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

Πιθανοτικοί Αλγόριθμοι-Η Γρήγορη ταξινόμηση σαν ένα παράδειγμα

Μελέτη και υλοποίηση

Πτυχιακη Εργασια

της

ΠΑΡΑΣΚΕΥΗΣ Γ. ΡΑΦΤΟΠΟΥΛΟΥ

Επιβλέπων: Γρηγόριος Καραγιώργος
Επίκουρος Καθηγητής

Σπάρτη, Νοέμβριος 2015

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

Πιθανοτικοί Αλγόριθμοι-Η Γρήγορη ταξινόμηση σαν ένα παράδειγμα

Μελέτη και υλοποίηση

Πτυχιακη Εργασια

της

ΠΑΡΑΣΚΕΥΗΣ Γ. ΡΑΦΤΟΠΟΥΛΟΥ

Επιβλέπων: Γρηγόριος Καραγιώργος
Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25α Νοεμβρίου 2015.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γρηγόριος Καραγιώργος	Ιωάννης Λιαπέρδος	Νικόλαος Κατσάκος-Μαυρομιχάλης
Επίκουρος Καθηγητής	Καθηγητής Εφαρμογών	Καθηγητής

Σπάρτη, Νοέμβριος 2015

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

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

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)

.....
Παρασκευή Ραφτοπούλου

Περίληψη

Αντικείμενο της παρούσης πτυχιακής εργασίας είναι η παρουσίαση του πιθανοτικού ή τυχαιοκρατικού αλγορίθμου της γρήγορης ταξινόμησης. Συγκεκριμένα ο αλγόριθμος που παρουσιάζεται είναι η τυχαιοκρατική ταχυταξινόμηση ο οποίος θα υλοποιηθεί στην γλώσσα προγραμματισμού C. Επίσης αναλύεται ο χρόνος εκτέλεσής του και στην συνέχεια γίνεται σύγκριση με τους αλγόριθμους της γρήγορης ταξινόμησης, της συγχωνευτικής ταξινόμησης και της ταξινόμησης σωρού. Όσον αφορά τους πιθανοτικούς αλγόριθμους έχουν αποδειχθεί χρήσιμοι για την απλότητα και την αποδοτικότητά τους. Ο πιθανοτικός αλγόριθμος της γρήγορης ταξινόμησης αποδεικνύεται ότι έχει χρόνο εκτέλεσης $O(n^2)$ στην χειρότερη περίπτωση και $O(nlogn)$ κατά μέσο όρο.

Λέξεις Κλειδιά

Πιθανοτικός αλγόριθμος, Τυχαιοκρατική Ταχυταξινόμηση, Γρήγορη Ταξινόμηση, Συγχωνευτική Ταξινόμηση, Ταξινόμηση Σωρού, χρόνος εκτέλεσης.

Abstract

The purpose of this thesis is the introduction of probabilistic or randomized algorithm of quick sort. Specifically the presented algorithm is the randomized quick sort which will be implemented in the programming language C. Also, the running time is analyzed and then compared with the algorithms of quick sort, merge sort and heap sort. Concerning the randomized algorithms, they have been demonstrated as useful for their simplicity and efficiency. The randomized algorithm of quick sort turns is proven that has execution time $\Theta(n^2)$ at worst case scenario and O(nlogn) on average.

Keywords

Probabilistic algorithm, Randomized quick sort, Quick sort, Merge sort, Heap sort, execution time.

στους γονείς μου, Γιώργο και Μαρία

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον επίκουρο καθηγητή κ. Γρηγόριο Καραγιώργο για την επίβλεψη αυτής της πτυχιακής εργασίας, την ευκαιρία που μου έδωσε να την εκπονήσω και την εξαιρετική συνεργασία που είχαμε. Επίσης ευχαριστώ θερμά τον καθηγητή Εφαρμογών κ. Ιωάννη Λιαπέρδο καθώς και την συμφοιτήτριά μου και φίλη μου Σταυρούλα Δαφνά για την πολύτιμη βοήθειά τους πάνω στο λογισμικό στο οποίο εργάστηκα. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου, τις αδερφές μου και τους φίλους μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Σπάρτη,Νοέμβριος 2015

Παρασκευή Ραφτοπούλου

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Πρόλογος

Η πτυχιακή αυτή εργασία εκπονήθηκε στα πλαίσια του προγράμματος σπουδών για την απόκτηση πτυχίου από το τμήμα Μηχανικών Πληροφορικής Τ.Ε. του Τ.Ε.Ι. Πελοποννήσου με έδρα την Σπάρτη, της σχολής Τεχνολογικών εφαρμογών. Στόχος της παρούσας πτυχιακής είναι να παρουσιάσει την πιθανοτική εκδοχή της ταχυταξινόμησης. Ο αλγόριθμος θα υλοποιηθεί στην γλώσσα προγραμματισμού C και θα υπολογισθεί ο χρόνος εκτέλεσής του.

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της εργασίας

Ο όρος αλγόριθμος είναι θεμελιώδης και πολύ σημαντικός στην επιστήμη της πληροφορικής. Ειδικότερα με τον όρο αλγόριθμο εννοείται μία απλά ορισμένη υπολογιστική διαδικασία, ουσιαστικά μία σειρά πεπερασμένων βημάτων ή εντολών η οποία δέχεται σαν είσοδο μία τιμή ή ένα σύνολο τιμών και επιστρέφει σαν έξοδο ένα αποτέλεσμα. Ένας αλγόριθμος μπορεί να χρησιμοποιηθεί ως εργαλείο για την επίλυση προβλημάτων και εκτέλεση άλλων διαδικασιών που παρουσιάζονται σήμερα και σε άλλους τομείς εκτός της πληροφορικής, όπως είναι η ιατρική, τα Μαθηματικά, η Οικονομία και πολλές άλλες. Η υπαρξη αλγορίθμων ωστόσο βοηθά τους προγραμματιστές να τους υλοποιήσουν με την βοήθεια διάφορων γλωσσών προγραμματισμού όπως είναι λ.χ. η C και η Prolog. Ωστόσο στην περίπτωση που ένας αλγόριθμος δέχεται μία είσοδο και το αποτέλεσμα που εξάγει δεν είναι ορθό, τότε δεν θεωρείται αλγόριθμος.

Η μελέτη της αναλυσης ενός αλγορίθμου αποτελεί μία πολύ σημαντική διαδικασία, καθώς μέσω αυτής υπολογίζεται η πολυπλοκότητά του, δηλαδή ο αριθμός των βημάτων που θα εκτελέσει έτσι ώστε να βρει το αποτέλεσμα. Η πολυπλοκότητα ενός αλγορίθμου μπορεί να είναι **καλύτερης** και **χειρότερης περίπτωσης** και για την πλήρη ανάλυσή του είναι απαραίτητη η μελέτη και των δύο αυτών περιπτώσεων. Η διαδικασία αυτή στην παρούσα πτυχιακή θα πραγματοποιηθεί για κάθε αλγόριθμο.

Ένα σημαντικό θέμα που προκύπτει πολλές φορές κατά την προσπάθεια επίλυσης προβλημάτων με την βοήθεια αλγορίθμων είναι η **ταξινόμηση**. Με τον όρο ταξινόμηση εννοείται ο τρόπος οργάνωσης και ιεράρχησης ή κατάταξης ενός συνόλου στοιχείων είτε αυτά είναι αριθμοί είτε γράμματα με στόχο την ευκολότερη διαδικασία επίλυσης ενός προβλήματος που δίνεται. Όπως στην καθημερινότητα η διαδικασία της ταξινόμησης είναι χρήσιμη λ.χ. για την αρχειοθέτηση εγγράφων και άλλων διαδικασιών έτσι και στην επιστήμη της πληροφορικής η ταξινόμηση χρησιμοποιείται για την διευκόλυνση της επίλυσης διαφόρων προβλημάτων, και για την διαδικασία αυτήν δημιουργήθηκαν οι **αλγόριθμοι ταξινόμησης.** Παράλληλα υπάρχουν και άλλοι αλγόριθμοι οι οποίοι δεν έχουν τον ίδιο τρόπο λειτουργίας με τους αλγόριθμους ταξινόμησης, αλλά βασίζονται στην τύχη. Αυτοί οι αλγόριθμοι ονομάζονται **πιθανοτικοί ή τυχαιοκρατικοί.**

- **Οι αλγόριθμοι Ταξινόμησης:** Όπως προαναφέρθηκε, δημιουργήθηκαν με σκοπό την ευκολότερη επίλυση ενός προβλήματος και είναι ευρέως γνωστοί, διαδεδομένοι

και χρησιμοποιημένοι ιδιαίτερα στον τομέα της Πληροφορικής. Παράλληλα χρησιμοποιούνται για την ταξινόμηση πολύπλοκων προβλημάτων. Για αυτόν τον λόγο κάθε αλγόριθμος ταξινόμησης χρησιμοποιεί την δική του μέθοδο για να τακτοποιήσει π.χ. ένα σύνολο αριθμών που του δίνεται, για παράδειγμα ο αλγόριθμος ταξινόμησης σωρού έχει σαν μέθοδο την επιλογή για να ταξινομήσει τα στοιχεία ενός σωρού.

- **Οι πιθανοτικοί αλγόριθμοι:** Πρόκειται για αλγόριθμους οι οποίοι βασίζονται στην τύχη. Αναλυτικότερα οι επιλογές που κάνουν είναι **τυχαίες** και η εξέλιξή του εξαρτάται από αυτές. Είναι απλοί αλγόριθμοι και συνήθως είναι ταχύτεροι από άλλους.

Στην παρούσα πτυχιακή εργασία θα παρουσιασθεί ο πιθανοτικός αλγόριθμος της γρήγορης ταξινόμησης. Στην συνέχεια θα αναλυθεί και θα αποδειχθεί ο χρόνος εκτέλεσής του στην καλύτερη και την χειρότερη περίπτωση. Έπειτα θα πραγματοποιηθεί η προγραμματιστική του υλοποίηση στην γλώσσα C καθώς και η σύγκρισή του με τους αλγόριθμους της συγχωνευτικής ταξινόμησης, γρήγορης ταξινόμησης και ταξινόμησης σωρού.

1.2 Δομή της εργασίας

Η πτυχιακή εργασία είναι σχεδιασμένη ως εξής:

Στο κεφάλαιο 2 γίνεται μία εισαγωγή στους πιθανοτικούς αλγόριθμους και στην διαδικασία παραγωγής τυχαίων αριθμών. Επίσης παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματα των αλγορίθμων αυτών.

Στο κεφάλαιο 3 ορίζεται επακριβώς η έννοια της πολυπλοκότητας ενός αλγορίθμου, της καλύτερης και χειρότερης περίπτωσης. Επίσης ορίζεται ο ασυμπτωτικός συμβολισμός η χρήση του οποίου γίνεται για πολύ μεγάλες εισόδους που δίνονται στον αλγόριθμο. Παράλληλα παρουσιάζονται συμβολισμοί όπως είναι οι Θ και Ω καθώς και μαθηματικές εκφρασεις με αυτούς.

Στο κεφάλαιο 4 παρουσιάζονται αλγόριθμοι ταξινόμησης. Για κάθε κάθε αλγόριθμο περιγράφεται η διαδικασία της λειτουργίας του, δίνεται ο χρόνος εκτέλεσής του και ένας ψευδοκώδικας.

Στο κεφάλαιο 5 εισάγεται ο αλγόριθμος της πιθανοτικής ταχυταξινόμησης και της διαδικασίας της διαμέρισής του. Στην συνέχεια αποδεικνύεται και υπολογίζεται ο χρόνος εκτέλεσής του και τέλος υλοποιείται στην γλώσσα προγραμματισμού C.

Το κεφάλαιο 6 αποτελεί μία εισαγωγή στην θεωρία των πιθανοτήτων, ορίζονται έννοιες όπως είναι ο δειγματικός χώρος, η πιθανότητα, τα σύνολα και παρουσιάζονται άλλες όπως είναι η μέση τιμη. Παράλληλα δε αναφέρονται και δύο πιθανοτικοί αλγόριθμοι, ο αλγόριθμος της πρόσληψης και της τυχαίας μετάθεσης συστοιχίας.

Στο κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα που προέκυψαν από την μελέτη της εργασίας.

Κεφάλαιο 2

Πιθανοτικοί Αλγόριθμοι

Το κεφάλαιο αυτό αποτελεί μία εισαγωγή στους πιθανοτικούς αλγορίθμους. Στο σημείο αυτό παρατίθενται πολύ συνοπτικά δύο παραλλαγές αυτών των αλγορίθμων. Στην συνέχεια γίνεται μία αναφορά στην παραγωγή των τυχαίων αριθμών με την χρήση και την βοήθεια των συναρτήσεων random() και srandom() και στο τέλος αναφέρονται πλεονεκτήματα και μειονεκτήματα των αλγορίθμων αυτών.

2.1 Πιθανοτικοί ή Τυχαιοκρατικοί Αλγόριθμοι

Οι Πιθανοτικοί ή Τυχαιοκρατικοι αλγόριθμοι υπάγονται σε μία κατηγορία αλγορίθμων οι οποίοι βασίζονται στην τύχη, δηλαδή συμπεριφέρονται με **τυχαίο τρόπο.** Συγκεκριμένα, οι αλγόριθμοι αυτοί είναι δυνατόν να εκτελεστούν n φορές, και παρόλα αυτά σε κάθε εκτέλεση να παράγουν διαφορετικό αποτέλεσμα. Η εφαρμογή τους είναι γνωστή σε θέματα λήψης αποφάσεων, όπως λ.χ. είναι η τοποθέτηση οκτώ βασιλισσών σε μία σκακιέρα, χωρίς η μία να απειλεί την άλλη. Υπάρχουν διαφορετικά είδη πιθανοτικών αλγορίθμων. Δύο από αυτά είναι:

- **Οι αλγόριθμοι Monte Carlo**: Πρόκειται για αλγόριθμους οι οποίοι εκτελούνται πάντα γρήγορα αλλά υπάρχει πιθανότητα η έξοδός τους να είναι λανθασμένη.

- **Οι αλγόριθμοι Las Vegas** : Πρόκειται για αλγόριθμους οι οποίοι διαβεβαιώνουν πως έχουν σύντομο χρόνο εκτέλεσης με μεγάλη πιθανότητα ενώ παράλληλα δίνουν πάντα σαν έξοδο ένα ορθό αποτέλεσμα [6].

2.2 Παραγωγή τυχαίων αριθμών

Γενικότερα ένας αλγόριθμος χαρακτηρίζεται ως πιθανοτικός εάν η συμπεριφορά του δεν καθορίζεται μόνο από την είσοδό του αλλά και από τις τιμές που προκύπτουν από μια συνάρτηση παραγωγής τυχαίων αριθμών, η οποία ονομάζεται **γεννήτρια τυχαίων αριθμών (random generator runtine).** Γεννήτριες τυχαίων αριθμών υπάρχουν σε κάθε γλώσσα προγραμματισμού πέρα της C . Η συνάρτηση αυτή είναι γνωστή ως συνάρτηση rand() . Υποθέστε πως έχετε μία συνάρτηση rand (m, n) η οποία επιστρέφει έναν ακέραιο αριθμό μεταξύ αυτών, έτσι ώστε οι ακέραιοι να είναι μεταξύ τους όλοι ισοπίθανοι.

Η συνάρτηση rand(m,n) σε αυτήν την περίπτωση θα δίνει είτε m είτε n με πιθανότητα $\frac{1}{2}$ για το κάθε στοιχείο. Ομοίως εάν θεωρήσετε την συνάρτηση rand(2,8) με μία κλήση της επιστρέφεται ένας από τους αριθμούς εντός του διαστήματός της συμπεριλαμβανομένων και των δύο άκρων της . Δηλαδή η συνάρτηση θα επιστρέφει έναν από τους αριθμούς 2, 3, 4, 5, 6, 7, 8 με πιθανότητα $\frac{1}{7}$ για τον καθένα από αυτούς.

Παράλληλα η συνάρτηση rand στηρίζεται στην λειτουργία μίας άλλης συνάρτησης που ονομάζεται srand() . Συγκεκριμένα η srand χρησιμοποιείται για να ορίσει ένα σημείο το οποίο θα ξεκινήσει την παραγωγή τυχαίων αριθμών από την συνάρτηση rand. Πιο αναλυτικά δίνει σε ένα πρόγραμμα την δυνατότητα κάθε φορά που εκτελείται να μην χρησιμοποιεί ίδιες ακολουθίες τυχαίων αριθμών διότι εάν δεν ορισθεί πριν από την διαδικασία παραγωγής τυχαίων αριθμών μία αρχική τιμή, τότε θα χρησιμοποιείται κάθε φορά μία συγκεκριμένη τιμή που θα ορίζεται αυτόματα από το σύστημα και θα είναι ίδια όσες φορές και να εκτελεσθεί η rand().

Αυτό σημαίνει πως κάθε φορά η ακολουθία τυχαίων αριθμών που θα παράγεται θα είναι ίδια, και για αυτόν ακριβώς τον λόγο η συνάρτηση srand θα καθορίζει μία νέα τιμή που θα μεταβάλλεται κάθε φορά έτσι ώστε τελικά η ακολουθία τυχαίων αριθμών που θα παράγεται να είναι διαφορετική [6].

2.3 Πλεονεκτήματα και Μειονεκτήματα Πιθανοτικών Αλγορίθμων

Όπως αναφέρθηκε σε παραπάνω ενότητα οι πιθανοτικοί-τυχαιοκρατικοί αλγόριθμοι βασίζονται στην τύχη και η εξέλιξή τους εξαρτάται από αυτές. Σαφώς και σε περίπτωση επιλογής της συγκεκριμένης κατηγορίας αλγορίθμων, ταυτόχρονα, υπάρχουν κάποια πλεονεκτήματα αλλά και μειονεκτήματα. Τα πλεονεκτήματα που λαμβάνουν χώρα είναι τα εξής: α)Χαρακτηρίζονται από απλότητα και κομψότητα, β)Συνήθως είναι πιο γρήγοροι από τους κοινούς ή ντετερμινιστικούς αλγόριθμους γ)Δίνουν λύσεις σε προβλήματα πάνω στα οποία δεν υπάρχει πλήρη γνώση των δεδομένων.

