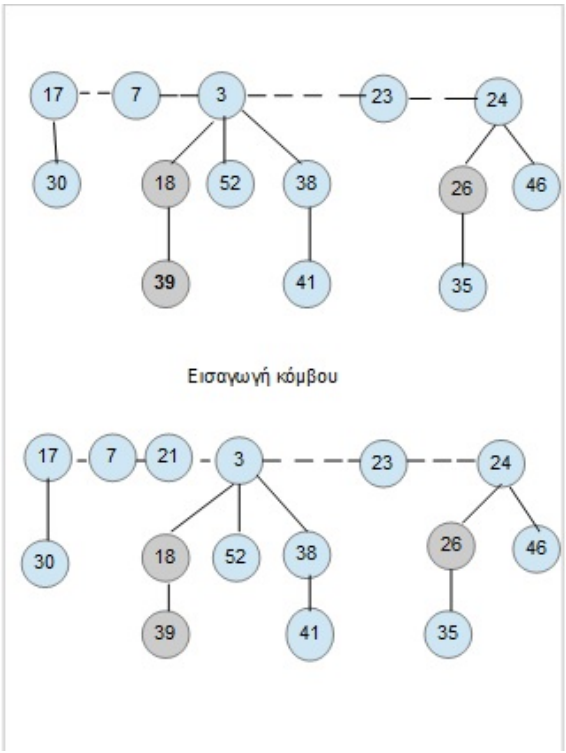
Στην συγκεκριμένη περίπτωση εισάγεται ένας νέος κόμβος χ στο σωρό Fibonacci με την προϋπόθεση ότι ο κόμβος έχει δεσμευτεί και το κλειδί[χ] έχει ήδη συμπληρωθεί. Στη συνέχεια στην εικόνα 3.4 βλέπουμε την εισαγωγή ενός νέου κόμβου με τιμή 21 που εισάγεται στο σωρό Fibonacci όπως εμφανίζεται στο κάτω μέρος της εικόνας. Έτσι ο νέος κόμβος εισάγεται στο ριζικό επίπεδο στα αριστερά της ρίζας η οποία είναι ο κόμβος με την τιμή 3 καθώς αυτός ο κόμβος βρίσκεται σε ένα μονοκομβικό δέντρο το οποίο είναι δομημένο κατά σωρού ελαχίστου και επίσης το δέντρο θεωρείται μη διατεταγμένο διωνυμικό δέντρο. Επιπλέον ο κόμβος δεν έχει άλλα παιδιά φύλλα και θεωρείται μη επισημασμένος.

Ένα άλλο σημαντικό που αφορά στην εισαγωγή κόμβου με σωρού σε Fibonacci είναι ότι δεν γίνεται κάποια ενοποίησης των δέντρων του σωρού. Επίσης για να προστεθεί ο νέος κόμβος στο ριζικό επίπεδο κάνει χρόνο εκτέλεσης $O(1)$ δηλαδή είναι σταθερός ο χρόνος. Ωστόσο για τον υπολογισμό του λογιστικού χρόνου για την εισαγωγή του κόμβου στο σωρό Fibonacci είναι ως εξής:

Ας υποθέσουμε ότι έχουμε το σωρό Η και το σωρό H' ο οποίος είναι ο τελικός σωρός Fibonacci. Οπότε θα έχουμε $t(H') = t(H) + 1$ καθώς επίσης $m(H') = m(H)$. Άρα το δυναμικό θα είναι

$$((t(H) + 1) + 2m(H)) - (t(H) + 2m(H)) = 1$$

Λόγω ότι είναι $O(1)$ το κόστος τότε ο χρόνος του λογιστικού κόστους είναι $O(1)$.



Εικόνα 3.4: *Εισαγωγή κόμβου σε σωρό Fibonacci*

Παρακάτω βλέπουμε τον αλγόριθμο της εισαγωγής ενός νέου κόμβου στο σωρό Fibonacci (βλπ. 3.1).

