Εργασία 1ου εξαμήνου στο μάθημα “Εισαγωγή στους υπολογιστές” - Αλγόριθμοι ταξινόμησης, Ομάδα 06

Κουτσουμάνης Παναγιώτης (1122069), Παπαλάμπρου Κωνσταντίνος (1122072)
https://github.com/konst3/python_project

Περιγραφή Project

Πρόβλημα Ταξινόμησης (Sorting Problem)
Το πρόβλημα της ταξινόμησης αποτελεί ένα από τα θεμελιώδη προβλήματα της επιστήμης των υπολογιστών. Συγκεκριμένα, δοθείσας μιας ακολουθίας στοιχείων, στόχος είναι η αναδιάταξή τους σε συγκεκριμένη σειρά (συνήθως αύξουσα ή φθίνουσα), βάσει μιας σχέσης διάταξης. Η ταξινόμηση αποτελεί κρίσιμη λειτουργία σε πληθώρα εφαρμογών, όπως η οργάνωση δεδομένων, η βελτιστοποίηση αλγορίθμων αναζήτησης, η ανάλυση δεδομένων και η επεξεργασία πληροφορίας.

Μονονηματική έναντι Παράλληλης Υλοποίησης
Στο πλαίσιο της σύγχρονης υπολογιστικής αρχιτεκτονικής, η επιλογή μεταξύ μονονηματικής (sequential/single-threaded) και παράλληλης (parallel) υλοποίησης των αλγορίθμων ταξινόμησης παρουσιάζει ιδιαίτερο ενδιαφέρον εφόσον μπορεί να υπάρξει θεωρητική επιτάχυνση των αλγορίθμων επειδή οι διεργασίες εκτελλούνται παράλληλα, όμως ταυτόχρονα υπάρχουν και πολλές προκλήσεις.

Μονονηματική Υλοποίηση
Στην μονονηματική προσέγγιση, οι εργασίες εκτελούνται σειριακά, με χρήση ενός μόνο νήματος εκτέλεσης (thread). Παρόλο που αυτή η προσέγγιση είναι απλούστερη στην υλοποίηση και τον έλεγχο ορθότητας, περιορίζεται από τη δυνατότητα αξιοποίησης μόνο ενός επεξεργαστικού πυρήνα, γεγονός που οδηγεί σε υψηλότερους χρόνους εκτέλεσης για μεγάλα σύνολα δεδομένων.

Παράλληλη Υλοποίηση
Η παράλληλη υλοποίηση επιδιώκει την αξιοποίηση πολλαπλών επεξεργαστικών πυρήνων, κατανέμοντας το υπολογιστικό φορτίο σε πολλαπλά νήματα που εκτελούνται ταυτόχρονα. Αυτή η προσέγγιση παρουσιάζει τα εξής πλεονεκτήματα και προκλήσεις:

Θεωρητικά Πλεονεκτήματα:
- Σημαντική μείωση του χρόνου εκτέλεσης για μεγάλα σύνολα δεδομένων
- Βέλτιστη αξιοποίηση των πολυπύρηνων αρχιτεκτονικών
- Βελτιωμένη απόδοση σε πραγματικές εφαρμογές

Προκλήσεις:
- Πολυπλοκότητα στην υλοποίηση και τον συγχρονισμό νημάτων
- Overhead λόγω διαχείρισης νημάτων και επικοινωνίας μεταξύ τους
- Ανάγκη για αλγορίθμους που υποστηρίζουν παραλληλοποίηση
- Πιθανά προβλήματα ανταγωνισμού (race conditions) και αδιεξόδων (deadlocks)

Από τις οποίες προκλήσεις προκύπτει πως τουλάχιστον στη γλώσσα προγραμματισμού python η υλοποίηση των παράλληλων αλγορίθμων δεν έχει αξιόλογο αποτέλεσμα εφόσον το “overhead” της είναι αυξημένο, λόγο της κακής διαχείρησης της μνήμης

