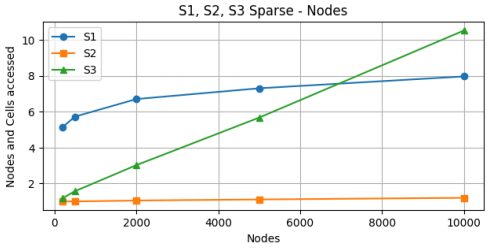
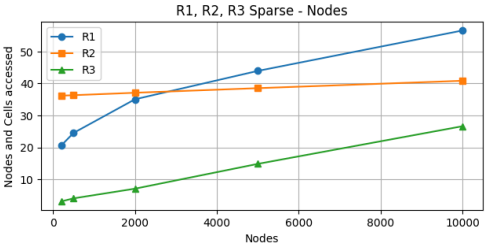
ΑΝΑΦΟΡΑ 2ου PROJECT ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ

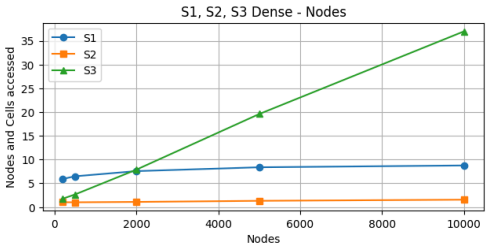
Σπύρος Πολυμένης 2023030021
Θεμιστοκλής Κόλλιας 2023030025

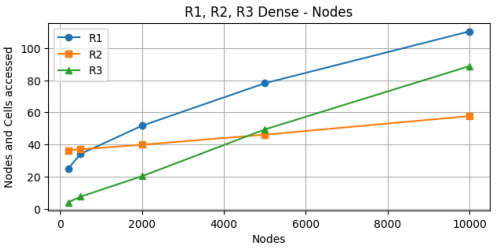
ΕΡΩΤΗΣΗ 1)

Διαγραμματα για Nodes:

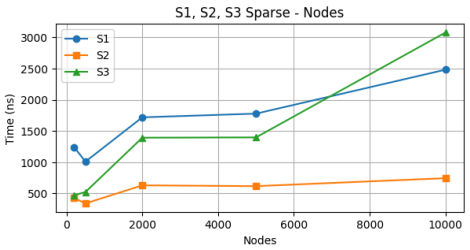


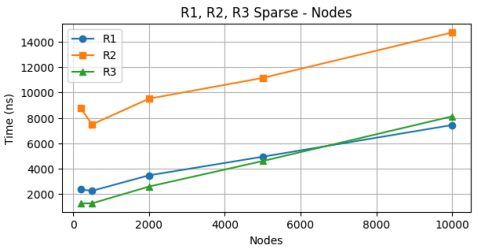


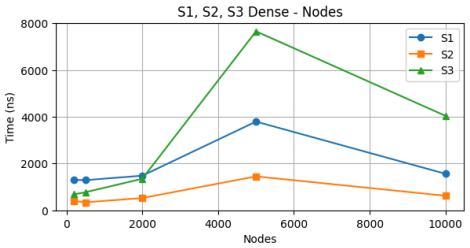


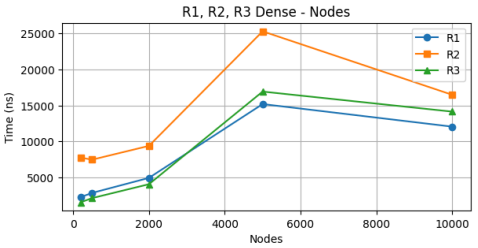


Διαγραμματα για χρονους:

