

Φασούλας Θεόδωρος
ΑΕΜ:2096
email: theodoft@csd.auth.gr

Γεωργιάδης Νικόλαος
ΑΕΜ:2043
email: nsgeorgi@csd.auth.gr

ΕΡΓΑΣΙΑ ΤΕΧΝΟΛΟΓΙΑΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Η εργασία αυτή έχει σαν στόχο την υλοποίηση κάποιων βασικών λειτουργιών ενός DBMS .Πιο συγκεκριμένα υλοποιούμε συναρτήσεις που αφορούν την ταξινόμηση , την απαλοιφή πολλαπλών εμφανίσεων και την σύνδεση με συγχώνευση.Ο κώδικας είναι γραμμένος σε C++ και οι συναρτήσεις λειτουργούν σε περιορισμένο χώρο μνήμης .Χρησιμοποίησα 3 διαφορετικές εκδοχές.Η πρώτη έχει σαν περιορισμένο χώρο μνήμης 400 MB και το σύνολο δεδομένων καταλαμβάνει 4 GB στο δίσκο(εφόσον τα αποτελέσματα είναι ορθά και σωστά για αυτό τον αριθμό των GB τότε θα έχουμε θετικά αποτελέσματα και για μεγαλύτερης τάξης δεδομένα.Εδώ βλέπουμε πως δουλεύουν οι συναρτήσεις μας σε μεγάλο σύνολο δεδομένων με αρκετά μεγάλο χώρο δεσμευμένο στη RAM. Στην δεύτερη εκδοχή έχουμε σαν χώρο μνήμης 200 MB και το σύνολο δεδομένων καταλαμβάνει 4GB.Εδώ δοκιμάζουμε κατά πόσο καλά δουλεύουν οι συναρτήσεις σε μικρότερο χώρο στην δευτερεύουσα μνήμη. Στην Τρίτη και τελευταία εκδοχή έχουμε μικρότερο σύνολο δεδομένων 2 GB με 400 MB στην δευτερεύουσα μνήμη.Εδώ εξετάζουμε πως δουλεύει ο αλγόριθμός έχοντας ένα μικρό σύνολο δεδομένων.

External Merge Sort

Mergesort(..)

Στην συνάρτηση αυτή πραγματοποιούμε εξωτερική ταξινόμηση.Αυτό το πετυχαίνουμε διαβάζοντας `nmem_blocks` από το αρχικό μας αρχείο ,αποθηκεύουμε τα δεδομένα στον buffer μας ο οποίος καταλαμβάνει X blocks ,ταξινομούμε τον buffer με την βοήθεια της συνάρτησης `Quicksort()` και δημιουργούμε `count_blocks/nmem_blocks` ταξινομημένα κομμάτια -αρχεία .Τέλος καλούμε την συνάρτηση `Merging()` όπου πέρνουμε αυτά τα ταξινομημένα τμήματα και τα συγχωνεύουμε με (`nmem_blocks`) –way Merge .

Αρχικά ξεκινάμε από την `MergeSort()` και μπαίνουμε σε έναν βρόγχο while όπου εκτελούμε επανάληψεις μέχρι να φτάσουμε στο τέλος του αρχείου.Σε κάθε επανάληψη ταξινομούμε τον πίνακα `buffer_rec[]` ο οποίος περιέχει ένα πλήθος από `num` στοιχεία και έτσι δημιουργούμε σε κάθε επανάληψη ένα ταξινομημένο καινούργιο τμήμα-αρχείο.

Στον επόμενο βρόγχο while ο οποίος είναι εσωτερικός του προηγούμενου while ,εκχωρούμε σε κάθε επανάληψη το πεδίο `num` από τον πίνακα `entries[]` στον πίνακα `buffer_rec[]`.Αυτό το κάνουμε για να ταξινομήσουμε όλα τα num πεδία του αρχείου και να δημιουργήσουμε ταξινομημένα τμήματα-αρχεία.Έτσι μέχρι να γεμίσει ο buffer μας δηλαδή μέχρι το

`inmem_blocks` θα εισάγουμε blocks από το αρχείο, Στη συνέχεια αφού γεμίσει ο πίνακας `buffer_rec[]` , θα τον ταξινομήσουμε.

