Εργασία 2
Question 1

Ονοματεπώνυμο: Βλάσης Μαξούτης
A.M: 11152400118

Σε αυτό το PDF βρίσκεται η απάντηση στο πρώτο ερώτημα της Εργασίας 2

Λειτουργεία Προγράμματος

Στο φάκελο question1 υπάρχει το C αρχείο "code.c". Η βασική λειτουργεία αυτού του προγράμματος είναι η ταξινόμηση ενός πίνακα με τη χρήση του αλγορίθμου Bubble Sort. Τα δεδομένα δίνονται ή μέσω της της γραμμής εντολών από το χρήστη, ή επιλέγονται τυχαία (ανάλογα με το 2ο όρισμα του προγράμματος. Το πρώτο όρισμα είναι το πλήθος των στοιχείων του πίνακα). Μία μικρή ανάλυση του κώδικα από την αρχή:
Αρχικά, συμπεριλαμβάνει τις βασικές βιβλιοθήκες "stdio.h" και "stdlib.h". Επιπλέον, θεωρεί ως αντικείμενο "Item" τους ακεραίους (μέσω τη χρήση typedef). Περιέχει επίσης 4 macros, τα οποία έχουν τις εξής λειτουργείες:

- key(A): Δίνει το "κλειδί" ενός αντικειμένου, που στην προκειμένη περίπτωση είναι ο ίδιος ο ακέραιος αριθμός.
- less(A, B): Συγκρίνει δύο κλειδιά A και B. Συγκεκριμένα, ελέγχει αν ισχύει η συνθήκη: $A < B$
- exch(A, B): Ανταλλάσει δύο "Item" A και B. Δηλαδή, ένα Item A παίρνει την τιμή ενός Item B, ενώ το B παίρνει την τιμή του A
- compexch(A, B): Συνδυάζει τα παραπάνω δύο macro για να ελέγξει αν το B είναι μικρότερο του A ($B < A$) και αν είναι, τότε ανταλλάσει τις τιμές τους (exch)

Έχει μία συνάρτηση με το ακόλουθο interface: void bubble(Item a[ ], int l, int r)
Η συνάρτηση αυτή δέχεται έναν πίνακα από Item, καθώς και δύο "δείκτες" l (left) και r (right) για το σωστό indexation του πίνακα (Για την ταξινόμηση του πίνακα, θα καλείται πάντα με l = 0 και r = N-1). Ο αλγόριθμος είναι απλός: Συγκρίνουμε κάθε στοιχείο με στοιχείο που βρίσκεται αριστερά του, και αν το αριστερό είναι μεγαλύτερο, τα εναλλάσουμε.
Η main απλώς επεξεργάζεται τα δύο ορίσματα του προγράμματος, δηλαδή το πλήθος των στοιχείων του πίνακα, και το αν ο πίνακας θα γεμίσει δεδομένα μέσω της γραμμής εντολών ή με τη χρήση της rand() (3-ψήφιοι αριθμοί) και στη συνέχεια καλεί την bubble sort για τον πίνακα αυτό και τυπώνει τον ταξινομημένο πίνακα.

Χρονική υπολογιστική πολυπλοκότητα

- bubble(): Η χρονική πολυπλοκότητα της συνάρτησης bubble είναι $O(n^2)$, όπου n το πλήθος των στοιχείων του δοσμένου πίνακα. Καταρχάς, να προαναφερθεί πως η compexch(A, B) έχει υπολογιστική πολυπλοκότητα O(1) καθώς πρόκειται για σταθερό αριθμό συγκρίσεων και αναθέσεων. Επιπλέον, Στη χειρότερη περίπτωση έχουμε πως $l = 0$ (έπεται ότι $l > 0$) και $r = n$ (έπεται ότι $r < n$ αλλιώς το πρόγραμμα θα αποτύχει, αλλά δεν έχει σημασία, καθώς το r είναι απλά ένας θετικός ακέραιος). Για να υπολογίσουμε το πλήθος των πράξεων στη χειρότερη περίπτωση, γράφουμε μια ισοδύναμη, ως προς το πλήθος των επαναλήψεων, μορφή για τον εσωτερικό βρόχο επανάληψης ως εξής:
  for (j = r; $j > i$; j–) ⇒ for (j = i+1; j≤ r; j++)
  Το πλήθος των επαναλήψεων που εκτελούνται είναι ίδιο και στις δύο περιπτώσεις, απλά στη δεύτερη περίπτωση το νούμερο αυτό μπορεί να γραφτεί με τη μορφή αθροίσματος, που είναι πιο βολικό.
  Προφανώς, σαν κώδικας, φυσικά και δεν είναι ισοδύναμες εκφράσεις, απλώς έχουν το ίδιο πλήθος επαναλήψεων.
  Επομένως, καθώς πρόκειται για δύο εμφωλιαζμένους βρόκους επανάληψης (nested loops), το συνολικό πλήθος των επαναλήψεων θα είναι:

$$\sum_{i=0}^{n-2} \sum_{j=i+1}^{n} 1 = \sum_{i=0}^{n-2} n - i = \frac{n(n+1)}{2} - 1 = \frac{n^2+n}{2} - 1$$

  Επομένως, η χρονική υπολογιστική πολυπλοκότητα της bubble είναι: $\boxed{O(n^2)}$.

- main(): Η main είναι πιο απλή. Τόσο στον if όσο και στον else κλάδο εκτελούνται n επαναλήψεις στο βρόχο επανάληψης, και στον βρόχο οι πράξεις είναι O(1) (ανάθεση ή rand()). Μετά, υπάρχει μία κλήση της bubble, καθώς και άλλο ένα for loop με n επαναλήψεις με O(1) πράξεις σε κάθε επανάληψη (printf()). Άρα, η χρονική υπολογιστική πολυπλοκότητα είναι O(n+n^2+n) = $\boxed{O(n^2)}$.