Τμήμα Μηχανικών Ηλεκτρονικών
Υπολογιστών και Πληροφορικής

Πολυδιάστατες Δομές Δεδομένων και Υπολογιστική Γεωμετρία

Υλοποίηση δομής QuadTree σε Scala

Ιωάννης Ορέστης Ρίζος – ΑΜ: 6190
Ανδρέας Αντωνίου – ΑΜ: 6261

Περιεχόμενα

1 Μεταγλώττιση και δημιουργία εκτελέσιμου

Όλες οι διαδικασίες σχετικές με την μεταγλώττιση των `scala` αρχείων, την δημιουργία πακέτου (.jar file) και το τρέξιμο σε development environment γίνεται με χρήση του εργαλείου `sbt`, το προεπιλεγμένο εργαλείο χτισίματος `scala` projects. Έτσι για την μεταγλώττιση χρειάζονται τα εξής εργαλεία εγκατεστημένα:

- `scala`
- `sbt`

Για να γίνει μεταγλώττιση των `.scala` αρχείων σε `.class` εκτελέσιμα αρχεία, ανοίγουμε ένα command-line terminal, στο κύριο κατάλογο του project (εκεί που βρίσκεται το `build.sbt` αρχείο) και εκτελούμε την εντολή `sbt compile`.

```
john@John-HP:~/Desktop/quadtree$ sbt compile
[info] Loading global plugins from /home/john/.sbt/1.0/plugins
[info] Loading project definition from /home/john/Desktop/quadtree/project
[info] Loading settings for project quadtree from build.sbt ...
[info] Set current project to quadtree (in build
file:/home/john/Desktop/quadtree/)
[info] Executing in batch mode. For better performance use sbt's shell
[info] Updating ...
[info] Done updating.
[info] Compiling 4 Scala sources to /home/john/Desktop/quadtree/target/scala-
2.12/classes ...
[info] Done compiling.
[success] Total time: 24 s, completed Jan 30, 2019 12:28:01 AM
```

Για την την δημιουργία `.jar` πακέτου για εύκολη διανομή του εκτελέσιμου κώδικα εκτελούμε την εντολή `sbt package`:

```
john@John-HP:~/Desktop/quadtree$ sbt package
[info] Loading global plugins from /home/john/.sbt/1.0/plugins
[info] Loading project definition from /home/john/Desktop/quadtree/project
[info] Loading settings for project quadtree from build.sbt ...
[info] Set current project to quadtree (in build
file:/home/john/Desktop/quadtree/)
[info] Packaging /home/john/Desktop/quadtree/target/scala-2.12/quadtree_2.12-
0.1.jar ...
[info] Done packaging.
[success] Total time: 2 s, completed Jan 30, 2019 12:36:09 AM
```

Για την εκτέλεση του `.jar` αρχείου εκτελούμε την εντολή `scala` ακολουθούμενη από το `.jar` αρχείο. π.χ.

```
john@John-HP:~/Desktop/quadtree$ scala target/scala-2.12/quadtree_2.12-0.1.jar
```

2 Εκτέλεση προγράμματος

2.1 Λειτουργίες εκτέλεσης εφαρμογής

Ο πυρήνας της υλοποίησης βρίσκεται στο αρχείο `QuadTree.scala` το οποίο υλοποιεί την δομή Quad Tree, και όλες τις λειτουργίες της. Έτσι μπορεί να γίνει χρήση της κλάσης αυτής ως βιβλιοθήκη σε κάποιο project. Ωστόσο έχει αναπτυχθεί και κλάση Main με σκοπό την δημιουργία εκτελέσιμου και την εκτέλεση ορισμένων λειτουργιών για παρουσίαση των δυνατοτήτων της δομής και έλεγχος της σωστής λειτουργίας της. Για την εκτέλεση του προγράμματος μπορεί να χρησιμοποιηθεί το .jar που δημιουργήθηκε πιο πάνω ή η εντολή `sbt run`.

```
john@John-HP:~/Desktop/quadtree$ sbt run
[info] Running Main
Usage: quadtree command

Commands:
 time [-n elements]:   Performs all basic operations (except range search) of n
                       elements, and prints the time to complete
 test [-i iterations]: Performs and validates all quadtree operations with random
                       data
 demo [-n elements]:   Performs a demo of all QuadTree operations with random
                       data. Prints the tree generated in .dot format
```