Αφού τελειώσουμε από τον παραπάνω βρόγχο καλούμε την συνάρτηση `Quicksort()` για να ταξινομήσουμε τον πίνακα `buffer_rec[]` με την μέθοδο της ταξινόμησης. Τέλος αφού ταξινομήσουμε και τον πίνακά μας , θα γράψουμε το εκάστοτε ταξινομημένο τμήμα σε ένα ξεχωριστό αρχείο . Αν τελειώσουμε και από το αρχικό while τότε είμαστε σε θέση να μεταβούμε στην επόμενη συνάρτηση την `Merging()`.

Merging(...)

Στη συνάρτηση αυτή αφού έχουμε δημιουργήσει όλα τα ταξινομημένα τμήματα-αρχεία , ήρθε η ώρα να τα συγχωνεύσουμε όλα σε ένα τελικό αρχείο. Αυτό θα το πραγματοποιήσουμε με αυτή τη συνάρτηση η οποία θα κάνει `[inmem_blocks - 1]` –Way Merge. Για παράδειγμα εάν διαθέτουμε 9 blocks στην κύρια μνήμη μας τότε θα εισάγουμε ένα μπλοκ από κάθε ένα από τα 9 ταξινομημένα αρχεία και θα τα εκχωρίσουμε στα 8 μπλοκ που έχουμε και το τελευταίο θα το χρησιμοποιήσουμε σαν αποθηκευτικό χώρο για τα αποτελέσματα του 8 Way-Merge. Στη συνέχεια θα εκτελέσουμε τον 8 Way-Merge και θα εκχωρίσουμε το αποτέλεσμα στο τελευταίο μπλοκ. Αν γεμίσει το μπλοκ αυτό, τότε γράφουμε τα αποτελέσματα στο αρχείο εξόδου. Την διαδικασία αυτή θα την εκτελέσουμε μέχρις ότου εκχωρίσουμε όλα τα μπλοκς από τα ταξινομημένα τμήματα , στον buffer μας. Αυτό θα είναι το δεύτερο πέρασμα. Την διαδικασία αυτή θα την κάνουμε μέχρι να μας έχει απομείνει μόνο ένα αρχείο το οποίο θα είναι και το τελικό αρχείο που θα έχει όλα τα μπλοκς δηλαδή όλα τα `num` πεδία ταξινομημένα .

```
While (current_level_files > 1 )
```

```
{  
  
.....  
  
}
```

Αρχικά στο βρόγχο αυτό σε κάθε επανάληψη αντιστοιχεί ένα πέρασμα. Δηλαδή εδώ μέσα συγχωνεύουμε και ταξινομούμε κάθε φορά όλα τα αρχεία του τρέχοντος επιπέδου. Αφού τα ταξινομήσουμε , τα αποθηκεύουμε σε καινούργια αρχεία. Αυτά τα καινούργια αρχεία τα ταξινομούμε στην επόμενη επανάληψη η οποία θα είναι το επόμενο πέρασμα. Την διαδικασία αυτή την ακολουθούμε μέχρις ότου στο τρέχον επίπεδο που θα βρισκόμαστε να υπάρχει μόνο ένα αρχείο το οποίο θα είναι και το τελικό ταξινομημένο αρχείο.

```
While (sorted_runs < current_level_files / (inmem_blocks - 1)
```

```
{  
  
.....  
  
}
```

Σε κάθε επανάληψη αυτού του βρόγχου κάνουμε μια 8-Way Merge και δημιουργούμε ένα καινούργιο ταξινομημένο αρχείο. Μέχρις ότου τα ταξινομημένα αυτά αρχεία να είναι λιγότερα από τον αριθμό των τρέχον αρχείων που έχω για συγχώνευση /(δια) των μπλοκς που έχω στην δευτερεύουσα μνήμη Ram.

Στον συγκεκριμένο βρόγχο ταξινομώ όλους τα **num** πεδία από τα μπλοκς που έχω στην μνήμη. Κάθε φορά θα τραβάω έναν αριθμό από κάθε μπλοκ θα τον βάζω σε ένα πίνακα **records[]** και θα τον ταξινομώ. Τον μικρότερο θα τον βάζω στο output. Στη συνέχεια θα φέρνω τον επόμενο αριθμό από το μπλοκ το οποίο βρήκα τον μικρότερο αριθμό στο προηγούμενο βήμα. Αυτό θα το κάνω συνέχεια μέχρι να αδειάσουν όλα τα μπλοκς. Αν αδειάσει ένα μπλοκ τότε φέρνω ένα άλλο από το αντίστοιχο αρχείο που πήρα το συγκεκριμένο. Αν πάλι αδειάσουν και όλα τα αρχεία τότε αυτό σημαίνει ότι εκτελέστηκε επιτυχίμως 1 sorted run και έτσι τελειώνει ο βρόγχος.