Αντικείμενο του Project
Το παρόν project επικεντρώνεται στη σχεδίαση και ανάπτυξη μιας εφαρμογής με γραφικό περιβάλλον χρήστη (GUI), η οποία προσφέρει τη δυνατότητα επιλογής και εκτέλεσης διαφόρων αλγορίθμων ταξινόμησης, τόσο σε μονονηματική όσο και σε παράλληλη υλοποίηση. Η εφαρμογή παρέχει στον χρήστη πρόσβαση σε ένα ευρύ φάσμα αλγορίθμων, συμπεριλαμβανομένων:

- **Bubble Sort** (μονονηματική και παράλληλη υλοποίηση)
- **Merge Sort** (μονονηματική και παράλληλη υλοποίηση)
- **Quick Sort** (μονονηματική και παράλληλη υλοποίηση με διαφορετικά σχήματα διαμέρισης - partition schemes)

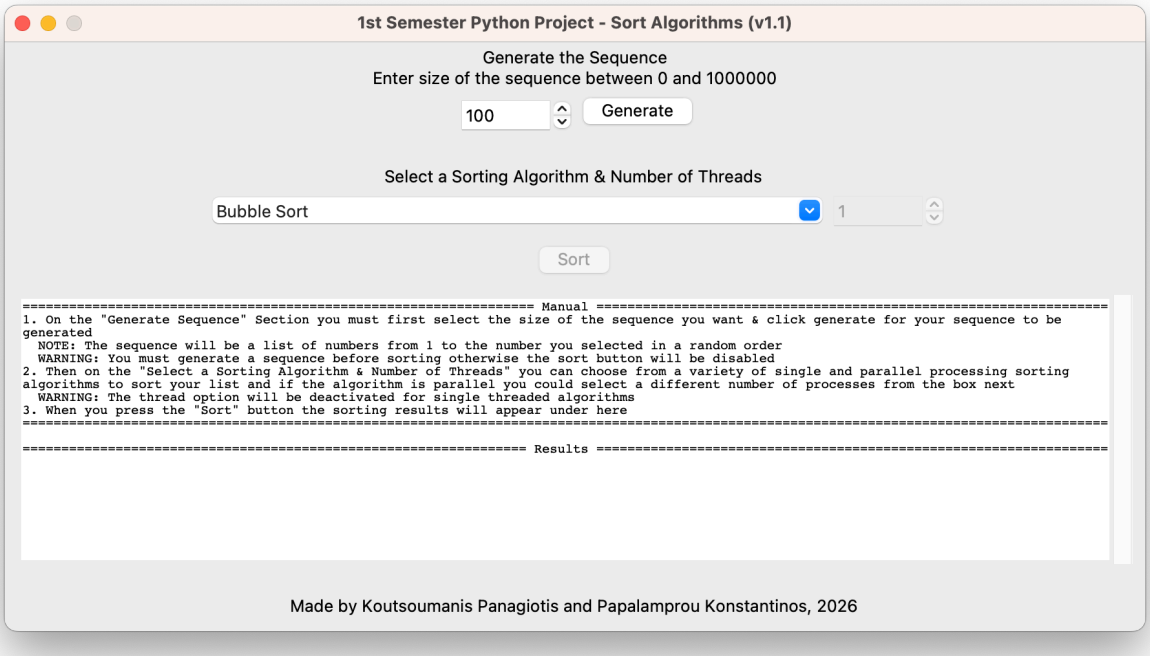
Λειτουργικότητα
Η εφαρμογή επιτρέπει στον χρήστη να:

- Επιλέξει τον επιθυμητό αλγόριθμο ταξινόμησης από την παρεχόμενη λίστα
- Καθορίσει τον τρόπο εκτέλεσης (μονονηματική ή παράλληλη υλοποίηση)
- Εφαρμόσει τον αλγόριθμο σε ακολουθίες τυχαίων αριθμών
- Καταγράψει και αναλύσει τις επιδόσεις κάθε αλγορίθμου
- Συγκρίνει την αποδοτικότητα μεταξύ διαφορετικών αλγορίθμων και προσεγγίσεων (μονονηματικών έναντι παράλληλων υλοποιήσεων)
- Αξιολογήσει την επιτάχυνση (speedup) που επιτυγχάνεται μέσω του παραλληλισμού

Στόχος
Ο σκοπός του project είναι η πειραματική αξιολόγηση και σύγκριση της υπολογιστικής πολυπλοκότητας και των επιδόσεων των αλγορίθμων υπό διαφορετικές συνθήκες εκτέλεσης, με έμφαση στην ανάδειξη των πλεονεκτημάτων και περιορισμών του παραλληλισμού στο πλαίσιο των αλγορίθμων ταξινόμησης.

