- Υλοποίηση με πίνακα γειτνίασης . Όταν έχουμε μια καινούργια ακμή ΒΓ ψαχνουμε να βρούμε όλες τις ακμές που είναι προσπελάσιμες από το Β (με κοστος 1) και τις ενώνουμε με όλες τις ακμές που είναι προσπελάσιμες απο το Γ(με κόστος 1).

- Υπάρχει μια 1-1 και επι αντιστοιχία ανάμεσα σε γραμμές (άρα και στήλες) του πίνακα και σε αρχεία json (π.χ γραμμή του πίνακα 100 αντιστοιχεί στο φάκελο άμαζον αρχείο: amazon12.json)

- Θα διαβάσουμε το dataset Χ και θα φτιάξουμε πίνακα 2D με μέγεθος όσα τα αρχεία του

- Κάθε εγγραφή θα είναι της μορφής filename//number

```
struct Directory {
        char * DirectoryName
        int numOfRows
        struct RBTree * RBTree(Η χρήση του δέντρου εξηγείται στη παρακάτω παράγραφο)
}
```

Για κάθε αρχείο φτιάχνουμε ένα binary Tree (ή ένα ισοζυγισμένο δέντρο, ακόμη καλύτερα λόγω πολυπλοκότητας - RBTree) το οποίο θα μας κανει την 1-1 αντιστοίχιση ανάμεσα σε όνομα αρχείου και θέση αρχείου στο φάκελο. Δηλαδη το αρχείο 29.json ειναι το 5ο αρχείο του φακέλου .Το κλειδί θα είναι η ονομασία-id του αρχείου και η τιμή σε κάθε κόμβο το νούμερο που αντιστοιχεί στην θέση του στο φάκελο
Αυτή η δομή σε συνδυασμό με το struct directory θα μας βοηθήσει να κάνουμε την αντιστοίχιση αναμεσα σε αρχείο και θέση στο πίνακα γειτνίασης . (Πχ ebay500.json βρίσκεται στη 104η γραμμή του πίνακα γειτνίασης)

29.json = 5

```
struct Clique {
        int ** malloc …. πίνακας γειτνίασης 2D μεγέθους όσα τα αρχεία του Χ
        struct Directory * directory // Κρατάμε πόσα αρχεία έχει ο κάθε πίνακας
                                                // Κάθε θέση του struct Directory αντιστοιχεί
                                                //  αναλογικά με τις θέσεις που καταλαμβάνουν
                                                // οι εγγραφές του στον πίνακα γειτνίασης
}
```