Sorted segments

Για τον υπολογισμό των sorted_segments σε κάθε πέρασμα που κάνουμε θα διαιρούμε τον συνολικό αριθμό των current αρχείων που έχουμε στο επιτεδό δια (/) των αριθμό των μπλοκς -1 που έχουμε διαθέσιμα στην μνήμη μας και στη συνέχεια θα αθροίζουμε το αποτέλεσμα αυτό.

Passes

Για τον υπολογισμό των περασμάτων κάθε φορά που διαβάζουμε όλα τα στοιχεία από τον δίσκο θα αυξάνουμε τον μετρητή.

IOs

Για τον υπολογισμό των IOs θα προσθέσουμε τον αριθμό των στοιχείων που διαβάζουμε + τον αριθμό των στοιχείων που γράφουμε και το αποτέλεσμα θα το πολλαπλασιάσουμε με τον αριθμό των περασμάτων.

Eliminate Duplicates (....)

Στόχος της συνάρτησης αυτής είναι να γίνει απαλοιφή των διπλότυπων του τελικού μας αρχείου που δημιουργήθηκε από το external sorting. Αρχικά εισάγουμε στην μνήμη ένα μπλοκ από το αρχείο. Με τον αλγόριθμο απαλοιφής διπλοτύπων διαγράφουμε τα διπλότυπα και μόλις φτάσουμε στο τέλος του μπλοκ γράφουμε τα αποτελεσματα μας σε ένα άλλο μπλοκ στην μνήμη μας (δηλαδή σε αυτή τη συνάρτηση δεσμεύουμε 2 μπλοκς στην Ram). Την παραπάνω διαδικασία την εκτελούμε μέχρι να αδειάσει το αρχείο εισόδου.

Unique Records

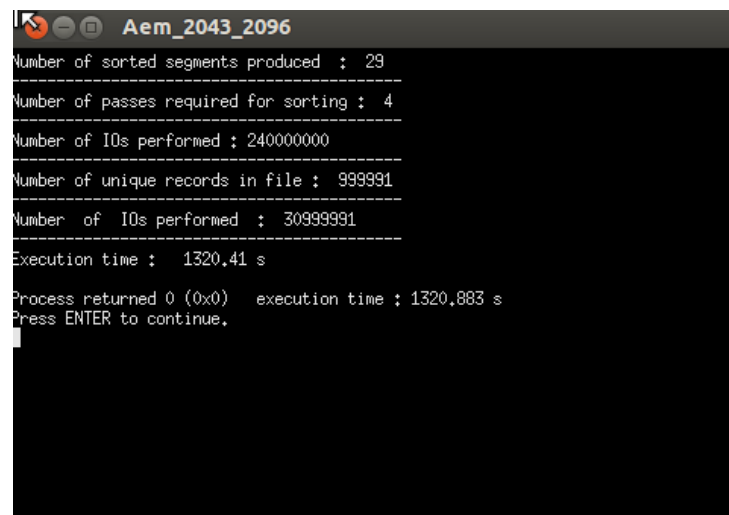
Για τον υπολογισμό των μοναδικών στοιχείων , κάθε φορά που ένα στοιχείο στο πίνακά μας είναι διαφορετικό από το αμέσως επόμενο του τότε θα αυξάνονται τα μοναδικά στοιχεία κατά 1 .

IOs

Για τον υπολογισμό των IOs θα προσθέσουμε τον αριθμό των στοιχείων που διαβάζουμε από το αρχείο εισόδου με τον αριθμό των μοναδικών στοιχείων που γράφουμε στο αρχείο εξόδου.

Αποτέλεσμα με 200 MB RAM και 4 GB είσοδο δεδομένων στο δίσκο για το 1^ο αρχείο.

Δηλαδή δεσμεύουμε 5 blocks στην Ram και το αρχείο αποτελείται από 100 blocks.



```
Aem_2043_2096
Number of sorted segments produced : 29
-----
Number of passes required for sorting : 4
-----
Number of IOs performed : 240000000
-----
Number of unique records in file : 999991
-----
Number of IOs performed : 30999991
-----
Execution time : 1320.41 s
Process returned 0 (0x0) execution time : 1320.883 s
Press ENTER to continue.
```

Αποτέλεσμα με 200 MB RAM και 4 GB είσοδο δεδομένων στο δίσκο για το 2^ο αρχείο.

