Δομές δεδομένων και αρχείων

Αναφορά 2ης εργασίας

Θωμάς Λάγκαλης

Απρίλιος 2023

1 Εισαγωγή

Η εργασία έχει σκοπό την εξοικείωση με δομές που χειρίζονται δεδομένα δύο διαστάσεων (δύο κλειδιά). Οι δομές αυτές είναι το **KdTree** και το **Point-Region (PR) Quad Tree.** Οι ενέργειες που υλοποιήθηκαν στις παραπάνω δομές είναι η **εισαγωγή κλειδιού (insetrion)** και η **αναζήτηση (search)**. Τέλος, στην αναφορά θα γίνει ανάλυση της απόδοσης του κάθε αλγορίθμου/δομής με βάση τις επιτυχημένες και αποτυχημένες αναζητήσεις που πραγματοποιήθηκαν και θα απαντηθούν τα παρακάτω ερωτήματα:

- Ποια είναι η έκφραση της πολυπλοκότητας του κάθε αλγορίθμου που χρησιμοποιήθηκε και ποιες είναι οι καμπύλες που τις εκφράζουν;
- Ποια δομή είναι πιο γρήγορη και γιατί;

1.1 Γενική δομή του κώδικα

Για κάθε μία δομή δημιουργήθηκε η ανάλογη κλάση που περιέχει τις κατάλληλες μεθόδους για εισαγωγή και αναζήτηση. Η κάθε κλάση περιέχει μία εμφωλευμένη κλάση Node η οποία αναπαριστά τον κάθε κόμβο του δέντρου. Οι κλάσεις Node περιέχουν στη γενική περίπτωση τις απαραίτητες πληροφορίες κάθε κόμβου, δηλαδή τα κλειδιά και τις αναφορές (references) στα παιδιά του. Αν είναι φύλλο, τότε οι αναφορές αυτές είναι null.

Τα κλειδιά αναπαριστούν συντεταγμένες στον δισδιάστατο χώρο (x, y). Για τον λόγο αυτόν έγινε μια ξεχωριστή κλάση **Coordinates** η οποία αναπαριστά τις συντεταγμένες. Με αυτόν τον τρόπο τα κλειδιά αποτελούν στειγμιότυπα αυτής της κλάσης.

Οι συνταρτήσεις κάθε δομής **insert()** και **search()** έχουν υλοποιηθεί με τη βοήθεια των συναρτήσεων insertHelp() και searchHelp() οι οποίες είναι private και δεν καλούνται έξω από την εκάστοτε κλάση. Με αυτόν τον τρόπο η κλήση των insert() και search() γίνεται πιο απλή (με μόνο όρισμα το key) και κομψή.

1.2 Αναζητήσεις με τυχαία κλειδιά

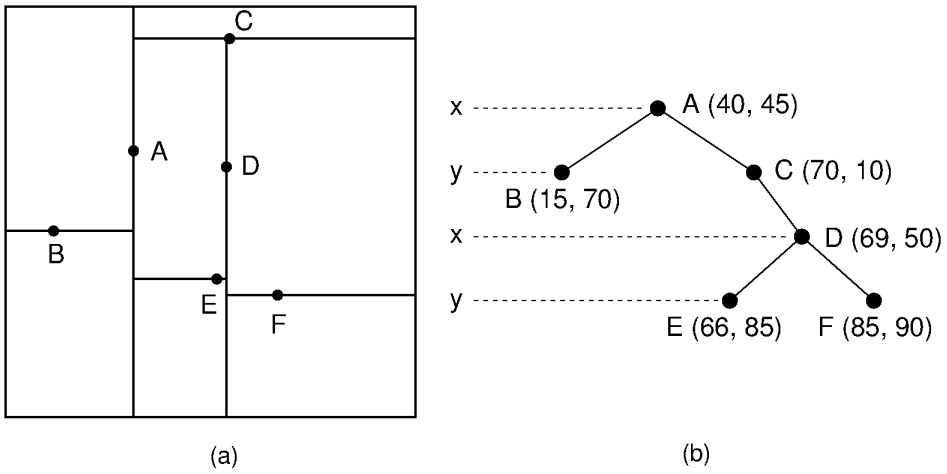
Για την εξαγωγή των αποτελεσμάτων πραγματοποιήθηκαν 100 αναζητήσεις με υπαρκτά και 100 αναζητήσεις με ανύπαρκτα κλειδιά για κάθε δομή με m εισαγωγές κλειδιών όπου το m παίρνει τις τιμές: 200, 500, 1000, 10000, 30000, 50000, 70000, 100000. Τα αποτελέσματα φαίνονται στην εικόνα 1.

