ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΟΠΤΙΚΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΤΑΞΙΝΟΜΗΣΗΣ

ΓΕΩΡΓΙΟΣ ΒΑΝΑΣΑΣ

ΕΠΙΒΛΕΠΩΝ: ΔΑΔΑΛΙΑΡΗΣ ΑΝΤΩΝΙΟΣ
ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΛΑΜΙΑ 2025

ΠΕΡΙΕΧΟΜΕΝΑ

1.ΕΙΣΑΓΩΓΗ

Η ταξινόμηση δεδομένων είναι ένα πολυσυζητημένο θέμα στον τομέα του προγραμματισμού και έχει απασχολήσει ένα ευρύ κοινό.
Οι αλγόριθμοι που μελετώνται είναι οι εξής:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Counting Sort
- Radix Sort
- Bucket Sort
- Shell Sort

Ο κάθε αλγόριθμος είναι μοναδικός, τόσο στο πως είναι γραμμένος σε κώδικα, αλλά και σε τι τύπο δεδομένων απευθύνεται, τι χώρο καταναλώνει, τι χρόνο απαιτεί και πόσο σταθερός είναι.

1.1 Σκοπός της Εργασίας

- Παρατήρηση των δέκα αλγορίθμων ταξινόμησης από κάθε όψη.
- Ανάλυση του θεωρητικού τους υπόβαθρου.
- Εξέταση στην πράξη με την εκτέλεση της εφαρμογής.
- Παρουσίαση της αποδοτικότητας των αλγορίθμων.
- Εμφανής οπτικοποίηση με ειδικό γράφημα.
- Προβολή της "αντίδρασης" κάθε αλγορίθμου σε συγκεκριμένα δεδομένα.

1.2 Δομή της Εργασίας

- Κεφάλαιο 1: Εισαγωγή στο θέμα, σκοπός και δομή.
- Κεφάλαιο 2: Θεωρητικό υπόβαθρο των αλγορίθμων ταξινόμησης.
- Κεφάλαιο 3: Τεχνολογίες και εργαλεία ανάπτυξης.
- Κεφάλαιο 4: Παρουσίαση αποτελεσμάτων και σύγκριση απόδοσης.
- Κεφάλαιο 5: Συμπεράσματα και προσωπική τοποθέτηση.

2. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ

2.1 Ανάλυση Αλγορίθμου

- Η ανάλυση αλγορίθμων εξετάζει την κατανάλωση πόρων και υπολογιστικής ισχύος.
- Οι "πόροι" αναφέρονται στον χρόνο εκτέλεσης και την μνήμη του προγράμματος.
- Δεν υπάρχει βέλτιστος αλγόριθμος για όλα τα σενάρια.
- Η εργασία επικεντρώνεται κυρίως στον χρόνο εκτέλεσης των αλγορίθμων.

2.2 Πολυπλοκότητα Χρόνου

Worst Case

- Αντίστροφα ταξινομημένος πίνακας.
- Μέγιστος αριθμός μετακινήσεων.
- Οι χρόνοι είναι μεγαλύτεροι.

Best Case

- Ταξινομημένος πίνακας.
- Πολύ χαμηλός χρόνος εκτέλεσης.
- Ελάχιστος αριθμός ενεργειών.

Average Case

- Τυχαία ταξινομημένος πίνακας.
- Πιο ρεαλιστικό σενάριο.
- Πραγματική απόδοση σε σύνολο δεδομένων.

2.3 Ρυθμός Αύξησης των Συναρτήσεων

Η ταχύτητα με την οποία αναπτύσσεται ο χρόνος εκτέλεσης ενός αλγορίθμου μέσω ασυμπτωτικής ανάλυσης:

Big-O Notation

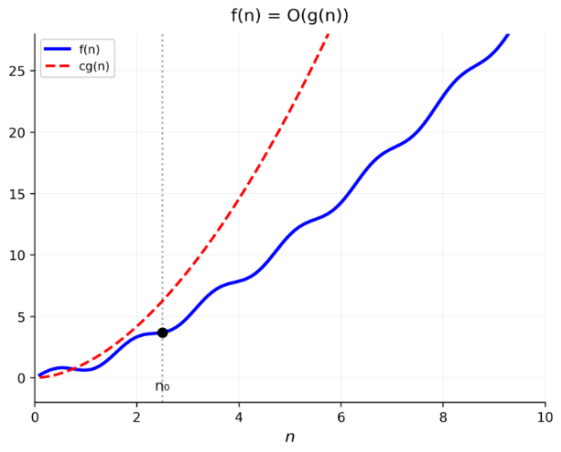
- Περιγραφή του ανώτερου ορίου μιας συνάρτησης.
- Μέγιστος αριθμός πόρων.

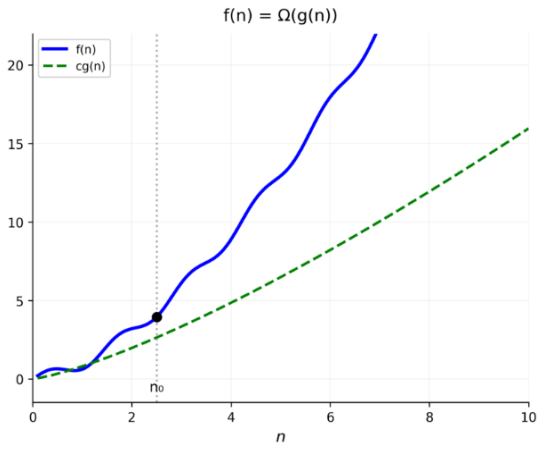
Big-Ω Notation

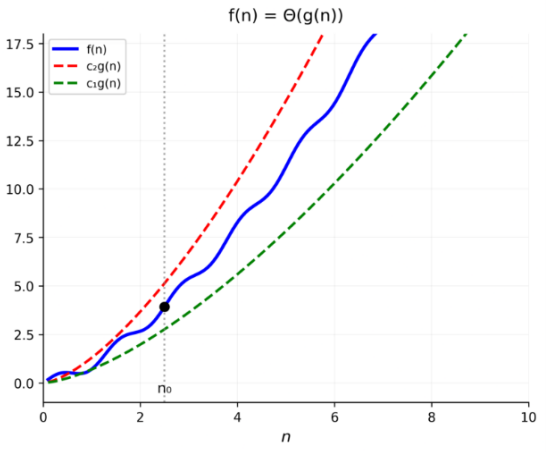
- Περιγραφή του κατώτερου ορίου μιας συνάρτησης.
- Ελάχιστος αριθμός πόρων.

Big-Θ Notation

- Περιγραφή μιας συνάρτησης με ανώτερο και κατώτερο όριο.



