ΥΣΒΔ

Εργασία 2

Ονοματεπώνυμο: Γεωργόπουλος Ιωάννης
Α.Μ.:1115201800026

Ονοματεπώνυμο: Δαμιανάκης Δαμιανός
Α.Μ.:1115201800306

SHT:

Γενική υλοποίηση:
Η SHT αποτελείται από ένα header info block που έχει το `SHT_info` το οποίο δείχνει στο πρώτο header block. Τα header blocks περιέχουν τις διευθυνσεις τών buckets αλλά και τον αριθμό των bytes στο block και τη θέση του επόμενου header block. Επίσης ο συνολικός αριθμός των buckets που μπορουν να αντιπροσωπεύει ένα header block είναι (512-8)/4=126 όπου 8 είναι ο χώρος που παίρνουν τα αλλα στοιχεία του header block, 512 ο συνολικός χώρος του header block και 4 το μέγεθος ενός integer. Τέλος υπάρχουν και τα buckets όπου έχουν 1 ή περισσότερα blocks. Σε αυτά τα blocks των buckets αποθηκεύονται τα records και άλλες χρήσιμες πληροφορίες για το block όπως τον αριθμό των records μέσα στο block, τον αριθμό των bytes που απομένουν μέσα στο block, το κλειδί του bucket και την θέση του επόμενου block αν υπάρχει. Αν δεν υπάρχει επόμενο block τότε η μεταβλητή θα έχει την τιμή -1.

Συναρτήσεις:
`SHT_CreateSecondaryIndex`: Στο SHT δημιουργούμε αρχικά ένα info header block που βρίσκεται στην θέση 0 των blocks. Αυτό περιέχει το `SHT_info` του αρχείου. Επίσης έχω προσθέσει και μια μεταβλητή (first_header_block) που περιέχει την θέση του πρώτου header block. Έπειτα δημιουργεί το πρώτο header block και βάζει τη θέση του στη μεταβλητή first_header_block του header info block που φτιάχτηκε προηγουμένως. Έπειτα φτιάχνω όσα buckets χρειάζονται ανάλογα με τα buckets που έχει δώσει ο χρήστης από την γραμμή εντολών και βάζω τις θέσεις των buckets στα header blocks επειδή όπως είπαμε το κάθε header block χωράει 126 θέσεις buckets αν δεν χωράνε οι θέσεις των buckets στο 1 header block δημιουργούνται και άλλα header blocks. Όταν γίνεται αυτό βάζω την θέση του καινούριου header block και στη μεταβλητή nextHeaderBlock του προηγούμενου.

`SHT_OpenSecondaryIndex`: Η open index ανοίγει το πρώτο μπλοκ του αρχείου, αντιγράφει το `SHT_info` σε ένα δείκτη και τον επιστρέφει.

`SHT_CloseSecondaryIndex`: Η close index απλά κλείνει το αρχείο και αποδεσμέβει τις δομές που δεσμεύτηκαν.

`SHT_SecondaryInsertEntry`: Ξεκινώ την διαδικασία για να βάλω το secondary record στο αρχείο. Αυτή η συνάρτηση ξεκινάει υπολογίζοντας το κλειδί του value σύμφωνα με το surname που δόθηκε. Προσθέτοντας όλους τους χαρακτήρες του surname μεταξύ τους και διαιρόντας αυτή την τιμη με τον αριθμό των buckets παίρνουμαι το κλειδί. Παίρνοντας το div, του κλειδιού δια το 126

(αριθμό θέσεων bucket που υποστηρίζει ένα header block) παίρνουμε ποιο header block έχει τη θέση του bucket με κλειδί το key που υπολογίσαμε. Και έπειτα παίρνουμε το key mod 126 για να δούμε που μέσα στο header block βρίσκεται η θέση του bucket με κλειδι το key που υπολογίσαμε. Αφου βρούμε τη θέση του bucket που αντιστοιχεί στο κλειδί ξεκινώ να ψάχνω κάθε μπλοκ μέσα στο bucket για κένη θέση που μπορεί να μπει το secondary record που θέλουμε να βάλουμε. Αν δεν υπάρχει θέση σε κανένα block του bucket τότε φτιάχνουμε ένα καινούργιο και βάζουμε το secondary record εκεί. Επίσης στο τελευταίο μπλοκ του bucket πριν φτιάξουμε το καινούργιο βαζουμε στη μεταβλητή nextBlock την θέση του καινούργιου block που μόλις φτιάξαμε. Τέλος όπου και να μπεί το secondary record επιστρέφουμε την θέση του block στο οποίο μπήκε.