Αλγοριθμος 3.1: *Αλγόριθμος εισαγωγής κόμβου σε σωρό Fibonacci [1]*

```
Εισαγωγή σε σωρό Fibonacci(H,x)
1. βαθμός[χ]=0
2. π[χ]=κενό
3. θυγατρικός=κενό
4. αριστερός[χ]=χ
5. δεξιός[χ]=χ)
6. Σήμανση[χ]=ψευδή
7.Συνδέουμε το ριζικό επιπέδο που περιέχει το χ με το ριζικό επιπέδο H
8. if min[H]=κενό ή κλειδί[χ]< (*κλειδί[min[H]]
9. then min[H]= χ
10. n[H]=n[H]+1
```

Αλγοριθμος 3.2: *Αλγόριθμος Συνένωση σωρών Fibonacci [1]*

```
Συνένωση σωρών Fibonacci(H1,H2)
1. H= ΚΑΤΑΣΚΕΥΗ ΣΩΡΟΥ Fibonacci()
2. min[H]=min[H1]
3. Συνδέουμε τα ριζικά επίπεδα των H2 και H
4.  if (min[H1]= κενό) ή
      (min[H2] διάφορο του κενού και κλειδί[min[H2]]< κλειδί [min[H1]])
5. then min[H]=min[H2]
6. n[H]=n[H1]+n[H2]
7. αποδεσμεύουμε τα αντικείμενα H1, H2
8. return H
```

3.3.3 Εύρεση του ελάχιστου κόμβου

Είδαμε ότι το στοιχείο $min[H]$ δηλώνει τον σωρό Fibonacci που έχει τον ελάχιστο κόμβο καθώς μπορεί να βρεθεί σε συνολικό χρόνο $O(1)$. Λόγω ότι το δυναμικό του σωρού Fibonacci δεν αλλάζει, το λογιστικό κόστος σε αυτή την περίπτωση της πράξης της εύρεσης του ελαχίστου κόμβου είναι ίσο με το πραγματικό κόστος δηλαδή $O(1)$.

3.3.4 Συνένωση δύων σορών Fibonacci

Η πράξη Συνένωση σωρών Fibonacci γίνεται με τον εξής τρόπο: συνενώνει τους σωρούς H_1 και H_2 Fibonacci, δηλαδή συνδέει τις ρίζες που βρίσκονται στο ριζικό επίπεδο H_1 και H_2 και στην συνέχεια ορίζει το νέο ελάχιστο κόμβο. Παρακάτω βλέπουμε αναλυτικά τον αλγόριθμο Συνένωση σωρών Fibonacci (βλπ. 3.2).

Έτσι η λειτουργία του συγκεκριμένου αλγόριθμου ακολουθείται ως εξής:

Με βάση τον αλγόριθμο 3.3 παρατηρείται πως από τις γραμμές του κώδικα 1 έως 3 συνδέονται οι ρίζες H_1 και H_2 οι οποίες βρίσκονται στο ριζικό επίπεδο και άρα έχουμε ένα νέο ριζικό επίπεδο H. Στη συνέχεια στις γραμμές 2,4 και 5 στον παραπάνω αλγόριθμο προσδιο-

Αλγοριθμος 3.3: *Αλγόριθμος εξαγωγή ελαχίστου από σωρό Fibonacci [1]*

```
 Εξαγωγή ελαχίστου από σωρό  Fibonacci()
1. z=min[H1]
2. if  z διάφορο του κενού
3.  then for κάθε θυγατρικό χ τουz
4. προσθέτουμε τον χ στο ριζικό επίπεδο του H
5.π(χ)=κενό
6. αφαιρούμε τον z από το ριζικό επίπεδο του H
7.    if z=δεξιός[z]
8.     then min[H]=κενό
9.     else min[H]=δεξιός[z]
10.            ΕΝΟΠΟΙΗΣΗ(H)
11.    n[H]= n[H]−1
12. return z
```

ρίζεται ο ελάχιστος κόμβος του H. Ωστόσο στην γραμμή 7 αποδεσμεύονται τα αντικείμενα H_1, H_2, και στην γραμμή 8 τελικά επιστρέφεται ο σωρός Fibonacci.

Για το υπολογισμό του δυναμικού το δυναμικό είναι 0 μιας και $t(H) = t(H_1) + t(H_2)$ και $m(H) = m(H_1) + m(H_2)$. Παίρνοντας την σχέση $\Phi(H) - (\Phi(H_1)+\Phi(H_2)) = (t(H) + 2m(H)) - ((t(H_1)_2 m(H_1)) + (t(H_2) + 2m(H_2))) = 0$.

Άρα το λογιστικό κόστος για την συνένωση σωρών Fibonacci είναι $O(1)$. Επιπλέον στην πράξη της συνένωσης σωρών δεν υπάρχουν κάποια ενοποίηση δέντρων.

3.3.5 Εξαγωγή ελάχιστου κόμβου

Η πράξη που ακολουθεί εξάγει από ένα σωρό Fibonacci τον κόμβο με το μικρότερο κλειδί και δείχνει πως είναι μία από τις δύσκολες και περίπλοκες πράξεις των σορών Fibonacci. Παρακάτω βλέπουμε τον αλγόριθμο για την εξαγωγή ελαχίστου από σωρό Fibonacci.