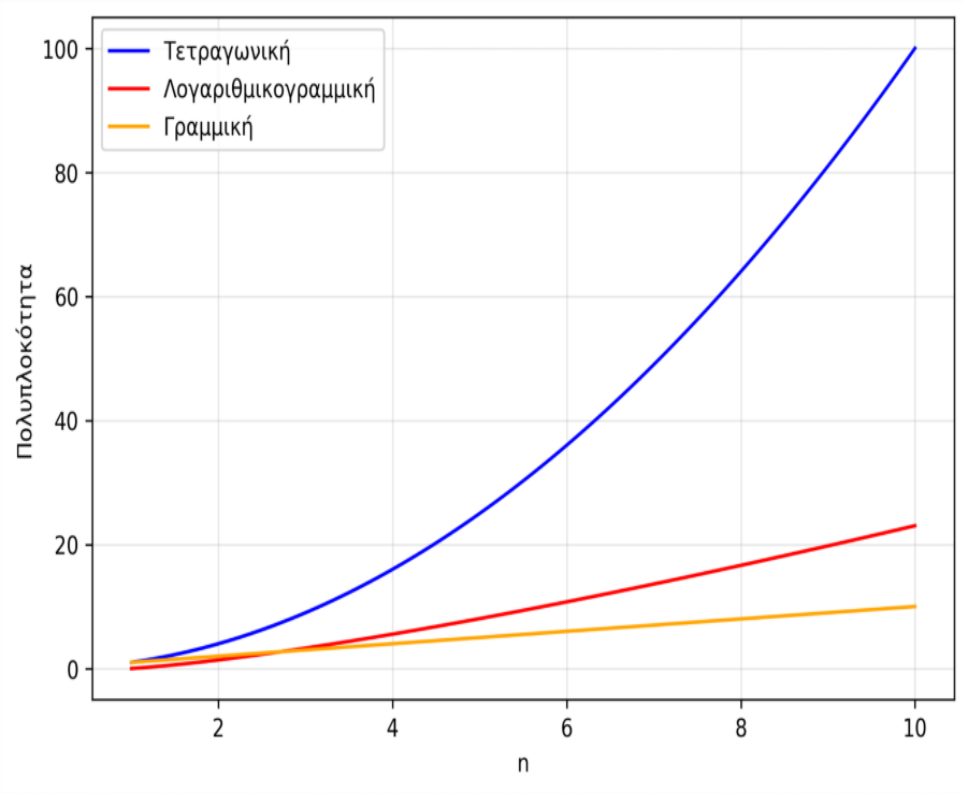




Standard Notation

Ο ρυθμός ανάπτυξης των αλγορίθμων ταξινόμησης είναι ο ρυθμός με τον οποίο αυξάνεται το κόστος ενός αλγορίθμου όσο μεγαλώνει το μέγεθος των δεδομένων που απαιτείται να ταξινομήσει.

- Γραμμικός O(n): Ο χρόνος εκτέλεσης αυξάνεται σύμφωνα με το μέγεθος των δεδομένων.
- Λογαριθμικογραμμικός O(nlogn): Ο χρόνος εκτέλεσης αυξάνεται πιο γρήγορα από τον γραμμικό, αλλά εξακολουθεί αποδοτικός.
- Τετραγωνικός $O(n^2)$: Ο χρόνος εκτέλεσης αυξάνεται πολύ πιο γρήγορα.



2.4 Χαρακτηριστικα Αλγορίθμων Ταξινόμησης

- Υπολογιστική Πολυπλοκότητα
  Πόσος χρόνος και μνήμη απαιτήθηκε από τον αλγόριθμο για την ολοκλήρωση της εργασίας.

- Πολυπλοκότητα Χώρου
  - Επί τόπου: Ο αλγόριθμος έχει σταθερό μέγεθος μνήμης O(1).
  - Εκτός τόπου: Χρησιμοποιεί μνήμη ανάλογα με τα δεδομένα του πίνακα.

- Σταθερότητα
  Η δυνατότητα διατήρησης της σχετικής σειράς των στοιχείων πίνακα όταν έχουν ίσες τιμές.

- Αναδρομικότητα
  Η ιδιότητα να καλεί ο αλγόριθμος τον εαυτό του με μικρότερες τιμές εισόδου.

- Εσωτερική και Εξωτερική Ταξινόμηση
  Η τοποθεσία αποθήκευσης των δεδομένων.

2.5 Θεωρία των Αλγορίθμων Ταξινόμησης

2.5.A Bubble Sort

Περιγραφή Λειτουργίας

- Συγκρίνει κάθε στοιχείο της λίστας με γειτονικά στοιχεία και πραγματοποιεί εναλλαγή αν δεν βρίσκονται σε σωστή σειρά.
- Για έναν πίνακα με n στοιχεία, απαιτείται να πραγματοποιηθούν n-1 συγκρίσεις στην πρώτη διέλευση.
- Το μεγαλύτερο στοιχείο του πίνακα μετακινείται σταδιακά προς το τέλος σε κάθε επανάληψη.

Απόδοση

Δεν θα χαρακτηριζόταν ιδιαίτερα καλός σε απόδοση, ειδικά σε μεγάλο όγκο δεδομένων.

Πλεονεκτήματα και Μειονεκτήματα

- Απλότητα και ευκολία υλοποίησης.
- Όχι κατάλληλος για μεγάλους πίνακες με πολλά δεδομένα πρός ταξινόμηση.

2.5.B Selection Sort

Περιγραφή Λειτουργίας

- Βρίσκει το μικρότερο και μεγαλύτερο στοιχείο του μη ταξινομημένου πίνακα και το ανταλλάσσει με το πρώτο στοιχείο του ταξινομημένου πίνακα.
- Τα στοιχεία που ταξινομούνται μεταφέρονται προς τα πάνω στον πίνακα και τα υπόλοιπα παραμένουν κάτω αταξινόμητα.
- Είναι αλγόριθμος επί τόπου, καθώς απαιτεί σταθερή ποσότητα μνήμης.

Απόδοση

- Απαιτούνται n-1 επαναλήψεις για πίνακα με n στοιχεία, με $O(n^2)$ συγκρίσεις και n-1 ανταλλαγές.
- Δεν είναι κατάλληλος για μεγάλους πίνακες με πολλά δεδομένα.

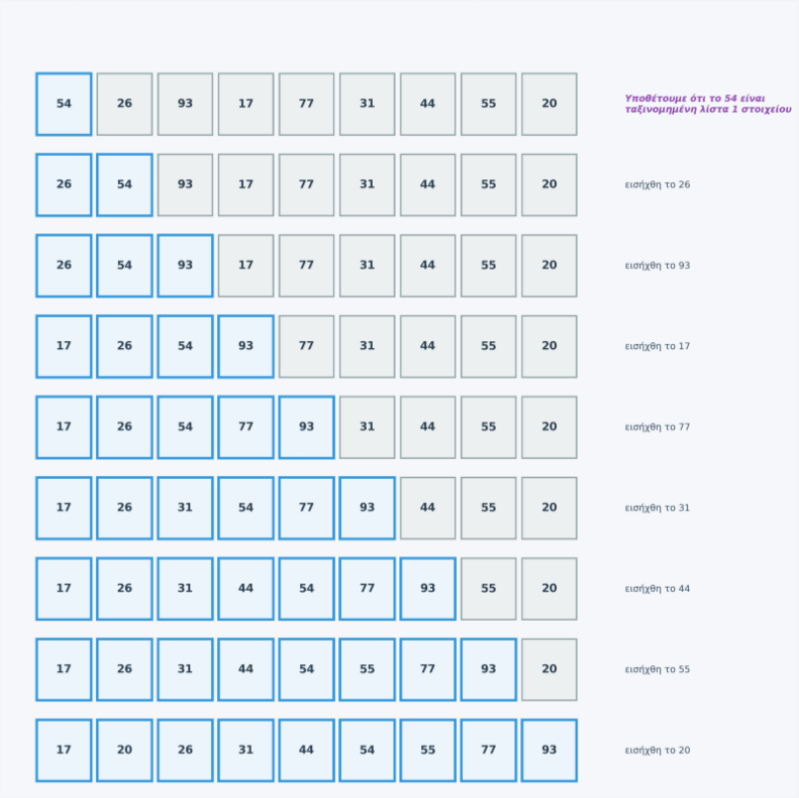
Πλεονεκτήματα και Μειονεκτήματα

- Είναι εύκολος και απλός στην υλοποίηση.
- Όχι ιδιαίτερα αποδοτικός σε μεγάλα μεγέθη.