Όταν γίνει εκτέλεση χωρίς την παροχή κάποιου command line argument, τότε τυπώνεται αυτό το κείμενο που δείχνει τις διαθέσιμες λειτουργίες και την παράμετρο που πρέπει να περαστεί για την εκτέλεση τής.

2.1.1 Λειτουργία time

Η λειτουργία time δημιουργεί ένα QuadTree αντικείμενο και εκτελεί όλες τις πράξεις του δέντρου (εκτός range search), χρονομετρώντας τον χρόνο εκτέλεσης της κάθε μιας. Η λειτουργία δέχεται και μια προαιρετική παράμετρο (-n αριθμός), αυτή του πλήθους στοιχείων που θα εισαχθούν στο δέντρο και φορών που θα πραγματοποιηθεί κάθε λειτουργία (αν δεν δοθεί, η προεπιλογή είναι 80.000 στοιχεία). Δύο παραδείγματα εκτέλεσης `sbt “run time”` και `sbt “run time -n 200000”` φαίνονται παρακάτω.

```
john@John-HP:~/Desktop/quadtree$ sbt "run time"
[info] Loading global plugins from /home/john/.sbt/1.0/plugins
[info] Loading project definition from /home/john/Desktop/quadtree/project
[info] Loading settings for project quadtree from build.sbt ...
[info] Set current project to quadtree (in build file:/home/john/Desktop/quadtree/)
[info] Running Main time
Performing and timing operations with 80000 elements.
Insertion of 80000 points: 608.800519 ms
Search of 80000 points: 324.929002 ms
Update of 80000 points: 355.604526 ms
3NN Search of 80000 points: 2970.777617 ms
Removal of 80000 points: 555.167311 ms
```

Εικόνα 1: Εκτέλεση λειτουργίας time, με τον προεπιλεγμένο αριθμό 80.000 στοιχείων.

```
john@John-HP:~/Desktop/quadtree$ sbt "run time -n 200000"
[info] Loading global plugins from /home/john/.sbt/1.0/plugins
[info] Loading project definition from /home/john/Desktop/quadtree/project
[info] Loading settings for project quadtree from build.sbt ...
[info] Set current project to quadtree (in build file:/home/john/Desktop/quadtree/)
[info] Running Main time -n 200000
Performing and timing operations with 200000 elements.
Insertion of 200000 points: 1269.341674 ms
Search of 200000 points: 738.426916 ms
Update of 200000 points: 878.40242 ms
3NN Search of 200000 points: 6228.438879 ms
Removal of 200000 points: 1242.253687 ms
```

Εικόνα 2: Εκτέλεση λειτουργίας time, με 200.000 στοιχεία.

> Παρατηρούμε την χρήση εισαγωγικών στα command-line arguments, όταν αυτό εκτελείται με χρήση του `sbt run`. Αν η εκτέλεση γίνει από το scala interpreter απ' ευθείας αυτά δεν χρειάζονται.

Οι χρόνοι που φαίνονται παραπάνω παράχθηκαν με την εκτέλεση του time σε Laptop με επεξεργαστή AMD A6-6410.