Γραφική Διεπαφή Χρήστη

Το πρόγραμμα απαρτίζεται από ένα γραφικό περιβάλλον που δημιουργήθηκε με την χρήση της βιβλιοθήκης tkinter.



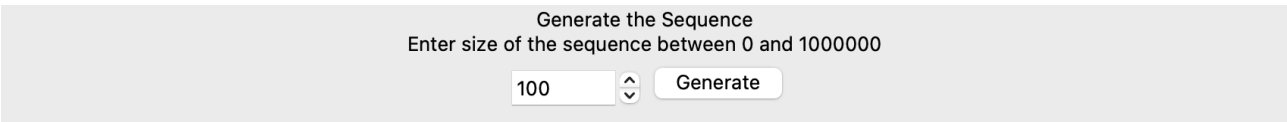
Manual

Καταρχάς στο πεδίο Manual αναφέρονται οι βασικές οδηγίες χρήσεις για την καθοδήγηση του χρήστη στην εφαρμογή

```
============================================================ Manual ============================================================
1. On the "Generate Sequence" Section you must first select the size of the sequence you want & click generate for your sequence to be
generated
  NOTE: The sequence will be a list of numbers from 1 to the number you selected in a random order
  WARNING: You must generate a sequence before sorting otherwise the sort button will be disabled
2. Then on the "Select a Sorting Algorithm & Number of Threads" you can choose from a variety of single and parallel processing sorting
algorithms to sort your list and if the algorithm is parallel you could select a different number of processes from the box next
  WARNING: The thread option will be deactivated for single threaded algorithms
3. When you press the "Sort" button the sorting results will appear under here
================================================================================================================================
```

Δημιουργία Τυχαίας Ακολουθίας

Ξεκινώντας ο χρήστης την εφαρμογή καλείται να δημιουργήσει μια ακολουθία τυχαίων αριθμών επιλέγοντας το μέγεθος της από το πεδίο spinbox (στο οποίο έχει εκπονηθεί κατάλληλη συνάρτηση *validate_spinbox* για τον έλεγχο της εγκυρότητας των στοιχείων του πεδίου κατά την είσοδο τους) και πατώντας το κουμπί **Generate** (το οποίο καλεί μια βοηθητική συνάρτηση που δημιουργεί μια λίστα από το 1 έως τον αριθμό που επέλεξε ο χρήστης και περιέχει τον κάθε αριθμό μόνο μια φορά)



Όταν η λίστα δημιουργηθεί θα υπάρξει κατάλληλο μήνυμα στο πεδίο των αποτελεσμάτων

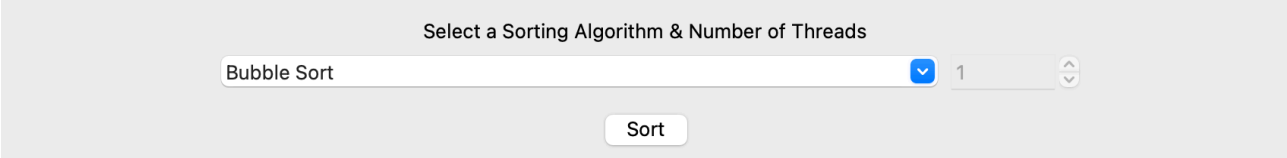
```
============================================================ Results ===========================================================

Generated new Sequence: [100, 94, 3, 85, 81, 68, 90, 32, 74, 75, 19, 83, 70, 10, 84, 49, 60, 31, 18, 39, 45, 20, 72, 73, 43, 29, 52, 42, 14,
55, 25, 7, 79, 96, 92, 89, 1, 61, 93, 77, 44, 80, 22, 56, 5, 6, 37, 59, 64, 57, 28, 69, 35, 47, 82, 50, 36, 58, 78, 13, 71, 86, 63, 87, 17,
34, 30, 38, 91, 66, 9, 98, 21, 23, 53, 8, 16, 67, 54, 27, 15, 88, 99, 26, 76, 97, 4, 48, 2, 62, 12, 51, 95, 40, 33, 24, 65, 41, 11, 46],
size: 100
```