2.5.Γ Insertion Sort

Περιγραφή Λειτουργίας

- Λειτουργεί με εισαγωγή κάθε στοιχείου στην κατάλληλη θέση της τελικής ταξινομημένης λίστας, συγκρίνοντάς το με τα γειτονικά του στοιχεία.
- Σε κάθε επανάληψη το ταξινομημένο τμήμα αυξάνεται κατά ένα στοιχείο μέχρι να ταξινομηθεί πλήρως ο πίνακας.
- Είναι αλγόριθμος επί τόπου και απαιτεί συγκεκριμένο χώρο για να λειτουργήσει.



Απόδοση

Είναι επαναληπτικός αλγόριθμος και απαιτούνται n-1 επαναλήψεις για n στοιχεία στον πίνακα.

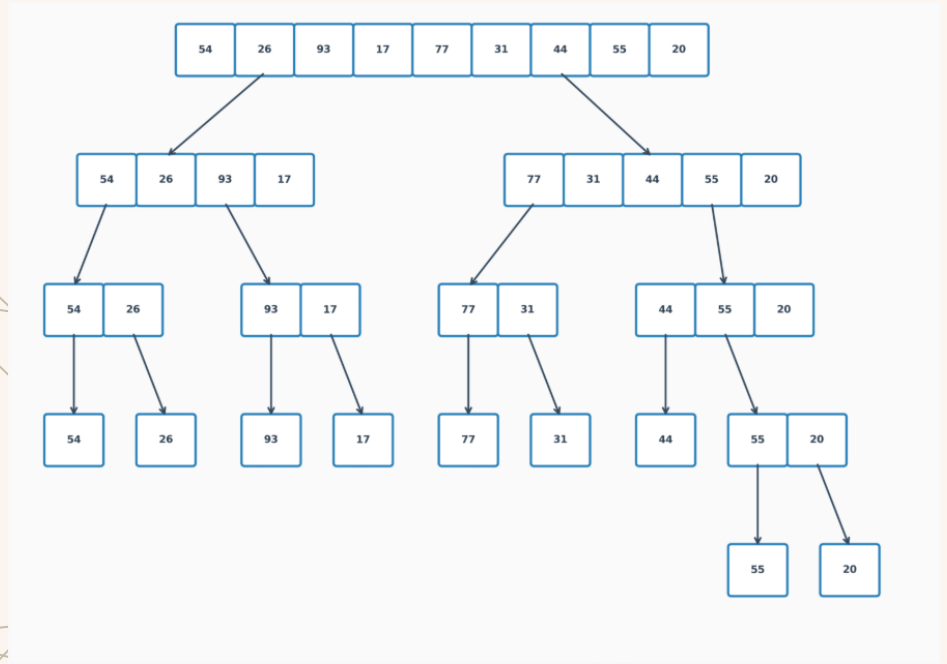
Πλεονεκτήματα και Μειονεκτήματα

- Απλότητα και υψηλή αποδοτικότητα σε μικρούς πίνακες.
- Αργός σε μεγάλα δεδομένα.

2.5.Δ Merge Sort

Περιγραφή Λειτουργίας

- Ο Merge Sort είναι ένας αλγόριθμος που χρησιμοποιεί τη μέθοδο "διαίρει και βασίλευε".
- Ο πίνακας χωρίζεται σε δύο υποπίνακες και κάθε υποπίνακας ταξινομείται αναδρομικά.
- Όταν και οι δύο υποπίνακες είναι ταξινομημένοι, συγχωνεύονται σε έναν ενιαίο, ταξινομημένο πίνακα.



Απόδοση

Αποδοτικός αλγόριθμος σε περιπτώσεις με επαρκή μνήμη στο σύστημα ώστε να λειτουργήσει απροβλημάτιστα.

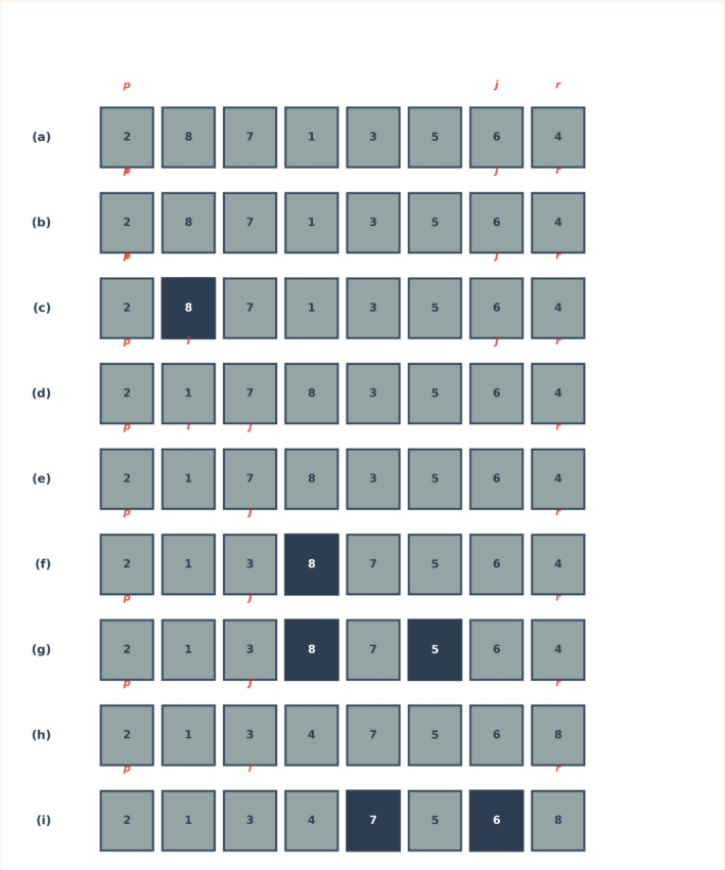
Πλεονεκτήματα και Μειονεκτήματα

- Κατάλληλος για δεδομένα που είναι αποθηκευμένα σε συνδεδεμένη λίστα.
- Απαιτεί αρκετή μνήμη και δεν συνιστάται για μικρότερους πίνακες λόγω της αναδρομικότητάς του.

2.5.E Quick Sort

Περιγραφή Λειτουργίας

- Ο Quick Sort είναι ένας αλγόριθμος που χρησιμοποιεί τη μέθοδο "διαίρει και βασίλευε".
- Επιλέγεται ένα στοιχείο ως "pivot" (άξονας) και αναδιοργανώνεται η λίστα έτσι ώστε τα μικρότερα στοιχεία να βρίσκονται αριστερά του pivot και τα μεγαλύτερα δεξιά.
- Εφαρμόζεται αναδρομικά η ίδια διαδικασία στις δύο υπολίστες, μέχρι κάθε υπολίστα να έχει μόνο ένα στοιχείο.



Απόδοση

Ο Quick Sort είναι από τους ταχύτερους αλγορίθμους ταξινόμησης. Ωστόσο όταν το pivot επιλέγεται σταθερά ως το πρώτο ή το τελευταίο στοιχείο, ένας ήδη ταξινομημένος πίνακας οδηγεί στη χειρότερη περίπτωση λειτουργίας.