Αρχικά κατά την διαδικασία της εξαγωγής ελαχίστου κόμβου στην αφαίρεση ελαχίστου από τον σωρό Fibonacc δημιουργείται ένας ριζικός κόμβος από κάθε παιδί ή φύλλο ή αλλιώς θυγατρικό του ελάχιστου κόμβου και αφαιρεί τον ελάχιστο κόμβο από τις ρίζες του ριζικού επιπέδου. Συνεχίζοντας ενοποιεί το ριζικό επίπεδο ενώνοντας τους ριζικούς κόμβους που είναι ισόβαθμοι μέχρι να μείνει τουλάχιστον ένας ριζικός κόμβος από κάθε κόμβο.

Όσον αφορά για την διαδικασία της ενοποίησης, ο αλγόριθμος που ακολουθείται παρακάτω περιγράφει τα βήματα της ενοποίησης (βλπ. 3.4).

Στην συνέχεια παρατηρούμε αλγόριθμο για την σύνδεση σωρού Fibonacci. (βλπ.3.5).

Στο σημείο αυτό θα αποδειχθεί ότι το λογιστικό κόστος κατά την διαδικασία της εξαγωγής του ελαχίστου σε σωρό Fibonacci ο οποίος αποτελείται από n κόμβους είναι $O(D(n))$. Επίσης το μέγεθος στο ριζικό επίπεδο όταν κληθεί η διαδικασία της ενοποίησης είναι τουλάχιστον $O(D(n) + 1t(H) - 1$, μιας και το ριζικό επίπεδο περιέχει $t(n)$ κόμβοι στο αρχικό επίπεδο καθώς συν και από τους κόμβους που είναι θυγατρικοί του κόμβου που εξάγεται. το σύνολο των οποίων είναι τουλάχιστον $D(n)$. Έτσι λοιπόν για κάθε επανάληψη που εκτελείται η while στον αλγόριθμο στις γραμμές από 6 έως 12, ένας από τους κόμβους στο ριζικό επίπεδο

Αλγοριθμος 3.4: *Αλγόριθμος ΕΝΟΠΟΙΗΣΗ [1]*

```
ΕΝΟΠΟΙΗΣΗ(H)
1. for i=0 εώς D(n[H])
2.   A[i]= κενό
3. για κάθε κόμβο w στο ριζικό επίπεδο του H
4.  χ=w
5.  d=βαθμός[χ]
6.   while A[d] είναι διάφορο του κενού
7.    y=A[d]
8.    if κλειδί[χ] > κλειδί[y]
9.     then εναλλαγή χ,y
10.    ΣΥΝΔΕΣΗ ΣΕ ΣΩΡΟ FIBONACCI(H,y, χ)
11.  A[d]= κενό
12.  d=d+1
13.  A[d]= χ
14.  min[H]= κενό
15. for i=0 εώς D(n[H])
16.   if A[i] είναι διάφορο του κενού
17.    then προσθέτουμε τον A[i] στο ριζικό επίπεδο του H
18. if min[H]=κενό ή κλειδί[A[i]]<κλειδί[min[H]]
19. then min[H]=A[i]
```

Αλγοριθμος 3.5: *ΑΛΓΟΡΙΘΜΟΣ ΣΥΝΔΕΣΗ ΣΕ ΣΩΡΟ Fibonacci [1]*

```
ΣΥΝΔΕΣΗ ΣΕ ΣΩΡΟ FIBONACCI(H,y,x)
1.  αφαιρούμε τον y από το ριζικό επίπεδο του H
2.    θέτουμε τον y θυγατρικό του χ, επαυξάνοντας το πεδίο βαθμός[χ]
3. σήμανση[y]= ΨΕΥΔΕΣ
```

ενώνεται με κάποιο άλλο κόμβο, οπότε λοιπόν το συνολικό έργο που τρέχει στο βρόχο *for* είναι το πολύ $D(n) + t(n)$. Άρα το συνολικό έργο μέχρι να εξάγει ο ελάχιστος κόμβος είναι $O(D(n) + t(n))$.