(Για την εξοικονόμηση χώρου στο πεδίο των αποτελεσμάτων οι σειρές που περιέχουν πάνω από 100 στοιχεία συντομεύονται στα πρώτα και τελευταία είκοσι)

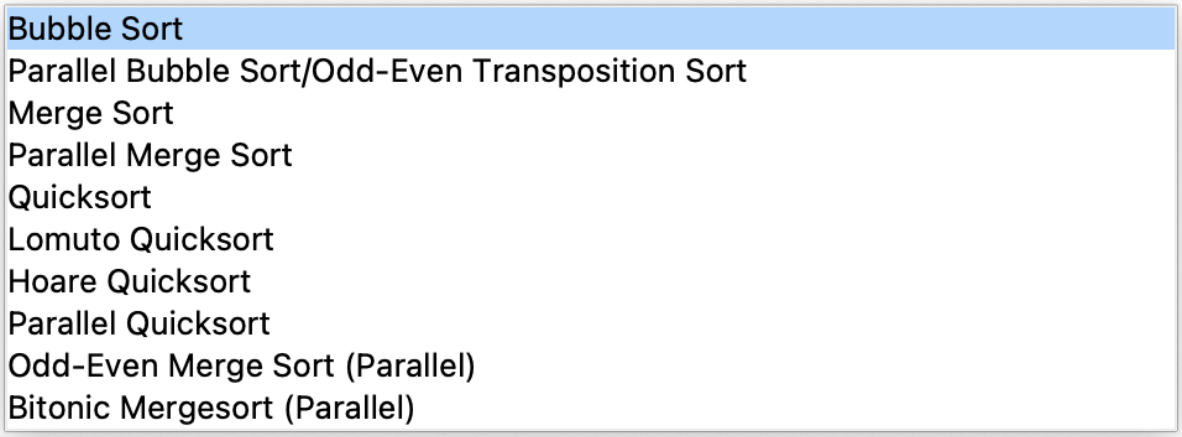
```
Generated new large Sequence: [326, 791, 526, 124, 58, 107, 708, 475, 594, 91, 122, 134, 585, 964, 77, 112, 877, 535, 195, 752]...[706, 408,
831, 888, 991, 805, 624, 704, 683, 924, 728, 340, 221, 968, 49, 138, 396, 216, 243, 975], size 1000
```

Επιλογή & Εκτέλεση Αλγορίθμου

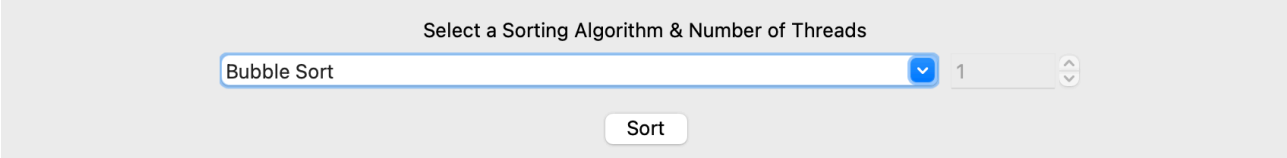
Ταυτόχρονα με την δημιουργία της ακολουθίας ενεργοποιείται το κουμπί **Sort**. Ως εκ τούτου, ο χρήστης μπορεί να προχωρήσει στο πεδίο επιλογής αλγορίθμου και πλήθους νημάτων, εφόσον είναι διαθέσιμα