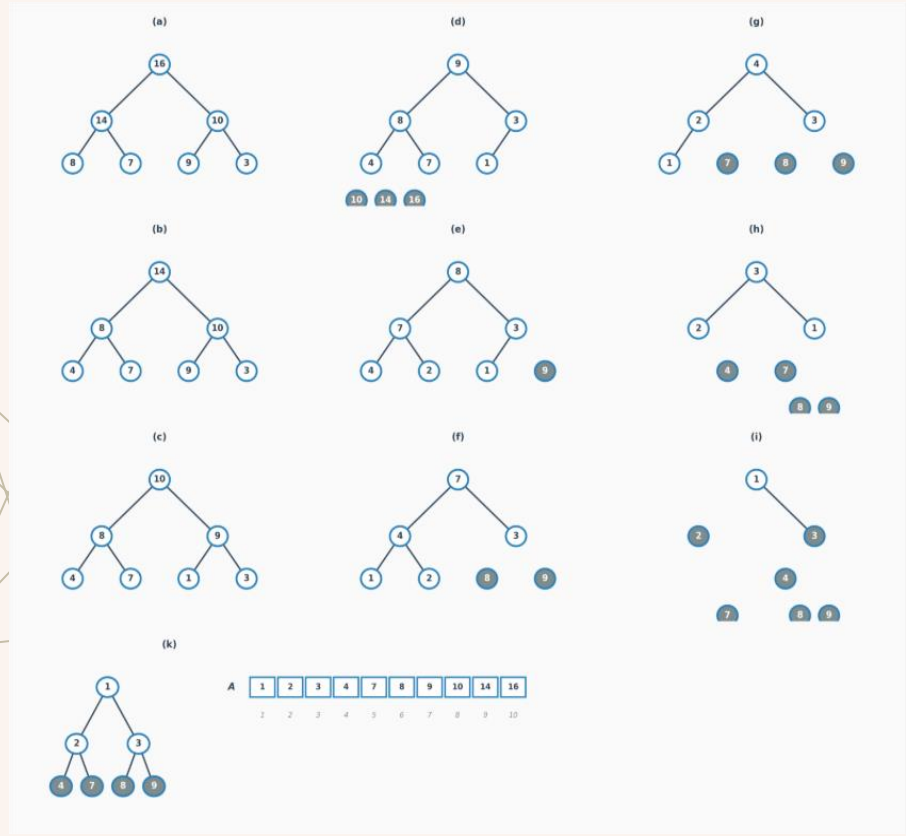
Πλεονεκτήματα και Μειονεκτήματα

- Πολύ μεγάλη ταχύτητα και αποδοτικότητα σε μεγάλους πίνακες, αλλά η αποδοτικότητα μειώνεται αν τα δεδομένα είναι ήδη ταξινομημένα ή ίσα.
- Καταλαμβάνει αρκετό χώρο σε μεγάλα δεδομένα

2.5.Z Heap Sort

Περιγραφή Λειτουργίας

- Ο Heap Sort χρησιμοποιεί μια δομή δεδομένων που ονομάζεται σωρός.
- Μετατρέπεται η λίστα σε μία μέγιστου σωρού, όπου το μεγαλύτερο στοιχείο βρίσκεται στη ρίζα, εξάγεται επαναληπτικά και τοποθετείται στο τέλος της λίστας και αλλάζει ο σωρός.
- Η διαδικασία συνεχίζεται μέχρι όλα τα στοιχεία του πίνακα να είναι ταξινομημένα.



Απόδοση

Αποδοτικός για ταξινόμηση μεγάλων δεδομένων, λόγω της αναδρομικότητάς και απαιτεί σταθερή ποσότητα πρόσθετου χώρου μνήμης.

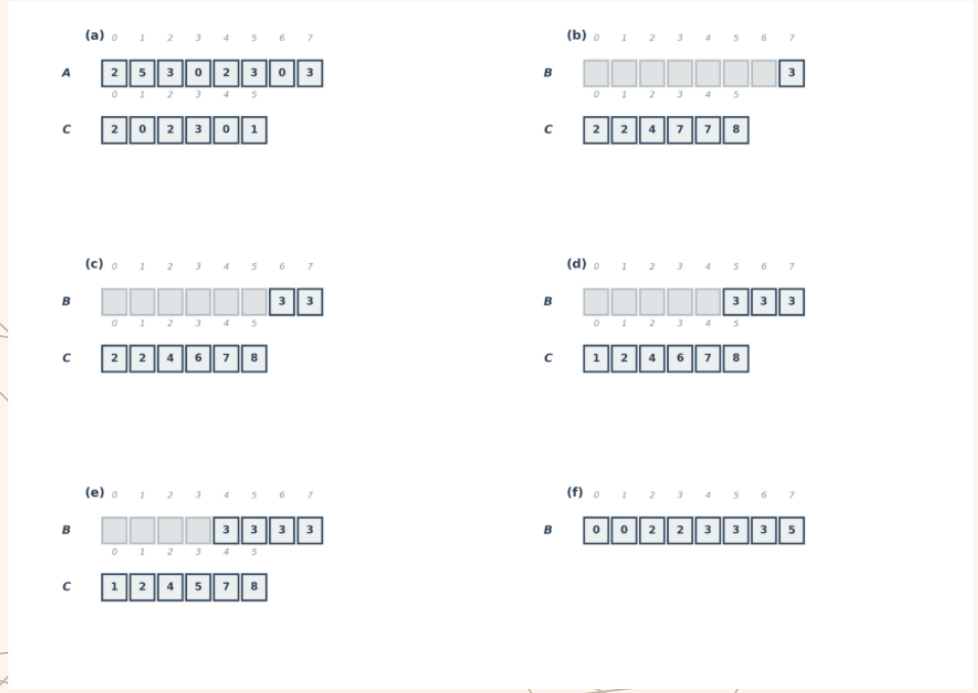
Πλεονεκτήματα και Μειονεκτήματα

- Είναι γρήγορος σε μεγάλους πίνακες με πολλά δεδομένα λόγω της αναδρομικότητάς του.
- Δεν είναι ιδιαίτερα γρήγορος σε συνδεδεμένες λίστες γιατί δυσκολεύεται να μετατρέψει συνδεδεμένες λίστες σε δομή σωρού.

2.5.Η Counting Sort

Περιγραφή Λειτουργίας

- Ο Counting Sort είναι μη συγκριτικός αλγόριθμος που λειτουργεί μετρώντας πόσες φορές εμφανίζεται κάθε διαφορετική τιμή στη λίστα.
- Δημιουργείται ένας πίνακας καταμέτρησης όπου κάθε θέση αντιστοιχεί σε μια τιμή και αποθηκεύονται οι φορές εμφάνισης.
- Χρησιμοποιείται αυτός ο πίνακας για να τοποθετηθεί κάθε στοιχείο στη σωστή θέση στην ταξινομημένη λίστα.



Απόδοση

Αποτελεί σταθερό αλγόριθμο που διατηρεί τη σχετική σειρά των στοιχείων με ίσες τιμές και είναι εύκολα υλοποιήσιμος.