	Kd Tree		PR Quad Tree	
M	successions(depth)	failed(depth)	successions(depth)	failed(depth)
200	8.95	10.35	4.95	3.85
500	9.12	11.39	5.48	4.60
1000	10.18	12.56	5.78	5.25
10000	16.10	17.75	7.47	6.81
30000	17.56	19.58	8.20	7.60
50000	19.36	20.82	8.71	7.98
70000	19.57	20.70	8.90	8.27
100000	19.03	22.68	9.22	8.55

Εικόνα 1: Αποτελέσματα αναζητήσεων. Μέσο βάθος για κάθε τιμή m

2 Kd Tree

Το Kd Tree είναι ένα δυαδικό δέντρο k διαστάσεων, τα κλειδιά του οποίου είναι k-διάστατα σημεία στον χώρο. Στην παρούσα εργασία, το k=2, δηλαδή τα κλειδιά έχουν δύο διαστάσεις (αποτελούν σημεία στον δισδιάστατο χώρο).



Εικόνα 2: Οπτική αναπαράσταση kd Tree (b) και των σημείων στον χώρο που περιέχει (a)

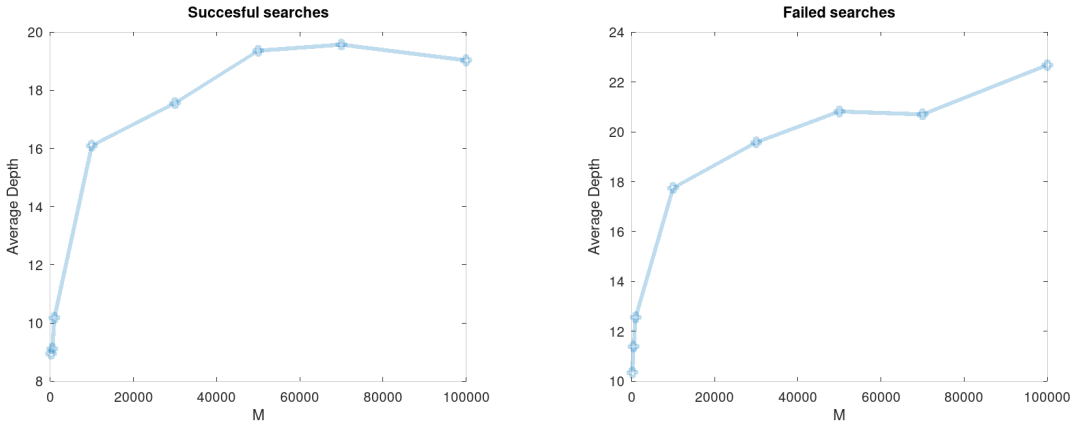
Όπως φαίνεται και στην εικόνα 1, οι επιτυχημένες αναζητήσεις έχουν μικρότερο μέσο βάθος από τις αποτυχημένες. Αυτό συμβαίνει διότι, όταν το κλειδί δεν υπάρχει η αναζήτηση φτάνει πάντα μέχρι τα φύλλα του δέντρου, ενώ όταν το κλειδί είναι υπαρκτό η αναζήτηση σταματάει στα φύλλα ή και νωρίτερα, ανάλογα με το που βρίσκεται το κλειδί.

Εφαρμόζοντας ασυμπτωτική ανάλυση η πολυπλοκότητα της αναζήτησης είναι O(logn) στη μέση περίπτωση και O(n) στη χειρότερη περίπτωση, αν το δέντρο είναι γραμμικό. Επομένως, το μέσο βάθος σε συνάρτηση με τον αριθμό των εγγραφών στο δέντρο μεταβάλλεται λογαριθμικά όπως φαίνεται στην Εικόνα 3

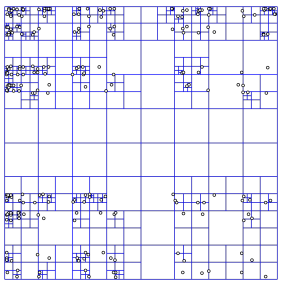
3 PR QuadTree

Τα PR (Point-Region) Quad Trees είναι δομές δένδρων που χρησιμοποιούνται για την αποθήκευση και τον χειρισμό των δεδομένων σε δισδιάστατο χώρο. Είναι μια βελτιωμένη μορφή των κλασικών Quad Trees, που επιτρέπουν την αναζήτηση τόσο των σημείων (points) όσο και των περιοχών (regions) στον χώρο.