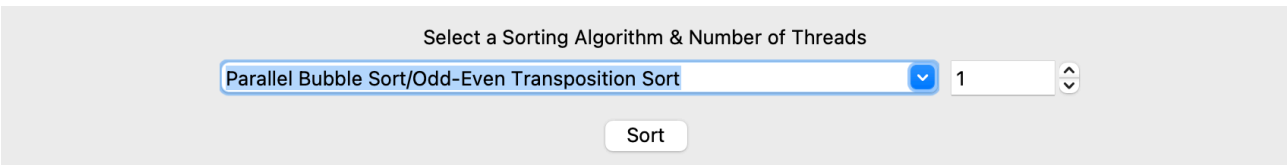


Συγκεκριμένα, η επιλογή αλγορίθμου γίνεται με την χρήση ενός *Combobox*



Με την επιλογή του κάθε αλγορίθμου (<<*ComboboxSelected*>> event) το πρόγραμμα ελέγχει εάν ο αλγόριθμος μπορεί να τρέξει παράλληλα (Σημείωση: Κάθε αλγόριθμος είναι δηλωμένος ως μια κλάση, η οποία περιέχει μια boolean μεταβλητή για το εάν ο αλγόριθμος είναι παράλληλος, την οποία μπορεί να διαβάσει μια βοηθητική συνάρτηση *toggle_box*). Με την βοήθεια της συνάντησης *toggle_box* το πεδίο επιλογής πλήθους νημάτων απενεργοποιείται και ενεργοποιείται αντίστοιχα, όπως φαίνεται και στις 2 παρακάτω εικόνες

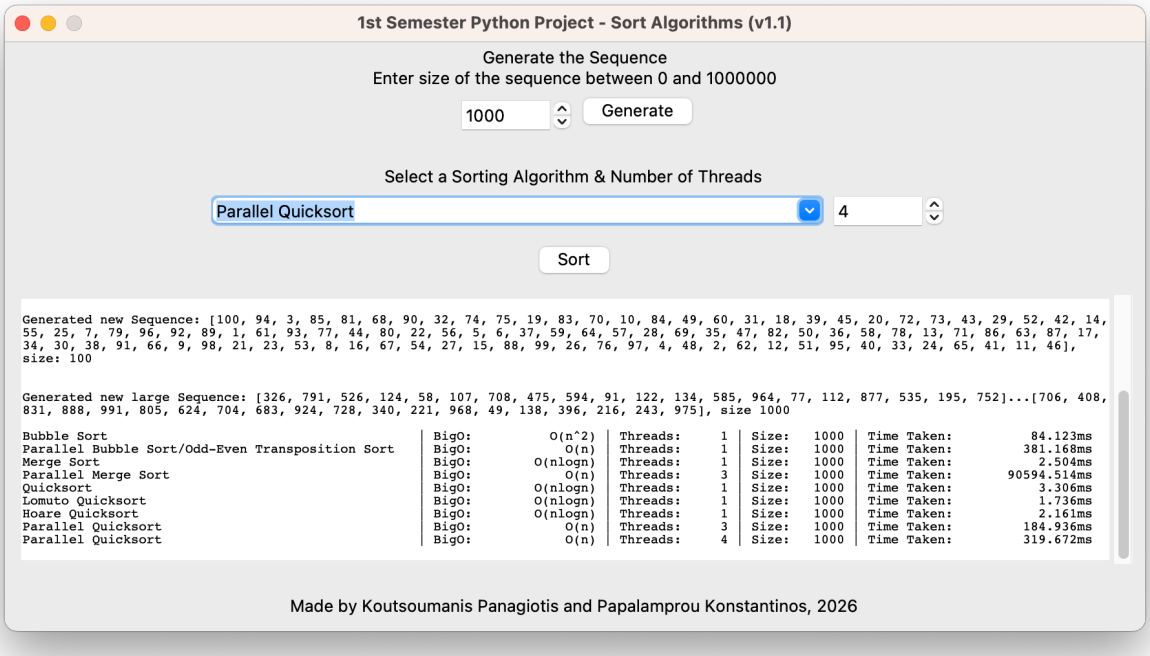




Στην επιλογή νημάτων το πρόγραμμα ανακυκλώνει την λογική του πρώτου *SpinBox*

Όταν ο χρήστης έχει κάνει τις επιλογές του μπορεί να πατήσει το κουμπί **Sort** για να τρέξει ο αλγόριθμος. Αναλυτικότερα χρησιμοποιείται ένας βοηθητικός διαχειριστής *run_sort()*, οποίος αντιστοιχεί μέσω ενός λεξικού *algoList* την επιλογή του χρήστη με το αντικείμενο της κλάσης του αλγορίθμου

