Πληροφορική & Τηλεπικοινωνίες

Υλοποίηση Συστημάτων Βάσεων Δεδομένων Χειμερινό Εξάμηνο 2024 – 2025

Διδάσκων Θεόφιλος Μαΐλης

Άσκηση 3 - Προθεσμία: 8/01/2025

Η 3η εργασία του μαθήματος, με ημερομηνία παράδοσης την Τετάρτη, 8 Ιανουαρίου, είναι προαιρετική. Η βαρύτητα των εργασιών διαμορφώνεται ως εξής: 1η: 0.5/10, 2η: 2.5/10, 3η: 2/10. Οι ομάδες που θα την ολοκληρώσουν θα βαθμολογηθούν με 50% από τις εργασίες και 50% από τη γραπτή εξέταση. Διαφορετικά, ο τελικός βαθμός θα προκύψει με συντελεστή 30% από τις εργασίες και 70% από τη γραπτή εξέταση.

Σκοπός της Εργασίας

Σκοπός αυτής της εργασίας είναι η κατανόηση του αλγορίθμου εξωτερικής ταξινόμησης με συγχώνευση σε ένα Συστήματα Βάσεων Δεδομένων, χρησιμοποιώντας τις βιβλιοθήκες που αναπτύξατε στις προηγούμενες εργασίες σας. Ο συγκεκριμένος αλγόριθμος επιτρέπει την ταξινόμηση των εγγραφών ενός αρχείου σωρού όταν δεν υπάρχει διαθέσιμη RAM μνήμη για την αποθήκευση ολόκληρου του αρχείου.

Ένας επιπλέον στόχος της εργασίας είναι η εξοικείωση των φοιτητών με τα μοντέλα γλωσσικής επεξεργασίας και η χρήση τους για την παραγωγή εκτελέσιμου κώδικα. Η υλοποίηση της εργασίας θα γίνει με υβριδικό τρόπο, που προκύπτει από τη συγγραφή κώδικα, αλλά και από την αλληλεπίδραση του φοιτητή με το μοντέλο γλωσσικής επεξεργασίας ChatGPT (https://chat.openai.com/). Το ChatGPT είναι ένα μοντέλο γλωσσικής επεξεργασίας που αναπτύχθηκε από την OpenAI, ανήκει στην οικογένεια των μοντέλων GPT (Generative Pre-trained Transformer) και σχεδιάστηκε ειδικά για τη δημιουργία κειμένου ή κώδικα μετά από προτάσεις και ερωτήσεις στα πλαίσια αλληλεπίδρασης του χρήστη. Το μοντέλο εκπαιδεύεται σε μια ποικιλία κειμένων από το διαδίκτυο και είναι ικανό να κατανοεί το πλαίσιο, να δημιουργεί συνεκτικές και συμφραζόμενα συναφή απαντήσεις και να εκτελεί διάφορες γλωσσικές εργασίες. Οι χρήστες μπορούν να αλληλεπιδρούν με το ChatGPT παρέχοντας ερωτήσεις ή προτάσεις, και το μοντέλο δημιουργεί απαντήσεις βασισμένες στα δεδομένα εκπαίδευσής του

Υλοποίηση

Ο αλγόριθμος εξωτερικής ταξινόμησης με συγχώνευση αποτελείται από το πέρασμα της ταξινόμησης και τα περάσματα, κατά τα οποία γίνεται η συγχώνευση ταξινομημένων τμημάτων που λέγονται συρμοί. Λεπτομέρειες του αλγορίθμου μπορούν να βρεθούν στις διαφάνειες της εργασίας, καθώς και στις διαφάνειες του μαθήματος. Οι συναρτήσεις και οι δομές για το στάδιο ταξινόμησης θα έχουν τα προθέματα sort_ και για το στάδιο συγχώνευσης merge_. Οι συναρτήσεις και οι δομές που σχετίζονται με ένα συρμό (chunk) θα έχουν το πρόθεμα CHUNK_. Θεωρούμε ότι ο αλγόριθμος ταξινομεί τις εγγραφές ενός αρχείου σωρού και θεωρούμε ότι υπάρχει υλοποιημένη η αντίστοιχη βιβλιοθήκη για το αρχείο σωρού.