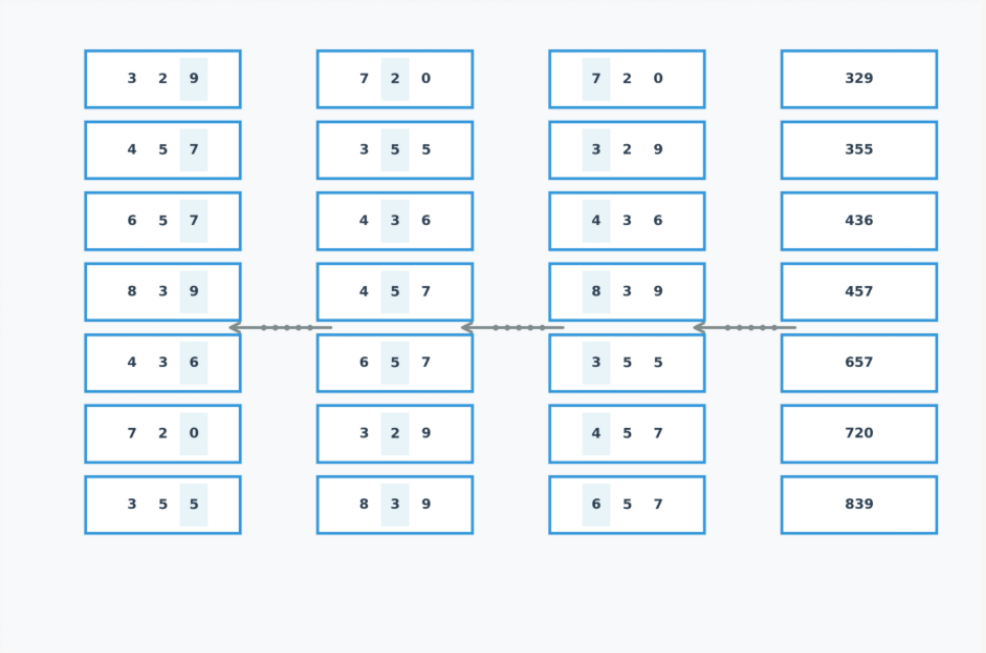
Πλεονεκτήματα και Μειονεκτήματα

- Χρησιμοποιεί τις τιμές κλειδιών ως δείκτες σε πίνακα διατηρώντας τη σχετική σειρά των στοιχείων με ίσα κλειδιά.
- Δεν είναι κατάλληλος για μεγάλα σύνολα δεδομένων και συμβολοσειρές.

2.5.Θ Radix Sort

Περιγραφή Λειτουργίας

- Δεν χρησιμοποιεί σύγκριση στοιχείων για την ταξινόμηση πίνακα, λειτουργεί ταξινομώντας τα στοιχεία με βάση τα κλειδιά τους.
- Ταξινομείται κάθε ψηφίο από τα στοιχεία εισόδου ξεκινώντας από το λιγότερο σημαντικό μέχρι το πιο σημαντικό.
- Αποτελεί έναν σταθερό αλγόριθμο καθώς διατηρείται η σχετική σειρά των στοιχείων με ίσα κλειδιά.



Απόδοση

Είναι ένας σταθερός αλγόριθμος όταν ο αριθμός ψηφίων δεν αλλάζει.

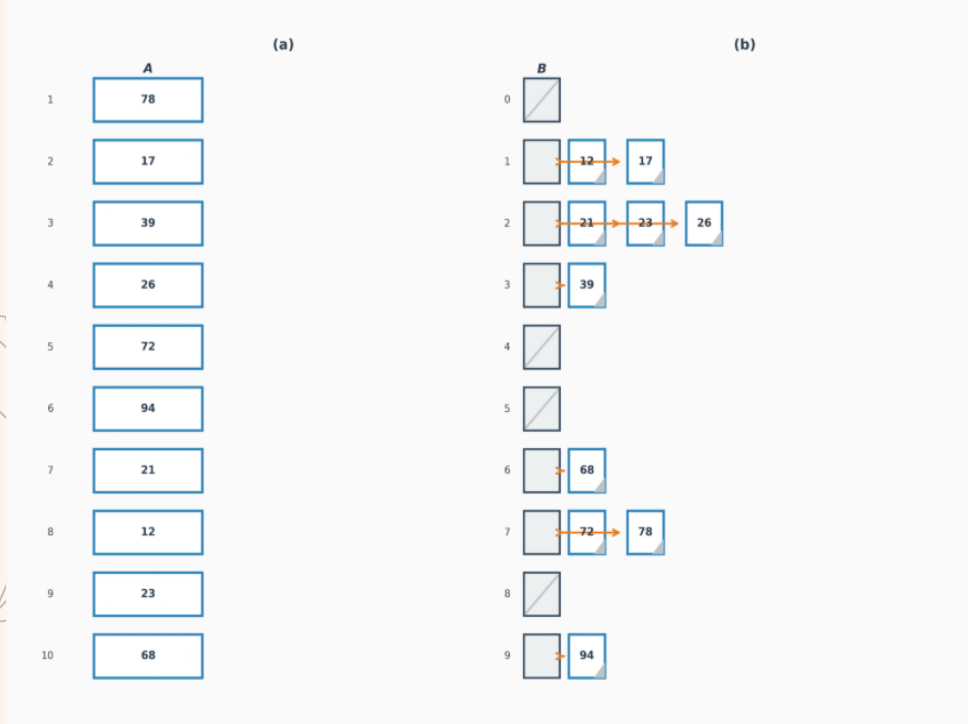
Πλεονεκτήματα και Μειονεκτήματα

- Η απόδοσή του δεν επηρεάζεται από τον τύπο και το μέγεθος των δεδομένων εισόδου.
- Είναι αρκετά δύσκολος στην υλοποίησή του, δεν είναι τόσο ευέλικτος και έχει μεγάλες απαιτήσεις στην κατανάλωση χώρου.

2.5.I Bucket Sort

Περιγραφή Λειτουργίας

- Λειτουργεί χωρίς τη χρήση συγκρίσεων και βασίζεται στη διανομή των στοιχείων του πίνακα σε buckets με βάση το εύρος των τιμών τους.
- Κάθε bucket ταξινομείται ξεχωριστά, με τη χρήση κάποιου βοηθητικού αλγορίθμου ταξινόμησης.
- Τα ταξινομημένα buckets συγχωνεύονται για την παραγωγή του τελικού ταξινομημένου πίνακα.



Απόδοση

Αποδοτικός όταν τα δεδομένα είναι ομοιόμορφα κατανεμημένα.