Αποτελέσματα
Μόλις ο αλγόριθμος τελειώσει τυπώνεται και ανάλογο αποτέλεσμα στο πεδίο των αποτελεσμάτων, το οποίο αναφέρει τόσο τον χρόνο εκτέλεσης όσο και το πλήθος των νημάτων, την πολυπλοκότητα του αλγορίθμου και το μέγεθος της ακολουθίας που ταξινομήθηκε

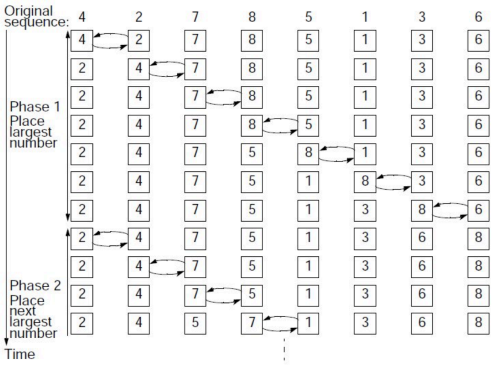


Επιπρόσθετα υπάρχει σύστημα debug που δείχνει σημαντικές πληροφορίες που μπορεί να χρειάζεται ο developer και απευθείας εκτέλεση του αλγορίθμου χωρίς χρήση της γραφικής διεπαφής στο *utils/utils.py*

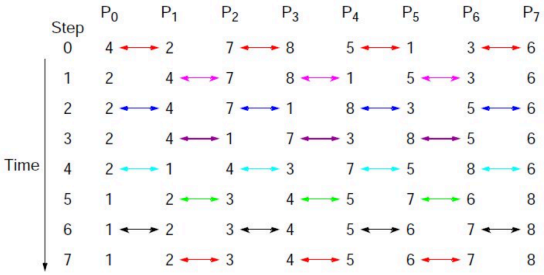
Οι Αλγόριθμοι

Bubble Sort

Ο αλγόριθμος Bubble Sort αποτελεί έναν από τους απλούστερους αλγορίθμους ταξινόμησης.



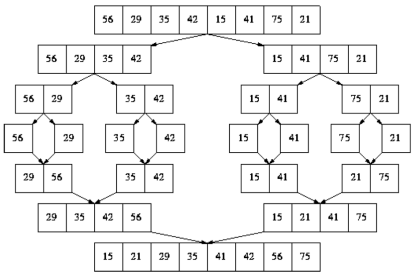
Μονονηματική Υλοποίηση: Λειτουργεί με επαναλαμβανόμενες διελεύσεις στη λίστα, συγκρίνοντας διαδοχικά ζεύγη γειτονικών στοιχείων και εναλλάσσοντάς τα εάν βρίσκονται σε λανθασμένη σειρά. Η διαδικασία επαναλαμβάνεται μέχρι να μην απαιτούνται περαιτέρω εναλλαγές, οπότε η λίστα θεωρείται ταξινομημένη.



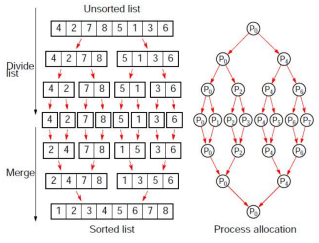
Παράλληλη Υλοποίηση: Κατασκευάστηκαν 2 μέθοδοι. Στην πρώτη η ακολουθία χωρίζεται σε κομμάτια και δημιουργείται ένα *Pool* με *Processes* τα οποία ταξινομούν το κάθε κομμάτι ξεχωριστά και στο τέλος τα ενώνουν. Στην δεύτερη χρησιμοποιείται ο αλγόριθμος Odd-Even Transposition Sort που συγκρίνει σε διαφορετικές φάσεις τα στοιχεία που βρίσκονται σε περιττή και άρτια θέση αντίστοιχα

Merge Sort

Μονονηματική Υλοποίηση: Ο Merge Sort είναι αλγόριθμος τύπου "διαίρει και βασίλευε" (divide and conquer) που χωρίζει αναδρομικά τη λίστα σε μικρότερες υπολίστες μέχρι να φτάσει σε μονάδες ενός στοιχείου, και στη συνέχεια συγχωνεύει (merge) αυτές τις υπολίστες με ταξινομημένο τρόπο.

