vff-struct

Συγγραφείς

- Συμεωνίδης Αναστάσιος (3088)

> Έφτιαξα το μεγαλύτερο μέρος της εργασίας.

- Οσμάν Ισμαήλ (3074)

> Προσπάθησε... Έφτιαξε μια έκδοση για arrays. Την έχω κρατήσει στο `src/structures/array/mainlib.*`

Για περισσότερες πληροφορίες μπορείτε να δείτε το `git log`.

Συνεργασία με άλλα άτομα

Για αυτή την εργασία προσπάθησα να συνεργαστώ με τον διπλανό μου. Κυριολεκτικά άρχισα να ρωτώ άτομα γύρω μου (διευκρίνηση: από την ομάδα ΕΛ/ΛΑΚ Πληρ.) μέχρι να βρω ένα άτομο το οποίο είναι διαθέσιμο. Τελικά βρήκα ένα τέτοιο άτομο (διευκρίνηση: ήταν ο δεύτερος που ρώτησα, δεν πήρε πολύ ώρα). Ωστόσο δεν υπήρχε αρκετή επικοινωνία μεταξύ μας, με αποτέλεσμα να φτάσουμε σε αυτό το σημείο. Github repository.

Μεταγλώττιση του προγράμματος

Θα ήθελα να προτείνω το CMake. Είναι πιο ευέλικτο και εύκολο στην διαχείριση μεγάλων προγραμμάτων από το make. Αυτό που κάνει είναι να παράγει τα κατάλληλα αρχεία (συνήθως Makefiles) έτσι ώστε να μπορεί να γίνει compile σε κάθε συνδυασμό συστήματος-μεταγλωττιστή. Δηλαδή χρειάζονται ελάχιστες αλλαγές στον πηγαίο κώδικα για να υποστηριχθεί ένα διαφορετικό λειτουργικό σύστημα. Λόγω αυτού μπορεί να χρειαστεί να κάνετε μερικές αλλαγές αν θέλετε να τρέξει στα Windows. Κυρίως προσθήκη του .exe στο όνομα του εκτελέσιμου αρχείου.

Λειτουργία του προγράμματος

Τα πιο ενδιαφέροντα: χαρακτηριστικά προγράμματος και αλγόριθμοι που χρησιμοποιήθηκαν.

Ξεκινώντας από την αρχή: η συνάρτηση main

Ο σκοπός της main είναι να διαβάσει τις παραμέτρους που δόθηκαν στο πρόγραμμα. Έπειτα χρησιμοποιώντας τις πληροφορίες αυτές δημιουργεί κατάλληλο ICommandable το οποίο τροφοδοτείται μαζί με κάποιες ροές εισόδου/εξόδου στον Executer. Ο Executer μετατρέπει τις εντολές κειμένου σε C++ function calls. Επίσης ο τρόπος λειτουργείας του προγράμματος μπορεί να αλλάξει αλλάζοντας τις παραμέτρους στην main. Για περισσότερες πληροφορίες μπορείτε να περάσετε `--help` στην main. Αυτή η οργάνωση βοήθησε αρκετά κατά τις αποσφαλματώσεις του προγράμματος(είχα τουλάχιστον τέσσερα σοβαρά προβλήματα με memory corruption και πολλά άλλα μικροπροβλήματα).

Η βοηθητική κλάση Dequeue

Κάνει ότι λέει το όνομα. Χρησιμοποιήθηκε για την μετατροπή αναδρομικών αλγορίθμων σε επαναληπτικούς. Δεν χρησιμοποίησα αναδρομικούς αλγορίθμους επειδή δεν μπορώ να ελέγχω πως δεσμεύεται η στοίβα του προγράμματος. Επίσης βρήκα αυτή την σχετική πληροφορία: τα προγραμμάτα στα Linux(x86_64) έχουν θεωρητική απεριόριστη στοίβα! Αλλά το περιορίζουν στα 8mb για προστασία(stack smashing detection).

DFS

Πρέπει να ήταν το δεύτερο πιο εύκολο κομμάτι αυτής της εργασίας. Έχω σκεφτεί δύο διαφορετικούς τρόπους για να κρατάω ποιές κορυφές έχουν επισκεφτεί. Ο πρώτος ήταν να κρατάω 2 bits ανά κορυφή. Το πρώτο bit για το αν υπάρχει η κορυφή και το δεύτερο για το αν έχει επισκεφτεί. Αυτός ο τρόπος έχει μειονεκτήματα στην περίπτωση που οι κορυφές δεν είναι διαδοχικές. Τελικά επέλεξα να κάνω δυαδική αναζήτηση σε ένα array από key-value.

Δομές δεδομένων

AVL

Ο ποιο δύσκολος από άποψης μου αλγόριθμος ήταν οι περιστροφές στα δυαδικά δέντρα. Χρειάστηκα ~3 μέρες για να τις καταλάβω. Τελικά τα κατάφερα και ο αλγόριθμος λειτουργεί ορθά(ελπίζω). Έχει μόνο μερικά memory leaks τα οποία δεν ξέρω από που προέρχονται. Επίσης λόγω της malloc της glib γίνεται κατάχρηση της μνήμης(32 bytes min allocation). Γενικότερα έχω σκεφτεί αρκετούς τρόπους που μπορούν να βελτιώσουν την απόδοση της δομής αυτής, όπως αποθήκευση των βαρών στο pointer στον γονέα, χρήση διαφορετικού allocator κτλ. Ωστόσο δεν μπορώ να βρω έναν 'τέλειο' τρόπο. Επίσης έμαθα ότι στην C++ το delete είναι διαφορετικό από delete[].

Arrays

Τα arrays τα γνωρίζω αρκετά καλλά, για αυτό και είναι αρκετά γρήγορα. Ο αλγόριθμος που χρησιμοποίησα έχει ελάχιστη πολυπλοκότητα O(logn) και O(nlogn) στην χειρότερη περίπτωση. Σε συνδυασμό με τις μικρές απαιτήσεις του σε μνήμη είναι σχετικά καλύτερος από τους άλλους δύο, σε συγκεκριμένες περιπτώσεις. Επίσης τα arrays δεν μπορούν να βελτιστοποιηθούν περαιτέρω.

Hash table

Το hash table νόμιζα αρχικά ότι θα ήταν το καλύτερο από τα υπόλοιπα δύο, αλλά μάλλον χρειάζεται λίγο καλύτερο tweaking. Δεν πρόλαβα να ελέγξω επίσης αν υπάρχουν conflicts. Επίσης ο τρόπος υλοποίησης μου υποφέρει από bandwidth conjection(?)-πολλές μετακινήσεις μεταξύ ram και cpu cache-. Νόμιζα ότι είχα κάποιες καλλές ιδέες(χρήση AVL για τα buckets και κάτι άλλο με merge πινάκων) αλλά τελικά υπήρχαν και άλλοι περιορισμοί που δεν υπολόγισα.