Όσον αφορά για το δυναμικό προτού εξάγει ο ελάχιστος κόμβος είναι $t(n) + 2m(H)$. Αντίθετα μετά την εξαγωγή του ελάχιστου κόμβου το δυναμικό είναι μικρότερο ή ίσο από το $(D(n) + 1) + 2m(H)$, μιας και έχουν μείνει τουλάχιστον $D(n) + 1$ ριζικοί κόμβοι. Επίσης όταν εκτελείται η πράξη της εξαγωγής κανένας από τους κόμβους δεν επισημαίνεται. Οπότε το λογιστικό κόστος είναι τουλάχιστον

$$O(D(n)+t(H))+((D(n)+1)+2m(H))-t(H)+2m(H)) == O(D(n))+O(t(H))-t(H) = O(D(n))$$

3.4 Μείωση κλειδιού και διαγραφή κόμβου

Στις προηγούμενες ενότητες είδαμε κάποιες από τις πράξεις που χρησιμοποιούνται για τους σωρούς Fibonacci. Στο σημείο αυτό θα δούμε άλλες πράξεις όπως η πράξη για την μείω-

Αλγοριθμος 3.6: *Μείωση κλειδιού σε σωρό Fibonacci [1]*

```
Μείωση κλειδιού σε σωρό FIBONACCI(H, x, k)
1. if k > κλειδί[x]
2.   then    σφάλμα (το νέο κλειδί είναι μεγαλύτερο από το τρέχον)
3. κλειδί[x]=k
4. y=p[x]
5. if y διάφορο του κενού και κλειδί[x] < κλειδί[y]
6. then Αποκοπή(H, x, y)
7.      Κλιμακωτή Αποκοπή(H, y)
8. if κλειδί[x] < κλειδί[min[H]]
9. then min[H]=x
```

Αλγοριθμος 3.7: *Αποκοπή [1]*

```
Αποκοπή(H, x, y)
1. Αφαιρούμε τον x από τον θυγατρικό κατάλογο του y μειώνοντας κατά μονάδα το πεδίο
βαθμός[y]
2. προσθέτουμε τον x στο ριζικό επίπεδο του H
3. π[x]=κένο
4. σύμανση[x]=Ψευδές
```

ση του κλειδιού σε ένα κόμβο σε ένα σωρό Fibonacci με λογιστικό χρόνο $O(1)$ καθώς επίσης η πράξη της διαγραφής από οποιαδήποτε κόμβο σε ένα σωρό Fibonacci αποτελούμενος από n κόμβους και με λογιστικό κόστος $O(D(n))$

3.4.1 Μείωση κλειδιού

Παρακάτω βλέπουμε τον αλγόριθμο για την μείωση κλειδιού σε ένα σωρό Fibonacci (βλπ. 3.6). Επίσης θεωρούμε ότι η εξαγωγή κάποιου κόμβου δεν μεταβάλλει κανένα από τα δομικά πεδία του εξαγόμενου κόμβου.

Συνεχίζοντας βλέπουμε τον αλγόριθμο για την διαδικασία της Αποκοπής (βλπ. 3.7).

Συνεχίζοντας βλέπουμε τον αλγόριθμο για την διαδικασία της κλιμακωτής αποκοπής (βλπ. 3.8).

Αλγοριθμος 3.8: *Κλιμακωτή Αποκοπή [1]*

```
Κλιμακωτή Αποκοπή(H, y)
1. z=p[y]
2. if z διάφορο του κενού
3. then if σήμανση[y]=ψευδές
4.   then σήμανση[x]=αληθές
5. else Αποκοπή(H, y, z)
6.   Κλιμακωτή Αποκοπή(H, z)
```

Η λειτουργία του αλγόριθμου μείωσης κλειδιού σε σωρό Fibonacci λειτουργεί με τον εξής τρόπο:

Όπως βλέπουμε στις γραμμές 1-3 στον αλγόριθμο δηλώνει ότι ο νέος κόμβος με το νέο κλειδί δεν θα είναι μεγαλύτερο από το τρέχων κλειδί του x, και στη συνέχεια γίνεται ανάθεση στο χ με το νέο κλειδί. Έτσι λοιπόν εάν ο x κόμβος ανήκει στο ριζικό επίπεδο δηλαδή είναι ριζικός κόμβος ή εάν το κλειδί κλειδί$[x]$ > κλειδί$[y]$, για το οποίο ισχύει ότι ο κόμβος y είναι πατέρας του x, τότε δεν χρειάζεται κάποια δομική μεταβολή, μιας και ότι δεν έχει παραβιαστεί η σειρά σωρού ελαχίστου. Αυτό φαίνεται στον αλγόριθμο στις γραμμές 4-5 όπου γίνεται έλεγχος αυτού του ενδεχόμενου. Όμως εάν η διάταξη του σωρού ελαχίστου έχει παραβιαστεί τότε θα υπάρχουν μεταβολές.

Στην συνέχεια στην γραμμή 6 γίνεται η αποκοπή του x. Με την διαδικασία της αποκοπής καταργείται η σύνδεση του x με του πατρικού που είναι ο y, και ο x αναπαριστά τον ριζικό κόμβο.

