Έχω υλοποιήσει δυναμικό hash table, για γρηγορότερες αναζητήσεις στο γράφο μου.Οπότε οι βασικές μου δομές είναι το Node που έχει ένα χαρακτηριστικό char *_id και ένα δείκτη στην κεφαλή της διπλά συνδεδεμένης λίστας που αποθηκεύω τις ακμές μου.
Η δομή μου edge έχει βάρος και δείκτη σε node ονόματι target που είναι το που καταλήγει μια ακμή.Επειδή είναι διπλά συνδεδεμένη έχει δείκτες prev και next.

Γενικά ο τρόπος που έχω οργανώσει τα αρχεία είναι σε layers δηλαδή, αν μπορεί να θεωρηθεί έτσι έχω τη main που δε κάνει κάτι απλά καλεί τον InputDirector.

Δεύτερο layer είναι οι συναρτήσεις επιπέδου API οι οποίες είναι υπεύθυνες για όλο το I/O δηλαδή για όλες τις εισόδους (αρχείο και prompt) και εξόδου.Εδώ έχω τρείς συναρτήσεις Input and Output Managers υπεύθυνοι για την επεξεργασία των αρχείων εισόδου εξόδου και τον InputDirector για τη γραμμή εντολών.

Οι Managers καλούν συναρτήσεις επιπέδου FileHandling για να διαχειριστούν αυτές το σωστό άνοιγμα αρχείων και οι Managers συνεχίζουν με την επεξεργασία των αρχείων δηλαδή το γράψιμο και το διάβασμα αντίστοιχα.

Ο InputDirector τώρα έχει πολύ βασικό ρόλο αφού μετά το κάλεσμα των managers για επεξεργασία των δοθέντων αρχείων, είναι υπεύθυνος για την σωστή ανάγνωση από τη γραμμή εντολής και με μια μεγάλη switch-case και την ανάθεση της κάθε εντολής στην σωστή case.Αυτό το κάνω τσεκάρωντας τον αριθμό των διαβασμένων λέξεων μέχρι το end of line.Θέλει πολύ προσοχή και διάλεξα να το κάνω έτσι για να έχω την ευελιξία να διαβάζω insert ή delete πχ και μετά ανάλογα πόσες λέξεις έχει μετά να καλώ τις σωστές συναρτήσεις.Οι συναρτήσεις που καλεί ο InputDirector είναι επιπέδου Repository.

Το Repository είναι πολύ σημαντικό επίπεδο διότι εδώ καλούνται οι κατάλληλες συναρτήσεις και δίνεται η κατάλληλη έξοδος για κάθε συνάρτηση.Οπότε εδώ διαχειρίζομαι το input που πήρα και το δίνω κατάλληλα στις “backend” συναρτήσεις ανάλογα την κάθε περίπτωση.Το Repository καλεί τις συναρτήσεις του κατώτατου επιπέδου.

Αυτές είναι είτε συναρτήσεις που αφορούν βασικές λειτουργίες του hash table είτε βασικές λειτουργίες πάνω στα nodes και edges μου.

Θα ήθελα να πω τώρα μερικά πράγματα για το hash table.Δεν είχα ξαναχρησιμοποιήσει hash table οπότε είναι η πρώτη φορά που το υλοποιώ έτσι βασίστηκα σε αυτό το tutorial: https://github.com/jamesroutley/write-a-hash-table .Έκανα και κάποιες αλλαγές όπου ήταν απαραίτητο αλλά ο κώδικας μου είναι κατά πολύ βασισμένος εκεί.Αυτό το έκανα διότι πριν δεν είχα ιδέα πως είναι ένα hash table μα τώρα ξέρω πολύ καλά, με βοήθησε να το καταλάβω σε βάθος.Στο hash table χρησιμοποιώ double hashing για να αποφύγω collisions και έχει αποδειχθεί πολύ χρήσιμο.Αλλιώς θα μπορούσα να βάλω και open addressing που είναι πιο απλό και απλά αν δε βρώ κενή θέση στο συγκεκριμένο index κάνω +1 μέχρι να βρώ κενή.Τώρα ξανακάνω hash που αλλάζει ανάλογα με ένα factor i που είναι οι προσπάθειες μου να βάλω την εγγραφή στο hash table.Η hash function και το double hashing είναι τα ίδια όπως στο tutorial.Επίσης, συμπεριέλαβα και load factor όπου αν είναι