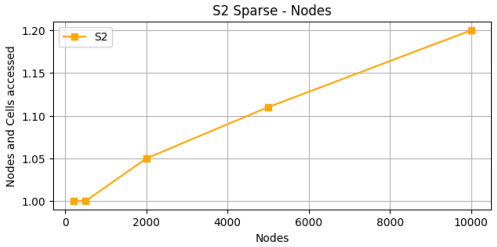


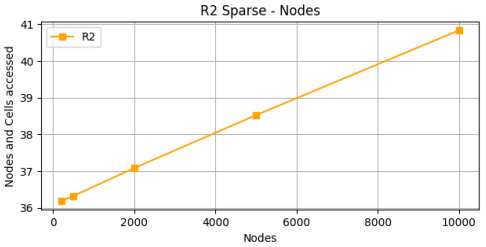


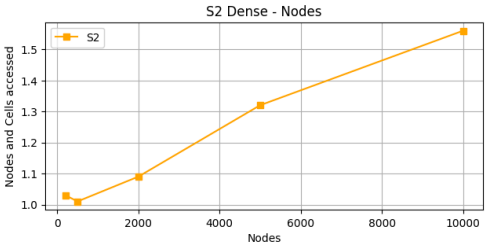


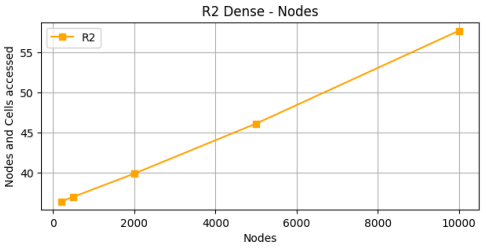


Παρακατω παρατιθεται σε ξεχωριστα διαγραμματα το SpatialHash1 (S2, R2) (Nodes - Nodes and Cells accessed) για να φανει καλυτερα η γραμμικοτητα της καμπυλης:









ΕΡΩΤΗΣΗ 2)

Οι καμπυλες επιπεδων δεν συμβαδιζουν παντα με τις καμπυλες χρονων. Αυτο συμβαινει γιατι σε καθε εκτελεση του προγραμματος εχουμε σταθερα σημεια στα οποια βρισκονται τα τερατα, οποτε ο αριθμος των κομβων θα ειναι παντα ιδιος. Αντιθετα οι χρονοι μπορει να επηρεαζονται απο πρακτικους παραγοντες, οπως οι διεργασιες του λειτουργικο συστηματος, οι οποιοι μπορει σε καποιες περιπτωσεις να καθυστερησουν τον χρονο εκτελεσης του κωδικα, ενω τα nodes που προσπελασονται θα παραμεινουν σταθερα.

ΕΡΩΤΗΣΗ 3)

Στο search (sparse & dense) το sh1 (B=4) ειναι πιο αποτελεσματικο απο το sh2 (B=32) επειδη καθε cell στο table εχει λιγοτερα nodes οποτε και κατα την προσπελαση τους ο αριθμος των nodes θα ειναι μικροτερος σε συγκριση με το sh2 οπου για να φτασουμε στο επιθυμητο node θα πρεπει να προσπελασουμε περισσοτερα nodes.

Στο range Search (sparse) το sh1 ειναι λιγοτερο αποτελεσματικο γιατι ψαχνοντας 10 θεσεις γυρω απο καθε σημειο ενω το cell που ανηκει ειναι 4*4 σημαινει οτι θα πρεπει να κανουμε access και αλλα cells ενω για το sh2 στις αν το σημειο που κανουμε το range Search δεν ειναι σε ακραιο σημειο, τοτε τα σημεια που ψαχνουμε θα βρισκονται μεσα στο ιδιο cell. (dense) ισχυει το ιδιο πραγμα μεχρι εναν συγκεκριμενο αριθμο τερατων (5000). Μετα οπως και στο search επειδη καθε cell εχει πολλες θεσεις τοτε παλι ο αριθμος των nodes που προσπελασονται θα ειναι μεγαλυτερος γιατι περναμε περισσοτερες θεσεις.

Οι χρονοι ειναι αντιστοιχοι των nodes. Δηλαδη στο search (sparse & dense) και range search(dense) οι χρονοι εκτελεσης του sh1 ειναι κατα μεσο ορο μικροτεροι απο αυτους του sh2. Ενω για το range Search(sparse) οι χρονοι του sh2 ειναι κατα μεσο ορο μικροτεροι.

