Sensor Challenge

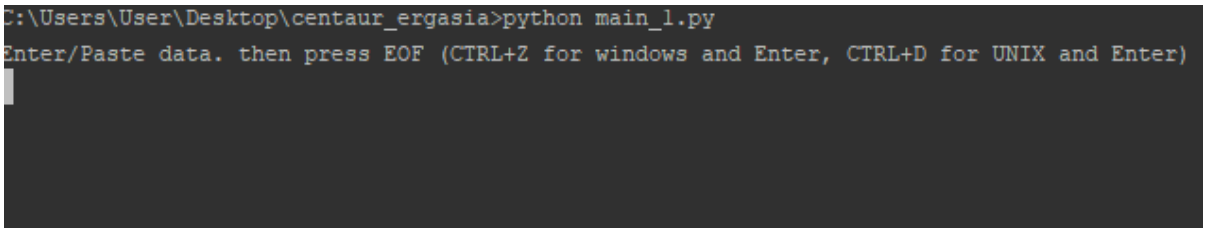
Για την υλοποίηση της συγκεκριμένης εργασίας χρησιμοποιήθηκε η **Python 3**, ο κώδικας που δημιουργήθηκε αποτελείται από 2 μέρη. Το πρώτο μέρος, αποτελεί την κλασσική υλοποίηση της ζητούμενης διαδικασίας ενώ το δεύτερο μέρος αποτελεί μια “optimized” υλοποίηση του αλγορίθμου με σκοπό την μείωση του αριθμού επαναλήψεων για τον υπολογισμό των τιμών που είναι μεγαλύτερες από τα κατώφλια που δίνονται.

Πρώτο Μέρος:

Το πρώτο μέρος αποτελείται από τα εξής αρχεία:

- function1.py: Περιέχει όλες τις συναρτήσεις που χρησιμοποιούνται από τα script του πρώτου μέρους.
- load_data.py: Περιέχει την συνάρτηση η οποία φορτώνει το αρχείο “data.txt”, χρησιμοποιείται και στο δεύτερο μέρος.
- main_1.py: Ο χρήστης δίνει είσοδο, σύμφωνα με την εκφώνηση, και η έξοδος είναι και αυτή σύμφωνα με την εκφώνηση.
- run_data1.py: Το αρχείο αυτό, αν εκτελεστεί εκτελεί τον υπολογισμό, για το αρχείο “data.txt”
- evaluate_count_elements.py: Το συγκεκριμένο αρχείο, χρησιμοποιεί την βιβλιοθήκη PyTest, και ελέγχει την ορθότητα της συνάρτησης count_elements του αρχείου function1.py, η οποία αποτελεί την βασική συνάρτηση που υπολογίζει τους αριθμούς πάνω από τα κατώφλια στην περίπτωση που διαβάζεται το αρχείο data.txt
- evaluate_run_me1.py: Το συγκεκριμένο αρχείο, χρησιμοποιεί την βιβλιοθήκη PyTest, και ελέγχει την ορθότητα της συνάρτησης run_me, χρησιμοποιείται από το αρχείο main_1.py, και είναι η βασική συνάρτηση που υπολογίζει τους αριθμούς που είναι πάνω από τα κατώφλια στην περίπτωση που δώσει ο χρήστης της μετρήσεις.

Οπότε για να θέσει ο χρήστης είσοδο, εκτελούμε το Python αρχείο **main_1.py** και έχουμε ως εξής:



Από ότι βλέπουμε εμφανίζεται το μήνυμα για να δώσει ο χρήστης είσοδο, η είσοδος θα είναι Multi line, οπότε αφου την εισάγουμε πατάμε **CTRL+Z +Enter.** Οι είσοδοι που παίρνει το πρόγραμμα είναι της μορφής:

2 9
45 46 47 48 52 60

S1 34 45 18 20 35 40 50 65 75

S2 87 89 80 78 90 38 32 45 58

Αφού εισάγουμε τα δεδομένα, πατάμε **CTRL+Z +Enter** έχουμε την έξοδο σύμφωνα με την ζητούμενη μορφή (άσπρο περίγραμμα):

```
Enter/Paste data. then press EOF (CTRL+Z for windows and Enter, CTRL+D for UNIX and Enter)
2 9
45 46 47 48 52 60
S1 34 45 18 20 35 40 50 65 75
S2 87 89 80 78 90 38 32 45 58
^Z
S1 3 3 3 3 2 2
S2 6 6 6 6 6 5
```

Για την εύρεση των τιμών που ξεπερνούν τα κατώφλια, από το αρχείο, “data.txt” εκτελούμε το αρχείο **run_data1.py**:

```
C:\Users\User\Desktop\centaur_ergasia>python run_data1.py
Total iterations:  4000000
```

Παρατηρούμε ότι για την εκτέλεση του αλγορίθμου, κλασσική εκδοχή (έλεγχος 4 κατωφλίων στις μετρήσεις για 10000 sensors) για αυτό και προκύπτουν **4000000** επαναλήψεις.