2.1.2 Λειτουργία test

Η λειτουργία test εκτελεί τα όλες τις λειτουργίες που υλοποιεί η δομή, με προκαθορισμένο πλήθος στοιχείων, και κρατάει τα στοιχεία σε μια ακόμα δομή (π.χ. HashMap). Στη συνέχεια ελέγχει τα αποτελέσματα που επιστρέφει η δομή του δέντρου με τα αποθηκευμένα, για να επαληθεύσει τη σωστή λειτουργία της δομής. Η λειτουργία δέχεται και μια προαιρετική παράμετρο (-i αριθμός), το πλήθος των φορών που θα τρέξει το κάθε test (αν δεν δοθεί, η προεπιλογή είναι 10 iterations). Ένα παράδειγμα εκτέλεσης της λειτουργίας test για 1 iteration με χρήση της εντολής `sbt "run test -i 1"` φαίνεται παρακάτω.

```
john@John-HP:~/Desktop/quadtree$ sbt "run test -i 1"
[info] Loading global plugins from /home/john/.sbt/1.0/plugins
[info] Loading project definition from /home/john/Desktop/quadtree/project
[info] Loading settings for project quadtree from build.sbt ...
[info] Set current project to quadtree (in build file:/home/john/Desktop/quadtree/)
[info] Running Main test -i 1
Running 1 iterations of all operation validations.
 - Iteration: 1 -
Passed Build Test
Passed Insertion Test
Passed Removal Test
Passed Update Test
Passed Range Search Test
Passed KNN Search Test
```

Εικόνα 3: Εκτέλεση λειτουργίας test, για 1 επανάληψη.

2.1.3 Λειτουργία demo

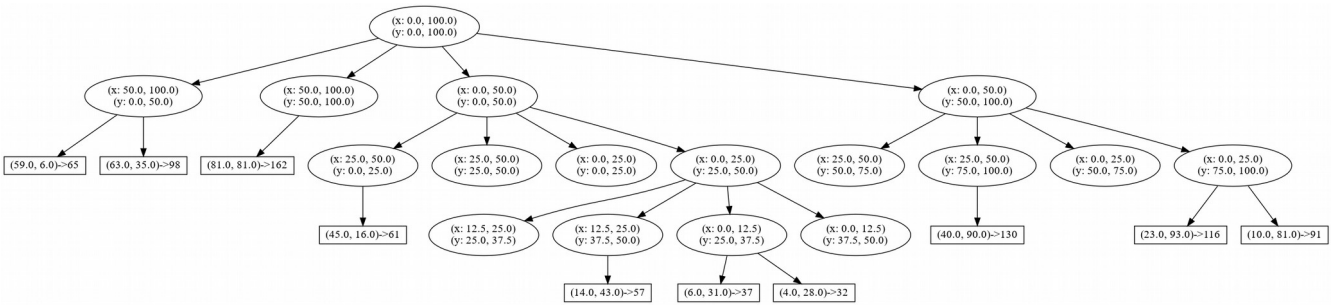
Τέλος παρέχεται και η λειτουργία demo η οποία δημιουργεί ένα στιγμιότυπο της κλάσης QuadTree, και εκτελεί όλες τις λειτουργίες, με έναν μικρό αριθμό στοιχείων (20 αν δεν δοθεί άλλος αριθμός με την προαιρετική παράμετρο `-n αριθμός`), και τυπώνει όλα τα αποτελέσματα. Η λειτουργία επίσης δημιουργεί και δύο αρχεία, tree.dot και tree.plot. Τα δύο αυτά αρχεία μπορούν να χρησιμοποιηθούν από το πρόγραμμα graphviz, και gnuplot, αντίστοιχα για να δημιουργήσουν μια γραφική οπτικοποίηση του δέντρου που παράχθηκε καθώς και του δισδιάστατου χώρου ορίων των κόμβων και των στοιχείων. Ένα παράδειγμα του εκτέλεσης του demo 10 στοιχείων με χρήση της εντολής `sbt "run demo -n 10"` φαίνεται παρακάτω.

```
john@John-HP:~/Desktop/quadtree$ sbt "run demo -n 10"
[info] Loading global plugins from /home/john/.sbt/1.0/plugins
[info] Loading project definition from /home/john/Desktop/quadtree/project
[info] Loading settings for project quadtree from build.sbt ...
[info] Set current project to quadtree (in build file:/home/john/Desktop/quadtree/)
[info] Running Main demo -n 10
Performing a demo of all operations with 10 elements.
[DEMO] Generated 10 points:
Point(23.0,93.0)
Point(59.0,6.0)
Point(6.0,31.0)
Point(45.0,16.0)
Point(14.0,43.0)
Point(4.0,28.0)
Point(10.0,81.0)
Point(81.0,81.0)
Point(40.0,90.0)
Point(63.0,35.0)

[DEMO] Inserting 10 points in the tree with data their x+y as Int...
[DEMO] Generating graphviz format .dot file as 'tree.dot' for tree visualization...
[DEMO] Generating gnu plot script file file as 'tree.plot' for tree visualization...


[DEMO] Searching for point Point(14.0,43.0)...
[DEMO] Tree result: 57

[DEMO] Updating point Point(14.0,43.0) value to 8...
[DEMO] Searching for point Point(14.0,43.0)...
[DEMO] Tree result: 8

[DEMO] Removing point Point(14.0,43.0) from tree...
[DEMO] Searching for point Point(14.0,43.0)...
[DEMO] Tree result: None

[DEMO] Range searching from x: (22, 44) y: (66, 88)...
[DEMO] Tree found 0 points:

[DEMO] Performing 2NN search of point Point(32.0,73.0)
[DEMO] Tree found 2 nearest neighbours:
(Point(23.0,93.0),116)
(Point(40.0,90.0),130)

[DEMO] COMPLETED
```

