Ilias-Elias Barmpar sdi1200118

Δημήτριος Κρικώνης sdi1400081

*added -lm on makefile

Για το Βημα 1 - Δημήτριος:

γενικα ισχυει:
first_block_num = ο αναγνωριστικός αριθμός block (block_num) του πρωτου μπλοκ
first_block = η θεση στον πινακα των blocks και των data οπου βρισκεται το block με αριθμο first_block_num
το ιδιο ισχυει για το last_block_num και το last_block

sort_file.c

SR_SortedFile:
-ελέγχουμε αν το bufferSize που δοθηκε ειναι εγκυρο
δηλαδη μεταξυ 3 και του μεγιστου BF_BUFFER_SIZE
-ανοιγουμε το input file με τα unsorted data
-δημιουργουμε ενα temp αρχειο step1.db στο οποίο περναμε
όλα τα blocks και τα records απο το input file
-βρισκουμε ποσα ειναι τα group των μπλοκ αναλογα με τον αριθμο των μπλοκ του αρχείου
-δημιουργουμε ενα πινακα με δεικτες σε block με μεγεθος bufferSize, αντιστοιχα εναν πινακα με τα data τους
και αρχικοποιούμε τα blocks
-για καθε group βρισκουμε τον αριθμο του πρωτο μπλοκ, το offset του πρωτου record σε αυτο,
τον αριθμο του τελευταιο μπλοκ και το το offset του τελευταίου record σε αυτο
-κανουμε pin στη μνήμη ολα τα blocks του group
-θετουμε τη θεση του πρωτου και του τελευταιου block στον πινακα
-καλουμε την quickSort με αυτα τα στοιχεια

step1_functions.c

record_cmp:
-συγκρινει δυο records αναλογα με το field που καθοριζεται απο το fieldNo

swap:
-αντιστρέφει τις θεσεις δυο records τα οποια βρίσκονται στο ιδιο η διαφορετικο μπλοκ

partition:
-
στο τελευταιο μπλοκ βρισκουμε την εγγραφη που βρισκεται στο high και την βαζουμε στο pivot ωστε να τη συγκρινουμε με καθε record
-
αρχικοποιηούμε ενα index και το μπλοκ στο οποιο βρισκεται, το οποιο θα ειναι η θεση που θα πηγαινουν τα records που ειναι μικροτερα απο το pivot
-παιρνουμε καθε μπλοκ απο τον πινακα και οριζουμε ποσα records αυτου του μπλοκ θα ελεγξουμε
-για καθε ενα απο αυτα τα record τα συκρινουμε με το pivot και αν ειναι μικροτερα:
  -αν το index δε βρισκεται στο τελος του μπλοκ μετακινουμε μια θεση το index και το κανουμε swap με αυτο το record
  -αλλιως προχωραμε στο επομενο μπλοκ το κανουμε index_block, κανουμε index το πρωτο record αυτου
  και κανουμε swap το index με το record που ειναι μικροτερο απο το pivot
-η διαδικασια συνεχιζεται για καθε record μεταξυ low και high και τελικα ολα τα records που βρισκονται
πριν το index ειναι μικροτερα απο το pivot και ολα οσα ειναι μετα ειναι μεγαλυτερα

-στο τελος κανουμε swap το pivot με το index
-επιστρεφουμε τη θεση index που βρισκεται το pivot και αριστερα του ειναι ολα μικροτερα και δεξια ολα μεγαλυτερα

quickSort:
-Η συνάρτηση λειτουργει αναδρομικά
-
ελεγχουμε αν εκει που εχουμε φτασει, η θεση του πρωτου μπλοκ στον πινακα των blocks (δηλαδη εμμεσα και ο αριθμος το υ μπλοκ) ειναι μικροτερος απο του δευτερου
ή αλλιως αν ειμαστε στο ιδιο μπλοκ αν το offset του πρωτου record ειναι μικροτερο απο του δευτερου για να συνεχισουμε τη διαδικασια της quickSort
-καλούμε τη συναρτηση partition η οποια μας επιστρέφει το μπλοκ και το offset του index
απο το οποιο αριστερα υπαρχουν ολα τα μικροτερα records και δεξια ολα τα μεγαλυτερα
-τωρα θελουμε να καλεσουμε αναδρομικα την quicksort για αυτα τα δυο κομματια
-για το αριστερο κομμάτι το low παραμενει αυτο που ηταν ενω το high παει στο προηγουμενο record απο το index
αν το index δεν ειναι στο πρωτο record καποιου μπλοκ οπου δεν υπαρχει προηγουμενο record,
αλλιως παμε στο προηγουμενο μπλοκ και βαζουμε το high στο τελευταιο record του
-καλουμε αναδρομικα την quicksort με αυτα τα στοιχεια για το αριστερο κομματι
-για το δεξι κομμάτι το high παραμενει αυτο που ηταν ενω για το low κανουμε αντιστοιχη διαδικασια με πριν
-καλουμε αναδρομικα την quicksort με αυτα τα στοιχεια για το δεξι κομματι

Για το Βημα 2 - Ilias-Elias :

Helper:
<> = actual header and source files
"groupsOf" = The maximum amount of blocks that can currently exist on a group.
"groupCount" = The ammount of groups there are in the input block we are using.