ΕΡΩΤΗΣΗ 4)

Για την αναζητηση ενος θηρίου σε ένα σημείο σε κόσμο όπου ο αριθμός των θηρίων είναι πολύ μεγαλύτερος από τα παραπάνω , θα προτιμούσαμε την υλοποίηση τη Spatial Hash και συγκεκριμενα για B = 4 . Ο λόγος είναι ότι η Spatial Hash υλοποιεί μια κατακερματισμένη προσπέλαση, δηλαδή υπολογίζει σε ποιο "κελί" του hash table ανήκει το σημείο με μια απλή μαθηματική πράξη(O(1)) συνεχίζει την αναζήτηση σε μία σχετικά μικρή λίστα μόνο αν το κελί είναι γεμάτο.Ετσι θα προσπελασει στην χειροτερη περιπτωση 16 θεσεις (4*4) ,ενω για για B = 32 θα προσπελασει στην χειροτερη περιπτωση 1024 θεσεις (32*32). Αντιθέτως, η αναδρομή στην QuadTree κοστίζει υπολογιστικά όσο προσθέτονται κόμβοι στο δέντρο και αυτό μεγαλώνει σε βάθος .

ΕΡΩΤΗΣΗ 5)

Η QuadTree προσφέρει εξαιρετική απόδοση σε αναζητήσεις περιοχών, ειδικά όταν τα σημεία είναι αραιά κατανεμημένα, καθώς χωρίζει περιοχές μόνο όταν χρειάζεται . Βέβαια , όταν βρισκόμαστε στην περίπτωση όπου τα δεδομένα είναι πυκνά κατανεμημένα , η QuadTree χάνει την αποτελεσματικότητά της γιατί η χωρική προσαρμογή γίνεται υπερβολική και βαθιά. Συνεπώς, επηρεάζεται άμεσα από το εάν τα δεδομένα ακολουθούν πυκνή ή αραιή κατανομή αλλά και από τον αριθμό των σημείων αυτών. Από την άλλη, η Spatial Hash με μικρό B (π.χ. B=4) έχει πολύ γρήγορη αναζήτηση σημείων λόγω του άμεσου κατακερματισμού, αλλά απαιτεί περισσότερη μνήμη λόγω των επιπλέον κελιών και χανει σε απόδοση όταν ψάχνουμε σε περιοχές. Αντίθετα, η Spatial Hash με μεγάλο B (π.χ. B=32) χρησιμοποιεί λιγότερη μνήμη καθώς έχουμε πιο συμπαγή πίνακας hash, αλλά συγκεντρώνει πολλά σημεία ανά κελί, κάτι που υποβαθμίζει την ακρίβεια και την ταχύτητα σε range Search. Τέλος , κάτι το οποίο επηρεάζει σημαντικά την απόδοση στις λειτουργίες Spatial Hash ειναι τα collisions με διπλανα κελιά, ειδικά στις αναζητήσεις εύρους ή κοντινών σημείων, καθώς αναγκάζουν τη δομή να προσπελάσει πολλά γειτονικά κελιά. Στην περίπτωση με μικρό μέγεθος κελιού (B=4), υπάρχουν περισσότερα κελιά συνολικά, άρα περισσότερες προσπελάσεις σε κελιά και συγκρίσεις, γεγονός που επιβαρύνει τον χρόνο και τη μνήμη, αν και οι λίστες ανά κελί είναι μικρές. Αντίθετα, όταν B=32, υπάρχουν λιγότερα κελιά και λιγότερες συνολικές προσπελάσεις, αλλά κάθε κελί περιέχει πολύ περισσότερα σημεία, οπότε οι συγκρίσεις μέσα σε κάθε κελί αυξάνονται.

ΕΡΩΤΗΣΗ 6)

Το μέγιστο βάθος στο οποίο μπορεί να φτάσει ένα QuadTree , όπως γνωρίζουμε και από την θεωρία , εξαρτάται άμεσα από το μέγεθος του κόσμου στον οποίο βρισκόμαστε (K) , αλλά και από το γεγονός ότι στη συγκεκριμένη περίπτωση απαιτώ ένα το πολύ σημείο ανα κόμβο και μόνο στα φύλλα. Συγκεκριμένα , η διαδικασία αρχίζει με τον κόμβο να διαιρείται στη μέση κατα μήκος και κατά πλάτος , άρα οι διαστάσεις κάθε παιδικού τετραγώνου είναι μισές από τον γονικό κόμβο. Η διάσπαση συνεχίζεται μέχρι το τετράγωνο να φτάσει σε μέγεθος 1×1, δηλαδή δεν μπορεί να χωριστεί άλλο και περιέχει μόνο μία θέση (πιθανό σημείο). Για K=1024, εφόσον με κάθε διάσπαση η πλευρά γίνεται $\frac{K}{2^d}$, έχω μέγιστο βάθος