Έτσι για να έχουμε επιτυχία στα επιθυμητά χρονικά φράγματα χρησιμοποιείται η διαδικασία της σήμανσης όπου καταγράφεται το ιστορικό του κάθε κόμβου.

Επίσης στην γραμμή 7 του αλγόριθμου της μείωσης κλειδιού σε σωρό Fibonacci εκτελείται η διαδικασία της κλιμακωτής αποκοπής όπου εκτελείται στο y. Έτσι λοιπόν εάν ο κόμβος y είναι ο ριζικός τότε όπως βλέπουμε στην γραμμή 2 του αλγόριθμου 3.8 της κλιμακωτής αποκοπής η διαδικασία επιστρέφει. Εάν όμως ο κόμβος y είναι μη επισημασμένος η διαδικασία τον επισημαίνει όπως βλέπουμε στην γραμμή 3 στον κώδικα του αλγόριθμου. Αντίθετα εάν ο κόμβος είναι επισημασμένος εννοεί ότι έχασε τον δεύτερο θυγατρικό και ο κόμβος y όπως φαίνεται στην γραμμή 5 με την διαδικασία της αποκοπής αποκόπτεται. Τέλος η διαδικασία της κλιμακωτής αποκοπής εκτελείται στην γραμμή 6 καλώντας αναδρομικά τον εαυτό της μέχρι να βρεθεί ένας ριζικός κόμβος είτε ένας μη επισημασμένος κόμβος. Έτσι μετά την περάτωση όλων αυτών των διαδικασιών ολοκληρώνεται η διαδικασία και επιστρέφει ο κόμβος $min[H]$ όπως φαίνεται στις γραμμές 8 και 9 στον αλγόριθμο της μείωσης κλειδιού σε σωρό Fibonacci.

Παρακάτω βλέπουμε ένα σχήμα με όλη την διαδικασία της μείωση κλειδιού σε σωρό Fibonacci.

Σχετικά με το κόστος του λογιστικού κόστους αποδεικνύεται ότι το λογιστικό κόστος της μείωσης κλειδιού σε ένα σωρό Fibonacci είναι $O(1)$. Έτσι η μείωση κλειδιού σε σωρό Fibonacci χρειάζεται $O(1)$ χρόνο μαζί με τον χρόνο που χρειάζεται η διαδικασία της κλιμακωτής αποκοπής. Έτσι εάν θεωρήσουμε ότι για μια δεδομένη χρονική στιγμή που εκτελείται η μείωση κλειδιού σε σωρό Fibonacci, η διαδικασία της κλιμακωτής αποκοπής καλεί αναδρομικά τον εαυτό της c φορές τότε το πραγματικό κόστος της μείωσης του κλειδιού σε σωρό Fibonacci είναι $O(c)$.

Ωστόσο θα υπολογισθεί η μεταβολή του δυναμικού. Θεωρούμε ότι H είναι ένας σωρός Fibonacci. Έτσι σε κάθε αναδρομή της διαδικασίας της κλιμακωτής αποκοπής γίνεται αποκοπή ενός επισημασμένου κόμβου. Οπότε υπάρχουν $t(H) + c$ δέντρα και τουλάχιστον $m(H) - c + 1$ επισημασμένοι κόμβοι. Άρα η μεταβολή του δυναμικού θα είναι μικρότερη ή ίση της ποσότητας. Οπότε προκύπτει ως εξής:

$((t(H) + c) + 2(m(H) - c + 2)) - (t(H) + 2m(H)) = 4 - c$

Ως συμπέρασμα προκύπτει ότι το λογιστικό κόστος της μείωσης κλειδιού σε σωρό Fi-

Αλγοριθμος 3.9: *Διαγραφή κόμβου [1]*

Διαγραφή κόμβου από σωρό`Fibonacci(H, x)`
1. Μείωση κλειδιού σε σωρό `Fibonacci(H, x)`
2. Εξαγωγή ελαχίστου από σωρό `Fibonacci(H)`

bonacci είναι τουλάχιστον $O(c) + 4 - c = O(1)$.

3.4.2 Διαγραφή κόμβου

Η διαδικασία της διαγραφής κόμβου σε ένα σωρό Fibonacci αποτελούμενος από n με λογιστικό χρόνο $O(D(n))$ θεωρείται απλή. Παρακάτω βλέπουμε τον αλγόριθμο 3.9 της διαγραφής κόμβου σε μορφή ψευδοκώδικα. Επίσης θεωρούμε ότι για κάποια δεδομένη στιγμή δεν υπάρχει κλειδί στο σωρό Fibonacci με τιμή - άπειρο.