Εικόνα 4: Εκτέλεση λειτουργίας demo με 10 τυχαία στοιχεία.

Για την παραγωγή της γραφική οπτικοποίησης του δέντρου πρέπει να είναι εγκατεστημένο το εργαλείο graphviz. Για την δημιουργία μιας εικόνας σε μορφή png εκτελείται η εξής εντολή.

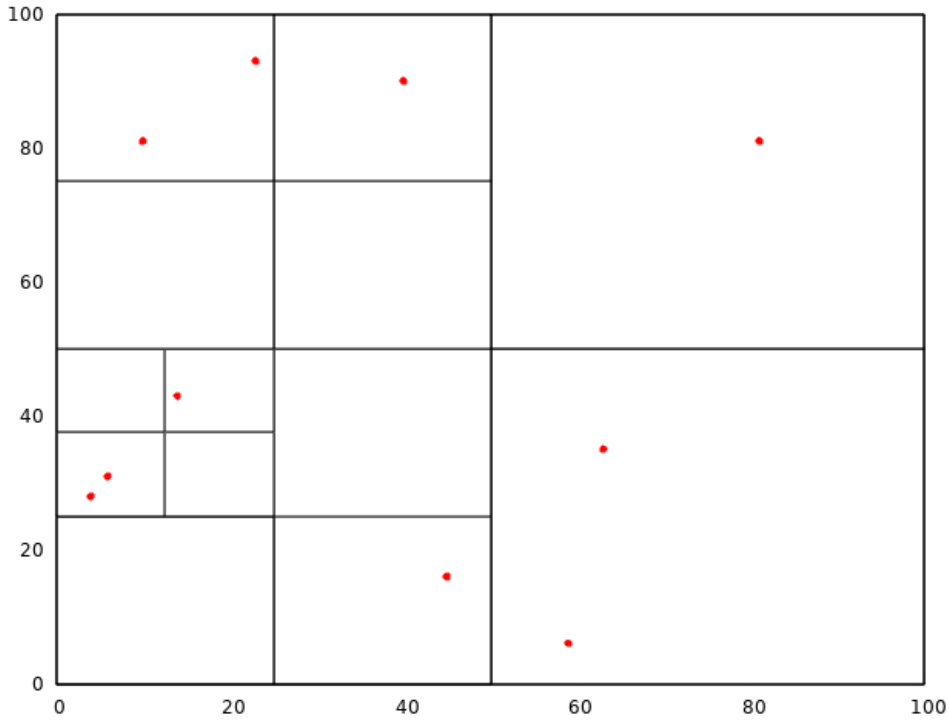
```
john@John-HP:~/Desktop/quadtree$ dot -Tpng tree.dot -o tree.png
```



Εικόνα 5: Οπτικοποίηση δέντρου που παράχθηκε στο παραπάνω demo.