Τα μειονεκτήματα που υπάρχουν είναι α) Ενώ σαν τυχαιοκρατικοί αλγόριθμοι δίνουν μία σωστή απάντηση, υπάρχει μία ελάχιστη πιθανότητα να δώσουν λάθος αποτέλεσμα β) Η πολυπλοκότητά τους κυμαίνεται ανάμεσα σε συγκεκριμένες τιμές και γ) Σε περίπτωση δημιουργίας ενός σφάλματος κατά την διάρκεια εκτέλεσής τους, τότε το σφάλμα είναι πολύ δύσκολο να εντοπισθεί [9] [3].

Κεφάλαιο 3

Στοιχεία εκτίμησης της πολυπλοκότητας

Σε αυτό το κεφάλαιο θα παρουσιασθεί η έννοια της πολυπλοκότητας ενός αλγορίθμου, οι χρόνοι καλύτερης και χειρότερης περίπτωσης καθώς και η έννοια του ασυμπτωτικού συμβολισμού, ο οποίος ορίζεται για πολύ μεγάλες εισόδους.

3.1 Πολυπλοκότητα και ανάλυση περιπτώσεων

Με τον όρο πολυπλοκότητα εννοείται ο χρόνος εκτέλεσης ενός αλγορίθμου, δηλαδη ο αριθμός των βημάτων που πρέπει να ακολουθήσει έτσι ώστε να δώσει το σωστό αποτέλεσμα. Στην περίπτωση που ο αλγόριθμος δώσει ένα λάθος αποτέλεσμα, τότε δεν είναι ορθός και κατά συνέπεια δεν θεωρείται αλγόριθμος. Ο χρόνος εκτέλεσης ενός αλγορίθμου είναι **καλύτερης** και **χειρότερης** περίπτωσης.

Ο χρόνος εκτέλεσης της **χειρότερης περίπτωσης** εκφράζει τον μέγιστο χρόνο για οποιαδήποτε είσοδο δίνεται στον αλγόριθμο. Το σίγουρο είναι πως σε περίπτωση που ο χρόνος χειρότερης περίπτωσης είναι γνωστός εξ αρχής, ταυτόχρονα είναι γνωστό και εμφανές πως ο αλγόριθμος δεν πρόκειται να εκτελέσει παραπάνω βήματα, δηλαδή να κάνει περισσότερο χρόνο. Χαρακτηριστικό του χρόνου εκτέλεσης της χειρότερης περίπτωσης είναι πως η εύρεση εναλλακτικής λύσης δεν αποτελεί εύκολη διαδικασία, δηλαδή η ανακάλυψη ενός χρόνου που να κάνει τον αλγόριθμο να εκτελεί λιγότερα βήματα δεν είναι εύκολη.

Ο χρόνος εκτέλεσης της **καλύτερης** περίπτωσης εκφράζει τον ελάχιστο χρόνο για οποιαδήποτε είσοδο δοθεί στον αλγόριθμο που εξετάζεται [1].

Ο χρόνος εκτέλεσης ενός αλγορίθμου μπορεί να εκφραστεί με την χρήση μαθηματικών συναρτήσεων, όπως είναι η $f(n)$ και η $g(n)$, οι οποίες βοηθούν στην προσέγγιση μιας ακριβούς τιμής της πολυπλοκότητάς του καθώς και με ασυμπτωτικό συμβολισμό, όπως είναι ο $O(n)$. Με την βοήθεια της ασυμπτωτικής εκτίμησης, είναι δυνατή η προσέγγιση της κατηγοριοποίησης του αποτελέσματος του χρόνου εκτέλεσης σε μία από τις παραπάνω δύο κατηγορίες. Η Ασυμπτωτική εκτίμηση αγνοεί σταθερές και εστιάζει σε τάξη μεγέθους χρόνου εκτέλεσης.

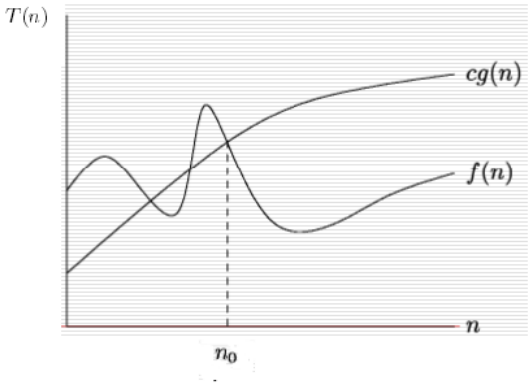
3.2 Ασυμπτωτικός συμβολισμός

3.2.1 Ο συμβολισμός Ο

Ο Ασυμπτωτικός συμβολισμός εκφράζει τα αποτελέσματα της ασυμπτωτικής εκτίμησης. Το σύμβολο Ο δηλώνει πως υπάρχει μόνο ένα ασυμπτωτικό φράγμα [1].

Θεωρώντας τις συναρτήσεις f και g οι οποίες παίρνουν πάντα πραγματικές θετικές τιμές τότε:

Ορισμός 3.1. *Όταν $O(f(n) = g(n)$, υπάρχουν θετικές σταθερές τέτοιες, ώστε για κάθε $n \geq n_0$ να ισχύει $0 < f(n) \leq c \cdot g(n)$. Για κάθε τιμή του n η οποία είναι μεγαλύτερη από το n_0 η τιμή της $f(n)$ βρίσκεται κάτω από την $c \cdot g(n)$ ή απλά συμπίπτει με αυτήν. Στο παρακάτω σχήμα 3.1 δίνεται η γραφική παράσταση του συμβολισμού $O(n)$ [1].*



Σχήμα 3.1: *Γραφική παράσταση συμβολισμού $f(n) = O(g(n)$ [10]*

Θεωρώντας επίσης τα σύμβολα Θ και Ω, ο και ω σαν ακριβείς εκτιμήσεις των τάξεων μεγέθους, παρουσιάζονται και οι αντίστοιχοι ορισμοί τους. [1]

3.2.2 Ο συμβολισμός Θ

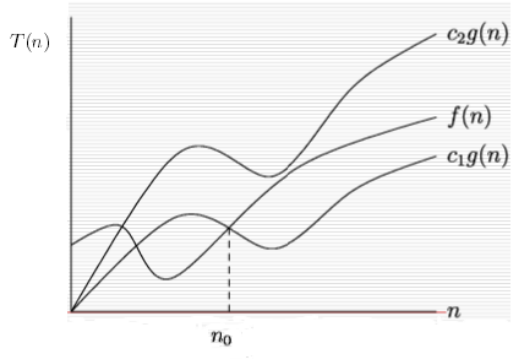
Έστω οι συναρτήσεις f και g, οι οποίες παίρνουν πάντα πραγματικές τιμές.

Ορισμός 3.2. *Όταν $\Theta(g(n)) = f(n)$, υπάρχουν θετικές σταθερές c_1, c_2 και n_0 τέτοιες ώστε να ισχύει $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ για κάθε $n \geq n_0$ [1].*

Μία συνάρτηση $f(n)$ ανήκει στο σύνολο $\Theta g(n)$ εάν υπάρχουν θετικές σταθερές όπως είναι η c_1 και η c_2 τέτοιες ώστε η συνάρτηση $f(n)$ να μπορεί να παρεμβάλλεται μεταξύ των γινομένων $c_1 \cdot g(n)$ και $c_2 \cdot g(n)$. Απο τα συμφραζόμενα προκύπτει πως $f(n) = \Theta(g(n)) \leftrightarrow f(n) = O(g(n))$. Με τον παραπάνω ορισμό διαπιστώνεται πως η συνάρτηση g(n) αποτελεί ένα **αυστηρά ασυμπτωτικό φράγμα** για την συνάρτηση $f(n)$. Σύμφωνα με τον παραπάνω ορισμό του συνόλου $\Theta(g(n))$ η κάθε συνάρτηση-μέλος $f(n)$ που ανήκει στον $\Theta(g(n)$ πρέπει να είναι ασυμπτωτικά μή αρνητική, δηλαδή από κάποια τιμή του n και πάνω η $f(n)$ να έχει μή αρνητική τιμή . Μία συναρτηση η οποία είναι θετική από κάποια τιμή του ορίσματός της και πάνω ονομάζεται **ασυμπτωτικά θετική** [11].

3.2.3 Ο συμβολισμός Ω

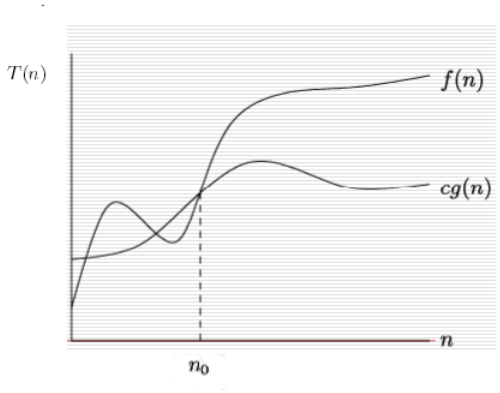
Ο συμβολισμός Ω ορίζει ένα ασυμπτωτικό κάτω φράγμα για μία συνάρτηση. Έτσι προκύπτει ο ορισμός :



Σχήμα 3.2: *Γραφική παράσταση συμβολισμού* $f(n) = \Theta(g(n))$ *[10]*

Ορισμός 3.3. *Όταν* $\Omega(g(n)) = f(n)$ *υπάρχουν θετικές σταθερές* c *και* n_0 *τέτοιες ώστε για κάθε* $n \geq n_0$ *να ισχύει* $0 \leq c \cdot g(n) \leq f(n)$*, δηλαδή η συνάρτηση* $f(n)$ *είναι ένα κάτω φράγμα για την συνάρτηση* $g(n)$*. [1]*

Η έκφραση $\Omega(f(n)) = g(n)$ απεικονίζεται γραφικά στο σχήμα 3.3, όπου αν παρατηρήσει κάποιος για όλες τις τιμές του n και πέρα από το n_0 η τιμή της συναρτησης $f(n)$ βρίσκεται πάνω από την συνάρτηση $c \cdot g(n)$ ή συμπίπτει με αυτήν [1].



Σχήμα 3.3: *Γραφική παράσταση συμβολισμού* $f(n) = \Omega(g(n))$ *[10]*

3.2.4 Ο συμβολισμός o

Ορισμός 3.4. *Όταν ισχύει* $o(g(n)) = f(n)$ *για οποιαδήποτε σταθερά* $c > 0$*, θα υπάρχει σταθερά* $n_0 > 0$ *τέτοια ώστε να ισχύει* $0 \leq f(n) \leq c \cdot g(n)$ *για κάθε* $n > n_0$ *[1].*

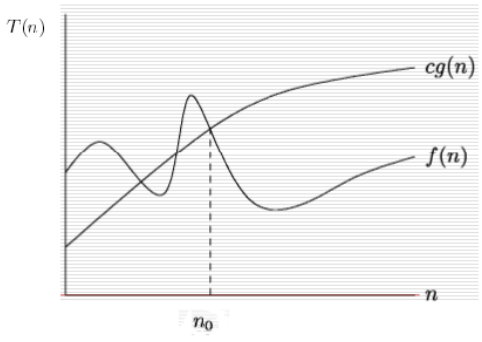
Η βασική διαφορά των συμβολισμών O και o περιγράφεται μέσω της σχέσης $f(n) = O(g(n))$, όπου το φράγμα $0 \leq f(n) \leq c \cdot g(n)$ περιγράφεται για κάποια σταθερά $c > 0$. Παράλληλα στην σχέση $f(n) = o(g(n))$ το φράγμα $0 \leq f(n) \leq c \cdot g(n)$ ισχύει για όλες τις θετικές σταθερές c.

Παράλληλα η συνάρτηση $g(n)$ στην τελευταία περίπτωση του συμβολισμού του o παίρνει πολύ μεγάλες τιμές, καθώς το n τείνει στο άπειρο. Το αντίθετο συμβαίνει με την συνάρτηση

$f(n)$ η οποία παίρνει όλο και μικρότερες τιμές στο σημείο που να θεωρείται ≪αμελητεα ≫. Η σχέση των δύο συναρτήσεων περιγράφεται από το παρακάτω όριο που ισούται με το μηδέν

$$\lim_{h\to\infty} \frac{f(n)}{g(n)} = 0$$

Στην γραφική παράσταση του σχήματος 3.4 απεικονίζεται η σχέση μεταξύ των συναρτήσεων $c \cdot g(n)$ και $f(n)$ [1].



Σχήμα 3.4: *[Γραφική παράσταση συμβολισμού $f(n) = o(g(n))$ [10]*

3.2.5 Ο συμβολισμός ω

Ο συμβολισμός w δηλώνει ένα κάτω φράγμα το οποίο δεν είναι ασυμπτωτικά αυστηρό. Γενικά, το $w(g(n))$ ορίζεται τυπικά ως το σύνολο που περιγράφεται από τον παρακάτω ορισμό:

Ορισμός 3.5. *Όταν $w(g(n)) = f(n)$ για οποιαδήποτε σταθερά $c > 0$, υπάρχει θετική σταθερά n_0 τέτοια ώστε $0 \leq c \cdot g(n) \leq f(n)$ για κάθε $n > n_0$ [1].*

Εξ ορισμού ισχύει πως η συνάρτηση $f(n)$ ανήκει στο $w(g(n))$ όταν και μόνο όταν η συνάρτηση $g(n)$ ανήκει στην $of(n)$. Επίσης η συνάρτηση $f(n)$ παίρνει πολύ μεγάλες τιμές καθώς το n τείνει στο άπειρο, κάτι το οποίο δεν ισχύει για την συνάρτηση $g(n)$ καθώς όσο το n τείνει στο άπειρο οι τιμές της μειώνονται. Η σχέση των δύο συναρτήσεων περιγράφεται από το παρακάτω όριο και η γραφική παράσταση δίνεται από το σχήμα 3.5 [1].

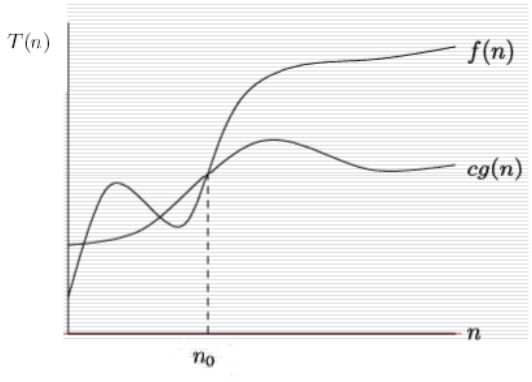
$$\lim_{h\to\infty} \frac{f(n)}{g(n)} = \infty$$

3.3 Ασυμπτωτικός συμβολισμός και μαθηματικές εκφράσεις

Ο ασυμπτωτικός συμβολισμός μπορεί να εκφρασθεί μέσω μαθηματικών σχέσεων. Μία μαθηματική εξίσωση η οποία περιέχει τον συμβολισμό $\Theta(n)$ γράφεται ως εξής:

$$3 \cdot n^2 + 2n + 1 = \Theta(n) + 4n \tag{3.1}$$

Σε μία τέτοια περίπτωση, ο ασυμπτωτικός συμβολισμός μπορεί να θεωρηθεί σαν μία συνάρτηση που δεν είναι ορισμένη σε κάποιο σύνολο [1]. Θεωρώντας ξανά την εξίσωση



Σχήμα 3.5: *Γραφική παράσταση συμβολισμού* $f(n) = w(g(n))$ *[10]*

$$3 \cdot n^2 + 2n + 1 = g(n) + 4n \tag{3.2}$$

και συγκρίνοντας τις δύο εξισώσεις συμπεραίνεται πως η συνάρτηση $g(n)$ είναι ορισμένη στο σύνολο $\Theta(n)$. Από τους ορισμούς του ασυμπτωτικού συμβολισμού προκύπτει πως ένα πλήθος n με $n >= 0$ μπορεί να εκφρασθεί συναρτήσει ενός από τους συμβολισμούς οι οποίοι συναντήθηκαν προηγουμένως, όπως είναι ο $\Omega(n^2)$ και άλλοι που αναφέρθηκαν παραπάνω.

Ειδικότερα εάν στο δεξί μέλος μίας εξίσωσης υπαρχει ένας συμβολισμός τότε το σύμβολο της ισότητας ισοδυναμεί με το σύμβολο του συνόλου. Δηλαδή εάν $n = \Omega(n^3) \iff n \in \Omega(n^3)$. Θεωρείστε την ακόλουθη εξίσωση

$$3 \cdot n^2 + \Theta(n) = O(n^3) \tag{3.3}$$

στην οποία εμπεριέχονται οι συμβολισμοί $O(n)$ και $O(n^3)$ στο πρώτο και στο δεύτερο μέλος αντίστοιχα. Η ισότητα μπορεί να ερμηνευθεί μέσω δύο συναρτήσεων $f(n)$ και $g(n)$ τέτοιες ώστε να ισχύει, δηλαδή να θεωρείται ισότητα [1]. Αναλυτικότερα για οποιαδήποτε συνάρτηση $f(n)$ στην οποία ανήκει στο σύνολο $O(n)$ υπάρχει κάποια άλλη συνάρτηση $g(n)$ η οποία ανήκει στο σύνολο $O(n^3)$ τέτοια ώστε

$$3 \cdot n^2 + f(n) = g(n) \tag{3.4}$$

Κεφάλαιο 4

Αλγόριθμοι Ταξινόμησης

Στο κεφάλαιο αυτό θα παρουσιασθούν αλγόριθμοι ταξινόμησης καθώς και οι μέθοδοι που χρησιμοποιεί ο καθένας από αυτούς κατά την διαδικασία της ταξινόμησης. Επίσης θα παρουσιασθούν οι χρόνοι εκτέλεσης αυτών των αλγορίθμων.

4.1 Ταξινόμηση αλγορίθμων

Με τον όρο ταξινόμηση ενός συνόλου στοιχείων καλείται η κατάταξή τους σε μία συγκεκριμένη σειρά, είτε τα στοιχεία αυτά είναι αριθμοί, είτε γράμματα, είτε αντικείμενα. Η ταξινόμηση σαν μορφή οργάνωσης ενός συνόλου πραγματοποιείται ούτως ώστε να γίνει πιο εύκολη η διαδικασία της αναζήτησης. Στην επιστήμη της Πληροφορικής ένας αλγόριθμος ταξινόμησης αποτελεί μία μορφή οργάνωσης και κατάταξης των δεδομένων που δίνονται με στόχο την ορθή εκτέλεση ενός προγράμματος. Γενικότερα οι αλγόριθμοι ταξινόμησης είναι πολύτιμοι για την κατάταξη στοιχείων ή δεδομένων ενός προβλήματος και την ευκολότερη με αυτόν τον τρόπο επίλυσή του.