Βιβλιοθήκη HP_

Συγκεκριμένα, θεωρούμε ότι στα πλαίσια της πρώτης άσκησης έχουν υλοποιηθεί οι ακόλουθες συναρτήσεις. Θεωρείστε την παραδοχή, ότι οι εγγραφές ξεκινούν από το **block 1** του αρχείου σωρού, ενώ στο **block 0** έχετε μεταδεδομένα.

```
/*The function HP_CreateFile is used to create and appropriately initialize an
empty heap file with the given fileName. If the execution is successful, it
returns 0; otherwise, it returns -1.*/

int HP_CreateFile(char *fileName);

/* The function HP_OpenFile opens the file with the name filename. The variable
*file_desc refers to the opening identifier of this file as derived from
BF_OpenFile.*/

int HP_OpenFile(char *fileName, int *file_desc);

/* The function HP_CloseFile closes the file identified by the descriptor
file_desc. If the operation is successful, it returns 0; otherwise, it returns
-1.*/

int HP_CloseFile(int file_desc);

/* The function HP_InsertEntry is used to insert a record into the heap file.
The identifier for the file is file_desc, and the record to be inserted is
specified by the record structure. If the operation is successful, it returns
1; otherwise, it returns -1.*/

int HP_InsertEntry(int file_desc, Record record);

/* The function HP_GetRecord is designed to retrieve a record from a heap file
specified by the file descriptor file_desc. It takes three parameters: blockId,
which indicates the block from which to retrieve the record, cursor, which
specifies the position of the record within that block, and record, which is a
pointer to a Record structure. The retrieved record will be stored in the memory
location pointed to by the record parameter.*/

int HP_GetRecord( int file_desc, int blockId, int cursor, Record* record);



/* The function HP_UpdateRecord updates or sets a record in a heap file specified
by the file descriptor file_desc. It takes four parameters: blockId, which
indicates the block where the record will be updated, cursor, which specifies
the position of the record within that block, and record, which is the new data
that will replace the existing record at the specified location. If the operation
is successful, the function returns 1; otherwise, it returns -1.*/

int HP_UpdateRecord(int file_desc, int blockId, int cursor,Record record);
```

```
/* The function HP_Unpin is designed to release the block identified by blockId
in the heap file associated with the descriptor file_desc. If the unpin is
successful, it returns 0; otherwise, it returns -1.*/

int HP_Unpin(int file_desc, int blockId);

// Prints all entries(records) stored in the heap file.

int HP_PrintAllEntries(int file_desc);

// Retrieves the current record count in a specified block.

int HP_GetRecordCounter(int file_desc, int blockId);

// Returns the identifier of the last block in the heap file.

int HP_GetIdOfLastBlock(int file_desc);

// Retrieves the number of records that can fit in a block of the heap file.

int HP_GetMaxRecordsInBlock(int file_desc);

// Prints all entries(records) contained in the specified block of the heap
file.

int HP_PrintBlockEntries(int file_desc, int blockId);
```

Σημείωση: Οι συναρτήσεις `HP_UpdateRecord` και `HP_GetRecord` προτείνονται για την απλοποίηση της υλοποίησης της εργασίας. Θα πρέπει να συνοδεύονται από το αντίστοιχο `HP_Unpin`.

Βιβλιοθήκη SORT_

```
/* Determines if two records should be swapped during sorting, returning true
if the order needs adjustment.*/

bool shouldSwap(Record* rec1,Record* rec2);

/* Sorts the contents of a file identified by 'file_desc' in chunks, where each
chunk contains 'numBlocksInChunk' blocks. The sorting is performed in-place
within each chunk, using an appropriate sorting algorithm.*/

void sort_FileInChunks(int file_desc, int numBlocksInChunk);

/* Sorts records within a CHUNK in ascending order based on the name and surname
of each person. */

void sort_Chunk(CHUNK* chunk);
```

Βιβλιοθήκη MERGE_

```
/* Merges every b chunks of size chunkSize from the input file to the specified
output file. The function takes input file and output file descriptors, chunk
size, and the number of chunks to merge. It should internally use a
CHUNK_Iterator and a CHUNK_RecordIterator.*/
```

```
void merge(int input_FileDesc, int chunkSize, int bWay, int output_FileDesc );
```