Για την παραγωγή των γραφικών οπτικοποίησης του δισδιάστατου χώρου των στοιχείων που περιέχει το δέντρο πρέπει να είναι εγκατεστημένο το εργαλείο gnu plot. Για την δημιουργία μιας εικόνας σε μορφή png εκτελείται η εξής εντολή.

```
john@John-HP:~/Desktop/quadtree$ gnuplot -p tree.plot
```



Εικόνα 6: Οπτικοποίηση του δισδιάστατου χώρου των σημείων (κόκκινες τελείες) και των ορίων των κόμβων του δέντρου (κάθε κόμβος έχει χωρητικότητά 2 σημείων πριν χωρίστει).

2.2 Διαδραστική εκτέλεση μέσω Scala REPL

Οι λειτουργίες που περιγράφηκαν παραπάνω προσφέρονται από την κλάση Main, περνώντας σε αυτήν την κατάλληλη παράμετρο. Ωστόσο όλες αυτές παράγουν τυχαία σημεία, το πλήθος των οποίων ορίζει ο χρήστης. Μπορούμε να εκμεταλλευτούμε το interactive shell που προσφέρει η scala, για να δημιουργήσουμε το δικό μας QuadTree από το command line και να εκτελέσουμε πράξεις σε αυτό, χωρίς να πειράξουμε των πηγαίο κώδικα που το ορίζει. Για να γίνει αυτό εκτελούμε την εντολή sbt console.

```
john@John-HP ~/Desktop/quadtree  sbt console
```