4.2 Η μέθοδος Διαίρει και Βασίλευε

Η συγκεκριμένη μέθοδος αποτελεί έναν από τους στόχους για την επίλυση προβλημάτων και χρησιμοποιείται κατά την διαδικασία της διαμέρισης ενός αλγορίθμου. Η ονομασία της προέκυψε από αρχαίους Ρωμαίους πολιτικούς οι οποίοι δεν σκέπτονταν ούτε σκόπευαν την δημιουργία αλγορίθμων ως τότε. Η μεθοδος στηρίζεται στο εξής: Διαιρείς τους εχθρούς σου, δημιουργείς συνθήκες ώστε να φιλονικούν μεταξύ τους και στην συνέχεια τους κατακτάς όλους έναν προς έναν. Η μέθοδος υλοποιεί την λύση του προβλήματος σε μικρότερα υποπροβλήματα τα οποία στην συνέχεια αναλύονται αναδρομικά. [6]
Συγκεκριμένα:

- Το πρόβλημα διαιρείται σε διάφορα υποπροβλήματα.
- Το κάθε πρόβλημα επιλύεται ξεχωριστά με τον ίδιο τρόπο και αναδρομικά.
- Συνδυάζονται οι λύσεις των προβλημάτων ώστε να γίνει η σύνθεση μίας αρχικής λύσης.

4.2.1 Ταξινόμηση με συγχώνευση Merge Sort

Η συγχωνευτική ταξινόμηση αποτελεί μία εφαρμογή της μεθόδου Διαίρει και βασίλευε. Αναλυτικότερα, σε μία ακολουθία και συγκεκριμένα σε έναν πίνακα n στοιχείων πραγματοποιεί διαίρεση σε δύο υποσύνολα του ίδιου μήκους $\frac{n}{2}$ το καθένα, στην συνέχεια ταξινομεί τα υποσύνολα αυτά καλώντας αναδρομικά τον εαυτό της και στο τέλος τα συγχωνεύει σε μία ακολουθία η οποία πλέον θα είναι ταξινομημένη. Πιο συγκεκριμένα κατά την διαδικασία της συγχωνευτικής ταξινόμησης:

- Γίνεται η διαίρεση του συνόλου των n στοιχείων που χωρίζονται σε 2 υποσύνολα, με $\frac{n}{2}$ στοιχεία. Δηλαδή, σε έναν πίνακα $n = 8$ στοιχείων που εφαρμόζεται η μέθοδος ο πίνακας θα χωρισθεί σε δύο υποπίνακες με μέγεθος $n = 4$ στοιχείων ο καθένας.
- Ταξινομείται το κάθε υποσύνολο, δηλαδή ο κάθε υποπίνακας ταξινομείται ξεχωριστά και αναδρομικά με την χρήση της συγχωνευτικής ταξινόμησης.
- Γίνεται συγχώνευση των δύο ταξινομημένων υποσυνόλων ώστε να σχηματισθεί το τελικό ταξινομημένο υποσύνολο [6].

Παράδειγμα 4.1. *Θεωρείστε του πίνακα του σχήματος 4.1. Ξεκινώντας την διαδικασία της ταξινόμησης και καλώντας τον αλγόριθμο αναδρομικά πραγματοποιούνται συνεχείς διαιρέσεις. Η διαίρεση του πίνακα σταματά όταν ο κάθε υποπίνακας που προκύπτει περιέχει μόνο ένα στοιχείο.*

1	2	3	4	5	6	7	8
3	2	4	10	7	5	8	9

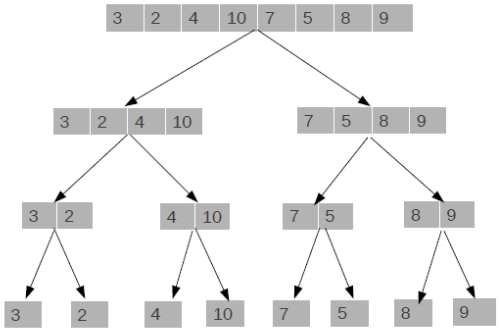
Σχήμα 4.1: *Πίνακας n=8 στοιχείων*

Ειδικότερα
εάν $p < r$ τότε γίνεται διαίρεση στους δύο πίνακες και αυτοί ταξινομούνται αναδρομικά, βλπ σχήμα 4.2.

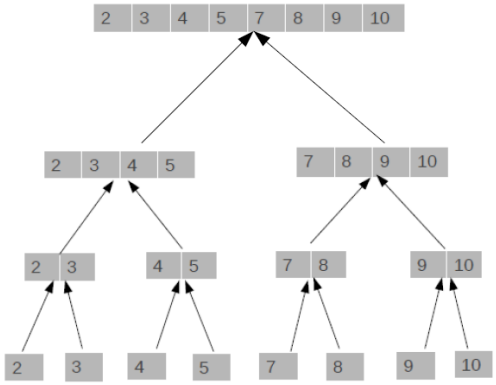
Συνεχίζοντας την διαδικασία, ο πίνακας τελικά ταξινομείται, μετά από διαδοχικές συνενώσεις, όπως φαίνεται και στο σχήμα 4.3.

Ένας ενδεικτικός αλγόριθμος ο οποίος μπορεί να χρησιμοποιηθεί για την συγχωνευτική ταξινόμηση περιγράφεται παρακάτω.

Χρόνος εκτέλεσης της συγχωνευτικής ταξινόμησης: Εάν $T(n)$ είναι ο χρόνος που απαιτείται για να ταξινομηθούν n στοιχεία μίας ακολουθίας τότε ο χρόνος εκτέλεσης αντίστοιχα για να ταξινομηθούν δύο υποακολουθίες είναι $2T(\frac{n}{2})$, όπου $\frac{n}{2}$ είναι το μέγεθος της κάθε υποακολουθίας.
Όταν $n > 1$, ο χρόνος που χρειάζεται η συνάρτηση Merge Sort για να ταξινομηθούν n στοιχεία ισούται με $\Theta(n)$.



Σχήμα 4.2: *1ο Δένδρο Αναδρομής για Συγχωνευτική Ταξινόμηση*



Σχήμα 4.3: *2ο Δένδρο Αναδρομής για Συγχωνευτική Ταξινόμηση*

Αναλυτικότερα, ο χρόνος εκτέλεσης της συγχωνευτικής ταξινόμησης εκφράζεται από τον τύπο $T(n) = 2T(\frac{n}{2}) + \Theta(n))$. Ωστόσο εάν $n = 1$ ο χρόνος εκτέλεσης που προκύπτει ισούται με $T(1) = \Theta(1)$ που είναι σταθερός, ο χρόνος εκτέλεσης της συγχωνευτικής ταξινόμησης θα ισούται με

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2T(\frac{n}{2}) + \Theta(n), & n > 1 \end{cases} \tag{4.1}$$

Από το Master Theorem το οποίο αναφέρεται στο παράρτημα, προκύπτει ο χρόνος εκτέλεσης που προκύπτει ισούται με $T(n) = O(nlogn)$ μόνο εφόσον υπάρχουν ίσες διαμερίσεις [29].

4.2.2 Η Ταχυταξινόμηση Quick Sort

Ο Αλγόριθμος της ταχυταξινόμησης αναπτύχθηκε το 1960 από τον φοιτητή Tomy Hoare, στην προσπάθειά του να ταξινομήσει ένα λεξικό μεταφρασμένων λέξεων το οποίο ήταν τοποθετημένο σε αλφαβητική σειρά, μέσα σε μία μαγνητική ταινία. Η ανάγκη της εργασίας πάνω σε ένα project από τον Hoare με την χρήση της μαγνητικής ταινίας τον οδήγησε στην δημιουργία του αλγορίθμου της Γρήγορης Ταξινόμησης, ο οποίος στην συνέχεια, υλοποιήθηκε και στην γλώσσα προγραμματισμού C, ως αλγόριθμος της γρήγορης ταξινόμησης [28].

Αλγοριθμος 4.1: *Συγχωνευτική Ταξινόμηση (A, p, r) [1]*

```
Συγχωνευτική Ταξινόμηση (A, p, r)
αν   p < r
τότε q ← ⌊(p + r)/2⌋
     Συγχωνευτική Ταξινόμηση (A, p, q)
     Συγχωνευτική Ταξινόμηση (A, q+1, r)
     Συγχώνευση (A, p, q, r)
```

Αλγοριθμος 4.2: *Ταχυταξινόμηση (A,p,r)[1]*

```
Ταχυταξινόμηση (A, p, r)
Αν   p < r
Τότε   q <–Διαμέριση (A, p, r)
       Ταχυταξινόμηση (A, p, q−1)
       Ταχυταξινόμηση (A, q+1, r)
```

Η Ταχυταξινόμηση Quick Sort στηρίζεται εξίσου στην τεχνική Διαίρει και βασίλευε. Πρόκειται για τον ταχύτερο αλγόριθμο ταξινόμησης, εξού και η ονομασία του. Αξίζει να σημειωθεί πως η μέθοδος αυτή είναι πιο αποδοτική όταν το σύνολο των στοιχείων που εξετάζονται δεν είναι ταξινομημένα, κάτι το οποίο δεν ισχύει όταν τα στοιχεία είναι σχεδόν ταξινομημένα. Σε αυτήν την μέθοδο, ο διαχωρισμός γίνεται αναδρομικά και πραγματοποιείται με την διαδικασία της διαμέρισης Partition καθώς και με την βοήθεια του στοιχείου $x = A[r]$ το οποίο χρησιμοποιείται ως οδηγός γύρω από τον οποίο θα διαμερισθεί η υποσυστοιχια $A[p, ..r]$.

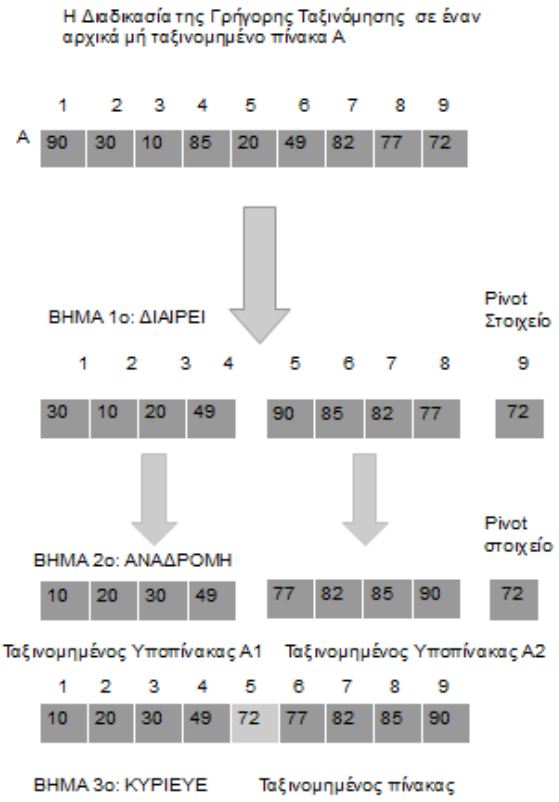
Αφού ο πίνακας χωρισθεί σε δύο τμήματα εκτός του στοιχείου διαχωρισμού, το πρώτο τμήμα που βρίσκεται αριστερά θα περιέχει τα στοιχεία εκείνα που είναι μικρότερα ή ίσα του στοιχείου αυτού. Αντίθετα, το δεύτερο τμήμα που βρίσκεται δεξιά θα περιέχει τα στοιχεία που είναι μεγαλύτερα ή ίσα του στοιχείου. Τελικά τα στοιχεία ταξινομούνται και συνενώνονται με τέτοιο τρόπο ώστε ο πίνακας να περιέχει αριστερά εκείνα τα οποία είναι μικρότερα ή ίσα του διαχωριστικού στοιχείου, ενδιάμεσα το διαχωριστικό στοιχείο και δεξιά εκείνα τα στοιχεία που είναι μεγαλύτερα είτε ίσα αυτού. Η διαδικασία της γρήγορης ταξινόμησης περιγράφεται με το σχήμα 4.4.

Η Ταχυταξινόμηση υλοποιείται μέσω της ακόλουθης διαδικασίας

Παράδειγμα 4.2. *Θεωρείστε πως θέλετε να ταξινομήσετε τον πίνακα του παρακάτω σχήματος 4.5.*

Αρχικά ξεκινώντας από τον αλγόριθμο Ταχυταξινόμηση (A, p, r) και εφόσον η συνθήκη ισχύει $(p < r)$ (εδώ $1 < 8$), ξεκινά η διαδικασία της διαμέρισης στον πίνακα (A, 1, 8). Τα βήματα σύμφωνα με τον αλγόριθμο που δίνεται παρουσιάζουται στο παρακάτω σχήμα 4.6.

Στην συνέχεια η διαμέριση τελειώνει και η διαδικασία της ταξινόμησης προχωρά και στην κάτω γραμμή του αλγόριθμου (στον αριστερό υποπίνακα που πρόκειται να ταξινομηθεί), δηλαδή στον πίνακα $(A, p, q-1)$ με $p = 1$ και $q-1 = 3$. Εν ολίγοις η ταξινόμηση ξεκινά στον αριστερό



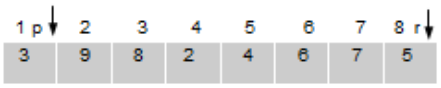
Σχήμα 4.4: *Η Γρήγορη Ταξινόμηση*

υποπίνακα (A, 1, 3). Σχηματικά φαίνεται παρακάτω 4.9.

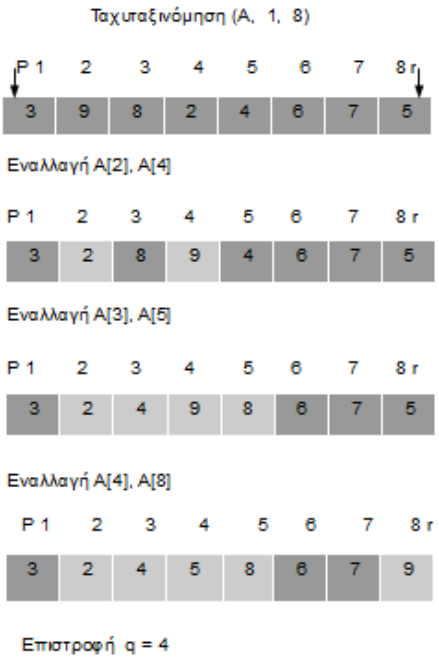
Όπως φαίνεται και στο σχήμα 4.9 η διαδικασία της διαμέρισης τερματίζεται χωρίς να γίνεται εναλλαγή θέσεων και επιστρέφει τερματίζοντας του αύξοντα αριθμό q = 3. Στην συνέχεια καλείται ξανά η Ταχυταξινόμηση (A, 1, 2) στον αριστερό υποπίνακα (όπως είναι γραμμοσκιασμένο με πιο σκούρο χρώμα). Στην συνέχεια η διαδικασία της ταχυταξινόμησης καλείται στον υποπίνακα (A, 5, 7) όπως φαίνεται και στο παρακάτω σχήμα 4.8.

Αλγοριθμος 4.3: *Διαμέριση (A, p, r)[1]*

```
Διαμέριση (A, p, r)
X <- A[r]
i <- p - 1
για  j <- p έως  r-1
            αν  A[j]<= X
                  τότε  i <- i + 1
                        εναλλαγή  A[i]<->A[j]
εναλλαγή  A[i+1]<->A[r]
επιστροφή  i+1
```



Σχήμα 4.5: *Αρχικός πίνακας 8 στοιχείων*



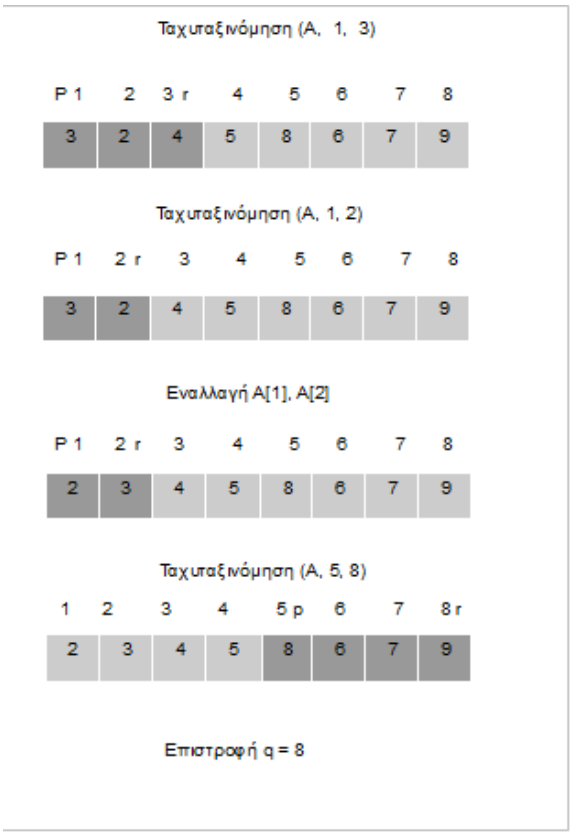
Σχήμα 4.6: *Ταχυταξινόμηση(A, 1, 8)*

4.2.3 Η χειρότερη περίπτωση για την Ταχυταξινόμηση

Ο βασικός παράγοντας στον χρόνο εκτέλεσης της ταχυταξινόμησης είναι ο χρόνος που δαπανάται στην διαδικασία διαμέριση. Η χειρότερη περίπτωση της ταχυταξινόμησης προκύπτει όταν το υποπρόγραμμα της διαμέρισης παράγει δύο προβλήματα με $n = 1$ στοιχεία και με 0 στοιχεία.