Τα PR Quad Trees χωρίζουν αναδρομικά τον χώρο σε τετραγωνικά κελιά (cells) που μπορούν να περιέχουν είτε ένα μόνο σημείο, είτε ένα υπόδεντρο με περισσότερα κελιά και σημεία (Εικόνα 4). Αυτό επιτρέπει την αποτελεσματική οργάνωση των δεδομένων σε ένα δένδρο, το οποίο μπορεί να αναζητηθεί αποδοτικά για την αναγνώριση και τον χειρισμό σημείων και περιοχών. Ένα από τα κύρια πλεονεκτήματα των PR Quad Trees είναι η δυνατότητα αποτελεσματικής αναζήτησης εντός περιοχών, αντί να χρειάζεται να εξεταστούν όλα τα κελιά ξεχωριστά. Επιπλέον, τα PR Quad Trees είναι ευέλικτα σε πολλά είδη εφαρμογών, όπως επεξεργασία εικόνας.



Εικόνα 3: Διαγράμματα που δείχνουν πως μεταβάλλεται το μέσο βάθος σε 100 αναζητήσεις σε σχέση με τον αριθμό των εγγραφών στο δέντρο για πετυχημένες αναζητήσεις (αριστερά) και αποτυχημένες αναζητήσεις (δεξιά) σε kd Tree.



Εικόνα 4: Τρόπος οργάνωσης πληροφορίας στο PR Quad Tree

Κάθε εσωτερικός κόμβος του QuadTree περιέχει τέσσερα παιδιά, δηλαδή κάθε περιοχή περιέχει τέσσερις ίσα χωρισμένες υποπεριοχές (τεταρτημόρια). Οι πληροφορίες/κλειδιά (σημεία) περιέχονται στα φύλλα του δέντρου ενώ οι εσωτερικοί κόμβοι δεν περιέχουν κλειδιά. Επομένως, η μόνη πληροφορία που περιέχουν οι εσωτερικοί κόμβοι είναι οι αναφορές στα τέσσερα παιδιά τους, που για λόγους ευκολίας έχουν ονομαστεί NorthWest, NorthEast, SouthWest, SouthEast.

3.1 Οργάνωση Κώδικα κλάσης PRQuadTree

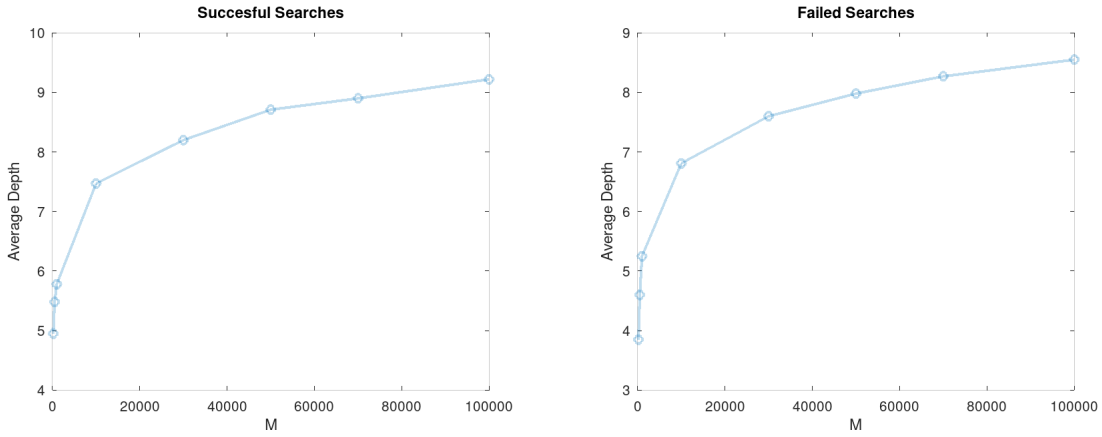
Η οργάνωση της κλάσης ακολουθεί παρόμοιο σχεδιασμό με αυτόν που περιγράφηκε παραπάνω. Δηλαδή, περιέχει μία εμφωλευμένη κλάση **Node** η οποία όμως διαφέρει με την αντίστοιχη της κλάσης kdTree. Συγκεκριμένα, εκτός από τις απαραίτητες πληροφορίες όπως το κλειδί και τις αναφορές στα παιδιά περιέχει και ορισμένες λειτουγρικότητες.