Σημείωση: σε κάθε πέρασμα, έχουμε σαν είσοδο ένα αρχείο που αποτελείται από k συρμούς (ταξινομημένα τμήματα), ο καθένας από τους οποίους περιέχει m blocks[1]. Όταν πραγματοποιήσουμε b-way συγχώνευση, θα έχουμε σαν αποτέλεσμα k/b συρμούς που ο καθένας θα αποτελείται από m*b blocks. Τα περάσματα σταματάνε όταν καταλήξουμε να έχουμε μόνο έναν συρμό. Θέλουμε για κάθε κύκλο να δημιουργείτε ένα καινούργιο αρχείο που θα περιέχει το ενδιάμεσο αποτέλεσμα.

Βιβλιοθήκη CHUNK_

Σημείωση: Η προτεινόμενη βιβλιοθήκη chunk είναι ενδεικτική, σε περίπτωση που ακολουθήσετε κάποια εναλλακτική προσέγγιση για την δημιουργία των συναρτήσεων για merge, sort η λύση σας θα θεωρηθεί αποδεκτή.

```
/* Represents a chunk of records in a file, defining the file descriptor, starting and ending block IDs, and the counts of records and blocks in the chunk. Useful for managing and sorting records within specific chunks. */

typedef struct {

    int file_desc;

    int from_BlockId;

    int to_BlockId;

    int recordsInChunk;

    int blocksInChunk;

} CHUNK;

/* Represents an iterator for traversing chunks within a file, storing the file descriptor, current block, last block ID, and the total number of blocks in each chunk. Useful for efficiently iterating over file chunks.*/

typedef struct {

    int file_desc;

    int current;

    int lastBlocksID;

    int blocksInChunk;

} CHUNK_Iterator;

/* Creates a ChunkIterator for efficient traversal of chunks within a file, specified by the file descriptor. The iterator is configured with a defined range of blocks (usually starting from block 1), along with the size of each chunk and the maximum number of records in each block.*/

CHUNK_Iterator CHUNK_CreateIterator(int fileDesc, int blocksInChunk);

/* Retrieves the next CHUNK in the sequence as per the provided CHUNK_Iterator. */

int CHUNK_GetNext(CHUNK_Iterator *iterator,CHUNK* chunk);
```

[1] Εκτός του τελευταίου συρμού

```
/* Retrieves the ith record from a CHUNK of blocks in a heap file. Returns 0 if successful, populating the 'record' parameter; otherwise, -1. Assumes sequential ordering of records within the chunk.*/

int CHUNK_GetIthRecordInChunk(CHUNK* chunk,  int i, Record* record);

/* Updates the ith record in a chunk. Returns 0 if successful; -1 if unsuccessful. Facilitates efficient and controlled updates within a chunk.*/

int CHUNK_UpdateIthRecord(CHUNK* chunk,  int i, Record record);

/* This function is used to print the records within a chunk.*/

void CHUNK_Print(CHUNK chunk);

/* Iterates through records in a CHUNK, encapsulating the id of the current block and a cursor in that block. */

typedef struct CHUNK_RecordIterator {

    CHUNK chunk;

    int currentBlockId;

    int cursor;

} CHUNK_RecordIterator;

/* Creates a record iterator for efficient traversal within a CHUNK. */

CHUNK_RecordIterator CHUNK_CreateRecordIterator(CHUNK *chunk);

/* Function to get the next record from the iterator. */

int CHUNK_GetNextRecord(CHUNK_RecordIterator *iterator,Record* record);
```

Έλεγχος ορθότητας του προγράμματός σας

Η main που σας δίνεται σε αυτή την εργασία είναι ενδεικτική. Καλείστε να δημιουργήσετε δικές σας συναρτήσεις/mains που θα αποδεικνύουν την ορθή λειτουργία του προγράμματός σας.