Έστω T(n) ο χρόνος εκτέλεσης χειρότερης περίπτωσης της διαδικασίας της γρήγορης ταξινόμησης. Εάν η συνάρτηση διαμέρισης Partition χωρίζει μία ακολουθία (λ.χ. έναν πίνακα) σε δύο μεγέθη k και $n - k$ τότε ο χρόνος θα δίνεται από την σχέση

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n - q - 1)) + \Theta(n)$$



Σχήμα 4.7: *Ταχυταξινόμηση(A, 1, 3)*

όπου q μία παράμετρος η οποία κυμαίνεται από το 0 έως το n-1. Υποθέτωντας ότι ισχύει

$$T(n) \leq cn^2$$

για κάποια σταθερά c, τότε προκύπτει

$$T(n) \leq \max_{0 \leq q \leq n-1} (c * q^2 + c(n-q-1)^2) + \Theta(n) = c \max_{0 \leq q \leq n-1} (q^2 + (n-q-1)^2) + \Theta(n)$$

Η έκφραση $q^2 + (n-q-1)^2$ παίρνει την μέγιστη τιμή της επί του διαστήματος

$$0 \leq q \leq n-1$$

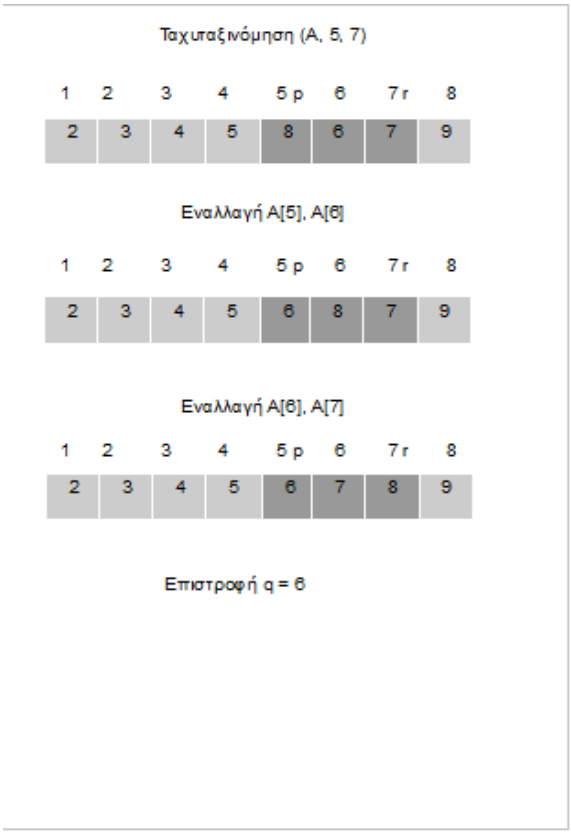
της παραμέτρου στα δύο ακραία σημεία της. Εφόσον η δεύτερη παράγωγος της σχέσης $q^2 + (n-q-1)^2$ είναι **θετική** προκύπτει το φράγμα

$$\max_{0 \leq q \leq n-1} ((q^2 + (n-q-1)^2) \leq (n-1)^2 = n^2 - 2n + 1$$

,

Έτσι για το φράγμα $T(n)$ προκύπτει

$$T(n) \leq c \leq n^2 - c \leq (2n-1) + \Theta(n) \leq c \cdot n^2$$



Σχήμα 4.8: *Ταχυταξινόμηση(A, 5, 7)*

Με αυτόν τον τρόπο επιλέγοντας μία αρκετά μεγάλη σταθερά c ο όρος $c \cdot (2n-1)$ επικρατεί του όρου $\Theta(n)$, επομένως $T(n) = O(n^2)$, συνεπώς ο χρόνος εκτέλεσης της χειρότερης περίπτωσης της ταχυταξινόμησης είναι $O(n^2)$. Ο χρόνος χειρότερης περίπτωσης της ταχυταξινόμησης αφορά την περίπτωση όπου η συστοιχία εισόδου είναι πλήρως ταξινομημένη [1].

4.2.4 Η Διαμέριση καλύτερης περίπτωσης για την Ταχυταξινόμηση

Η καλύτερη περίπτωση της διαδικασίας της διαμέρισης προκύπτει όταν παράγει δύο προβλήματα με μέγεθος $\frac{n}{2}$ το καθένα, δεδομένου ότι το ένα έχει μέγεθος $\left\lfloor \frac{n}{2} \right\rfloor$ και το άλλο $\left\lceil \frac{n}{2} \right\rceil - 1$. Στην περίπτωση αυτή η ταχυταξινόμηση εκτελείται πολύ πιο γρήγορα. Η αναδρομική σχέση που δίνεται για τον χρόνο εκτέλεσης είναι

$$T(n) \leq 2T(\frac{n}{2}) + \Theta(n)$$

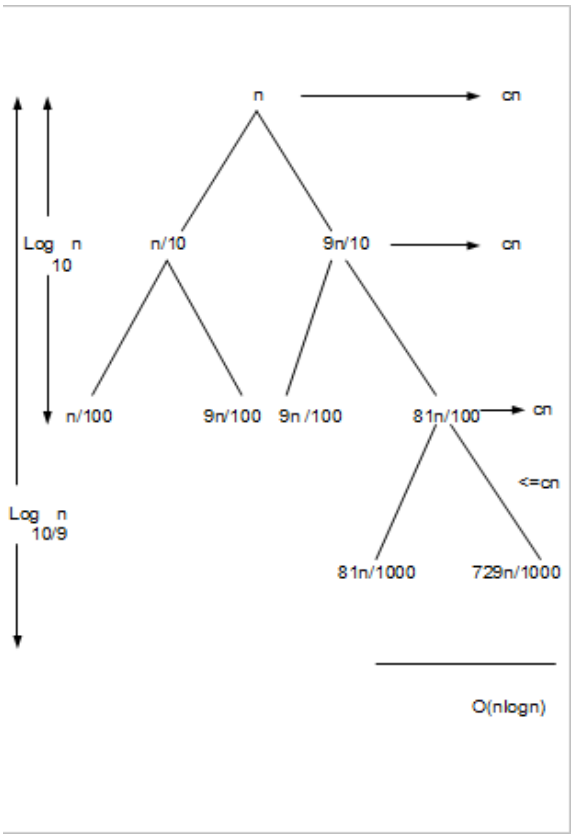
όπου σύμφωνα με το κεντρικό θεώρημα η λύση της είναι $T(n) = O(nlogn)$[1] .

4.2.5 Η ισομερής διαμέριση για την Ταχυταξινόμηση

Ο χρόνος εκτέλεσης τσ μέσης περίπτωσης της ταχυταξινόμησης αγγίζει κατά πολύ περισσότερο την καλύτερη παρά την χειρότερη περίπτωση. Υποθέτωντας πως ο αλγόριθμος διαμέρισης δίνει πάντοτε έναν διαχωρισμό 9 προς 1 ο οποίος εξ' αρχής φαίνεται αρκετά ανισομερής, ο χρόνος εκτέλεσης της ταχυταξινόμησης θα δίνεται από την αναδρομική σχέση

$$T(n) \leq T(\frac{9n}{10}) + T(\frac{n}{10}) + c \cdot n$$

Η σταθερά c εμπεριέχεται στον όρο $\Theta(n)$. Παρατηρώντας το δένδρο, το κάθε επίπεδό του έχει κόστος $c \cdot n$ μέχρις ότου να φθάσει σε μία συνοριακή συνθήκη σε βάθος $logn = \Theta(logn)$, όπου σε αυτό το σημείο τα επίπεδα έχουν κόστος μικρότερο ή ίσο του γινομένου $c \cdot n$. Συνεπώς σε έναν τέτοιο διαχωρισμό η γρήγορη ταξινόμηση εκτελείται σε χρόνο $O(nlogn)$. Αυτό συνεπάγεται πως οποιοσδήποτε διαχωρισμός με σταθερή αναλογία δίνει ένα δένδρο αναδρομής με βάθος $\Theta(logn)$, το κόστος σε κάθε επίπεδο είναι $O(n)$. Συμπερασματικά προκύπτει πως οποισδήποτε διαχωρισμός χαρακτηρίζεται από σταθερή αναλογία, χρόνος εκτέλεσης είναι $O(nlogn)$ [1].



Σχήμα 4.9: *Δένδρο Αναδρομής της Διαδικασίας της γρήγορης ταξινόμησης*

Αλγοριθμος 4.4: *Ταξινόμηση σωρού (A) [1]*

```
Ταξινόμηση σωρού  (A)

Κατασκευή σωρού μεγίστου (A)
για  i<–  μήκος[A] αντίστροφα- έως 2
          εναλλαγή A[1]  <–>  A[ i ]
          πλήθος-σωρού[A]  <–  πλήθος-σωρού [A] - 1
          Αποκατάσταση σωρού μεγίστου (A, 1)
```

4.3 Η ταξινόμηση σωρού Heap Sort

Ο Αλγόριθμος της heap sort είναι ένας αλγόριθμος ταξινόμησης και εφευρέθηκε από τον JWJ Williams ως μία χρήσιμη δομή δεδομένων. Η μέθοδος είναι λιγότερο αποτελεσματική από τον αλγόριθμο γρήγορης ταξινόμησης (quick sort) καθώς απαιτεί περισσότερο χρόνο κατά την εκτέλεσή του [6]. Η διαδικασία ταξινόμησης σωρού έχει ως εξής:

- Βήμα 1ο:Αρχικά κατασκευάζεται μία συστοιχία εισόδου, συνήθως ένας πίνακας στοιχείων. Στην συνέχεια ο πίνακας αυτός αναδιατάσσεται σε σωρό.

- Βήμα 2ο: Μετά το 1ο βήμα ο πίνακας είναι ένας σωρός στην 1η θέση του οποίου τοποθετείται ο κόμβος με την μεγαλύτερη τιμή. Τα υπόλοιπα υποδένδρα του σωρού, δηλαδή τα δένδρα και τα υποδένδρα των παιδιών της ρίζας θα είναι επίσης σωροί. Στο σημείο αυτό πραγματοποιείται ανταλλαγή του πρώτου στοιχείου του σωρού με το τελευταίο. Με αυτόν τον τρόπο το μεγαλύτερο στοιχείο του σωρού θα βρίσκεται απομονωμένο στην τελευταία θέση.

- Βήμα 3ο:πραγματοποιείται η διαδικασία της **Αποκατάστασης σωρού μεγίστου** και

- Βήμα 4ο: Επαναλαμβάνονται τα δύο προηγούμενα βήματα (Η ανταλλαγή των στοιχείων και η αποκατάσταση σωρού μεγίστου) μέχρις ότου να μείνουν μόνο δύο στοιχεία στον σωρό τα οποία εναλλάσσονται στο τέλος.

Η Ταξινόμηση σωρού υλοποιείται μέσω της διαδικασίας 4.4.

Χρόνος εκτέλεσης της ταξινόμησης σωρού: Η κλήση της διαδικασίας **κατασκευή σωρού μεγίστου** απαιτεί χρόνο $O(n)$ καθώς και κάθε μία από τις $n-1$ κλήσεις της διαδικασίας **Αποκατάσταση σωρού μεγίστου** απαιτεί χρόνο $O(logn)$. Συνεπώς η διαδικασία της **Ταξινόμησης σωρού** απαιτεί χρόνο $O(nlogn)$ [29].

Κεφάλαιο 5

Ο Πιθανοτικός αλγόριθμος της ταχυταξινόμησης

Σε προηγούμενο κεφάλαιο αναφέρθηκε ο αλγόριθμος της ταχυταξινόμησης ο οποίος στηρίζεται στην μέθοδο Διαίρει και Βασίλευε (Devide and conquer). Ο αλγόριθμος εκτελείται κάθε φορά αναδρομικά με διαδοχικές διαιρέσεις με το κάλεσμα της συνάρτησης Partition και την επιλογή του στοιχείου $A[r]$ όπου A είναι ένα σύνολο ή μία υποσυστοιχία και $A[r]$ το στοιχείο-οδηγός γύρω από το οποίο θα πραγματοποιηθεί η διαμέριση του συνόλου ή της υποσυστοιχίας . Σε αυτό το κεφάλαιο θα αναλυθεί εκτενώς η τυχαιοκρατική εκδοχή της ταχυταξινόμησης με την μέθοδο Διαίρει και Βασίλευε αλλά με την επιλογή μίας τυχαίας θέσης $A[i]$. Έπειτα θα πραγματοποιηθεί υπολογισμός του χρόνου εκτέλεσης του ιδίου αλγορίθμου και τέλος η προγραμματιστική υλοποίησή του μέσω της γλώσσας προγραμματισμού c. Παράλληλα στο τέλος του κεφαλαίου παρουσιάζεται η υλοποίηση του αλγόριθμου της πιθανοτικής γρήγορης ταξινόμησης για 100 στοιχεία.

5.1 Πιθανοτική εκδοχή του αλγορίθμου γρήγορης ταξινόμησης

Η εν λόγω ταξινόμηση εκτελείται με τον ίδιο ακριβώς τρόπο με τον οποίο εκτελείται και η ταχυταξινόμηση. Η μόνη διαφορά ωστόσο, έγκειται στο γεγονός πως στην συγκεκριμένη μέθοδο, γίνεται η επιλογή μίας τυχαίας θέσης $A[i]$ της συστοιχίας που δίνεται (εδώ ενός πίνακα) και όχι της θέσης $A[r]$ όπως συμβαίνει στην ταχυταξινόμηση. Η τυχαιοκρατική εκδοχή της ταχυταξινόμησης χρησιμοποιείται κυρίως για αρκετά μεγάλες εισόδους. Παράλληλα η διαδικασία της διαίρεσης της συστοιχίας που δίνεται σε τμήματα πραγματοποιείται με την διαδικασία της τυχαιοκρατικής διαμέρισης, στην συνέχεια της ανταλλαγής του τυχαίου στοιχείου της θέσης $A[i]$ που επιλέχθηκε με το στοιχείο της θέσης $A[r]$ και στην συνέχεια με την επιστροφή της διαδικασίας της διαμέρισης. Η Τυχαιοκρατική διαμέριση υλοποιείται μέσω της ακόλουθης διαδικασίας 5.2.

Η τυχαιοκρατική ταχυταξινόμηση καλεί αντί της διαμέρισης την τυχαιοκρατική διαμέριση.

Παράδειγμα 5.3. *Θεωρείστε πως θέλετε να ταξινομήσετε του πίνακα του σχήματος 5.1*

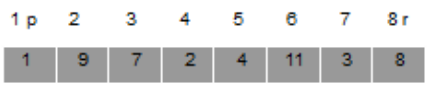
Ακολουθώντας πάντα τα βήματα του αλγόριθμου αρχικά εφόσον η συνθήκη $p < r$ ισχύει (αφού $1 < 8$) καλείται η διαδικασία της Τυχαιοκρατικής Ταχυταξινόμησης για τον πίνακα A, p, r. Η Διαδικασία περιγράφεται στο σχήμα 5.2.

Αλγοριθμος 5.1: *Τυχαιοκρατική Διαμέριση (A, p, r) [1]*

```
Τυχαιοκρατική Διαμέριση   (A, p, r)
i <– Τυχαία (p, r)
εναλλαγή A[r] <–> A[i]
επιστροφή Διαμέριση   (A, p, r)
```

Αλγοριθμος 5.2: *Τυχαιοκρατική Ταχυταξινόμηση (A, p, r) [1]*

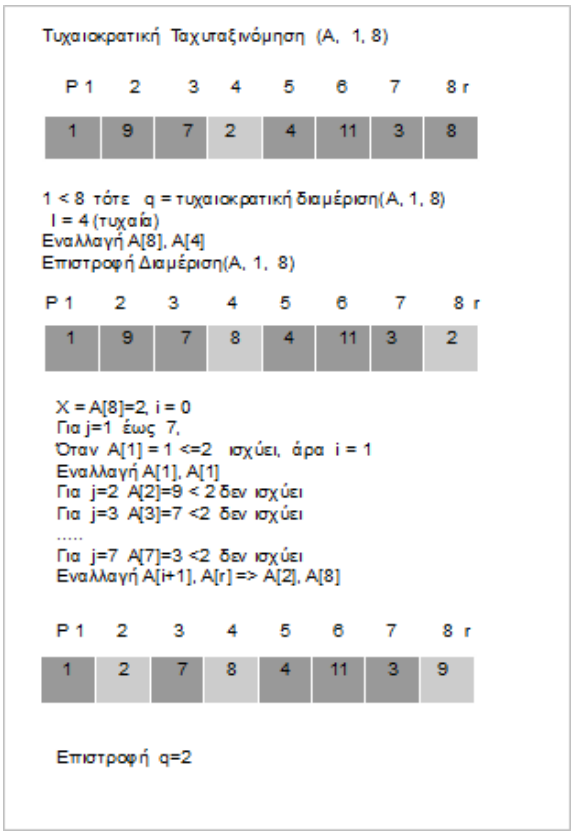
```
Τυχαιοκρατική Ταχυταξινόμηση (A, p, r)
αν p < r
        τότε  q <– Τυχαιοκρατική Διαμέριση   (A, p, r)
              Τυχαιοκρατική Ταχυταξινόμηση (A, p, q–1)
              Τυχαιοκρατική Ταχυταξινόμηση (A, q+1, r)
```



Σχήμα 5.1: *Πίνακας προς ταξινόμηση*

Σε αυτό το σημείο η διαδικασία της διαμέρισης σταματά και επιστρέφεται ο αριθμός $q = 2$. Στην συνέχεια καλείται η Τυχαιοκρατική ταχυταξινόμηση για τον πίνακα (A, q+1, r), δηλαδή για τον πίνακα (A, 3, 8). Στα παρακάτω σχήματα 5.3 και 5.5 διακρίνονται τα βήματα που εκτελεί ο αλγόριθμος κάθε φορά.

Η διαδικασία της τυχαιοκρατικής ταχυταξινόμησης σταματά όταν επιστραφεί ο αριθμός $q = 6$. Στην συνέχεια καλείται η τυχαιοκρατική ταχυταξινόμηση στον πίνακα (A, p, q-1), δηλαδή στον πίνακα (A, 3, 5). Εν τέλει η διαδικασία τερματίζεται όταν οι τιμές των δεικτών p και r γίνουν ίσοι ή γενικότερα, όταν σταματήσει να ισχύει η συνθήκη $p < r$. Στο σημείο αυτό πίνακας ταξινομείται, και επιστρέφεται ο αριθμός $q = 4$, όπως φαίνεται στο παρακάτω σχήμα 5.5. Στην περίπτωση αυτή όπου ο πίνακας με τα στοιχεία που δίνονται δεν είναι ταξινομημένα (αρχικά) εκτελείται ο ελάχιστος αριθμός των βημάτων οπότε και πρόκειται για την καλύτερη περίπτωση της τυχαιοκρατικής ταχυταξινόμησης.