Από εδώ έχουμε παροχή σε όλες της λειτουργίες που ορίζονται στην κλάση QuadTree.scala και στις παραμέτρους που μπορούν να περαστούν. Για να γίνει πιο εύκολα κατανοητός ο τρόπος χρήσης της δομής από το REPL, ακολουθούν παραδείγματα εκτέλεσής τους των λειτουργιών στο παρακάτω πλαίσιο. Μια σύντομη περιγραφή για τη κάθε λειτουργία βρίσκεται σε μορφή σχόλιου πάνω από τη κάθε μία. Όπου υπάρχει το prompt “scala>”, αναγράφεται αυτό που εισήχθηκε από τον χρήστη, ενώ στην επόμενη γραμμή τα αποτελέσματα που επιστρέφονται.

```
john@John-HP ~/Desktop/quadtree  sbt console
Welcome to Scala 2.12.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_202).
Type in expressions for evaluation. Or try :help.



// Δημιουργία δέντρου όπου κάθε κόμβος χωράει 2 σημεία, με κέντρο το 0,0 και
// διαστάσεις απο -50,-50 έως 50,50. Τα values των σημείων είναι τύπου String
scala> val q = new QuadTree[String](K=2, center=Point(0,0), halfDim=50)
q: QuadTree[String] = QuadTree@21710383

// Εισαγωγή σημείου 1,5 και τιμή “String1”
scala> q.insert(Point(1,5), "String1")
// Επιστράφηκε τιμή true ως ένδειξη επιτυχίας εισαγωγής
res1: Boolean = true

scala> q.insert(Point(5,5), "String2")
res2: Boolean = true

scala> q.insert(Point(5,50), "String2")
res3: Boolean = true

scala> q.insert(Point(-49, 13), "String3")
res5: Boolean = true

scala> q.insert(Point(-9, 0), "String3")
res6: Boolean = true

scala> q.insert(Point(41, 14), "String3")
res7: Boolean = true
```

```
// Αναζήτηση σημείου 41, 14 στο δέντρο.
scala> q.search(Point(41, 14))


// Επιστράφηκε τιμή String3
res8: Option[String] = Some(String3)

// Διαγραφή σημείου 41,14
scala> q.remove(Point(41, 14))
// Επιστρέφει true ως ένδειξη επιτυχημένης διαγραφής σημείου
res9: Boolean = true

// Αναζήτηση σημείου 41, 14 στο δέντρο.
scala> q.search(Point(41,14))
// Επιστρέφει None
res10: Option[String] = None

// Δημιουργία dot αρχείου με όνομα ‘dotfile.dot’ για οπτικοποίηση δέντρου
// q, το όνομα της μεταβλητής δέντρου που δημιουργήθηκε πιο πάνω
scala> Demo.generateDotFile(q, "dotfile.dot")


// Δημιουργία gnu plot αρχείου με όνομα ‘tree.plot’ για οπτικοποίηση χώρου
scala> Demo.generatePlotFile(q, "tree.plot") //q, η μεταβλητή δέντρου

// Εκτύπωση δέντρου στο τερματικό
scala> q.dfsPrint
{(x: -50.0 to 50.0), (y: -50.0 to 50.0)} -> {(x: 0.0 to 50.0), (y: -50.0 to 0.0)}
|  {(x: 0.0 to 50.0), (y: 0
.0 to 50.0)}  |  {(x: -50.0 to 0.0), (y: -50.0 to 0.0)}  |  {(x: -50.0 to 0.0),
(y: 0.0 to 50.0)}  |
{(x: 0.0 to 50.0), (y: -50.0 to 0.0)} ->
{(x: 0.0 to 50.0), (y: 0.0 to 50.0)} -> {(x: 25.0 to 50.0), (y: 0.0 to 25.0)}  |
{(x: 25.0 to 50.0), (y: 25.0
to 50.0)}  |  {(x: 0.0 to 25.0), (y: 0.0 to 25.0)}  |  {(x: 0.0 to 25.0), (y:
25.0 to 50.0)}  |
{(x: 25.0 to 50.0), (y: 0.0 to 25.0)} ->
{(x: 25.0 to 50.0), (y: 25.0 to 50.0)} ->
{(x: 0.0 to 25.0), (y: 0.0 to 25.0)} ->
{(x: 0.0 to 25.0), (y: 25.0 to 50.0)} ->
{(x: -50.0 to 0.0), (y: -50.0 to 0.0)} ->
{(x: -50.0 to 0.0), (y: 0.0 to 50.0)} ->

// Αναζήτηση 1 κοντινότερου γείτονα (δεύτερη παράμετρος), από το σημείο 0,0
scala> q.knnSearch(Point(0,0), 1)



// Επιστράφηκε λίστα που περιέχει ένα σημείο 1, 5 με περιεχόμενο String1
```

```
res14: scala.collection.mutable.ListBuffer[(Point, String)] =
ListBuffer((Point(1.0,5.0),String1))


// Αναζήτηση εύρους από x: -100 έως 0 και y: -100 έως 0
scala> q.rangeSearch(Point(-100,-100), Point(0,0))
// Επιστράφηκε μια λίστα που περιέχει μόνο ένα σημείο το -9, 0
res15: scala.collection.mutable.ListBuffer[(Point, String)] = ListBuffer((Point(-
9.0,0.0),String3))


// Εκτέλεση λειτουργίας ενημέρωσης στο σημείο -49, 13 με την τιμή “updatedString”
scala> q.update(Point(-49, 13), "updatedString")
res19: Boolean = true


// Αναζήτηση του σημείου.
scala> q.search(Point(-49, 13))
// Επιστρέφεται η ενημερωμένη τιμή
res20: Option[String] = Some(updatedString)

scala>
```