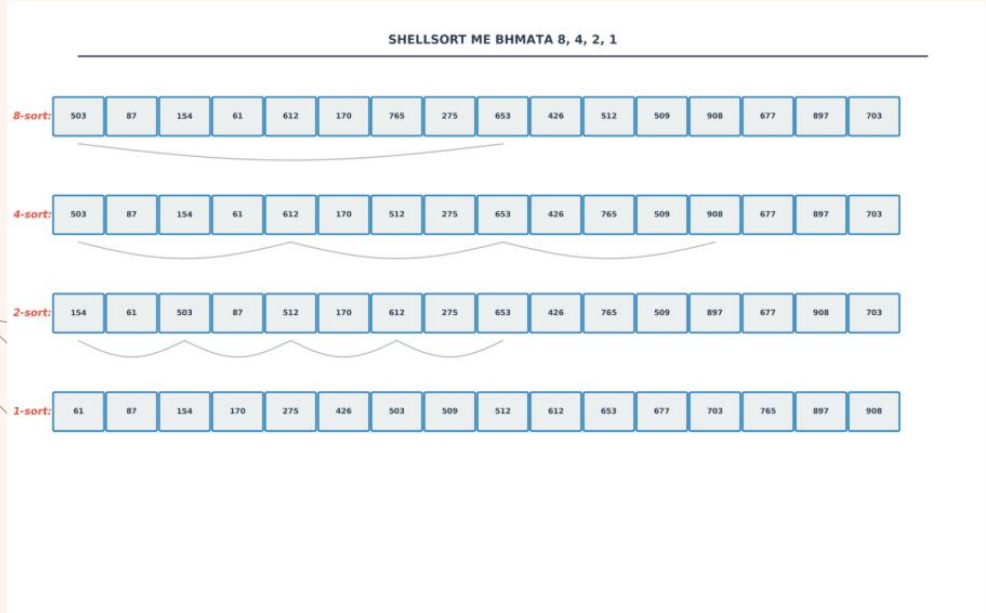
Πλεονεκτήματα και Μειονεκτήματα

- Ιδιαίτερα γρήγορος όταν τα δεδομένα είναι ομοιόμορφα κατανεμημένα.
- Μη αποδοτικός σε μεγάλους πίνακες λόγω του βοηθητικού αλγορίθμου και απαιτεί σημαντική ποσότητα μνήμης για την αποθήκευση των buckets.

2.5.K Shell Sort

Περιγραφή Λειτουργίας

- Αντί για σύγκριση και εναλλαγή γειτονικών στοιχείων, συγκρίνονται στοιχεία που βρίσκονται σε συγκεκριμένη απόσταση μεταξύ τους.
- Η απόσταση μειώνεται σταδιακά όσο προχωρά η διαδικασία ταξινόμησης.
- Πραγματοποιείται μετακίνηση απομακρυσμένων στοιχείων και στη συνέχεια με τη βοήθεια του Insertion Sort να ολοκληρωθεί η ταξινόμηση.



Απόδοση

- Η απόδοση εξαρτάται από τα βήματα που κάνει ώστε να ταξινομήσει τον πίνακα.
- Ταξινομεί επί τόπου και καταλαμβάνει μικρό χώρο μνήμης.

Πλεονεκτήματα και Μειονεκτήματα

- Γρήγορος αλγόριθμος, απλός σε υλοποίηση με μικρή κατανάλωση μνήμης και αποδοτικός.
- Εξάρτηση της απόδοσης από την επιλογή των βημάτων και ασταθής λόγω πιθανής αλλαγής σειράς ίσων στοιχείων.

3. ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

Παρουσιάζεται μία εφαρμογή οπτικοποίησης αλγορίθμων ταξινόμησης για πίνακα ακεραίων γραμμένη σε Python. Σκοπός ήταν η κατανόηση σε βάθος των λειτουργιών των αλγορίθμων μέσω γραφικής αναπαράστασης και μέτρηση χρόνου για τον κάθε αλγόριθμο ξεχωριστά.

Πλεονεκτήματα της Γλώσσας

- Απλή σύνταξη που επιτρέπει την υλοποίηση αλγορίθμων με σαφή τρόπο και εύκολη συντήρηση.
- Πλούσια συλλογή βιβλιοθηκών (Tkinter για γραφική διεπαφή, Matplotlib για οπτικοποίηση).
- Δυναμική τυποποίηση (εστίαση στους αλγορίθμους χωρίς δήλωση τύπων δεδομένων).
- Αυτόματη διαχείριση μνήμης που μειώνει τις πιθανότητες σφάλματος.
- Διερμηνευόμενη γλώσσα (γρήγορη εκτέλεση χωρίς μεταγλωττιστή).

3.2 Αρχιτεκτονική Συστήματος

3.2.Α Επίπεδο Αλγορίθμων

- Όλοι οι αλγόριθμοι αποθηκεύονται σε dictionary "ALGORITHMS", όπου κάθε κλειδί αντιστοιχεί στο όνομα του αλγορίθμου και η τιμή στην αντίστοιχη συνάρτηση.
- Εύκολη επανάληψη όλων των αλγορίθμων και προσθήκη νέων.
- Κάθε αλγόριθμος έχει την ίδια δομή: λαμβάνει τον πίνακα χρήστη, καταγράφει τα βήματα, πραγματοποιεί την ταξινόμηση και επιστρέφει τον ταξινομημένο πίνακα.

3.2.Β Επίπεδο Παρουσίασης

- Κλάση GUI για την αλληλεπίδραση με τον χρήστη και παροχή περιβάλλοντος.
- Ο χρήστης εισάγει το μέγεθος του πίνακα, ελάχιστη και μέγιστη τιμή στοιχείων, επιλογή διπλότυπων αριθμών και τύπο ταξινόμησης (αύξουσα, τυχαία, φθίνουσα).
- Buttons για την εκτέλεση του κώδικα και την εμφάνιση του γραφήματος.

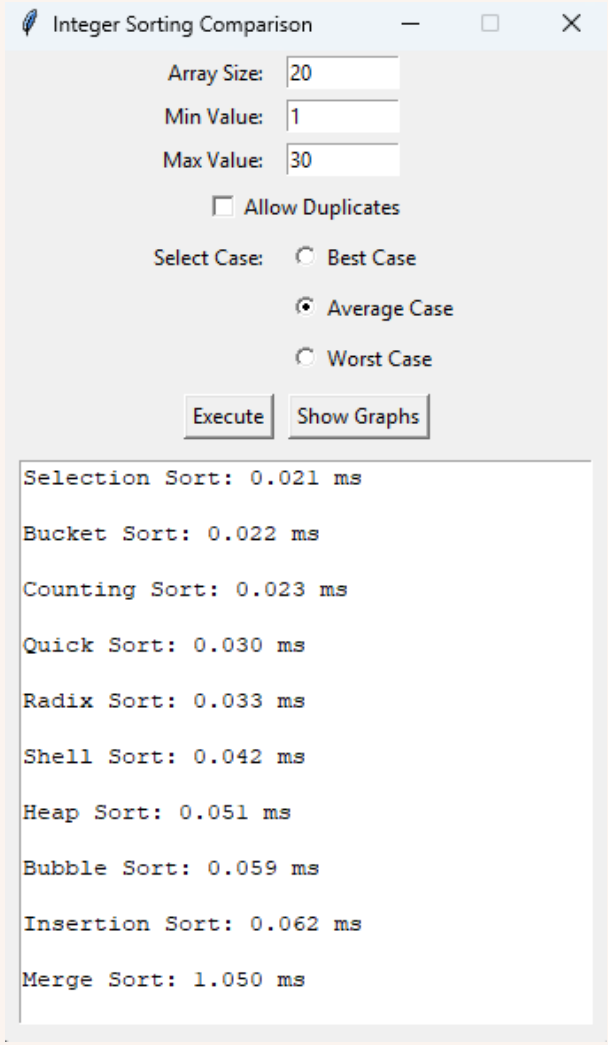
3.2.Γ Επίπεδο Οπτικοποίησης

- Κλάση Graph για δυνατότητα εξαγωγής της γραφικής αναπαράστασης κάθε αλγορίθμου ταξινόμησης.
- Γράφημα για την παρακολούθηση της πορείας του αλγορίθμου από τον αρχικό πίνακα μέχρι την πλήρη ταξινόμηση.