Σχήμα 5.2: *Τυχαιοκρατική Ταχυταξινόμηση (A, 1, 8)*

5.2 Χειρότερη περίπτωση για την τυχαιοκρατική ταχυταξινόμηση

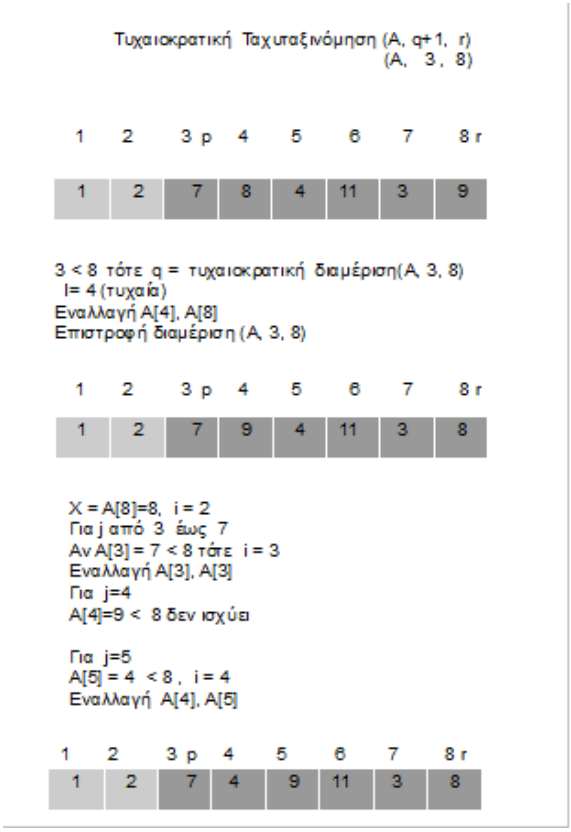
Ο χρόνος εκτέλεσης ή απλά η πολυπλοκότητα του αλγορίθμου της ταχυταξινόμησης όπως και της τυχαιοκρατικής ταχυταξινόμησης εξαρτάται από τον χρόνο που εκτελείται στην διαδικασία διαμέριση (Partition). Κάθε φορά που καλείται η διαμέριση επιλέγεται ένα τυχαίο στοιχείο random, το οποίο δεν χρησιμοποιείται στις επόμενες κλήσεις της συνάρτησης. Έστω ότι $T(n)$ ο χρόνος εκτέλεσης της χειρότερης περίπτωσης της διαδικασίας της τυχαιοκρατικής ταχυταξινόμησης. Η αναδρομική σχέση που περιγράφει τον συγκεκριμένο χρόνο εκτέλεσης θα είναι

$$T(n) = \max_{0 \le q \le n-1} (T(n-1) + T(n-q-1) + \Theta(n) \tag{5.1}$$

,

όπου q είναι μία παράμετρος που κυμαίνεται από το 0 έως το $n-1$, διότι η διαδικασία της διαμέρισης (Partition) παράγει δύο υποπροβλήματα μεγέθους $n-1$. Υποθέτωντας πως ισχύει $T(n) \le c \cdot n^2$ για κάποια σταθερά c, και αντικαθιστώντας την σχέση αυτή στην 5.1 προκύπτει πως

$$T(n) \le \max_{0 \le q \le n-1} (c \cdot q^2 + c \cdot (n-q-1)^2) + \Theta(n) = c \cdot \max 0 \le q \le n-1 (q^2 + (n-q-1)^2 + \Theta(n)$$



Σχήμα 5.3: *Τυχαιοκρατική Ταχυταξινόμηση (A, 3, 8)*

Η έκφραση $q^2+(n-q-1)^2)$ παίρνει την μέγιστη τιμή της επί του διαστήματος $\leq q \leq n-1$, δεδομένου ότι η δεύτερη παράγωγός της ως προς q είναι θετική. Με βάση τα παραπάνω η παρατήρηση αυτή δίνει το φράγμα

$$\max 0 \leq q \leq n-1((q^2+(n-q-1)^2) \leq (n-1)^2 = n^2-2n+1$$

,

Ομοίως όσον αφορά το φράγμα του $T(n)$ ισχύει

$$T(n) \leq c \leq n^2 - c \leq (2n-1) + \Theta(n) <= c \cdot n^2$$

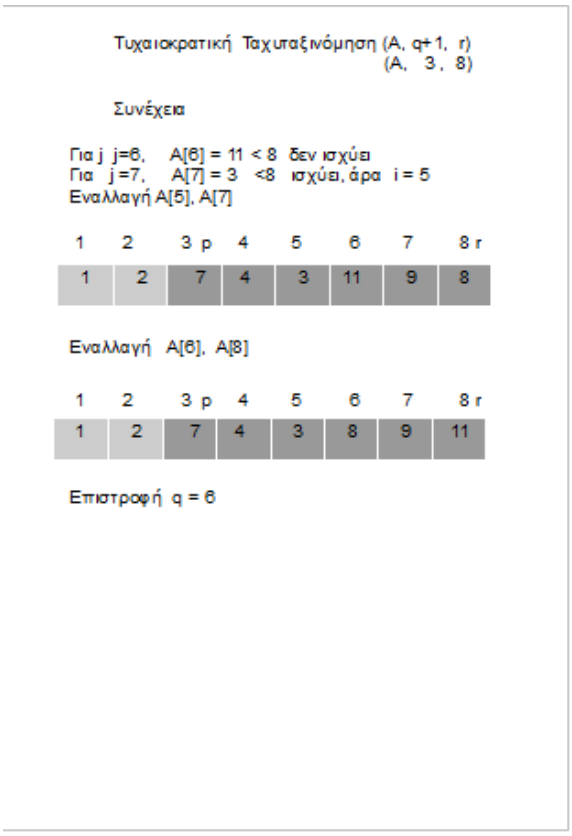
, όπου c μία σταθερά.

Ο χρόνος εκτέλεσης χειρότερης περίπτωσης της πιθανοτικής ταχυταξινόμησης που προκύπτει είναι $T(n) = O(n^2)$ [1].

Γενικότερα η χειρότερη περίπτωση της τυχαιοκρατικής ταχυταξινόμησης αφορά την περίπτωση που η συστοιχία εισόδου είναι πλήρως ταξινομημένη.

Παράδειγμα 5.4. *Θεωρείστε τον πίνακα του σχήματος 5.6*

Ακολουθώντας πάντα τα βήματα της τυχαιοκρατικής ταχυταξινόμησης η διαδικασία περιγράφεται στο σχήμα 5.7. Στο τέλος της διαδικασίας της διαμέρισης, επιστρέφεται ο αύξοντας αριθμός $q = 4$ και τα στοιχεία των θέσεων A[4] και A[8] εναλλάσσονται, όπως φαίνονται στο

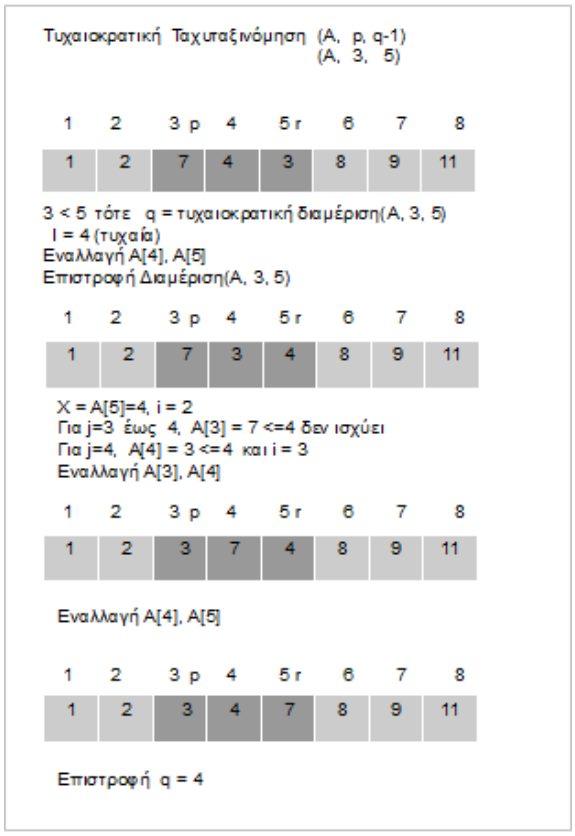


Σχήμα 5.4: *Τυχαιοκρατική Ταχυταξινόμηση (A, 3, 8)*

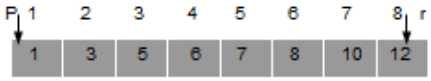
σχήμα 5.8. Παρατηρείται πως ο πίνακας με το τέλος της διαδικασίας της διαμέρισης προκύπτει όπως ακριβώς ήταν και στην αρχή. Εδώ θα πρέπει να τονισθεί πως η διαδικασία δεν τερματίζεται σε αυτό το σημείο, αλλά στο σημείο που οι δείκτες p και r θα γίνουν ίσοι. Δηλαδή η διαδικασία τυχαιοκρατικής ταχυταξινόμησης θα επαναληφθεί για τον πίνακα $(A, p, q-1)$, δηλαδή για τον πίνακα (A, 1, 3), στην συνέχεια για τον πίνακα $(A, q+1, r)$ δηλαδή για τον (A, 5, 8) και στο τέλος για τον πίνακα $(A, q+1, r)$ δηλαδή για τον (A, 7, 8). Ο πίνακας μετά το τέλος των διαδικασιών αυτών προκύπτει όπως ακριβώς ήταν και στην αρχή. Με τον τρόπο αυτόν περιγράφεται ο μέγιστος αριθμός των βημάτων της διαδικασίας της τυχαιοκρατικής ταχυταξινόμησης, που αντιστοιχεί στην χειρότερη περίπτωση. Η Τυχαιοκρατική ταχυταξινόμηση σε αυτήν την περίπτωση εκτελείται n φορές, συνεπώς ο χρόνος εκτέλεσής της θα ισούται με $O(n^2)$. Για την τυχαιοκρατικη ταχυταξινόμηση η επιλογή τυχαίου οδηγού κάνει την χειρότερη περίπτωση λιγότερο πιθανή.

5.3 Αναμενόμενος χρόνος εκτέλεσης για την τυχαιοκρατική ταχυταξινόμηση

Εάν σε κάθε επίπεδο της αναδρομής ο διαχωρισμός που πραγματοποιεί η τυχαιοκρατικη διαμέριση τοποθετεί στην μία πλευρά της ένα σταθερό ποσοστό στοιχείων, το δένδρο αναδρομής θα έχει βάθος $\Theta(logn)$ ενώ παράλληλα σε κάθε επίπεδο του δένδρου εκτελείται έργο $O(n)$. Ακόμα και εάν προσθεθούν επιπλέον επίπεδα με τον ανισομερή διαχωρισμό μεταξύ των επιπέδων αυτών, ο συνολικός χρόνος θα παραμείνει $O(nlogn)$.



Σχήμα 5.5: *Τυχαιοκρατική Ταχυταξινόμηση (A, 3, 5)*



Σχήμα 5.6: *Αρχικός πίνακας*

Για την ανάλυση του χρόνου εκτέλεσης της τυχαιοκρατικής ταχυταξινόμησης είναι απαραίτητη η κατανόηση της λειτουργίας της διαδικασίας της διαμέρισης [1].

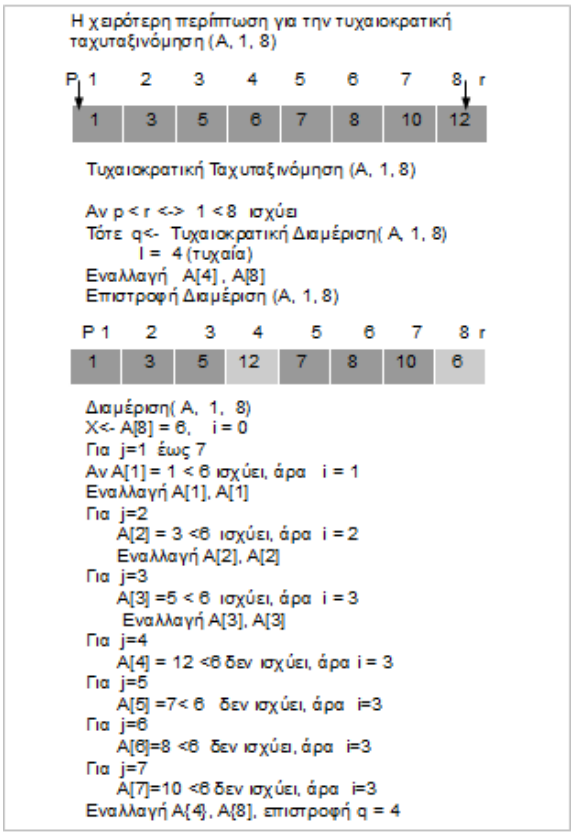
Με την βοήθεια των δείκτριων τυχαίων μεταβλητών ξεκαθαρίζεται πότε ο αλγόριθμος συγκρίνει τα δύο στοιχεία της συστοιχίας και πότε όχι. Έστω οι δείκτριες τυχαίες μεταβλητές

$Xi,j = \{Z_i$ συγκρίνεται με $Z_j\}$

όπου εξετάζεται κατά πόσο η σύγκριση λαμβάνει χώρα οποιαδήποτε στιγμή κατά την εκτέλεση του αλγορίθμου και όχι κατά την διάρκεια μίας μόνο επανάληψης ή κλήσης της διαδικασίας της διαμέρισης. Εφόσον δύο στοιχεία συγκρίνονται μεταξύ τους το πολύ μία φορά, το συνολικό πλήθος των συκρίσεων θα δίνεται από το άθροισμα των επιμέρους ≪περιπτώσεων≫ των δείκτριων τυχαίων μεταβλητών που αναφερθηκαν παραπάνω. Συγκεκριμένα

$$x = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_i j$$

Χρησιμοποιώντας την αναμενόμενη τιμή των δύο μελών προκύπτει:



Σχήμα 5.7: *α. Η Χειρότερη περίπτωση για την τυχαιοκρατική ταχυταξινόμηση*

Σχήμα 5.8: *β. Η Χειρότερη περίπτωση για την τυχαιοκρατική ταχυταξινόμηση*

$$E(X) = E[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_ij] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_ij] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}$$

$$I = \{Z_i \text{ συγκρίνεται με } Z_j\} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P_r\{Z_i \text{ συγκρίνεται με } Z_j\}$$

Για να υπολογισθεί λοιπόν ο αριθμός των συγκρίσεων x αρκεί να υπολογισθεί η πιθανότητα $P_r\{Z_i$ συγκρίνεται με $Z_j\}$. Στην περίπτωση αυτή γίνεται υπόθεση ότι το κάθε στοιχείο-οδηγός επιλέγεται τυχαία και ανεξάρτητα από τα άλλα. Επειδή τα στοιχεία εισόδου είναι εξ υποθέσεως διαφορετικά, από την στιγμή που θα επιλεγεί κάποιος οδηγός x οι τιμές του οποίου να κυμαίνονται ανάμεσα στο Z_i και στο Z_j, τα Z_i και Z_j είναι αδύνατον να συγκριθούν μεταξύ τους στο μέλλον. Από την άλλη πλευρά εάν επιλεγεί το Z_i ως οδηγός πριν από όλα τα άλλα στοιχεία του συνόλου Z_ij, τότε το Z_ij θα συγκριθεί με όλα τα υπόλοιπα στοιχεία του συνόλου αυτού εκτός από τον εαυτό του. Το ίδιο ακριβώς θα συμβεί εάν επιλεγεί το στοιχείο

Z_j.

Ωστόσο τα στοιχεία αυτά συγκρίνονται μεταξύ τους όταν το 1ο στοιχείο που επιλέγεται ως διαχωριστικό (ή οδηγός) είτε είναι το ένα είτε το άλλο, δηλαδή εάν και μόνο εάν είναι είτε το Z_i είτε το Z_j. Θα εξετασθεί λοιπόν η πιθανότητα να συμβεί αυτό το ενδεχόμενο. Πριν από την στιγμή που επιλέγεται κάποιο στοιχείο σαν οδηγός από το σύνολο Z_ij, όλο το Z_ij ανήκει στο ίδιο διαμέρισμα. Συνεπώς οποιοδήποτε στοιχείο έχει την ίδια πιθανότητα να επιλεγεί σαν οδηγός. Με δεδομένο ότι το Z_ij έχει έχει $j - i + 1$ στοιχεία και οι αδηγοί επιλέγονται τυχαία και ανεξάρτητα μεταξύ τους, η πιθανότητα να επιλεγεί πρώτο ως οδηγός οποιοδήποτε συγκεκριμένο στοιχείο είναι

$Pr\{Z_i$ συγκρίνεται με $Z_j\}$ =$Pr\{Z_i$ ή z_j επιλέγεται πρώτο ως οδηγός από το z_ij= $Pr\{Z_i$ επιλέγεται πρώτο ως οδηγός από το z_ij+$Pr\{Z_j$ επιλέγεται πρώτο ως οδηγός από το $z_ij = \frac{1}{j-1+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}$

Η δεύτερη ισότητα έπεται από το γεγονός ότι τα δύο γεγονότα είναι ασυμβίβαστα. Συνεπώς η σχέση που υπολογίσθηκε προηγουμένως γίνεται

$$E[x] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n-1} O(nlogn) \simeq O(nlogn)$$

Συμπερασματικά μέσω της τυχαιοκρατικής διαμέρισης ο αναμενόμενος χρόνος εκτέλεσης της γρήγορης ταξινόμησης είναι $O(nlogn)$ κάτι που ισχύει σε περίπτωση που οι τιμές των στοιχείων της εισόδου είναι διαφορετικές μεταξύ τους [1].

5.4 Υλοποίηση του αλγορίθμου της τυχαιοκρατικής ταχυταξινόμησης

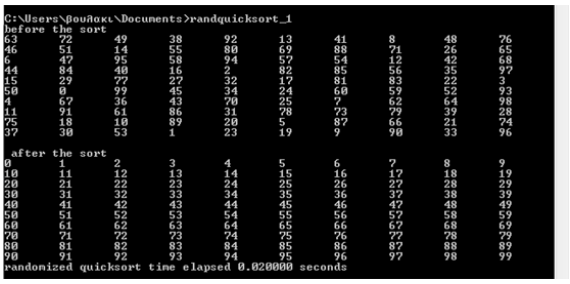
Το πρόγραμμα 5.3 της τυχαιοκρατικής ταχυταξινόμησης που απεικονίζεται (σελίδα 49) είναι υλοποιημένο στην γλώσσα προγραμματισμού c. Η τυχαιοκρατική ταχυταξινόμηση καλεί την τυχαιοκρατική διαμέριση κατά την οποία χρησιμοποιείται ένα τυχαία επιλεγμένο στοιχείο ενός πίνακα A με 10 στοιχεία. Η τυχαιοκρατική διαμέριση με την σειρά της καλεί την διαμέριση η οποία επιστρέφει έναν οδηγό, και με βάση αυτόν καλείται η τυχαιοκρατική ταχυταξινόμηση όπου εφαρμόζεται και στα δύο τμήματα που έχουν δημιουργηθεί (μικρότερα του οδηγού και μεγαλύτερα). Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου να γίνει πλήρης ταξινόμηση. Η random shuffle χρησιμοποιείται για να δοθεί σαν είσοδος μία τυχαία σειρά των αριθμών προς ταξινόμηση.