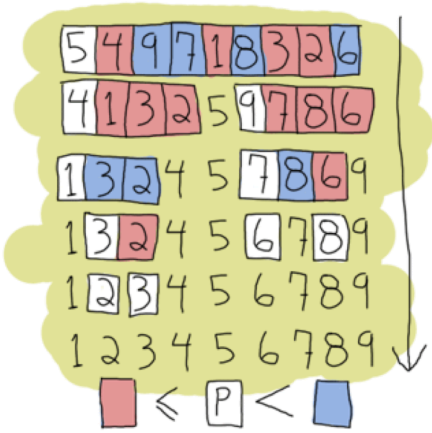


Παράλληλη Υλοποίηση: Η φύση του αλγορίθμου ευνοεί τον παραλληλισμό του, λόγω της διακλαδισμένης φύσης του, εφόσον μπορεί το κάθε *Process* να διαιρεί την λίστα και να στέλνει μέσω *Pipes* την υπόλοιποι σε άλλα process για να συνεχιστεί η διαδικασία και τελικά γίνεται συγχώνευση τον δεδομένων του κάθε *Process*

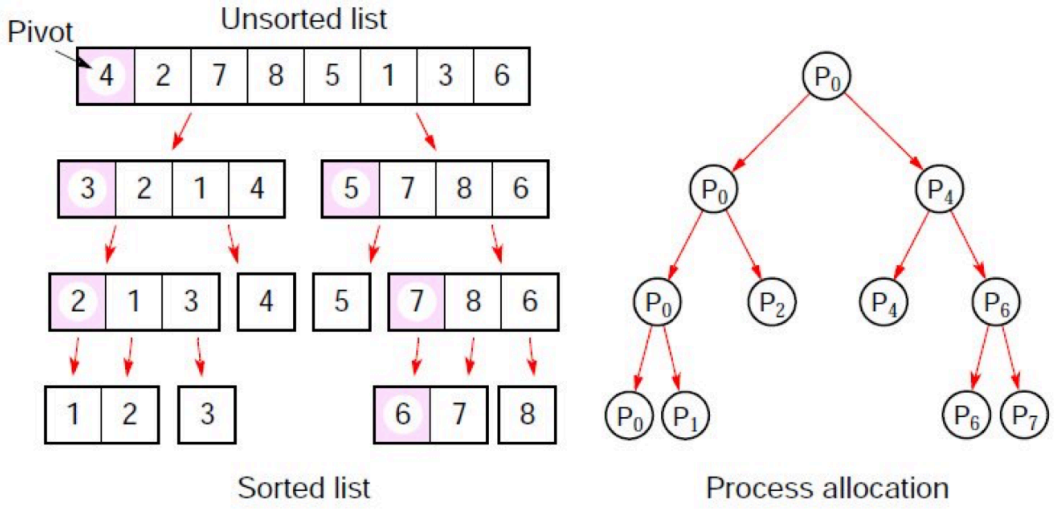


Quick Sort

Μονονηματική Υλοποίηση: Ο Quick Sort είναι ένας αποδοτικός αλγόριθμος τύπου "διαίρει και βασίλευε" που επιλέγει ένα στοιχείο ως "pivot" και διαμερίζει τη λίστα σε δύο υπολίστες: στοιχεία μικρότερα και μεγαλύτερα από το pivot. Η διαδικασία εφαρμόζεται αναδρομικά στις υπολίστες και τελικά τα μεμονωμένα στοιχεία τοποθετούνται στην κατάλληλη σειρά.



Παράλληλη Υλοποίηση: Οι δύο υπολίστες που προκύπτουν μετά το partitioning μπορούν να ταξινομηθούν ανεξάρτητα σε διαφορετικά νήματα, επιτυγχάνοντας σημαντική παραλληλοποίηση.