Η κλάση Node δεν παριστάνει σημείο στον χώρο όπως στο kdTree αλλά περιοχή (region). Για τον λόγο αυτόν περιέχει την μέθοδο **containsPoint()** η οποία ελέχει αν η συντεταγμένη που δέχεται ως όρισμα βρίσκεται μέσα στην περιοχή που παριστάνει το Node. Επίσης, περιέχει και την **numberOfKeysContaining()** η οποία επιστρέφει τον ακέραιο αριθμό των κλειδιών που περιέχονται στην περιοχή. Να σημειωθεί, πως μία περιοχή μπορεί να περιέχει παραπάνω από ένα κλειδί (σημείο στον χώρο) πράγμα που ρυθμίζεται από την global σταθερά **QT_NODE_CAPACITY** που έχει οριστεί για τις ανάγκες της εργασία ίση με 1.

Μία τετράγωνη περιοχή στον χώρο παριστάνεται από την εμφωλευμένη κλάση **Rectangle** η οποία περιέχει πεδία: τις συντεταγμένες του κέντρου του τετραγώνου και δύο ακεραίους, την απόσταση από το κέντρο μέχρι κάποια οριζόντια πλευρά και την απόσταση από το κέντρο μέχρι κάποια κατακόρυφη πλευρά. Κάθε φορά που υποδιαιρείται ένα τετράγωνο προσαρμόζονται ανάλογα οι τιμές των νέων τετραγώνων που δημιουργούνται.

3.2 Πολυπλοκότητα και αποτελέσματα αναζήτησης

Η πολυπλοκότητα του αλγορίθμου της αναζήτησης είναι **O(logn)** στη μέση περίπτωση και **O(n)** στη χειρότερη περίπτωση όπως και στο kd Tree.



Εικόνα 5: Μεταβολή του μέσου βάθους σε 100 αναζητήσεις σε σχέση με τον αριθμό των εγγραφών στο δέντρο για πετυχημένες αναζητήσεις (αριστερά) και αποτυχημένες αναζητήσεις (δεξιά) σε kd Tree.

4 Συμπεράσματα

Όπως γίνεται αντιληπτό στις εικόνες 3 και 5, οι καμπύλες που εκφράζουν το μέσο βάθος σε συνάρτηση με τον αριθμό εγγραφών είναι λογαριθμικές και στις δύο περιπτώσεις δέντρων. Αυτό δικαιολογεί και την ταύτιση στην πολυπλοκότητα των αλγορίθμων αναζήτησης. Αυτό που αλλάζει όμως είναι το τελικό μέσο βάθος που φτάνει ο κάθε αλγόριθμος αναζήτησης για κάθε δέντρο (εικόνα 1). Στο Quad Tree το μέσο βάθος είναι εμφανώς μικρότερο από αυτό του kd Tree. Οφείλεται, στο γεγονός ότι στο Quad Tree κάθε κόμβος περιέχει τέσσερα παιδιά με αντίθεση το kd Tree που περιέχει δύο παιδιά. Επομένως, και στις δύο περιπτώσεις το μέσο βάθος μεταβάλλεται λογαριθμικά με την αύξηση του αριθμού των εγγραφών όμως η βάση του λογαρίθμου είναι 4 στο Quad Tree και 2 στο kd Tree.

4.1 Ποια δομή είναι πιο γρήγορη;

Η απάντηση αυτού του ερωτήματος δεν είναι σαφής. Από τα αποτελέσματα των αναζητήσεων μπορεί να φαίνεται πως το μέσο βάθος είναι μικρότερο στο Quad Tree όμως δεν είναι καθοριστικός παράγοντας για τον προσδιορισμό της ταχύτητας του αλγορίθμου. Αυτό ισχύει διότι, τα βήματα που εκτελούνται σε κάθε κόμβο κατά την αναζήτηση είναι διαφορετικά σε κάθε δομή. Αυτό το γεγονός είναι ικανό να επηρεάσει σημαντικά την ταχύτητα της αναζήτησης. Ο πιο καλός τρόπος να καθοριστεί ποιος αλγόριθμος είναι πιο γρήγορος είναι να χρονομετρηθούν οι αναζητήσεις τους και να υπολογιστεί ο μέσος όρος. Επομένως, η κατάλληλη απάντηση στο ερώτημα είναι ότι η ταχύτητα της δομής εξαρτάται από την επεξεργασία των δεδομένων που κάνει η κάθε δομή στην αναζήτηση.

5 Πηγές

- BST code implemented for KdTree: https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/
- Search in BST: https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/
- Την παραγωγή τυχαίων κλειδιών από την εκφώνηση του προηγούμενου project.
- Random Generator (nextInt()): https://docs.oracle.com/javase/8/docs/api/java/util/Random.html#nextDouble–
- MultiCounter από τις διαφάνειες του φροντιστηρίου.
- Wikipedia QuadTree: https://en.wikipedia.org/wiki/Quadtree