$dmax = log_2 1024 = 10$

ΕΡΩΤΗΣΗ 7)

Το μεγιστο μηκος της λιστας σε καθε κελι του hashtable ειναι B*B. Στην περιπτωση δηλαδη που το B ειναι 4, το μεγιστο μηκος της λιστας ειναι 16, ενω για B = 32 το μεγιστο μηκος ειναι 1024. Αυτο συμβαινει γιατι για να εχουμε το μεγιστο μηκος της λιστας θα πρεπει να εχουμε ενα τερας σε καθε node του κελιου. Οταν το κελι ειναι 4*4 αν υποθεσουμε οτι εχουμε σε καθε node ενα τερας τοτε θα εχουμε 16 τερατα. Αντιστοιχα και οταν το κελι ειναι 16*16.

ΕΡΩΤΗΣΗ 8)

Παρακάτω φαίνονται τα ονόματα τεράτων που τυπώνονται στην κονσόλα στα συγκεκριμένα δοσμένα σημεία αλλά και γύρω από το δοσμένο σημείο (900, 688) . Χρησιμοποιούνται και οι 3 υλοποιήσεις :

2e. i)

Search result for specific points using sh1 (B=4):

Found (1018, 558): Monster: Torgimok1
Found (992, 518): Monster: Neknimok3
Found (594, 646): Monster: Myrrathok3
Found (581, 836): Monster: Grimelur
Found (718, 827): Monster: Gorrezrak1
Found (633, 930): Monster: Skelrarax3

Search result for specific points using sh2 (B=32):

Found (1018, 558): Monster: Torgimok1
Found (992, 518): Monster: Neknimok3
Found (594, 646): Monster: Myrrathok3
Found (581, 836): Monster: Grimelur
Found (718, 827): Monster: Gorrezrak1
Found (633, 930): Monster: Skelrarax3

Search result for specific points using QuadTree:

Found (1018, 558): Monster: Torgimok1
Found (992, 518): Monster: Neknimok3
Found (594, 646): Monster: Myrrathok3
Found (581, 836): Monster: Grimelur
Found (718, 827): Monster: Gorrezrak1
Found (633, 930): Monster: Skelrarax3

2e. ii)

Range search result for (900, 688) using sh1 (B=4):
(891, 679): Lazginax2
(890, 687): Zulregrim2
(890, 689): Morkamar2
(892, 683): Balunir3
(893, 692): Skellonir3
(893, 698): Morzunir3
(897, 678): Thrazuthul
(899, 681): Gorzuven2
(896, 681): Narulven
(896, 685): Oggzukesh2
(909, 695): Fenzunul

Range search result for (900, 688) using sh2 (B=32):
(893, 692): Skellonir3
(893, 698): Morzunir3
(890, 689): Morkamar2
(890, 687): Zulregrim2
(892, 683): Balunir3
(891, 679): Lazginax2
(909, 695): Fenzunul
(899, 681): Gorzuven2
(896, 685): Oggzukesh2
(896, 681): Narulven
(897, 678): Thrazuthul

Range search result for (900, 688) using QuadTree:
(891, 679): Lazginax2
(892, 683): Balunir3
(890, 687): Zulregrim2
(890, 689): Morkamar2
(893, 692): Skellonir3
(893, 698): Morzunir3
(897, 678): Thrazuthul
(896, 681): Narulven
(899, 681): Gorzuven2
(896, 685): Oggzukesh2
(909, 695): Fenzunul

ΠΗΓΕΣ:

Χρησιμοποιηθηκε το GitHub Copilot για την ομοιομορφη εμφανιση αποτελεσματων στους πινακες και μας εδωσε ιδεες για την υλοποιηση των συγκεκριμενων ανζητησεων στην main και για την χρηση του `table.computeIfAbsent(index, k -> new ArrayList<>())` στην SpatialHash. Επιπλέον έγινε χρήση του ChatGpt και για βοήθεια στην σωστή και εύστοχη υλοποίηση της αναδρομής στην QuadTree καθώς και για τον λόγο που ο χειρισμός της γίνεται καλύτερα έχοντας δύο κλάσεις QuadTree και QuadTreeNode αντί για μία.