3.3 Τεχνολογίες και Βιβλιοθήκες

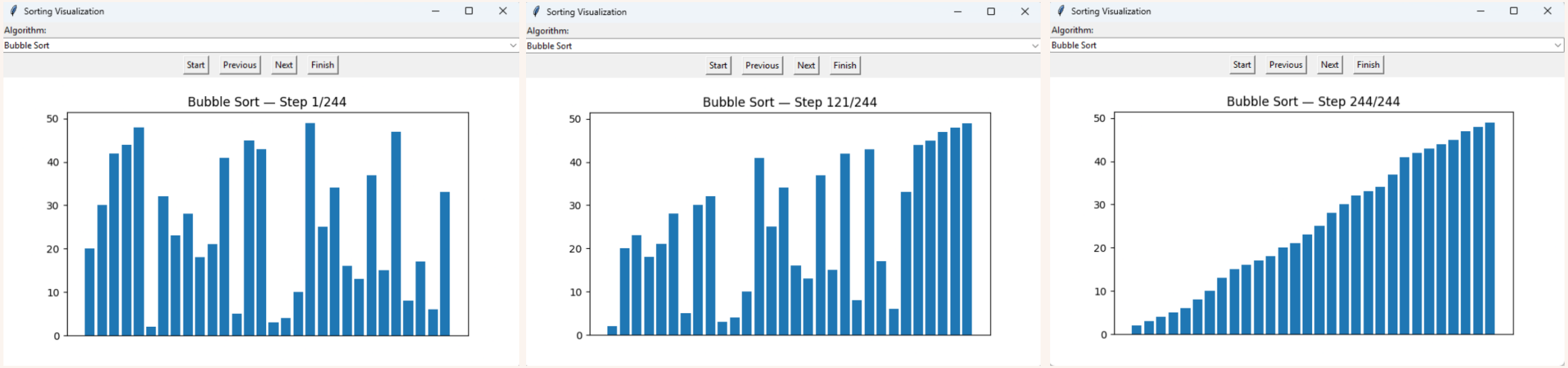
3.3.Α Βιβλιοθήκη για τη Γραφική Διεπαφή

- Για το επίπεδο παρουσίασης χρησιμοποιείται η βιβλιοθήκη Tkinter, η οποία είναι ενσωματωμένη στη γλώσσα Python.
- Το Tkinter αποτελεί την προεπιλεγμένη βιβλιοθήκη για γραφική διεπαφή (GUI).
- Δεν απαιτεί εξωτερική εγκατάσταση, γεγονός που την καθιστά βολική για εφαρμογές που απαιτούν φορητότητα.
- Διαθέτει μεγάλη γκάμα γραφικών στοιχείων: Label, Entry, Button, Checkbutton, Radiobutton, Text, Combobox και Frame.
- Υποστηρίζει διαχειριστές διάταξης (pack, grid, place) για την τοποθέτηση των γραφικών στοιχείων στο παράθυρο της εφαρμογής.



3.3.Β Βιβλιοθήκη για την Οπτικοποίηση

- Για το επίπεδο οπτικοποίησης χρησιμοποιείται η βιβλιοθήκη Matplotlib.
- Είναι ανοιχτού κώδικα με συντήρηση από μεγάλη κοινότητα προγραμματιστών.
- Υποστηρίζει πολλούς τύπους διαγραμμάτων: γραμμικά, ραβδογράμματα, ιστογράμματα, διαγράμματα διασποράς, πίτας, τρισδιάστατα και άλλα.
- Τα διαγράμματα εξάγονται σε διάφορες μορφές (PNG, PDF, SVG, EPS).
- Στην εφαρμογή χρησιμοποιείται για δημιουργία ραβδογραμμάτων που δείχνουν την κατάσταση του πίνακα σε κάθε βήμα ταξινόμησης, με κάθε ράβδο να αντιστοιχεί σε στοιχείο του πίνακα.



3.3.Γ Πρόσθετες Βιβλιοθήκες

Εκτός από Tkinter και Matplotlib, η εφαρμογή χρησιμοποιεί τις βιβλιοθήκες Time και Random.

Βιβλιοθήκη Time

- Παρέχει συναρτήσεις για εργασία με χρόνο και χρονομέτρηση.
- Στην εφαρμογή, χρησιμοποιείται η time.perf_counter() για καταγραφή του χρόνου εκτέλεσης κάθε αλγορίθμου με υψηλή ακρίβεια.
- Τα αποτελέσματα εμφανίζονται ταξινομημένα με βάση τον χρόνο εκτέλεσης.

Βιβλιοθήκη Random

- Παρέχει συναρτήσεις για δημιουργία τυχαίων αριθμών και τυχαία επιλογή ή ανακατάταξη στοιχείων.
- Χρησιμοποιείται random.randint() όταν επιτρέπονται διπλότυπα και random.sample() όταν δεν επιτρέπονται.
- Η random.shuffle() χρησιμοποιείται για την τυχαία ανακατάταξη του πίνακα στο Average Case.

3.4 Λειτουργίες και Υλοποίηση

3.4.Α Σύστημα Καταγραφής Βημάτων

- Χρησιμοποιείται συνάρτηση για την παρακολούθηση κάθε αλγορίθμου ταξινόμησης και την καταγραφή όλων των ενδιάμεσων καταστάσεων.
- Γίνεται σύγκριση της τρέχουσας κατάστασης με την προηγούμενη και αποθηκεύεται το στιγμιότυπο μόνο σε περίπτωση αλλαγής στοιχείου.
- Τα δεδομένα αποθηκεύονται σε λίστα, όπου κάθε στοιχείο περιέχει αντίγραφο της τρέχουσας κατάστασης του πίνακα.
- Κάθε αλγόριθμος καταγράφει την αρχική κατάσταση, την πρόοδο μετά από κάθε αλλαγή και τον τελικό ταξινομημένο πίνακα.

3.4.Β Διαχείριση και Επαλήθευση Εισόδου

- Πραγματοποιείται έλεγχος της εγκυρότητας των στοιχείων που εισάγει ο χρήστης.
- Ελέγχεται ότι ο χρήστης εισάγει ακέραιους αριθμούς απο 1 μέχρι 500.
- Ελέγχεται ότι η μέγιστη τιμή είναι μεγαλύτερη από την ελάχιστη και ότι το εύρος επαρκεί για δημιουργία πίνακα χωρίς διπλότυπα (όταν δεν επιτρέπονται).
- Σε περίπτωση λάθους, εμφανίζεται κατάλληλο μήνυμα χωρίς να χρειάζεται επανεκκίνηση της εφαρμογής.

3.4.Γ Δημιουργία Δεδομένων Εισόδου

Δημιουργούνται τρεις τύποι δεδομένων για αξιολόγηση των αλγορίθμων σε διαφορετικές συνθήκες:

- Best Case: Πίνακας ήδη ταξινομημένος (καλύτερη περίπτωση).
- Average Case: Τυχαία σειρά στοιχείων (πιο ρεαλιστικό σενάριο).
- Worst Case: Αντίστροφα ταξινομημένος πίνακας (μέγιστη πολυπλοκότητα).

3.4.Δ Διαδοχική Εκτέλεση Αλγορίθμων

- Η διαδοχική εκτέλεση γίνεται με αναδρομική μέθοδο για ανταποκρίσιμο GUI.
- Κάθε αλγόριθμος παίρνει τα ίδια δεδομένα για να υπάρχει δίκαιη σύγκριση.
- Αποτελέσματα εμφανίζονται σε μορφή πίνακα και γραφήματος με την ολοκλήρωση.