`SHT_SecondaryGetAllEntries`: Αυτή η συνάρτηση ξεκινάει υπολογίζοντας το κλειδί του value σύμφωνα με το surname που δόθηκε. Προσθέτοντας όλους τους χαρακτήρες του surname μεταξύ τους και διαιρόντας αυτή την τιμη με τον αριθμό των buckets παίρνουμαι το κλειδί. Μετά παιρνοντας το κλειδί δια το 126 (αριθμό θέσεων bucket που υποστηρίζει ένα header block) παίρνουμε ποιο header block έχει τη θέση του bucket με κλειδί το key που υπολογίσαμε. Και έπειτα παίρνουμε το key mod 126 για να δούμε που μέσα στο header block βρίσκεται η θέση του bucket με κλειδί το key που υπολογίσαμε. Όταν βρούμε τη θέση του bucket ψάχνουμε σε όλα τα blocks του bucket για το record με surname == value. Αν βρω το record τότε παίρνω την τιμή του block id στο οποίο βρίσκεται το αντίστοιχο record στο primary hash file και ψάχνω σε αυτό το block του primary hash για το record με surname == value. Όταν το βρω εμφανίζω τα στοιχεία του record και επιστρέφω των αριθμό των blocks που χρειάστηκε να διαβάσω στο secondary hash file για να το βρω. Αν δεν βρω το record σε οποιοδήποτε στάδιο της αναζήτησης ή αν εμφανιστεί κάποιο error τότε επιστρέφω -1.

HT:

<u>Γενική υλοποίηση:</u>
Η HT αποτελείται από ένα header info block που έχει το `HT_info` το οποίο δείχνει στο πρώτο header block. Τα header blocks περιέχουν τις διευθυνσεις τών buckets αλλά και τον αριθμό των bytes στο block και τη θέση του επόμενου header block. Επίσης ο συνολικος αριθμός των buckets που μπορουν να αντιπροσωπεύει ένα header block είναι (512-8)/4=126 όπου 8 είναι ο χώρος που παίρνουν τα αλλα στοιχεία του header block, 512 ο συνολικός χώρος του header block και 4 το μέγεθος ενός integer. Τέλος υπάρχουν και τα buckets όπου έχουν 1 η περισσότερα blocks. Σε αυτά τα blocks των buckets αποθηκεύονται τα records και άλλες χρήσιμες πληροφορίες για το block όπως τον αριθμό των records μέσα στο block, τον αριθμό των bytes που απομένουν μέσα στο block, το κλειδί του bucket και την θέση του επόμενου block αν υπάρχει. Αν δεν υπάρχει επόμενο block τότε η μεταβλητή θα έχει την τιμή -1.
Συναρτήσεις:

`HT_CreateIndex`: Στο HT δημιουργούμε αρχικά ένα info header block που βρίσκεται στην θέση 0 των blocks. Αυτό περιέχει το `HT_info` του αρχείου. Επίσης έχω προσθέσει και μια μεταβλητή (first_header_block) που περιέχει την θέση του πρώτου header block. Έπειτα δημιουργεί το πρώτο header block και βαζει τη θέση του στη μεταβλητή firtst_header_block του header info block που φτιαχτηκε προηγουμένως. Έπειτα φτιάχνω όσα buckets χρειάζονται ανάλογα με τα buckets που έχει δώσει ο χρήστης από την γραμμή εντολών και βάζω τις θεσεις των buckets στα header blocks επειδή όπως είπαμε το κάθε header block χωράει 126 θέσεις buckets αν δεν χωράνε οι θέσεις των buckets στο 1 header block δημιουργούνται και άλλα header blocks. Όταν γίνεται αυτό βάζω την θέση του καινούριου header block και στη μεταβλητή nextHeaderBlock του προηγούμενου.

`HT_OpenIndex`: Η open index ανοίγει το πρώτο μπλοκ του αρχείου, αντιγράφει τον HT_info σε ένα δείκτη και τον επιστρέφει.

`HT_CloseIndex`: Η close index απλά κλείνει το αρχείο.