-Contribution-

<sort_file>
SR_SortedFile:
-Creates a second file and allocates as many blocks as the first one has.

-Creates arrays for the blocks and the info necessary for their operations.

-Setting internal/external conditions:
  <general_use_functions>
  switchIntegers: Simple integer switch.

  strictRoundUp: Rounds a double by only going up. (e.g 1.00...001 -> 2)

  setExternalCondition: Sets the amount of iterations by computing the
      logarithm of blockCount/bufferSize with a
      base of bufferSize-1 and then using strictRoundUp.
  setInternalCondition: Sets the amount of iterations by computing the
      division of blockCount by groupsOf, rounding it,
      which makes "groupCount" and then dividing it with
      (bufferSize-1) and rounding it.
  setFieldOffSet: Simple case dependant setting.

  printers: Some simple block/record related printers.

-Merging process: We've got our two temp files and we switch them, having one act as the inputfile
    and the other as the outputfile whilst merging. At the end, we check which one is
    the final one (rename it accordingly) and we delete the other one.

<buffer>
bufferSetup: Initializes and pins the input blocks our buffer will be using
on it's sorting phase. If there aren't anymore blocks and our
buffer still has input blocks available we just null our dataArray
and inputInfoArray to know when to avoid such groups.

T

buffersortexplanation: For every external iteration the first output block is
already pinned, so we begin by finding the minimum of
the records based on the fieldNo then we proceed to
copying it from the responsible input block to our output
block and finally we run two checks for the inputblock
affected and the output block.

<bufferSort>
sortFindMin: A minimum search that skips empty blocks, once at the start
(if an input block is empty its recordCount is equal to 0)
and if our min is now bigger than the ammount of available
input blocks (bufferSize-1) we've reached the end. Otherwise
we do a linear minimum search starting from minimum, whilst
skipping once more, input blocks that have been nulled.

sortCheckInputBlock: Checks the affected input block once we know there are no
other records left. If there are other blocks it gets
the next and updates the necessary info. If there aren't
this block gets nulled.

sortCheckOutputBlock: We memcpy its outputRecordCount now that we know this block
is about to be unpinned, and update the nec. info. If this
isn't our last output block, we get the next output block.

Για τις υπόλοιπες συναρτησεις του sort_file.c:

SR_CreateFile
- ορίζεται και αρχικοποιηται ένα BF_Block (metadatablock)
- δημιουργεία BF αρχείου με καταλληλο ελεγχο, ανοιγμα του αρχειου και δεσμευση ενός μπλοκ
- εισαγωγη του στοιχειου SRF στο πρωτο μλποκ ως αναγνωριστικο ότι προκειται για sort file
- ορισμός του μπλοκ ως βρωμικο, ξεκαρφίτσωμα και κλεισιμο BF αρχειου

SR_OpenFile
- Άνοιγμα στο BF επιπεδο με το filename του SR αρχειου
- Έλεγχος αν πρόκειται για SR αρχείο, δηλαδή αν το πρώτο μπλοκ του περιέχει το HPF

SR_InsertEntry
- παίρνουμε δυο περιπτώσεις:
1. το αρχειο περιεχει μονο το πρωτο μπλοκ με την πληροφορια για το sort file.
Σε αυτή την περιπτωση δεσμευεται καινουριο μπλοκ, οπου ο αριθμος των εγγραφων του γινεται 1
για την εγγραφη που θα εισαχθεί (θετουμε το offset στο χωρο που πιανει ο αριθμος των εγγραφών στην αρχη του μπλοκ)
2. το αρχειο περιέχει και αλλα μπλοκ με εγγραφές.
Βρίσκουμε ποσα είναι τα μπλοκ (και αρα τον αριθμο του τελευταιου) και ποσες εγγραφες εχει το τελευταίο.
Εδώ παίρνουμε δυο υποπεριπτώσεις.
a. η εγγραφή να χωράει στο τελευταιο μπλοκ.
Θετουμε το offset στο χωρο που πιανουν οι προηγουμενες εγγραφες
και αυξάνουμε τον αριθμο των εγγραφων στην αρχη του μπλοκ.
b. η εγγραφή να μη χωράει στο τελευταιο μπλοκ.
Αποδεσμευουμε το τελευταιο μπλοκ και εισαγουμε ένα καινουριο, δρώντας ανάλογα με την πρωτη πειπτωση.

- εισάγουμε τα μελη της εγγραφης στο μπλοκ, που ειτε δημιουργήσαμε ειτε πηραμε αναλογα με την περιπτωση, και με βαση το αντίστοιχο offset που προέκυψε από μια από τις περιπτώσεις.
- θετουμε το μπλοκ βρωμικο και το αποδεσμευουμε.

SR_PrintAllEntries
- βρίσκουμε τον αριθμό των μπλοκ του αρχειου
- για κάθε μπλοκ θετεουμε το offset στο χωρο που πιανει ο αριθμος των εγγραφών στην αρχη του μπλοκ και ξεκιναμε να τυπώνουμε κάθε εγγραφη αυξανοντας κάθε φορά το offset κατά το μεγεθος της εγγραφης, μεχρι να φτασουμε στο τελος του μπλοκ
- αποδεσμευουμε κάθε μπλοκ