3.4.Ε Σύστημα Χρονομέτρησης

- Χρησιμοποιείται η time.perf_counter() για μεγαλύτερη ακρίβεια.
- Ο χρόνος αποθηκεύεται σε milliseconds και εμφανίζεται σε αύξουσα σειρά.

3.4.Ζ Σύστημα Οπτικοποίησης

- Ανοίγει με το κουμπί "Show Graph" και υπάρχει επιλογή αλγορίθμου με Combobox.
- Τα βήματα εναλλάσσουν δυναμικά ανάλογα με το μέγεθος του πίνακα.
- Πλοήγηση μεταξύ βημάτων με τα κουμπιά "Previous" και "Next".

4. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

4.2 Δοκιμή σε Μικρό Πίνακα

- Σε μικρό πίνακα 20 στοιχείων οι χρόνοι εκτέλεσης όλων των αλγορίθμων είναι πολύ μικροί.
- Οι αλγόριθμοι $O(n^2)$ (Bubble Sort, Selection Sort, Insertion Sort) έχουν πολύ καλή απόδοση σε μικρό πίνακα.
- Οι αλγόριθμοι O(nlogn) (Merge Sort, Quick Sort, Heap Sort, Shell Sort) έχουν καλύτερη θεωρητική απόδοση, αλλά το πλεονέκτημα δεν είναι εμφανές σε μικρό πίνακα.

Αλγόριθμος	Best Case	Average Case	Worst Case
Insertion Sort	0.007	0.034	0.977
Selection Sort	0.012	0.019	0.020
Bubble Sort	0.012	0.033	0.074
Bucket Sort	0.013	0.018	0.026
Counting Sort	0.014	0.021	0.023
Shell Sort	0.015	0.031	0.039
Quick Sort	0.021	0.026	0.032
Radix Sort	0.024	0.025	0.030
Merge Sort	0.034	0.049	0.060
Heap Sort	0.045	0.039	0.076

- Οι αλγόριθμοι O(n) (Counting Sort, RadixSort, Bucket Sort) δεν εμφανίζουν μεγάλη διαφορά στον μικρό πίνακα.
- Συμπερασματικά, σε μικρό πίνακα όλοι οι αλγόριθμοι φαίνονται γρήγοροι.

4.3 Δοκιμή σε Μεσαίο Πίνακα

- Στον πίνακα μεσαίου μεγέθους παρατηρούνται πιο έντονες διαφορές μεταξύ των αλγορίθμων.
- Οι αλγόριθμοι με $O(n^2)$ πολυπλοκότητα δυσκολεύονται με 100 στοιχεία.
- Οι αλγόριθμοι O(nlogn) έχουν μεγάλο πλεονέκτημα σε σχέση με τους $O(n^2)$.
- Οι O(n) αλγόριθμοι ξεχωρίζουν καθαρά και είναι εξαιρετικά γρήγοροι.

Αλγόριθμος	Best Case	Average Case	Worst Case
Insertion Sort	0.032	1.755	2.852
Bucket Sort	0.052	0.069	0.066
Counting Sort	0.053	0.072	0.072
Radix Sort	0.088	0.082	0.078
Shell Sort	0.134	0.443	0.383
Selection Sort	0.135	0.187	0.193
Bubble Sort	0.165	6.765	2.625
Merge Sort	0.247	0.728	0.873
Quick Sort	0.310	0.237	0.258
Heap Sort	1.461	0.434	0.366

4.4 Δοκιμή σε Μεγάλο Πίνακα

- Στον πίνακα μεγάλου μεγέθους η εικόνα της πολυπλοκότητας είναι απόλυτα ξεκάθαρη.
- Οι αλγόριθμοι $O(n^2)$ πολυπλοκότητας έχουν πολύ μεγάλο χρόνο εκτέλεσης.
- Οι αλγόριθμοι O(nlogn) έχουν πλεονέκτημα και παραμένουν σταθεροί και αποδοτικοί.
- Οι O(n) ξεχωρίζουν ως βέλτιστοι, με ελάχιστους χρόνους ανεξαρτήτως μεγέθους δεδομένων.

Αλγόριθμος	Best Case	Average Case	Worst Case
Insertion Sort	0.282	246.630	475.456
Radix Sort	0.373	0.297	0.451
Bucket Sort	0.418	1.191	1.254
Counting Sort	0.679	1.150	1.343
Shell Sort	2.221	17.392	12.463
Merge Sort	2.743	12.365	15.324
Selection Sort	3.282	7.019	3.948
Bubble Sort	4.081	169.782	370.722
Quick Sort	5.278	5.783	4.716
Heap Sort	9.255	10.917	10.726

4.5 Συγκριτική Ανάλυση

- Τα πειραματικά αποτελέσματα του Average Case συνδέονται με τη θεωρητική πολυπλοκότητα.
- Οι αλγόριθμοι $O(n^2)$ παρουσιάζουν μεγάλη αύξηση χρόνου εκτέλεσης.
- Οι αλγόριθμοι O(nlogn) επιβεβαιώνουν τη θεωρία, με τον Quick Sort ως τον ταχύτερο της κατηγορίας.
- Οι αλγόριθμοι O(n) είναι οι πιο γρήγοροι σε ακέραιους με περιορισμένο εύρος με τον Radix Sort να έχει την καλύτερη συνολική απόδοση σε μεγάλα μεγέθη.
- Σε μικρό πίνακα η θεωρητική πολυπλοκότητα δεν δείχνει την πραγματική απόδοση.
- Σε μεγάλο μέγεθος πίνακα η θεωρητική πολυπλοκότητα γίνεται ο κύριος παράγοντας και επαληθεύεται πλήρως.

Αλγόριθμος	20 στοιχεία (ms)	100 στοιχεία (ms)	500 στοιχεία (ms)
Bucket Sort	0.018	0.069	1.191
Selection Sort	0.019	0.187	7.019
Counting Sort	0.021	0.072	1.150
Radix Sort	0.025	0.082	0.297
Quick Sort	0.026	0.237	5.783
Shell Sort	0.031	0.443	17.392
Bubble Sort	0.033	6.765	169.782
Insertion Sort	0.034	1.755	246.630
Heap Sort	0.039	0.434	10.917
Merge Sort	0.049	0.728	12.365

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η εργασία είχε ως βασικό στόχο την υλοποίηση και γραφική αναπαράσταση των αλγορίθμων ταξινόμησης. Διαπιστώθηκε ότι η θεωρία κάθε αλγορίθμου εξαρτάται από παράγοντες όπως:

- Τη διάταξη των δεδομένων του πίνακα.
- Τον τρόπο χρήσης της μνήμης.
- Τον τρόπο υλοποίησης του.

- Από τα πειραματικά αποτελέσματα διαπιστώθηκε πως οι απλοί αλγόριθμοι μπορούν να είναι αποτελεσματικοί σε μικρούς πίνακες, ενώ σε μεγάλα δεδομένα υπάρχουν αποδοτικότερες εναλλακτικές.
- Με την οπτικοποίηση ήταν δυνατή η κατανόηση των αποτελεσμάτων και παρουσιάστηκε η εξέλιξης κάθε αλγορίθμου στην ταξινόμηση πίνακα.
- Η εργασία πέτυχε τον στόχο της και μπορεί να θεωρηθεί ένα σημαντικό εργαλείο μάθησης και κατανόησης αλγορίθμων ταξινόμησης.

ΕΡΩΤΗΣΕΙΣ;