`HT_InsertEntry`: Αυτή η συνάρτηση ξεκινάει ψάχνοντας με τη βοήθεια της `HT_GetAllEntries` αν υπάρχει ήδη κάποιο record μέσα στο αρχείο με το ιδιο id με το record που θέλουμε να βάλουμε. Αν ήδη υπάρχει τότε εκτυπώνω αντίστοιχο μήνυμα. Αλλιώς ξεκινώ την διαδικασία για να βάλω το record στο αρχείο. Συνεχίζουμε υπολογίζοντας το κλειδί του value που δόθηκε. Παίρνοντας το div, του κλειδιού δια το 126 (αριθμό θέσεων bucket που υποστηρίζει ένα header block) παίρνουμε ποιο header block έχει τη θέση του bucket με κλειδι το key που υπολογίσαμε. Και έπειτα παίρνουμε το key mod 126 για να δούμε που μέσα στο header block βρίσκεται η θέση του bucket με κλειδι το key που υπολογίσαμε. Αφου βρούμε τη θέση του bucket που αντιστοιχεί στο κλειδί ξεκινώ να ψάχνω κάθε μπλοκ μέσα στο bucket για κένη θέση που μπορεί να μπει το record που θέλουμε να βάλουμε. Αν δεν υπάρχει θέση σε κανένα block του bucket τότε φτιάχνουμε ένα καινούργιο και βάζουμε το record εκεί. Επίσης στο τελευταίο μπλοκ του bucket πριν φτιάξουμε το καινούργιο βαζουμε στη μεταβλητή nextBlock την θέση του καινούργιου block που μόλις φτιάξαμε. Τέλος όπου και να μπεί το record επιστρέφουμε την θέση του block στο οποίο μπήκε.

`HT_DeleteEntry`: Αυτή η συνάρτηση ξεκινά ψαχνοντας με τη βοήθεια της `HT_GetAllEntries` αν υπάρχει record με το δοσμένο value για id. Αν δεν υπάρχει αυτό σημαίνει ότι δεν μπορεί να σβηστεί οπότε εκτυπώνεται αντίστοιχο μήνυμα. Αν υπάρχει η `HT_GetAllEntries` θα επιστρέψει το id του block στο οποίο βρίσκεται το record. Η delete θα πάρει αυτό το id και θα ανοίξει το αντίστοιχο block. θα ψάξει μέσα στο block για το record. Όταν το βρει θα κοιτάξει αν βρίσκεται στην τελευταία θέση του block. Αν ναι θα μειώσει τον αριθμό των records στο block και θα αυξήσει το χώρο που υπολείπεται στο block κατα το μέγεθος του record. Με αυτό τον τρόπο ουσιαστικά το record σβήνεται. Αν όμως το record δεν βρίσκεται στην τελευταία θέση του block τότε παίρνω το τελευταίο record του block και το βάζω στη θέση αυτού που θέλω να σβήσω. Μειώνω τώρα τον αριθμό τον records στο block και αυξάνω την χωρητικότητα του block κατα το μέγεθος του record. Τέλος επιστρέφω 0 ως τιμή επιτυχίας. Αν για κάποιο λόγο δεν βρει το record στο block επιστρέφει -1.

`HT_GetAllEntries`: Αυτή η συνάρτηση ξεκινάει υπολογίζοντας το κλειδί του value που δόθηκε. Παίρνοντας το div του κλειδιού δια το 126 (αριθμό θέσεων bucket που υποστηρίζει ένα header block) παίρνουμε ποιο header block έχει τη θέση του bucket με κλειδι το key που υπολογίσαμε. Και έπειτα παίρνουμε το key mod 126 για να δούμε που μέσα στο header block βρίσκεται η θέση του bucket με κλειδι το key που υπολογίσαμε.Όταν βρούμε τη θέση του bucket ψάχνουμε σε όλα τα blocks του bucket για το record με id == value. Αν βρω το record τότε επιστρέφω το το id του block στο οποίο βρίσκεται. Αν δεν υπάρχει σε κανένα από τα blocks του bucket τότε επιστρέφω -1.

Μεταγλώττιση και εκτέλεση:

Για την μεταγλώττιση πρέπει στο φάκελο που είναι το Makefile να πατήσετε make. Αυτό μεταγλωτίζει τον κώδικα και το εκτελέσιμο αρχείο τοποθετείται στο φάκελο bin. Για την εκτέλεση του προγράμματος εκτελείται η εντολή “./bin/main (int numOfRec) SHT (int numOfBucket)” όπου το (int numOfRec) είναι ένας ακέραιος και συγκεκριμένα μπορεί να πάρει τιμές (1,5,10,15) και

καθορίζει το αρχείο από το οποίο θα διαβάσει το πρόγραμμα μας. Επίσης το (int numOfBucket) είναι ο αριθμός των bucket και πρέπει να είναι ακέραιος αριθμός θετικος. Δηλαδή αν θέλω να τρέξω το πρόγραμμα με αρχείο εισόδου /examples/records1K.txt και 500 buckets θα τρέξω τη εντολή ./bin/main 1 SHT 500. Η αν θέλω να τρέξω το πρόγραμμα με αρχείο εισόδου το /examples/records10K.txt και 1000 buckets θα γράψω ./bin/main 10 SHT 1000 . Το τελευταίο όρισμα (1000) είναι ο αριθμός των bucket και πρέπει να είναι ακέραιος αριθμός θετικος.