Δηλαδή δεσμεύουμε 5 blocks στην Ram και το αρχείο αποτελείται από 100 blocks.

```
Aem_2043_2096
Number of sorted segments produced : 29
-----
Number of passes required for sorting : 4
-----
Number of IOs performed : 240000000
-----
Number of unique records in file : 999996
-----
Number of IOs performed : 30999996
-----
Execution time : 1837.03 s
Process returned 0 (0x0)   execution time : 1837.563 s
Press ENTER to continue.
```

Το αποτέλεσμα για τον πρώτο αριθμό των IOs είναι για το external sorting ενώ το αποτέλεσμα για το 2^ο αριθμό των IOs είναι για το remove duplicates.

Αποτέλεσμα με 400 MB RAM και 2 GB είσοδο δεδομένων στο δίσκο για το 1^ο αρχείο.

Δηλαδή δεσμεύουμε 10 blocks στην Ram και το αρχείο αποτελείται από 50 blocks.

```
Aem_2043_2096
Number of sorted segments produced : 6
-----
Number of passes required for sorting : 2
-----
Number of IOs performed : 60000000
-----
Number of unique records in file : 999994
-----
Number of IOs performed : 15999994
-----
Execution time : 357.459 s
Process returned 0 (0x0)   execution time : 357.941 s
Press ENTER to continue.
```

Αποτέλεσμα με 400 MB RAM και 2 GB είσοδο δεδομένων στο δίσκο για το 2^ο αρχείο.

Δηλαδή δεσμεύουμε 10 blocks στην Ram και το αρχείο αποτελείται από 50 blocks.

```
Aem_2043_2096
Number of sorted segments produced : 6
-----
Number of passes required for sorting : 2
-----
Number of IOs performed : 60000000
-----
Number of unique records in file : 999992
-----
Number of IOs performed : 15999992
-----
Execution time : 327.239 s

Process returned 0 (0x0)   execution time : 329.049 s
Press ENTER to continue.
```

Το αποτέλεσμα για τον πρώτο αριθμό των IOs είναι για το external sorting ενώ το αποτέλεσμα για το 2^ο αριθμό των IOs είναι για το remove duplicates.

Αποτέλεσμα με 400 MB RAM και 4 GB είσοδο δεδομένων στο δίσκο για το 1^ο αρχείο.

Δηλαδή δεσμεύουμε 10 blocks στην Ram και το αρχείο αποτελείται από 100 blocks.

```
Aem_2043_2096
Number of sorted segments produced : 13
-----
Number of passes required for sorting : 3
-----
Number of IOs performed : 180000000
-----
Number of unique records in file : 999994
-----
Number of IOs performed : 30999994
-----
Execution time : 1142.57 s

Process returned 0 (0x0)   execution time : 1143.458 s
Press ENTER to continue.
```

Αποτέλεσμα με 400 MB RAM και 4 GB είσοδο δεδομένων στο δίσκο για το 2^ο αρχείο.

Δηλαδή δεσμεύουμε 10 blocks στην Ram και το αρχείο αποτελείται από 100 blocks.

```
Aem_2043_2096
Number of sorted segments produced : 13
-----
Number of passes required for sorting : 3
-----
Number of IOs performed : 180000000
-----
Number of unique records in file : 983232
-----
Number of IOs performed : 19283232
-----
Execution time : 871.889 s
Process returned 0 (0x0)   execution time : 872.701 s
Press ENTER to continue.
```

Το αποτέλεσμα για τον πρώτο αριθμό των IOs είναι για το external sorting ενώ το αποτέλεσμα για το 2^ο αριθμό των IOs είναι για το remove duplicates.

MergeJoin(...)

Στη συνάρτηση αυτή εκτελούμε την πράξη της σύνδεσης ανάμεσα σε 2 αρχεία .Αρχικά δεσμεύουμε 2 μπλοκ στην μνήμη μας .Το πρώτο για να εισάγουμε κάθε φορά ένα μπλοκ από το πρώτο αρχείο και το δεύτερο για να εισάγουμε από το δεύτερο αρχείο.Αφού κάνουμε τις συγκρίσεις σε κάθε στοιχείο ανάμεσα στα 2 αρχεία , αν τελειώσει ένα μπλοκ από ένα αρχείο τότε θα φέρουμε το αμέσως επόμενο από το αντίστοιχο αρχείο.Αν τελειώσουν όλα τα μπλοκ από ένα αρχείο τότε τελειώνει και η συνάρτηση.