- Πειραματιστείτε με τις διαφορετικές παραμέτρους της εξωτερικής ταξινόμησης με συγχώνευση. Για μεγάλο αριθμό εγγραφών, πόσα περάσματα έχω όταν χρησιμοποιήσω ταξινόμηση σε συρμούς των 5 και συγχώνευσης ανά 2 (2-way συγχώνευση); Πόσα περάσματα έχω όταν χρησιμοποιήσω ταξινόμηση σε συρμούς των 5 και συγχώνευση ανά 10 (10-way συγχώνευση);

Αρχεία που σας δίνονται

Στα αρχεία που σας δίνονται, θα βρείτε δύο φακέλους που περιέχουν το project που θα πρέπει να επιστρέψετε. Το project έχει την παρακάτω δομή:

- bin: Περιλαμβάνει τον κώδικα των εκτελέσιμων που δημιουργούνται.
- build: Περιέχει όλα τα object files που δημιουργούνται κατά τη μεταγλώττιση.
- include: Περιέχει όλα τα αρχεία κεφαλίδας που θα έχει το project σας, όπως τα αρχεία 'bf.h' και 'hp_file.h.'
- lib: Περιλαμβάνει όλες τις βιβλιοθήκες που θα χρειαστείτε για το project σας, όπως το αρχείο 'libbf.so,' που αποτελεί τη βιβλιοθήκη για το επίπεδο BF.
- src: Περιλαμβάνει τα αρχεία κώδικα (.c) της εφαρμογής σας.
- examples: Σε αυτόν τον φάκελο θα βρείτε ένα αρχείο main ('sort_main.c'), το οποίο περιέχει κώδικα που εκτελεί τις συναρτήσεις που πρέπει να υλοποιήσετε.

- Επίσης, σας παρέχεται ένα αρχείο Makefile για το αυτόματο compile των κωδικών σας.

Πράγματα που πρέπει να προσέξετε

- Όταν θα κάνετε χρήση των συναρτήσεων **`HP_GetRecord`** και **`HP_UpdateRecord`** θα πρέπει να κάνετε, όταν χρειάζεται και χρήση της **`HP_Unpin`**. Διαφορετικά θα γεμίσει ο Buffer σας.

Παράδοση εργασίας

- Η εργασία αυτή είναι αυστηρά ομαδική (2 ή 3 ατόμων).
- Γλώσσα υλοποίησης: C / C++ χωρίς χρήση βιβλιοθηκών που υπάρχουν μόνο στην C++.
- Περιβάλλον υλοποίησης: Linux (gcc 5.4+).
- Μοντέλο γλωσσικής επεξεργασίας: ChatGPT 3.5.
- Περιβάλλον εξέτασης: Η εργασία θα εξεταστεί στα linux του di.uoa και καλείστε να έχετε ελέγξει ότι εκτελείται ορθώς στο συγκεκριμένο περιβάλλον.

Παραδοτέα

1. Όλος ο φάκελος μαζί με αρχείο Makefile που θα κάνει link τον κώδικά σας με την ήδη υπάρχουσα main αλλά και με τις υπόλοιπες που θα υλοποιήσετε.
2. Ένα README που:
    a. Θα εξηγείτε τις σχεδιαστικές επιλογές ή τυχόν παραδοχές που έχετε κάνει (μισή σελίδα).
    b. Θα αναφέρετε τυχόν δυσλειτουργίες του προγράμματος σας που έχετε παρατηρήσει.
    c. Για κάθε συνάρτηση ή δομή, να μας δώσετε το αντίστοιχο σύνδεσμο που περιέχει την συνομιλία σας με το ChatGPT από την οποία προέκυψε η συγκεκριμένη συνάρτηση ή δομή. Ο σύνδεσμος αυτός προκύπτει από την κεντρική σελίδα του ChatGPT χρησιμοποιώντας το κουμπί στο δεξί μέρος μιας συνομιλίας:

    

    Προσοχή! ΜΗΝ διαγράψετε την συνεδρία γιατί θα χαθεί και η συγκεκριμένη συνομιλία με αποτέλεσμα να χάσετε μονάδες από την βαθμολογία σας.

    d. Για κάθε συνάρτηση θα αναφέρετε συνοπτικά τις αλλαγές που κάνατε στον παραγόμενο από το ChatGPT κώδικα.
- Η παράδοση θα γίνεται ανά ομάδα εργασίας.