Για να γίνει ο έλεγχος της ορθότητας, χρήση PyTest έχουμε ως εξής:

1.
python evaluate_count_elements.py

py.test evaluate_count_elements.py

2.
python evaluate_run_me1.py

py.test evaluate_run_me1.py

Οπότε όσο αφορά το πρώτο μέρος,

- main_1.py: Εκτελώντας αυτό το αρχείο, ο χρήστης δίνει είσοδο
- run_data1.py: Εκτελώντας αυτό το αρχείο, εκτελείται ο αλγόριθμος για το αρχείο “data.txt”
- evaluate_count_elements.py, evaluate_run_me1.py: Τεστάρουν τις κυριότερες συναρτήσεις.

Δεύτερο Μέρος:

Το δεύτερο μέρος αποτελείται από τα εξής αρχεία:

- optimized_functions.py: Περιέχει τις συναρτήσεις που χρησιμοποιούνται για το δευτερο μερος
- main_2.py: Αν εκτελεστεί, ο χρήστης δίνει την είσοδο σε μορφή σύμφωνα με αυτή της εκφώνησης, και παίρνουμε ως έξοδο το πλήθος των μετρήσεων που είναι πάνω από τα δεδομένα κατώφλια
- run_data_optimized.py: Αν εκτελεστεί, φορτώνεται το αρχείο data.txt, και εκτελείται η διαδικασία για τα δεδομένα στο αρχείο, αποτελεί μια βελτιστοποιημένη έκδοση της κλασσικής υλοποίησης.

- evaluate_second_run.py: Ελέγχει την ορθότητα της συνάρτησης **run_me2** του αρχείου optimized_functions, είναι η συνάρτηση που καλεί όλες τις υπόλοιπες και χρησιμοποιείται στην περίπτωση που γίνεται ο έλεγχος στο αρχείο data.txt αλλά και όταν εισάγει ο χρήστης δεδομένα.

Οπότε για να θέσει ο χρήστης είσοδο εκτελεί το αρχείο **main_2.py**:

Η είσοδος που θέτουμε είναι της μορφής
2 9

45 46 47 48 52 60

S1 34 45 18 20 35 40 50 65 75

S2 87 89 80 78 90 38 32 45 58

Έχουμε ως εξής, πάλι η έξοδος είναι στο άσπρο περίγραμμα

```
Enter/Paste data. then press EOF (CTRL+Z for windows and Enter, CTRL+D for UNIX and Enter)
2 9
45 46 47 48 52 60
S1 34 45 18 20 35 40 50 65 75
S2 87 89 80 78 90 38 32 45 58
^Z
S1 3 3 3 3 2 2
S2 6 6 6 6 6 5
```

Για την εύρεση των τιμών που ξεπερνούν τα κατώφλια, από το αρχείο, “data.txt” εκτελούμε το αρχείο **run_data_optimized.py**:

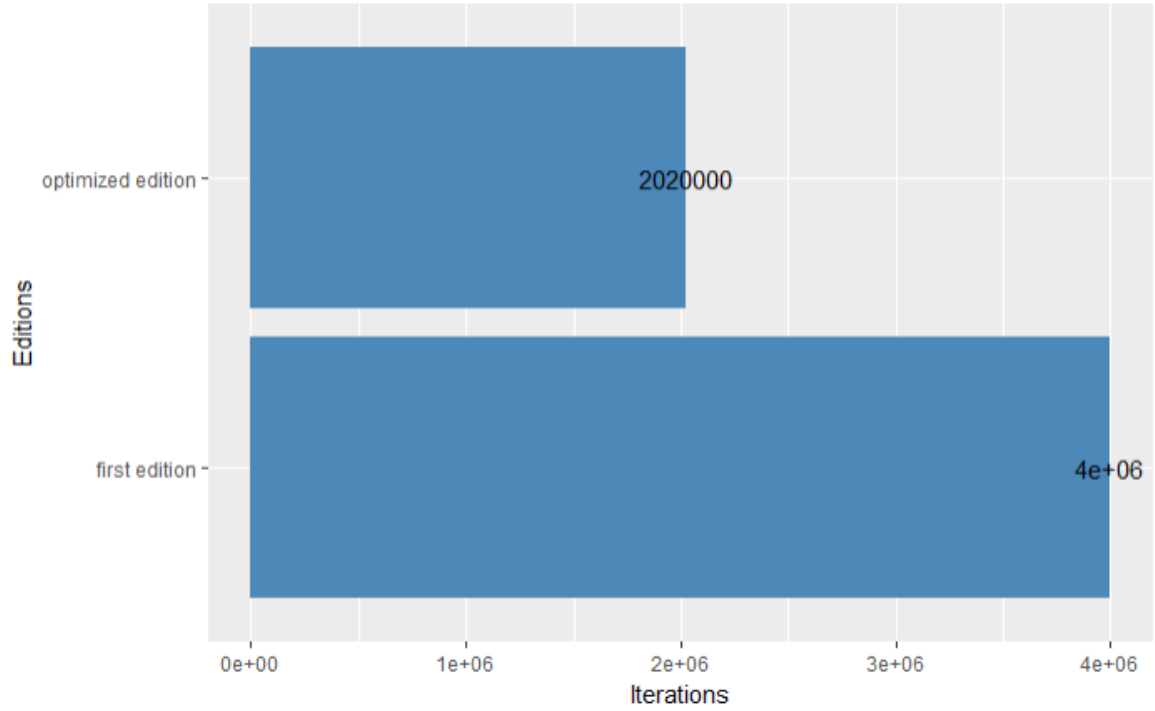
```
C:\Users\User\Desktop\centaur_ergasia>python run_data_optimized.py
Total iterations:  2020000
```