κάτω από 10% γεμάτος ο hash table τον κάνω resizedown ενώ αν είναι γεμάτος πάνω από 70% τον κάνω resizeup τον μεγαλώνω δηλαδή να χωράει κιάλλες εγγραφές.
Αυτό είναι καλό γιατί μειώνονται και τα collisions και είναι πιο σωστά κατανεμημένες οι εγγραφές στο hash table.
Αυτό που μου έκανε ιδιαίτερη εντύπωση είναι ότι παρότι μοιάζει πολύ με το tutorial ο κώδικάς μου με ταλαιπώρησε πολύ μέχρι να δουλέψει και μου φαγε πολύ χρόνο που θα μπορούσα να χρησιμοποιήσω για να φτιάξω κι άλλες συναρτήσεις.Εννοώ έφαγα πολύ χρόνο στο να φτιάξω το hash table να δουλεύει καλά σε σχέση με το να είχα απλά άλλη μια συνδεδεμένη λίστα.Βέβαια έμαθα παραπάνω πράγματα και δε το μετανιώνω.

Έκανα και κάποιες προσπάθειες για την circlefind αλλά ήταν άκαρπες οπότε δεν συμπεριέλαβα κώδικα.Ουσιαστικά η ιδέα ήταν να κάνω ενα DFS αλλά δεν πρόλαβα να το κάνω στην πράξη.Έκανα κάποιες προσπάθειες με αναδρομικές συναρτήσεις που έλεγχαν αν το target->_id είναι ίσο με το start->_id αλλά να πηγαίνει σε βάθος δηλαδή για το start node να διατρέχει όλη την συνδεδεμένη του λίστα, αλλά σε κάθε edge να πηγαίνει και το target node και να ψάχνει και από εκεί και στη δική του συνδεδεμένη λίστα, αλλά δεν κατάφερα να το υλοποιήσω.

Γενικά συμβουλεύτηκα πολύ το site geeksforgeeks για τη συνδεδεμένη λίστα και ύστερα λίγο stackoverflow.

Όταν τρέχω το πρόγραμμα με valgrind μου λέει ότι δεν έχω leaks παραμόνο κάποια unconditional jumps και κάποια που βασίζονται σε uninitialized values.Με λίγα λόγια αυτά ήταν αναμενόμενα και δε νομίζω ότι είναι σοβαρά λάθη, συμβαίνουν επειδή ελέγχω αν κάποιοι pointers είναι null.

Έτρεξα το πρόγραμμά μου με όλα τα InputFiles και δείχνει να ανταποκρίνεται καλά.
Αργεί λιγάκι σε κάποια σημεία αλλά τα βγάζει πέρα.Επίσης, μια δυο φορές μου πέταξε segmentation για κάποιο λόγο αλλά μετά το έτρεξα με valgrind και μου βγαζε ότι είναι όλα μια χαρά.

Αν ξέχασα να πω/εξηγήσω κάτι ρωτήστε με στην προφορική εξέταση.Ελπίζω να τα κάλυψα όλα.Θα αφήσω και ένα φάκελο με τα output files όπως μου τα βγαλε τώρα να τα έχουμε τότε και θα αφήσω και τα test files να μη τα ξανακατεβάζουμε για διευκόλυνση.

PS. Βελτιώσεις:
*Η συνδεδεμένη λίστα θα μπορούσε να μην είναι διπλά συνδεδεμένη.
*Θα μπορούσα στο hash table να έκανα καλύτερη διαχείρηση για το πως θα μεγαλώνει και θα μικραίνει και να απλοποιούσα την hash function γιατί είναι καλή για λίγες εγγραφές αλλά χρησιμοποιεί pow() που μπορεί να αυξήσει αρκετά τον χρόνο εκτέλεσης.