Η συγκεκριμένη πράξη της διαγραφής κόμβου σε σωρό Fibonacci είναι ανάλογη με την διαγραφή κόμβου σε ένα διωνυμικό σωρό. Στην περίπτωση συτή ο κόμβος x είναι ο ελάχιστος κόμβος του σωρού αναθέτωντας σε αυτόν το κόμβο μία μοναδική τιμή η οποία είναι η ελάχιστη τιμή κλειδιού - άπειρου. Συνεχίζοντας ο κόμβος x εξάγεται δηλαδή αφαιρείται από τον σωρό με την βοήθεια της διαδικασίας εξαγωγής ελαχίστου από ένα σωρό Fibonacci.

Επίσης όσον αφορά για τον λογιστικό χρόνο της πράξης της διαγραφής κόμβου είναι ίσο με το άθροισμα του λογιστικού χρόνου $O(1)$ της μείωσης κλειδιού σε σωρού Fibonacci και του λογιστικού χρόνου $O(D(n))$ της πράξης της εξαγωγής ελαχίστου από ένα σωρό Fibonacci [2].

Κεφάλαιο 4

Υλοποίηση

4.1 Υλοποίηση σωρών Fibonacci με την γλώσσα προγραμματισμού C

Στην ενότητα αυτή βλέπουμε την υλοποίηση ενός σωρού Fibonacci με την γλώσσα προγραμματισμού *C*.

Αλγοριθμος 4.1: *Αλγόριθμος Συνένωση σωρών Fibonacci [6]*

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<malloc.h>
#define NIL 0
int node;
/*                                    *
struct fibheap_node
{
struct fibheap_node *parent;
struct fibheap_node *left;
struct fibheap_node *right;
struct fibheap_node *child;
int degree;
int mark;
int key;
};
struct fibheap_node *min,*min1;
/*                                  Fibonacci */
void create_fibonacci()
{
min->parent=NIL;
min->key=NIL;
min->left=NIL;
min->right=NIL;
min->child=NIL;
node=0;
}
/*                                                  Fibonacci */
void fibinsert(int val)
{
struct fibheap_node *fheap;
fibheap=malloc(sizeof(struct fibheap_node));
fibheap->key=val;
fibheap->parent=NIL;
fheap->left=NIL;
fheap->right=NIL;
fheap->child=NIL;
if(min->key!=NIL)
{
fheap->right=min;
fheap->left=min->left;
(min)->left=fheap;
(fheap->left)->right=fibheap;
if(val<min->key)
min=fheap; }
else
{
min=fibheap;
min->left=min;
min->right=min;
}
node++;
}
```

Αλγοριθμος 4.2: *Αλγόριθμος Συνένωση σωρών Fibonacci [6]*