Στην εικόνα 5.1 απεικονίζεται το εκτελεσιμο αρχείο της τυχαιοκρατικης ταχυταξινόμησης.

Εικόνα 5.1: *Τυχαιοκρατική Ταχυταξινόμηση*

5.5 Υλοποίηση του αλγορίθμου της τυχαιοκρατικής ταχυταξινόμησης για την ταξινόμηση 100 αριθμών

Παρακάτω απεικονίζεται το πρόγραμμα 5.4 (σελίδα 50) της τυχαιοκρατικής ταχυταξινόμησης το οποίο ταξινομεί 100 αριθμούς. Ο αλγόριθμος εκτελείται 10 φορές και κάθε φορά βρίσκει έναν χρόνο εκτέλεσης.Ο τρόπος που μετριέται ό χρόνος εκτέλεσης είναι με τη χρήση της συνάρτησης του συστήματος clock() μία φορά για την έναρξη της μέτρησης (στην αρχή του προγράμματος) που αποθηκεύεται στην μεταβλητή start time και μετά στο τέλος του προγράμματος που ο χρόνος αποθηκεύεται στην μεταβλητή end time. Η μεταβλητή *elapsed − time* ισούται με την διαφορά *endtime − starttime* διαιρούμενη με το clocks per second που είναι ο αριθμός των tic (ο ρυθμός του ρολογιού) ο οποίος μας δίνει τον αριθμό των δευτερολέπτων. Μετά από εκτέλεση 10 φορών του αλγορίθμου βρίσκονται 10 χρόνοι εκτέλεσης, προστίθενται μεταξύ τους και το τελικό άθροισμα διαιρείται με το 10. Το τελικό αποτέλεσμα είναι και ο μέσος όρος. Στην εικόνα 5.2 απεικονίζεται το εκτελέσιμο αρχείο κατά την εκτέλεση του προγράμματος της τυχαιοκρατικής ταχυταξινόμησης που ταξινομεί 100 αριθμούς καθώς και ο χρόνος εκτέλεσής του για 1 φορά. Πριν από την ταξινόμηση υπάρχουν 100 ακέραιοι αριθμοί σε τυχαία σειρά. Μετά οι αριθμοί ταξινομούνται με την χρήση του προγράμματος randquicksort που βασίζεται στην τυχαιοκρατική ταχυταξινόμηση.



Εικόνα 5.2: *Πρώτη εκτέλεση του αλγορίθμου*

Στην εικόνα 5.2 φαίνεται πως ο πρώτος χρόνος εκτέλεσης ισούται με 0.02. Οι υπόλοιποι χρόνοι εκτέλεσης απεικονίζονται στον παρακάτω πίνακα.

Εκτέλεση	Χρόνος
1η	0.02
2η	0.02
3η	0.02
4η	0.01
5η	0.03
6η	0.02
7η	0.02
8η	0.02
9η	0.01
10η	0.02

Πίνακας 5.1: *Πίνακας χρόνων εκτέλεσης*

Αθροίζοντας τους χρόνους του παραπάνω πίνακα και στην συνέχεια διαιρώντας το αποτέλεσμα με το 10 προκύπτουν:

$$0.02 + 0.02 + 0.02 + 0.01 + 0.03 + 0.02 + 0.02 + 0.02 + 0.01 + 0.02 = 0.19$$

$$\frac{0.19}{10} = 0.019$$

που είναι και ο μέσος όρος για δέκα στοιχεία.

Αλγοριθμος 5.3: *Πρόγραμμα της Τυχαιοκρατικής ταχυταξινόμησης [30]*

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define max 10
void random_shuf(int B[]){
    srand(time(NULL));
    int i, j, tmp;
    for (i = max - 1; i > 0; i--){
        j = rand()%(i + 1);
        tmp = B[i];
        B[i] = B[j];
        B[j] = tmp;
    }}
 void swap(int *a, int *b){
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}
int Partition(int B[], int p, int r){
    int pivotInd = p + rand()%(r - p + 1);
    int pivot;
    int i = p - 1;
    int j;
    pivot = B[pivotInd];
    swap(&B[pivotInd], &B[r]);
    for (j = p; j < r; j++){
    if (B[j] < pivot){
            i++;
            swap(&B[i], &B[j]);
        }}
    swap(&B[i+1], &B[r]);
    return i + 1;}
 void q_s(int B[], int p, int q){
    int j;
    if (p < q){
        j = Partition(B, p, q);
        q_s(B, p, j-1);
        q_s(B, j+1, q);
    }}
int main(){
    int i;
    int B[max];
    for (i = 0; i < max; i++)
        B[i] = i;
    random_shuf(B);
    printf("before the sort\n");
    for (i = 0; i < max; i++)
         printf("%d \t", B[i]);
    q_s(B, 0, max-1);
    printf("\n after the sort \n");
    for (i = 0; i < max; i++)
         printf("%d \t", B[i]);
    return 0;
    }
```

Αλγοριθμος 5.4: *Πρόγραμμα για ταξινόμηση 100 στοιχείων [30] [34]*

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define max 100
//* H random_shuf exei san eisodo max arithmous kai tous vazei se tuxaia seira
void random_shuf(int B[])  {
srand(time(NULL));
int i, j, tmp;
    for (i = max - 1; i > 0; i--)
    {       j = rand()%(i + 1);
         tmp = B[i];
        B[i] = B[j];
        B[j] = tmp;     }  }
//* H swap xrhsimopoieitai gia antallagh twn thesewn stoixeiwn  sthn Partition
void swap(int *a, int *b)
{   int tmp;
   tmp = *a;
    *a = *b;
    *b = tmp;  }
//*H Partition epistrefei ton odhgo diamerishs gia enan dedomeno pinaka
//*Xrhsimopoiei mia tuxaia thesh gia ton odhgo

int Partition(int B[], int p, int r)
{    int pivotInd = p + rand()%(r - p + 1);
     int pivot;
     int i = p - 1;
     int j;
     pivot = B[pivotInd];
     swap(&B[pivotInd], &B[r]);
     for (j = p; j < r; j++)
 {   if (B[j] < pivot)
        {   i++;
   swap(&B[i], &B[j])     }  }
  swap(&B[i+1], &B[r]);
  return i + 1;  }
//* H q_s einai h tuxaiokratikh taxutaxinomhsh
    void q_s(int B[], int p, int q)
{   int j;
    if (p < q)  {
    j = Partition(B, p, q);
   q_s(B, p, j-1);
   q_s(B, j+1, q);    }  }

//* To kurio programma afou dhmiourgei enan pinaka
100 arithmwn xrhsimopoiei thn sunarthsh random_shuf gia na tous valei se tuxaia seira.
Arxikopoiei to roloi tou systimatos. Kalei thn
quicksort (q_s h opoia kalei me thn seira ths tin partition)
Sto telos kaleitai xana to roloi gia na shmeiwsei ton
termatismo tou programmatos. Ypologizetai o xronos
diekeperewsis kai tuponetai o taxinomhmenos pinakas
 int main()  {
 int i;
 int B[max];
     clock_t end_time, start_time;
     double time_elapsed;

 for (i = 0; i < max; i++)
      B[i] = i;
     random_shuf(B);
     printf("before the sort\n");
      for (i = 0; i < max; i++)
          printf("%d \t", B[i]);
    start_time = clock();
    q_s(B, 0, max-1);
    printf("\n after the sort \n");
  for (i = 0; i < max; i++)
         printf("%d \t", B[i]);
  end_time = clock();
  time_elapsed = (double)(end_time - start_time)/CLOCKS_PER_SEC;
  printf("randomized quicksort time elapsed %f seconds\n",time_elapsed); return 0;
  }
```

Κεφάλαιο 6

Βασικές έννοιες και προβλήματα πιθανοτήτων

Το κεφάλαιο αυτό αποτελεί μία εισαγωγή στις πιθανότητες, στις πράξεις μεταξύ συνόλων καθώς και στην αναφορά τυχαιοκρατικών αλγορίθμων. Η πιθανοτική ανάλυση αναφέρεται στην επίλυση προβλημάτων με την χρήση των πιθανοτήτων ή μεγεθών που εντάσσονται στις πιθανότητες, όπως είναι λ.χ. η μέση τιμή. Στις επόμενες ενότητες αναφέρονται εκτενώς βασικές εισαγωγικές έννοιες πιθανοτήτων και το κεφάλαιο κλείνει με την μελέτη και την αναφορά δύο τυχαιοκρατικών αλγορίθμων.

6.1 Δειγματικός χώρος και πειράματα τύχης

Για την μελέτη της θεωρίας πιθανοτήτων είναι απαραίτητη η εξέταση των τυχαίων (ή στοχαστικών) φαινομένων. Τα φαινόμενα αυτά ονομάζονται πειράματα τύχης και χαρακτηριστικό τους είναι πως σε μία εκτέλεσή τους δεν είναι δυνατή η βεβαιότητα του αποτελέσματος το οποίο θα εμφανισθεί. Ωστόσο είναι εφικτή η καταγραφή των δυνατών αποτελεσμάτων που θα προκύψουν από την μελέτη του πειράματος. Τα αποτελέσματα αυτά παρουσιάζονται μέσω ενός δειγματικού χώρου ή δειγματοχώρου [13].

Ορισμός 6.6. *Δειγματικός χώρος ενός τυχαίου πειράματος καλείται το σύνολο των δυνατών αποτελεσμάτων που προκύπτουν από μία εκτέλεσή του. Συμβολίζεται με κεφαλαία γράμματα όπως A ή S. Ο δειγματικός χώρος ενός πειράματος μπορεί να είναι:*

- *Διακριτός, δηλαδή να έχει πεπερασμένο ή αριθμήσιμο πλήθος στοιχειωδών ενδεχομένων όπως είναι μία ακολουθία αριθμών $Y = 1, 2, 3, 4, \ldots, N$ ή*
- *Συνεχής, όπου τα στοιχεία του να σχηματίζουν διαστήματα, όπως η ισότητα $Y \geq 0$, δηλαδή η μεταβλητή Y στην συγκεκριμένη περίπτωση να παίρνει μόνο τιμές θετικές.*

Τα στοιχεία ενός δειγματικού χώρου ονομάζονται **δειγματικά σημεία** ενώ τα υποσύνολα ενός δειγματικού χώρου ονομάζονται ενδεχόμενα και συμβολίζονται και αυτά με κεφαλαία γράμματα όπως A ή B. Απλά ή στοιχειώδη ενδεχόμενα ονομάζονται εκείνα τα οποία αποτελούνται από έναν μόνο δειγματικό χώρο. Σύνθετο ενδεχόμενο καλείται ένα ενδεχόμενο το οποίο περιέχει περισσότερα του ενός στοιχεία [13]. Θεωρείστε πως έχετε ένα νόμισμα και το ρίχνετε δύο φορές. Σημειώστε με K το ενδεχόμενο κατά την ρίψη του νομίσματος να εμφανισθεί κεφαλή και Γ το ενδεχόμενο να εμφανισθεί γράμματα. Ο δειγματικος χώρος του πειράματος θα είναι:

$$S = \{KK, K\Gamma, \Gamma K, \Gamma\Gamma\}$$

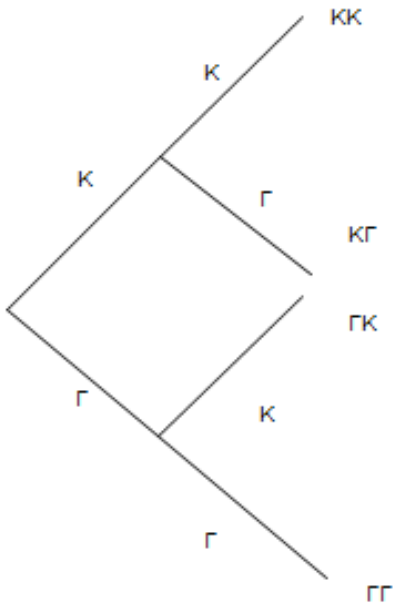
Το ενδεχόμενο να εμφανισθούν δύο φορές τα ίδια αποτελέσματα περιγράφεται από τον δειγματικό χώρο

$$X = \{ΚΚ, ΓΓ\}$$

ενώ, το ενδεχόμενο να εμφανίζεται μία κεφαλή και μία γράμματα περιγράφεται από τον δειγματικό χώρο

$$Y = \{ΚΓ, ΓΚ\}$$

Πολλές φορές η καταγραφή των αποτελεσμάτων ενός πειράματος τύχης είναι εφικτή με την χρήση των δενδροδιαγραμμάτων όπως απεικονίζεται στην παρακάτω εικόνα 6.1.



Εικόνα 6.1: *Δενδροδιάγραμμα παραδείγματος 6.1*

Το δενδροδιάγραμμα του παραπάνω παραδείγματος μπορεί να σχεδιασθεί και για την εκτέλεση ενός πειράματος για την ρίψη ενός νομίσματος 3, 4 ή n φορές.
Πείραμα τύχης μπορεί να θεωρηθεί και η γέννηση ενός παιδιού. Σε αυτήν την περίπτωση ο δειγματικός χώρος του πειράματος θα είναι

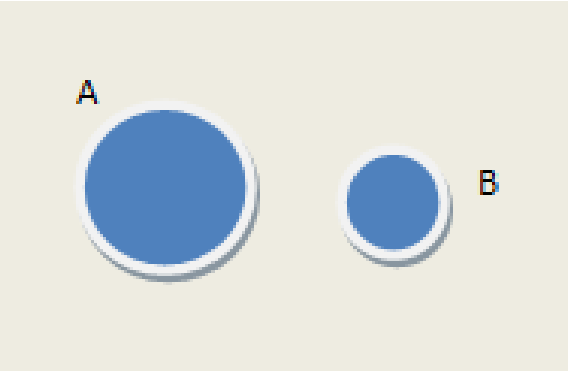
$$S = \{A, K\}$$

καθώς ένα παιδί μπορεί να γεννηθεί αγόρι(Α) ή κορίτσι(Κ).

6.2 Πράξεις μεταξύ συνόλων

Οι σχέσεις μεταξύ ενδεχομένων ενός δειγματικου χώρου μπορούν να σχεδιασθούν και με την βοήθεια των διαγραμμάτων Venn. Υποθέστε πως έχετε δύο υποσύνολα (ή ενεδεχόμενα)

ανεξάρτητα μεταξύ τους τα οποία ανήκουν σε ένα σύνολο S. Έπεται ότι τα υποσύνολα A και B ανήκουν στο S όπως φαίνεται και στο παρακάτω σχήμα 6.1.



Σχήμα 6.1: *Διάγραμμα Venn το οποίο απεικονίζει δύο ανεξάρτητα ενδεχόμενα A και B*

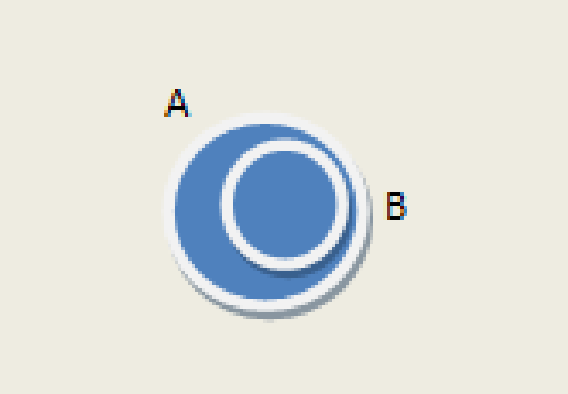
Υπάρχουν και άλλες πράξεις οι οποίες περιγράφουν τις σχέσεις μεταξύ των συνόλων και βοηθούν στην απόδειξη μαθηματικών θεωρημάτων που στηρίζονται στην πιθανοτική ανάλυση [4].

Ορισμός 6.7. ***Τομή****: Καλείται το σύνολο των στοιχείων τα οποία ανήκουν και στο A και στο B και συμβολίζεται με* $A \cap B$.

Ορισμός 6.8. ***Ένωση****: Καλείται το σύνολο των στοιχείων που ανήκουν ή στο A ή στο B ή και στα δύο σύνολα. Συμβολίζεται με* $A \cup B$.

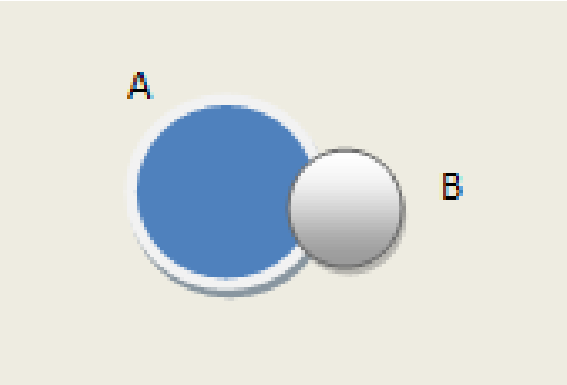
Ορισμός 6.9. ***Συμπλήρωμα*** *: Εάν το B είναι υποσύνολο του A, τότε το υποσύνολο* $A - B$ *καλείται συμπλήρωμα του B ως προς το A.*

Το συμπλήρωμα σχηματικά απεικονίζεται παρακάτω, βλπ σχήμα 6.2.



Σχήμα 6.2: *Διάγραμμα Venn το οποίο απεικονίζει το συμπλήρωμα των δύο συνόλων A και B*

Ορισμός 6.10. ***Διαφορά****: Συμβολίζεται με* $A - B$ *και περιέχει όπως φαίνεται και στο σχήμα 6.3 όλα τα στοιχεία του A που δεν ανήκουν στο B.*



Σχήμα 6.3: *Η Διαφορά των δύο υποσυνόλων Α και Β*

6.3 Ορισμοί Πιθανοτήτων

Ορισμός 6.11. *Υποθέτωντας πως ένας δειγματικός χώρος S περιέχει n ενδεχόμενα τα οποία είναι ισοπίθανα μεταξύ τους, η πιθανότητα να εμφανισθεί ένα ενδεχόμενο Α δίνεται από την σχέση*

$$P(A) = \frac{|A|}{|S|}$$

Όπου Α είναι το πλήθος των ενδεχομένων του συνόλου Α και S είναι το πλήθος των στοιχείων του δειγματοχώρου S.