Σχήματα Διαμέρισης (Partition Schemes): Η εφαρμογή υποστηρίζει διαφορετικές στρατηγικές επιλογής pivot και διαμέρισης, όπως:

- Lomuto partition scheme
- Hoare partition scheme

Συμπεράσματα

Αν και λόγω περιορισμένου χρόνου δε μπορέσαμε να αναπτύξουμε πλήρως και να βελτιστοποιήσουμε τους αλγορίθμους μας, καταλήξαμε σε ορισμένα συμπεράσματα μέσα από τις συγκρίσεις που κάναμε με διάφορα μεγέθη σειρων, αλγορίθμους και πλήθος νημάτων:

- Κάποιοι αλγόριθμοι τρεχουν πιο γρήγορα από άλλους: Εκτός από τις διάφορες πολυκλοκότητες (O(N), O(NlogN), $O(N^2)$), κάποιοι αλγόριθμοι συγκρίσιμης πολυπλοκότητας τρέχουν πιο γρήγορα από άλλους, ανάλογα με την διάταξη της τυχαίας αρχικής σειράς και τον τρόπο υλοποίησης)

- Το πλήθος των νημάτων και ο χρόνος εκτέλεσης δεν έχουν θετική συσχέτιση: Έπειτα από κάποιο σημείο, το πλήθος νημάτων μειώνει την ταχύτητα εκτέλεσης διότι απαιτείται παραπάνω χρόνος για την δημιουργία τους.

- Η παράλληλη εκτέλεση δεν είναι ωφέλιμη σε μικρότερες λίστες: Λόγω του προαναφερθέντος χρόνου εκκίνησης των νημάτων, μικρότερες λίστες δεν επωφελούνται του παραλληλοποίησης διότι ο single-threaded αλγόριθμος ολοκληρώνει την εκτέλεση πολύ πιο γρήγορα.

- Η Python, ως μία high level γλώσσα δεν είναι ιδιαίτερα αποδοτική για την εκτέλεση πολλών μικρών διεργασιών και τη διαίρεσή τους. Για τέτοιας μορφής προβλήματα είναι καλύτερο να χρησιμοποιούνται εργαλεία που να επικοινωνούν πιο άμεσα με το υλικό στο οποίο εκτελείται το πρόγραμμα (π.χ. compiled προγράμματα ή precompiled shared libraries)

Διαίρεση εργασιών

Αναμέναμε να είμαστε 5 άτομα στην ομάδα και να μοιράσουμε τον φόρτο αναλόγως λόγο περιορισμένου χρόνου, όμως εν τέλει η εργασία εκπονήθηκε από δύο.

Κωνσταντίνος Παπαλάμπρου (@konst3):
Υλοποίηση των περισσότερων αλγορίθμων (bubblesort, parallel bubble sort, odd even transposition sort, merge sort, parallel merge sort), βασικός προγραμματισμός του UI (generation of sequence) και συνεισφορά σε μεγαλύτερο μέρος στις έγγραφες αναφορές|

Πάνος Κουτσουμανής (@devnol):
Αρχική σύλληψη διάταξης UI, ροή προγράμματος, υλοποίηση αλγορίθμου Quicksort, διαχειριστής αλγορίθμων, δοκιμές (debugging and unit testing for utils) και αποσφαλμάτωση κώδικα, proofreading κειμένων

Οι εργασίες του καθενός εμφανίζονται αναλυτικά στο αποθετήριο git της εργασίας:

https://github.com/konst3/python_project στα commits και στο README.

Σημεία που μας δυσκόλεψαν περισσότερο έγιναν από κοινού σε οποιοδήποτε από τα δύο προφίλ στο github.

Χρήση AI:
Το μεγαλύτερο μέρος της εργασίας έγινε χωρίς τη χρήση τεχνητής νοημοσύνης, εκτός από σημεία ακραίας αγανάκτισης και απελπισίας. Τα κείμενα έχουν συγγραφεί κυρίως χειροκίνητα και έχει γίνει χρήση του μοντέλου Claude Sonnet για την πύκνωσή τους.