```
int main ()
{
int option,n,i,m;
clrscr();
min=NIL;
while(1)
{printf("\nmenu\n");
printf("1:create fibonacci heap\n");
printf("2:insert in fibonacci heap\n");
printf("3: find min in fibonacci heap \n");
printf("4:display\n");
printf("5: exit \n");
scanf ("%d",&option);
switch(option)
{
case 1 :create_fib();
break;
case 2: printf("\nenter the element= ");
scanf("%d",&n);
Finsert(n);
break;
case 3: findmin();
break;
case 4: display(min1);
break;
case 5 :exit(1);
default: printf("\nwrong choice... try again \n ");
  }}}
```

Κεφάλαιο 5

Σύγκριση με άλλες δομές

Στο κεφάλαιο αυτό θα συγκριθούν οι σωροί Fibonacci με άλλες δομές δεδομένων όπως είναι αυτών των δυωνυμικών σορών καθώς επίσης θα παρουσιασθούν πλεονεκτήματα και μειονεκτήματα.

5.1 Σύγκριση σορών Fibonacci με τους δυωνυμικούς σωρούς

Στο κεφάλαιο 3 έγινε μία μελέτη στους σωρούς Fibonacci οι οποίοι παρουσιάζουν καλύτερα χρονικά φράγματα στις πράξεις τους, καθώς επίσης χρησιμοποιούν λογιστικά χρονικά φράγματα για τον χρόνο εκτέλεσης.

Επίσης υπάρχει και μία άλλη κατηγορία των δομών δεδομένων οι οποίοι παρομοιάζουν με τους σωρούς Fibonacci και χρησιμοποιούν τις ίδιες πράξεις όπως κατασκευή σωρού, εισαγωγή σωρού, εισαγωγή ελάχιστου κόμβου, συνένωση κόμβου ή συγχώνευση, μείωση κλειδιού, διαγραφή κόμβου. Αυτές οι δομές δεδομένες λέγονται διωνυμικοί σωροί οι οποίοι σε σύγκριση με τους σωρούς Fibonacci παρουσιάζουν κάποιες διαφορές. Επίσης οι διωνυμικοί σωροί λέγονται συγχωνεύσιμοι σωροί.

Παρακάτω βλέπουμε τον ορισμό ενός διωνυμικού σωρού:

Ορισμός 5.7. *Ένας διωνυμικός σωρός H είναι ένα σύνολο αποτελούμενα από διωνυμικά δέντρα για τα οποία ισχύουν ώς εξής:*

- *Τα διωνυμικά δέντρα στο σωρό H έχουν την ιδιότητα σωρού ελαχίστου έτσι ώστε το κλειδί ενός κόμβου να είναι μεγαλύτερο ή ίσο του κλειδιού του κόμβου που είναι ο πατρικός. Έτσι στην περίπτωση αυτή ένα δέντρο θα λέγεται δομημένο κατά ελαχίστου.*
- *Για κάθε $k > 0$, όπου k είναι ένας ακέραιος θετικός, υπάρχει τουλάχιστον ένα διωνυμικό δέντρο στο σωρό H όπου ο βαθμός ριζικού κόμβου είναι k.*

Παραπάνω είδαμε τι είναι ένας δυωνυμικός σωρός. Τώρα θα δούμε τι είναι ένα δυωνυμικό δέντρο. Οπότε ένας ορισμός για τα διωνυμικά δέντρα είναι ως εξής:

Ορισμός 5.8. *Ένα διωνυμικό δέντρο είναι ένα διατεταγμένο δέντρο το οποίο ορίζεται αναδρομικά. Ένα δυωνυμικό δέντρο έχει 2^k κόμβους και k ύψος.*

Έτσι λοιπόν συγκρίνοντας τους σωρούς Fibonacci με τους διωνυμικούς σωρούς προκύπτουν κάποιες διαφορές μεταξύ τους:

Όσον αφορά για τους διωνυμικούς σωρούς έχουν προκύψει τα εξής:

- Ο χρόνος εκτέλεσης στην χειρότερη περίπτωση που παρουσιάζουν οι δυωνυμικοί σωροί στις πράξεις τους είναι $O(logn)$, όπου n δηλώνει το σύνολο των στοιχείων σε ένα σωρό.
- Στην διαδικασία της συγχώνευσης, ο χρόνος εκτέλεσης που χρειάζεται η συνένωση είναι $O(logn)$.
- Η πράξη της συνένωσης στην χειρότερη περίπτωση χρειάζεται χρόνο εκτέλεσης $\Theta(n)$.

Επιπλέον στο παρακάτω πίνακα βλέπουμε τους χρόνους εκτέλεσης από τις πράξεις που χρησιμοποιούν οι δυωνυμικοί σωροί και οι σωροί Fibonacci στην χειρότερη περίπτωση. Ωστόσο οι δυωνυμικοί σωροί που χρησιμοποιούν τις ίδιες πράξεις έχουν χρόνο εκτέλεσης $O(logn)$ στη χειρότερη περίπτωση. Αντίθετα οι σωροί Fibonacci έχουν μία βέλτιστη και καλύτερη εκδοχή σε σχέση με τους δυωνυμικούς σωρούς. Έτσι η επίδοση του χρόνου μετριέται μέσω των λογιστικών φραγμάτων. Επίσης συμβολίζουμε με n στον πίνακα το σύνολο των στοιχείων των σωρών για κάποια χρονική στιγμή που τρέχει η πράξη.

Διαδικασία	Δυωνυμικός σωρός	Σωρός Fibonacci