Η παραπάνω πρόταση αποτελεί τον κλασικό ορισμό της πιθανότητας [13].

Βασιζόμενοι στο παρακάτω δενδροδιάγραμμα 6.4 προηγούμενου παραδείγματος, έστω πως ζητείται η πιθανότητα να έρθει δύο φορές κορρώνα «Κ».

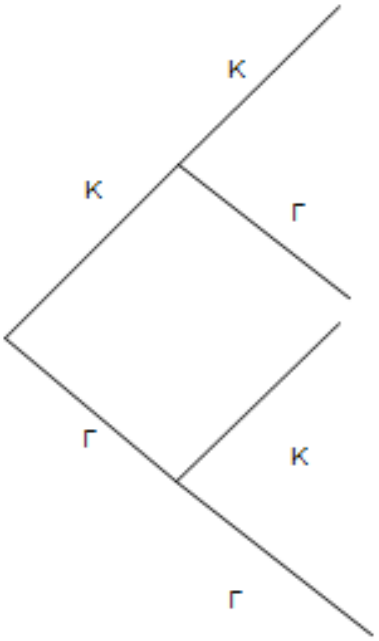
Ορισμός 6.12. *Σε κάθε γεγονός ενός δειγματικού χώρου αντιστοιχίζεται ένας πραγματικός αριθμός P(A). Η P καλείται ως μία πραγματική συνάρτηση ορισμένη στην κλάση c γνωστή ως συνάρτησης πιθανότητας. Η P(A) καλείται ως πιθανότητα εφόσον ικανοποιεί τα παρακάτω αξιώματα:*
Α) Για κάθε γεγονός της κλάσεως c είναι $P(A) \geq 0$.
Β) Για το γεγονός του δειγματοχώρου της κλάσεως είναι $P(S) = 1$.
Γ) Εάν τα γεγονότα A_1, *Β της κλάσεως c είναι ασυμβίβαστα τότε* $P(A \cup B) = P(A) + P(B)$

Η παραπάνω πρόταση αποτελεί τον αξιωματικό ορισμό της πιθανότητας [4].

6.4 Σημαντικά Θεωρήματα

Με βάση τον αξιωματικό ορισμό της πιθανότητας που αναφέρθηκε σε προηγούμενη ενότητα, προκύπτουν διάφορα θεωρήματα, όπως τα παρακάτω [4].

Θεώρημα 6.1. *Εάν P(A) είναι η πιθανότητα να συμβεί το γεγονός Α, τότε η πιθανότητα αυτή βρίσκεται ανάμεσα στο 0 και στο 1, δηλαδή* $0 \leq P(A) \leq 1$.
Απόδειξη



Σχήμα 6.4: *Δενδροδιάγραμμα*

Από τον αξιωματικό ορισμό της πιθανότητας προκύπτει ότι $P(A) \geq 0$ *για κάθε γεγονός* A*. Επίσης για το γεγονός του δειγματικού χώρου* S *της κλάσεως* c *είναι* $P(S) = 1$*, συνεπώς* $P(S) = 1$

.

Θεώρημα 6.2. ***Μονοτονία πιθανότητας****: Εάν* A *και* B *είναι δύο ενδεχόμενα ενός δειγματικού χώρου* S *με* B *να είναι υποσύνολο του* A*, τότε θα ισχύει* $P(B) \leq P(A)$ *και για κάθε σύνολο ή ενδεχόμενο ενός δειγματικού χώρου* S *ισχύει ότι* $P(A) \leq 1$*.*

Απόδειξη

Α)Εφόσον $B = A$*, τότε θα ισχύει* $P(A-B) = P(A)-P(B)$*. Αφού* $P(A-B) \geq 0$ *τότε* $P(B) \leq P(A)$

.

Β)Αφού το A *είναι υποσύνολο του* S *τότε η πιθανότητα του είναι* $P(A) \leq 1$.

Θεώρημα 6.3. *Εάν* A *και* B *είναι δύο ενδεχόμενα, τότε* $P(A \cup B) = P(A) + P(B) - P(AB)$*.*

Απόδειξη

Η Ένωση δύο συνόλων A *και* B *ενός δειγματικού χώρου μπορεί να περιγραφεί από την σχέση* $A \cup B = AB' \cup B$*. Χρησιμοποιώντας την προσθετική ιδιότητα θα ισχύει:* $P(A \cup B) = P(AB' \cup B) = P(AB') + P(B) = P(A) - P(AB) + P(B) = P(A) + P(B) - P(AB)$*.*

Θεώρημα 6.4. ***Νόμος του συμπληρωματικού τετραγώνου*** $P(A') = 1 - P(A)$*.*

Απόδειξη *Εάν* S *είναι ένας δειγματοχώρος και* A*,* A' *δύο ενδεχόμενα ανεξάρτητα μεταξύ τους, που ανήκουν στο* S*, τότε το σύνολο των στοιχείων που ανήκουν ή στο* A *ή στο* A' *ή και στα δύο θα ισούται με το σύνολο του δειγματικού χώρου, δηλαδή* $A \cup A' = S$*, και λαμβάνοντας υπόψην*

πως τα ενδεχόμενα A και A′ είναι ξένα μεταξύ τους προκύπτει ότι $P(A) + P(A') = 1$*, συνεπώς από αυτήν την σχέση προκύπτει αντίστοιχα* $P(A') = 1 - P(A)$.

6.5 Διακριτές και συνεχείς μεταβλητές

Όπως αναφέρθηκε σε προηγούμενη ενότητα ο δειγματικός χώρος ενός πειράματος μπορεί να είναι είτε διακριτός, δηλαδή να αποτελείται από πεπερασμένο ή αριθμήσιμο πλήθος ενδεχομένων, είτε συνεχής, δηλαδή να αποτελείται από ενδεχόμενα το πλήθος των οποίων σχηματίζουν διαστήματα. Υποθέτωντας πως ένα νόμισμα ρίχνεται δύο φορές, ο δειγματικός χώρος που προκύπτει θα περιγράφεται από τον παρακάτω πίνακα 6.1 Εάν συμβολισθεί με Κ το πλήθος των όψεων «κεφάλι» που αντιστοιχούν σε κάθε υποσύνολο τότε

Ενδεχόμενα Δειγματοχώρου	ΚΚ	ΚΓ	ΓΚ	ΓΓ
Κ	2	1	1	0

Πίνακας 6.1: *Πίνακας Παραδείγματος*

Έτσι προκύπτει $K = 2$ όταν το K εμφανίζεται δύο φορές, $K = 1$ όταν το K εμφανίζεται το πολύ μία φορά, $K = 0$ το K δεν εμφανίζεται καμία φορά. Η τυχαία μεταβλητή K στο παραπάνω παράδειγμα παίρνει τις τιμές $= 0, 1, 2, \ldots, n$ όπου n είναι ο αριθμός των ρίψεων του νομίσματος. Η τυχαία μεταβλητή Κ σε αυτήν την περίπτωση ονομάζεται διακριτή, εφόσον παίρνει ένα αριθμήσιμο ή πεπερασμένο πλήθος τιμών.

6.6 Μέση τιμή

Θεωρώντας μία διακριτή τυχαία μεταβλητή Χ με τιμές $X_1 + X_2 + X_3 + .. + X_3$ μπορεί να ορισθεί η έννοια της **αναμενόμενης μέσης τιμής ή μαθηματικής ελπίδας** [4] Θέτοντας ως $E(X)$ την μέση τιμή προκύπτει

$$E(X) = X_1 \cdot P(X = x_1) + x_2 \cdot P(X = X_2) + .. + X_n \cdot P(X = X_n)$$

Αντικαθιστώντας όπου

$$P(X = x_1) = f(x_1)$$

η νέα σχέση που προκύπτει είναι η

$$E(X) = x_1 \cdot f(x_1) + x_2 \cdot f(x_2) + .. + X_n \cdot f(X_n) = \sum X_n \cdot f(x)$$

Η αναμενόμενη μέση τιμή μίας τυχαίας συνεχούς μεταβλητής θα ισούται με το ολοκλήρωμα της συνάρτησης πυκνότητας-πιθανότητας $f(x)$ για τιμές που ανήκουν στο διάστημα των πραγματικών αριθμών, δηλαδή για κάθε $x \in \mathbf{R}$.

6.7 Μέση τιμή για συναρτήσεις τυχαίων μεταβλητών

Α)Εάν $x = f(x)$ και $Y = g(x)$ δύο τυχαίες μεταβλητές, τότε

$$E(X) = X \cdot P(X = x) + Y \cdot P(X = x) + .. + N \cdot P(X = n) = \sum P(X = x)$$

Εφόσον X και Y είναι δύο τυχαίες διακριτές μεταβλητές με συναρτήσεις πιθανότητας–πυκνότητας $f(x)$ και $g(x)$ αντίστοιχα, τότε η αναμενόμενη μέση τιμή θα δίνεται από την σχέση

$$E(X) = f(x_1) \cdot g(x_1) + f(x_2) \cdot g(x_2) + f(x_3) \cdot g(x_3) + .. + f(x_n) \cdot g(x_n) = \sum g(x) \cdot f(x)$$

Στην περίπτωση που η X είναι μία συνεχής τυχαία μεταβλητή με πιθανότητα $f(x)$ τότε

$$E(g(x)) = \int_{-\infty}^{\infty} f(x) \cdot g(x)$$

Εάν οι δύο συνεχείς τυχαίες μεταβλητές X και Y έχουν μία κοινή συνάρτηση πιθανότητας-πυκνότητας η οποία δέχεται τιμές X και Y, δηλαδή $f(x, y)$, η μέση αναμενόμενη τιμή θα είναι

$$E(g(x, y)) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) f(x, y)$$

6.8 Ιδιότητες της μέσης τιμής

Ιδιότητα 1η Εάν $E(X)$ είναι η αναμενόμενη μέση τιμή μίας τυχαίας μεταβλητής X και μία σταθερά c, τότε η μέση τιμή θα ισούται με

$$E(c \cdot X) = c \cdot E(X)$$

Ιδιότητα 2η

Έστω X και Y δύο τυχαίες μεταβλητές. Ο υπολογισμός του αθροίσματος της μέσης τιμής αυτών των μεταβλητών που προκύπτει είναι το άθροισμα της αναμενόμενης μέσης τιμής της άλλης μεταβλητής Y.

$$E(X + Y) = E(X) + E(Y)$$

Ιδιότητα 3η

Έστω X και Y δύο τυχαίες ανεξάρτητες μεταβλητές. Ο υπολογισμός του γινομένου των δύο αυτών μεταβλητών προκύπτει πολλαπλασιάζοντας την μέση τιμή της μίας τυχαίας μεταβλητής X με την μέση τιμή της άλλης μεταβλητής Y, δηλαδή

$$E(X \cdot Y) = E(X) \cdot E(Y)$$

6.9 Το πρόβλημα της πρόσληψης

Έστω πως ένα γραφείο επιθυμεί να προσλάβει έναν νέο γραμματέα. Μετά από πολλές προσπάθειες, απευθύνεται σε ένα γραφείο εύρεσης εργασίας με το οποίο γίνεται συμφωνία

Αλγοριθμος 6.1: *ΟΔΗΓΟΣ ΠΡΟΣΛΗΨΗΣ (n) [1]*

```
ΟΔΗΓΟΣ ΠΡΟΣΛΗΨΗΣ (n)

βέλτιστος <- 0
για   i<-1 έως n
          εξετάζουμε τον υποψήφιο   i
           αν ο υποψήφιος  i  είναι καταλληλότερος από τον βέλτιστος
              τότε βέλτιστος <- i
              προσλαμβάνουμε τον υποψήφιο i
```

για την καθημερινή αποστολή ενός διαφορετικού υπαλλήλου για την θέση εργασίας. Το γραφείο εξετάζει καθημερινά έναν υποψήφιο με διαφορετικά προσόντα ελέγχοντάς τα και συγκρίνοντάς τα πάντα με τα προσόντα των προηγούμενων υπαλλήλων. Τονίζεται πως στόχος του γραφείου είναι να έχει ανά πάσα στιγμή τον καλύτερο δυνατό υπάλληλο. Σαφώς η πρόσληψη ενός νέου υποψηφίου ειναι κάτι που κοστίζει περισσότερο σε συνδυασμό με την απόλυση του τωρινού και την καταβολή ενός ποσού στο γραφείο εύρεσης εργασίας. Τα παραπάνω βήματα του αλγόριθμου 6.1 εκφράζουν την στρατηγική πρόσληψης σε μορφή κώδικα. Ξεκινώντας στο βήμα 1 ορίζεται ένας εικονικός υποψήφιος με αριθμό 0 ο οποίος έχει λιγότερα προσόντα από τους υπόλοιπους.

Στην πραγματικότητα, αυτό που προέχει είναι ο υπολογισμός του κόστους, το οποίο προκύπτει από την εξέταση ενός υποψηφίου και στην συνέχεια από την πρόσληψη ενός νέου. Δεδομένου πως εξετάζονται n υποψήφιοι το κόστος της εξέτασης είναι c_i, και είναι χαμηλότερο από το κόστος της πρόσληψης που είναι ch. Εάν θεωρηθεί m ο αριθμός των ατόμων που προσλαμβάνοται, τότε το συνολικό κόστος που θα προκύψει είναι $O(nc_i + mch)$. Στην καλύτερη περίπτωση του προβλήματος πρόσληψης προκύπτει εάν προσληφθεί ένας μόνο υπάλληλος για την θέση, που είναι άλλωστε και το ζητούμενο.Σε αυτήν την περίπτωση το κόστος θα είναι $O(Ch)$. Η χειρότερη περίπτωση του προβλήματος, ωστόσο, προκύπτει εάν προσληφθούν όλοι οι υποψήφιοι που εξετάζονται κατά αύξουσα σειρά προσόντων.Εδώ το κόστος θα είναι $O(nch)$, όπου n ο αριθμός των υποψηφίων[1].

6.10 Τυχαία μετάθεση συστοιχίας

Ορισμένοι αλγόριθμοι χρησιμοποιούν πολλές φορές την πιθανότητα στην είσοδο ενός προβλήματος ή διατεταγμένης συστοιχίας αριθμών, όπως η παρακάτω: A=[5,6,7,8]

Η συστοιχία A περιέχει ένα n αριθμών, όπου εδώ n=4, με μικρότερο αριθμό προοτεραιότητας το n=5 εφόσον υπάρχουν τέσσερα στοιχεία. Σε αυτήν την περίπτωση γίνεται εφικτή η κατασκευή μίας τυχαίας μετάθεσης συστοιχίας. Αναλυτικότερα,ορίζοντας έναν τυχαίο αριθμό προτεραιότητας A[i] και κατασκευάζοντας τον πίνακα προτεραιοτήτων A=[28,9,30,27] με μικρότερο αριθμό τον n=9 και στην συνέχεια κατά αύξουσα σειρά τους αριθμούς $n = 17, n = 28, n = 30$, προκύπτει ο πίνακας συστοιχίας C=[6,8,5,7].

Παρακάτω δίνονται τα βήματα του αλγορίθμου για την παραπάνω διαδικασία της ταξι-

Αλγοριθμος 6.2: *ΤΑΞΙΝΟΜΙΚΗ ΜΕΤΑΘΕΣΗ (Α) [1]*

```
ΤΑΞΙΝΟΜΙΚΗ ΜΕΤΑΘΕΣΗ (Α)

n <- μήκος [Α]
για i <- 1 έως n
        P[i] = ΤΥΧΑΙΑ (1, n^3)
ταξινομούμε την Α, χρησιμοποιώντας ως κλειδιά τα στοιχεία της P
επιστροφή Α
```

νόμησης μετάθεσης.

Στον παραπάνω κώδικα επιλέγεται σύμφωνα με την διαδικασία της ταξινόμησης μετάθεσης ένα στοιχείο $A[i]$ μεταξύ του 1 και του n^3. Η επιλογή του συγκεκριμένου διαστήματος διακύμανσης τιμών γίνεται για την αύξηση της πιθανότητας όλων των αριθμών προτεραιότητας στην συστοιχία A να είναι μοναδικοί. Ωστόσο η εκτέλεση του κώδικα αποτελεί μία εξαιρετικά χρονοβόρα διαδικασία.

6.11 Δείκτριες τυχαίες μεταβλητές

Στην πιθανοτική ανάλυση γίνεται τις περισσότερες φορές απαραίτητη η χρήση των δείκτριων τυχαίων μεταβλητών. Οι δείκτριες τυχαίες μεταβλητές αποτελούν μία χρήσιμη μέθοδο μετατροπής των πιθανοτήτων σε αναμενόμενες τιμές και αντιστρόφως. Γενικά η χρήση των δείκτριων τυχαίων μεταβλητών μπορεί να γίνει πιο κατανοητή με το υπόδειγμα ενός πειράματος. Υποθέτωντας πως ένας φοιτητής επιθυμεί να προσδιορίσει τον αναμενόμενο αριθμό των κεφαλών Κ που προκύπτουν από την ρίψη ενός νομίσματος, ορίζει έναν δειγματικό χώρο S και ένα ενδεχόμενο Α. Στην περίπτωση του πειράματος που θέλει να πραγματοποιήσει ορίζει ως δειγματικό χώρο τις αναμενόμενες ρίψεις του νομίσματος,

$$S = \{K, \Gamma\}$$

όπου με K συμβολίζεται η κεφαλή του νομίσματος και Γ τα γράμματα, καθώς και P την πιθανότητα η ρίψη του νομίσματος να είναι Κ ή Γ, δηλαδή

$$P(K) = P(\Gamma) = \frac{1}{2}$$

Επιπλέον, ο μαθητής ορίζει μία τυχαία δείκτρια μεταβλητή Χ με την οποία θα εκφράζει την περίπτωση να προκύψει κεφαλή. Έτσι προκύπτει πως η τιμή της μεταβλητής θα ισούται με 1 εάν το αποτέλεσμα της ρίψης είναι κεφαλή, και 0 εάν είναι γράμματα. Συνεπώς προκύπτει ότι

$$X(K) = \begin{cases} 1, & P(K) = K \\ 0, & P(\Gamma) = \Gamma \end{cases} \tag{6.1}$$

Εάν ο μαθητής ορίσει ως $E(X)$ την αναμενόμενη τιμή της δείκτριας μεταβλητής Χ τότε θα υπολογίσει τον αριθμό των κεφαλών που αναμένεται.

$$E(X) = 1 \cdot P(K) + 0 \cdot P(\Gamma) = 1 \cdot P(K) = 1 \cdot \frac{1}{2} = \frac{1}{2} \tag{6.2}$$

Έτσι, υπολογίζοντας την αναμενόμενη τιμή ο μαθητής φθάνει στο συμπέρασμα πως ο αριθμός των κεφαλών που προκύπτουν στην ρίψη ενός νομίσματος ισούται με $\frac{1}{2}$ [4].