Παρατηρούμε τώρα οτι οι επαναλήψεις, που χρειάστηκαν για την καταμέτρηση των μετρήσεων που είναι πάνω από τα κατώφλια “έπεσε” στις **2020000** επαναλήψεις. Αυτό οφείλεται στον τρόπο που υλοποιήθηκε ο αλγόριθμος. Πιο συγκεκριμένα, όταν εισάγουμε τις μετρήσεις ενός sensor και τα κατώφλια, ο αλγόριθμος τα ταξινομεί και τις 2 λίστες από το μικρότερο στο μεγαλύτερο. Έτσι λοιπόν το πρώτο κατωφλι, το οποίο είναι και το μικρότερο αριθμητικά, θα προσπελάσει όλη την λίστα μέχρις ότου να βρει στοιχεία που είναι μεγαλύτερα από αυτό και αποθηκεύει την θέση του της μέτρησης στην λίστα που είναι αμέσως μικρότερη από το κατώφλι και έστω ότι την ονομάζουμε i. Έτσι στην επόμενη επαναλήψη, τα επόμενο κατώφλι που είναι μεγαλύτερο αριθμητικά από το πρώτο κατώφλι θα ξεκινήσει τον υπολογισμό από την θέση “i”. Ομοίως και τα επόμενα κατώφλια, με αυτόν τον τρόπο μειώνουμε σημαντικά τις επαναλήψεις. Παρακάτω ακολουθεί ένα παράδειγμα που περιγράφει με πιο απτό τρόπο την διαδικασία.

Έστω ότι έχουμε τις μετρήσεις **[18,20,34,35,40,45,50,65,75]** και τα κατώφλια **[45,46,47,48,52,60]** και ότι το πρόγραμμα τα έχει ταξινομημένα. Το πρώτο κατώφλι “45” διατρέχει την λίστα βρίσκει ότι 3 στοιχεία είναι μεγαλύτερα από αυτό και αποθηκεύει την θέση “5”,η αρίθμηση ξεκινάει από το 0, ως την θέση του αμέσως μικρότερου η ίσου στοιχείου. Έτσι λοιπόν το κατώφλι “46”, θα ξεκινήσει τον υπολογισμό από την θέση 5 και μετά, γιατί τα προηγούμενα στοιχεία είναι σίγουρα μικρότερα. Το κατώφλι “46” στην συνέχεια επιστρέφει την θέση “5” (θέση του αμέσως μικρότερου η ίσου στοιχείου) και βρίσκει 3 μεγαλύτερα στοιχεία από αυτό. Ομοίως τα κατώφλια “47”, ”48”. Όταν φτάσουμε στο κατώφλι 52, αυτό αρχίζει την αναζήτηση από την θεση “5” που επέστρεψε το

κατώφλι “48”, βρίσκει 2 στοιχεία που είναι μεγαλύτερα από αυτό και επιστρέφει την θέση “6” που είναι η θέση στην λίστα των μετρήσεων του αμέσως μικρότερου η ίσου αριθμού και ούτω καθεξής.

Το παρακάτω διάγραμμα αναπαριστά το πλήθος των επαναλήψεων για την κλασσική και για την βελτιστοποιημένη εκδοχή του προγράμματος.



Για να γίνει ο έλεγχος εγκυρότητας της συνάρτησης run_me2 του αρχείου optimized_functions.py, η οποία κάνει τον έλεγχο ποιες μετρήσεις είναι πάνω από τα κατώφλια που δίνονται, κάνουμε τα εξής:

python evaluate_second_run.py

py.test evaluate_second_run.py

Οπότε για το δεύτερο μέρος έχουμε ως εξής:

- main_2.py: Εκτελώντας αυτό το αρχείο, ο χρήστης δίνει είσοδο, και έχουμε το αποτέλεσμα.
- run_data_optimized.py: Εκτελώντας αυτό το αρχείο, εκτελείται ο ζητούμενος υπολογισμός πιο αποδοτικά.
- evaluate_second_run.py: Εκτελώντας αυτό το αρχείο, γίνεται ο έλεγχος ορθότητας της συνάρτησης run_me2

ΣΗΜΕΙΩΣΗ

Επειδή ταξινομούμε και την λίστα με τα κατώφλια, του αρχείου data.txt, τα αποτελεσματα για κάθε σενσορα δεν εχουν την σειρά που είχαν και στην πρώτη έκδοση. Για παράδειγμα, τώρα για τον πρώτο σενσορα παίρνουμε **[100, 0, 0, 0]** καθώς η θέση των thresholds έχει αλλάξει, λόγω ταξινόμησης και είναι [0,1,2,4] ενώ η αρχική μορφή ήταν [2,1,0,4]