Κατασκευή σωρού	$\Theta(1)$	$\Theta(1)$
Εισαγωγή	$O(logn)$	$\Theta(1)$
Ελάχιστο	$O(logn)$	$\Theta(1)$
Εξαγωγή ελαχίστου	$\Theta(logn)$	$O(logn)$
Συνένωση	$\Theta(logn)$	$\Theta(1)$
Μείωση κλειδιού	$\Theta(logn)$	$\Theta(1)$
Διαγραφή	$\Theta(logn)$	$O(logn)$

Πίνακας 5.1: *Χρόνος εκτέλεσης πράξεων στη χειρότερη περίπτωση*

Επίσης όσο οι διωνυμικοί σωροί όσο και οι σωροί Fibonacci δεν έχουν καλές επιδόσεις στην πράξη της Αναζήτηση όπου η αναζήτηση ενός κόμβου με κάποιο κλειδί είναι πιθανόν να απαιτεί περισσότερο χρόνο με αποτέλεσμα να είναι χρονοβόρα η πράξη της αναζήτησης.

5.2 Πλεονεκτήματα και μειονεκτήματα

Στην τρέχουσα ενότητα θα καταγραφούν κάποια πλονεκτήματα και μειονεκτήματα στους διωνυμικούς σωρούς και σωρούς Fibonacci.

Ένα πλεονέκτημα των σορών Fibonacci σε σύγγριση με των διωνυμικών σορών είναι ότι οι σωροί Fibonacci οι οποίοι χρησιμοποιούν τις ίδιες πράξεις με τους διωνυμικούς σωρούς είναι ότι οι πράξεις οι οποίες δεν περιέχουν την διαγραφή κάποιου στοιχείου έχουν λογιστικό χρόνο $O(1)$ ενώ οι δυωνυμικοί σωροί στην χειρότερη περίπτωση έχουν $O(logn)$ χρόνο στις πράξεις εισαγωγή, ελαχίστου, εξαγωγή ελαχίστου, συνένωση, μείωση κλειδιού και διαγραφή σε συγχωνεύσιμο σωρό.

Επίσης ένα άλλο βασικό πλεονέκτημα που παίζει σημαντικό ρόλο είναι η συνάρτηση δυναμικού η οποία χρησιμοποιείται για αντισταθματική ανάλυση.

Ένα μειόνεκτημα στους σωρούς Fibonacci και στους δυωνυμικούς σωρούς είναι ότι δεν παρουσιάζουν καλές επιδόσεις για την πράξη της αναζήτησης η οποία παρομοιάζει χρονοβόρα στην εύρεση κόμβου με κάποιο δεδομένου κλειδί.

Κεφάλαιο 6

Επίλογος

Στο κεφάλαιο αυτό γίνεται μία σύνοψη της πτυχιακής εργασίας με τα συμπεράσματα.

6.1 Συμπεράσματα

Στην πτυχιακή εργασία μελετήθηκαν οι βασικές πράξεις που αφορά τους σωρούς Fibonacci οι οποίοι μοιάζουν με τους διωνυμικούς σωρούς. Γενικά είδαμε ότι οι σωροί Fibonacci είναι συλλογές από δέντρα. Έτσι τα συμπεράσματα που έχουμε από τις πράξεις των σωρών Fibonacci είναι ότι τα φράγματα που παρουσιάζουν είναι σημαντικά αφού παρουσιάζουν καλύτερα φράγματα για αυτές τις πράξεις. Επιπλέον για τις πράξεις Εισαγωγή, Ελάχιστο και συνένωση στους σωρούς Fibonacci προκύπτει ότι έχουν λογιστικό χρόνο εκτέλεσης $O(1)$ ενώ οι πράξεις Εξαγωγή ελαχίστου, διαγραφή έχουν λογιστικό χρόνο $O(logn)$.

Βιβλιογραφία

[1] Ronald L. Rivest Clifford Stein Thomas H. Cormen, Charles E. Leiserson. *Εισαγωγή στους Αλγορίθμους*. 2013.

[2] Ι. Παπουτσής. *Εισαγωγή στις Δομές Δεδομένων και στους Αλγόριθμους, Υλοποιήση σε C*. ΑΘ. ΣΤΑΜΟΥΛΗΣ, Αθήνα, 2010.

[3] Νικόλαος Μισυρλής. *Δομές Δεδομένων με C*. 2008.

[4] Umesh Vazirani Sanjou Dasgupta, Christos Papadimitriou. *Αλγόριθμοι*. ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ, 2009.

[5] Levitin Anany. *Ανάλυση και σχεδίαση αλγορίθμων*. ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ ΥΙΟΙ Α.Ε, 2008.

[6] *C code for fibonacci Heap*. http://www.coders-hub.com/2013/04/c-code-for-fibonacci-heap-tree.html#.Wacpn9FLfIV/.

[7] *Algorithm*. https://en.wikipedia.org/wiki/Algorithm.

[8] *Fibonacci heap*. https://en.wikipedia.org/wiki/Fibonacci_heap.

[9] *Heap (data structure)*. https://en.wikipedia.org/wiki/Heap_(data_structure).

[10] *Binomial hap*. https://en.wikipedia.org/wiki/Binomial_heap.

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
Εισαγωγή	Insertion
Αναζήτηση	Searching
Διαγραφή	Deletion
Ταξινόμηση	Sorting
Προσπέλαση	Access
Συγχώνευση	Mergin
Ντετερμινιστικός πολυωνυμικός χρόνος	Deterministic Polynomial Time