Κεφάλαιο 7

Επίλογος

Στο κεφάλαιο αυτό καταγράφονται όλα τα συμπεράσματα που προέκυψαν από την πτυχιακή αυτή εργασία.

7.1 Συμπεράσματα

Στην παρούσα πτυχιακή εργασία μελετήθηκε ο αλγόριθμος της τυχαιοκρατικής ταχυταξινόμησης του οποίου υπολογίσθηκε ο χρόνος εκτέλεσης.

Σε προηγούμενα κεφάλαια παρουσιάστηκαν οι χρόνοι εκτέλεσης των αλγορίθμων συγχωνευτικής ταξινόμησης, γρήγορης ταξινόμησης ταξινόμησης σωρού και της πιθανοτικής γρήγορης ταξινόμησης. Στον εν λόγω πίνακα παρουσιάζονται συνοποτικά οι χρόνοι εκτέλεσης 7.1.

Συμπερασματικά, ο αλγόριθμος της συγχωνευτικής ταξινόμησης έχει πολυπλοκότητα καλύτερης και χειρότερης περίπτωσης $O(nlogn)$ και η μέθοδος που χρησιμοποιεί για να ταξινομεί είναι η συγχώνευση.

Αλγόριθμος	καλύτερη περίπτωση	χειρότερη περίπτωση	Μέθοδος Ταξινόμησης
Ταξινόμηση με συγχώνευση	$O(nlogn)$	$O(nlogn)$	Συγχώνευση
Ταξινόμηση σωρού	$O(nlogn)$	$O(nlogn)$	Επιλογή
Ταχυταξινόμηση	$O(nlogn)$	$O(n^2)$	Διαμέριση
Τυχαιοκρατική ταχυταξινόμηση	$O(nlogn)$	$O(n^2)$	Τυχαιοκρατική διαμέριση

Πίνακας 7.1: *Συνοπτικός πίνακας πολυπλοκότητων αλγορίθμων ταξινόμησης*

Ο αλγόριθμος του σωρού χρησιμοποιεί ως κύρια μέθοδο την επιλογή και η διαδικασία ταξινόμησης περιγράφεται σε προηγούμενη ενότητα. Ο χρόνος εκτέλεσης καλύτερης και χειρότερης περίπτωσης ισούται με $O(nlogn)$.

Ο αλγόριθμος της ταχυταξινόμησης έχει πολυπλοκότητα καλύτερης περίπτωσης $O(nlogn)$ και χειρότερης περίπτωσης $O(n^2)$ και η μέθοδος που χρησιμοποιεί για να ταξινομήσει είναι η διαμέριση.

Ο αλγόριθμος της τυχαιοκρατικής ταχυταξινόμησης χρησιμοποιεί την τυχαιοκρατική διαμέριση σαν διαδικασία ταξινόμησης. Η μόνη διαφορά του αλγόριθμου αυτού με τον αλγόριθμο της ταχυταξινόμησης έγκειται στο γεγονός ότι στην τυχαιοκρατική ταχυταξινόμηση η θέση που επιλέγεται κάθε φορά είναι τυχαία σε αντίθεση με την Ταχυταξινόμηση στην οποία επιλέγεται πάντα το στοιχείο οδηγός $x = A[r]$.

Ο χρόνος εκτέλεσης καλύτερης περίπτωσης της πιθανοτικής γρήγορης ταξινόμησης είναι κατά μέσο όρο $O(nlogn)$ ενώ ο χρόνος εκτέλεσης της χειρότερης περίπτωσής της είναι $O(n^2)$.

Παραρτήματα

Παράρτημα Α΄

Αναδρομικές Σχέσεις- Master Theorem

Πολλές φορές, η μελέτη της πολυπλοκότητας ενός αλγορίθμου απαιτεί την χρήση εξισώσεων και άλλων σχέσεων για την διευκόλυνση του υπολογισμού της. Σε αυτήν την περίπτωση γίνεται αναγκαία η χρήση εξισώσεων, που είναι γνωστές ως αναδρομικές και χρησιμοποιούνται από αλγόριθμους που καλούν την αναδρομή. Επίσης η ανάγκη για τον ακριβή καθορισμό του χρόνου εκτέλεσης ενός αλγορίθμου οδήγησε στην ανακάλυψη ενός θεωρήματος που είναι γνωστό ως θεώρημα του Κυρίαρχου όρου ή Master Theorem.

Α΄.1 Αναδρομικές σχέσεις

Αναδρομική σχέση καλείται μία σχέση ή ισότητα της μορφής

$$2 \cdot T(\frac{n}{2}) + n$$

,

η οποία βοηθά στην ευκολότερη έκφραση της μελέτης του χρόνου εκτέλεσης ή όπως λέγεται, της πολυπλοκότητας ενός αλγορίθμου. Στην περίπτωση αυτή, της ανάλυσης δηλαδή της πολυπλοκότητας ενός αλγορίθμου εμφανίζεται πολύ συχνά και ιδιαίτερα όταν μελετάται ένας αναδρονικός αλγόριθμος, δηλαδή ένας αλγόριθμος ο οποίος καλεί τον εαυτό του.Στην ενότητα αυτήν θα παρουσιαστούν δύο βασικές μέθοδοι για την επίλυση των αναδρομικών αυτών εξισώσεων, συγκεκριμένα αυτές είναι α)Η Μέθοδος της επανάληψης και
β)Η Μέθοδος της αντικατάστασης [23] [24].

Α΄.1.1 Η Μέθοδος της επανάληψης

Θεωρείστε την αναδρομική σχέση

$$T(n) = \begin{cases} 2 \cdot T(\frac{n}{2}) + 2n, & k \geq 2 \\ 2, & n = 2 \end{cases} \quad (\text{Α΄.1})$$

Με την μέθοδο της επανάληψης, αυτό που πραγματοποιείται ουσιαστικά είναι η ανάπτυξη της σχέσης σε ένα άθροισμα και στην συνέχεια, το άθροισμα αυτό υπολογίζεται μέσω της κλειστής του σχέσης.[24]. Αναλυτικότερα, δουλεύοντας στην παραπάνω εξίσωση προκύπτει:

$$T(n) = 2n+2T(\frac{n}{2}) = 2n+\frac{2n}{2}+2^n+T(\frac{n}{4}) = .. = 2n+\frac{2n}{2}+2^2\cdot\frac{n}{2^2}+..+2^{i-1}\cdot\frac{n}{2^{i-1}}+2^i\cdot T(\frac{n}{2^i}) = 2n\cdot i+2^i\cdot T(\frac{n}{2^i})$$

Θέτοντας όπου

$$i = logn - 1$$

προκύπτει ότι

$$T(n) = n \cdot logn$$

Α΄.1.2 Η μέθοδος της αντικατάστασης

Η μέθοδος της αντικατάστασης χρησιμοποιεί την μαθηματική επαγωγή για να αποδείξει ότι η λύση είναι σωστή [24]. Θεωρώντας την σχέση

$$T(n) = T(\frac{n}{2}) + \Theta(n)$$

, με αρχική συνθήκη

$$T(1) = \Theta(1)$$

Η παραπάνω σχέση θα χρησιμοποιηθεί έτσι ώστε να αποδειχθεί πως

$$T(n) = \Theta(nlogn)$$

Χρησιμοποιώντας την παραπάνω σχέση επαγωγικά υποθέτωντας πως ισχύουν οι σχέσεις

$$c_1 \cdot (\frac{n}{2}) \cdot log(\frac{n}{2}) \leq T(\frac{n}{2}) \quad \text{(Α΄.2)}$$

και

$$T(\frac{n}{2}) \leq c_2 \cdot \frac{n}{2} \cdot log(\frac{n}{2}) \quad \text{(Α΄.3)}$$

Πρέπει να αποδειχθεί η σχέση

$$c_1 \cdot n \cdot logn \leq T(n) \leq c_2 \cdot n \cdot logn \quad \text{(Α΄.4)}$$

Χρησιμοποιώντας την σχέση που υποθετικά ισχύει και δουλεύοντας πάνω στις σχέσεις $A'2$ και $A'3$ προκύπτει:

$$T(n) \geq 2 \cdot c_1(\frac{n}{2}) \cdot log(\frac{n}{2}) + c_1 \cdot n = c_1 \cdot n \cdot (logn - 1) + c_1 \cdot n = c_1 \cdot n \cdot logn \quad \text{(Α΄.5)}$$

Για την δεύτερη σχέση $A'3$ προκύπτει:

$$T(n) \leq 2 \cdot c_2(\frac{n}{2}) \cdot log(\frac{n}{2}) + c_2 \cdot n = c_2 \cdot n \cdot (logn - 1) + c_2 \cdot n = c_2 \cdot n \cdot logn \quad (A'.6)$$

Δουλεύοντας πάνω στις σχέσεις $A'2$ και $A'3$ η σχέση $A'4$ αποδείχθηκε. Συνεπώς

$$T(n) = T(\frac{n}{2}) + \Theta(n)$$

Α΄.2 Το θεώρημα του Κυρίαρχου όρου

Το θεώρημα του Κυρίαρχου όρου ή Master Theorem αποτελεί έναν εύκολο τρόπο για την επίλυση των αναδρομικών σχέσεων της μορφής

$$T(n) = a \cdot T(\frac{n}{b}) + O(n^d)$$

όπου a και b (που είναι η βάση του λογαρίθμου) και d τρεις σταθερές [11].

Μέσω του Θεωρήματος αυτού διακρίνονται τρεις διαφορετικές περιπτώσεις:

1η περίπτωση

Εάν $d > log_b a$ τότε

$$T(n) = O(n^d)$$

2η περίπτωση

Εάν $d = log_b a$ τότε

$$T(n) = O(n^d logn)$$

3η περίπτωση

Εάν $d < log_b a$ τότε

$$T(n) = O(n^{log a} logn)$$

Βιβλιογραφία

[1] Ronald L. Rivest Clifford Stein Thomas H.Cormen, Charles E.Leiserson. *Introduction to Algorithms*. University edditions of Crete, Reading, Massachusetts, 2η έκδοση, 2013.

[2] Umesh Vazirani Sanroy Dasgupta, Christos Papadimitriou. *Algorithms*. Kleidarithmos, Reading, Massachusetts, 3η έκδοση, 2011.

[3] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Kleidarithmos, Reading, Massachusetts, 1η έκδοση, 2006.

[4] Murray Spiegel. *Probability and Statistics*. mcGraw Hill, Reading, Massachusetts, 1η έκδοση, 1978.

[5] Φώτω Αφράτη,Γιώργος Παπαγεωργίου, Τίμος Ασλανίδης. *Αλγόριθμοι:Μέθοδοι Σχεδίασης και Ανάλυσης Πολυπλοκότητας*. Συμμετρία, Αθήνα, 2η έκδοση, 2006.

[6] Ιωάννης Παπουτσής. *Εισαγωγή στις Δομές Δεδομένων και στους Αλγόριθμους, Υλοποίηση σε c++*. Αθ.Σταμούλης , Αθήνα, 1η έκδοση, 2010.

[7] Π.Κατσαρός. *Σχεδίαση Αλγορίθμων, σημειώσεις*. Πανεπιστήμιο Μακεδονίας, Μακεδονία, 1η έκδοση, 2010.

[8] Δρ Ηλίας Κ. Σάββας. *Αλγόριθμοι και Πολυπλοκότητα, σημειώσεις*. ΤΕΙ Λάρισας, Λάρισα, 2005.

[9] Σ. Ζάχος,Δ. Φωτάκης. *Πιθανοτικοί Αλγόριθμοι, σημειώσεις*. Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, ΕΜΠ, Αθήνα, 2010.

[10] Σ. Ζάχος,Δ. Φωτάκης. *Ασυμπτωτικός Συμβολισμός, σημειώσεις*. Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, ΕΜΠ, Αθήνα, 2010.

[11] Γ.Καραγιώργος. *Ασυμπτωτικός Συμβολισμός, σημειώσεις*. ΑΤΕΙ Πελοποννήσου,Τμήμα Μηχανικών Πληροφορικής ΤΕ, Σπάρτη, 2010.

[12] Σ. Ζάχος,Δ. Φωτάκης. *Γρήγορη ταξινόμηση, σημειώσεις*. Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, ΕΜΠ, Αθήνα, 2010.

[13] Μ. Κούτρας. *Εισαγωγή στις Πιθανότητες, θεωρία και εφαρμογές*. Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης, Πανεπιστήμιο Πειραιά, Αθήνα, 2004.

[14] Χ.Αναγνώστου. *Ανάλυση, επεξεργασία και παράσταση των αλγορίθμων ταξινόμησης Heap Sort*. Τμήμα Εφαρμοσμένης Πληροφορικής ΠΜΣ Ειδίκευσης, Πανεπιστήμιο Μακεδονίας, Θεσσαλονίκη, 2009.

[15] Δ.Μιχαήλ. *Αλγόριθμοι και πολυπλοκότητα,ανάλυση αλγορίθμων*. Τμήμα Πληροφορικής και Τηλεματικής,Χαροκόπειο Πανεπιστήμιο, Αθήνα, 2009.

[16] Ε.Ούτσιος. *Αλγόριθμοι και Δομές Δεδομένων,σημειώσεις θεωρίας*. Τμήμα Πληροφορικής και επικοινωνιών,Τεχνολογικό Εκπαιδευτικό Ίδρυμα Σερρών, Σέρρες, 2004.

[17] Τ. Δημητρίου. *Παράλληλοι Αλγόριθμοι και software*. Τμήμα Μηχανικών Υπολογιστών και Μηχανικών Υπολογιστών,, Σέρρες, 2001.

[18] Β.Ζησιμόπουλος. *Αλγόριθμοι και πολυπλοκότητα*. Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο, Αθήνα, 2008.

[19] Α.Βακάλη,Η.Ιωαννίδης,Ν.Ιωαννίδης,Χ.Κοίλιας,Κ.Μάλαμας,Ι.Μανωλόπουλος,Π.Πολίτης. *Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον*. Παιδαγωγικό ιστιντούτο, Αθήνα, 2008.

[20] Δ. Φωτάκης. *Αναζήτηση, Ταξινόμηση, επιλογή*. Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων, Σχολή Θετικών Επιστημών, Πανεπιστήμιο Αιγαίου, Σάμος, 2008.

[21] Γ.Καραγιώργος. *Quick sort, ασκήσεις*. ΑΤΕΙ Πελοποννήσου,Τμήμα Μηχανικών Πληροφορικής ΤΕ, Σπάρτη, 2010.

[22] Α.Πλιάτσιος. *Μελέτη αλγορίθμων ταξινόμησης στην κύρια μνήμη, πτυχιακή εργασία*. Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Σχολή θετικών επιστημών, Τμήμα Πληροφορικής , Θεσσαλονίκη, 2011.

[23] Δ.Ζαρολιάγκης. *Σημειώσεις για την επίλυση Αναδρομικών σχέσεων*. Πανεπιστήμιο Πατρών, Σχολή Τεχνολογικών επιστημών, Τμήμα Μηχανικών Η/Υ και Πληροφορικής , Πάτρα, 2004.

[24] Δ. Φωτάκης. *Επίλυση Αναδρομικών Σχέσεων*. Πανεπιστήμιο Αιγαίου, Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων, Σάμος, 2004.

[25] *Probabilistic analysis of algorithms*. http://en.wikipedia.org/wiki/Probabilistic_analysis_of_algorithms,.

[26] *Probabilistic Quick Sort, Tasos Dhmhtriou*. http://www0.dmst.aueb.gr/damianos/algo%5Cquicksort.pdf.

[27] *Randomized Algorithms*. http://www.cs.uoi.gr/~loukas/courses/Algorithms_and_Data_Structures/index.files/randomized_algorithms.pdf,.

[28] *Quick sort*. https://en.wikipedia.org/wiki/Quicksort.

[29] *sorting*. http://www.softlab.ntua.gr/~fotakis/data_structures/sorting.pdf.

[30] *programm of random quick sort*. http://www.sanfoundry.com/c-program-implement-quick-sort-using-randomization/.

[31] *Αλγόριθμος*. https://el.wikipedia.org/wiki/%CE%91%CE%BB%CE%B3%CF%8C%CF%81%CE%B9%CE%B8%CE%BC%CE%BF%CF%82.

[32] *ειδικά θέματα:πιθανοτικοί αλγόριθμοι*. https://eclass.uoa.gr/courses/D420/,.

[33] *Αλγόριθμοι*. https://openeclass.teimes.gr/modules/document/file.php/DEMES154/%CE%94%CE%B9%CE%B1%CF%86%CE%AC%CE%BD%CE%B5%CE%B9%CE%B5%CF%82/Algo%CE%99%CE%A7v.pdf,.

[34] *Time Functions*. https://www.google.gr/search?q=%CF%83%CF%85%CE%BD%CE%AC%CF%81%CF%84%CE%B7%CF%83%CE%B7+clock&ie=utf-8&oe=utf-8&gws_rd=cr&ei=hTFTVqqGNcGuadH5rLAN,.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
π.χ.	παραδείγματος χάρην
ΤΕΙ	Τεχνολογικό Εκπαιδευτικό Ίδρυμα

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
Συγχωνευτική Ταξινόμηση	Merge sort
Ταχυταξινόμηση	Quick Sort
Τυχαιοκρατική Ταχυταξινόμηση	Randomized Quick sort
Ταξινόμηση Σωρού	Heap sort
Αλγόριθμοι Μόντε Κάρλο	Monte Carlo Algorithms
Αλγόριθμος Λας Βέγκας	Las Vegas Algorithms
Πιθανότητα	Probability
Διαίρει και Βασίλευε	Divide and Conquer
Διαμέριση	Partition
Πολυπλοκότητα	Complexity
Μάστερ Θεώρημα	Master Theorem