Στον βρόγχο αυτό μέχρι να φτάσουμε στο τέλος ενός από τα 2 αρχεία , θα συγκρίνουμε τα num στοιχεία μεταξύ τους και άμα αυτά είναι ίσα τότε είναι pair και θα αποθηκεύουμε το αποτέλεσμα στο output.Αν δεν είναι ίσα τότε θα ανεβάζουμε τον δείκτη του μπλοκ στο οποίο ήταν μικρότερος ο αριθμός σε σχέση με τον άλλο.

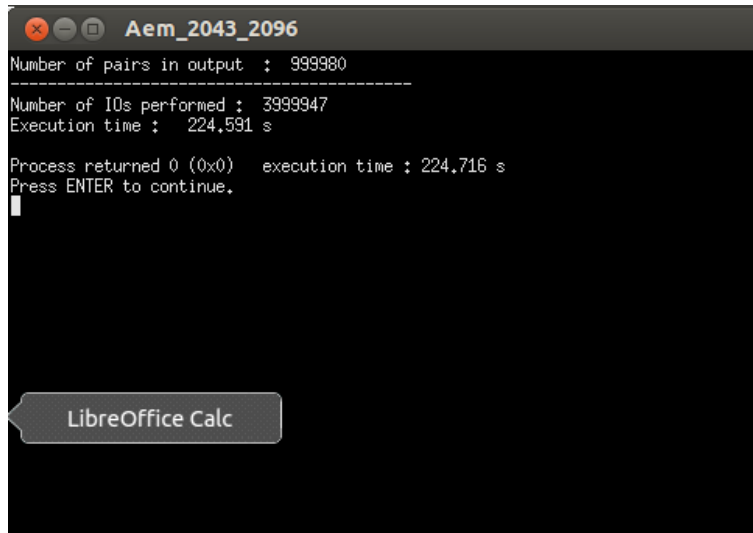
Pairs

Για τον υπολογισμό των ζευγαριών , κάθε φορά που ένα στοιχείο από το ένα αρχείο είναι ίσο με το άλλο τότε θα το προσθέτω στο συνολικό άθροισμα.

Ios

Για τον υπολογισμό των Ios θα προσθέσω τον αριθμό των στοιχείων που διαβάζω και από τα 2 αρχεία με τον συνολικό αριθμό από τα ζευγάρια που θα γράψω στο αρχείο εξόδου.

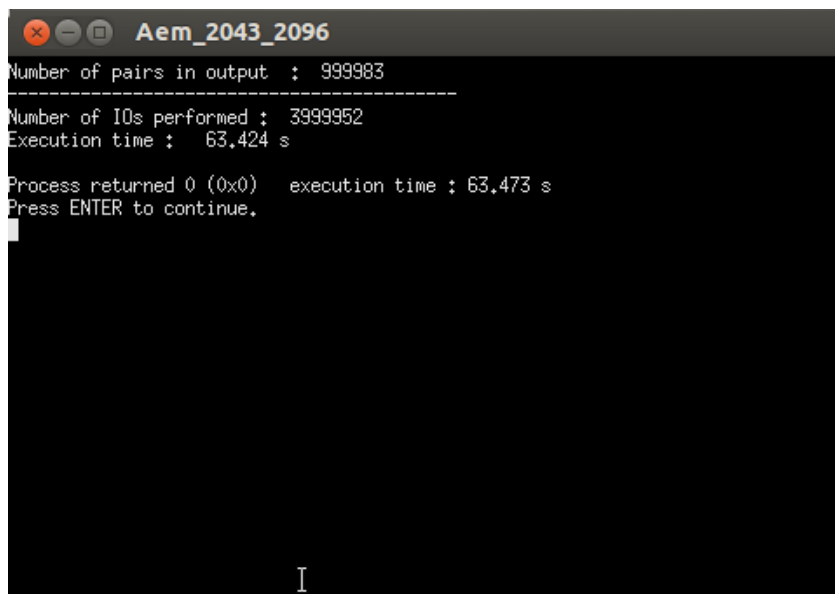
Αποτέλεσμα με 200 MB RAM και 4 GB είσοδο δεδομένων στο δίσκο για το κάθε αρχείο.



```
Aem_2043_2096
Number of pairs in output : 999980
-----
Number of IOs performed : 3999947
Execution time : 224.591 s

Process returned 0 (0x0) execution time : 224.716 s
Press ENTER to continue.
```

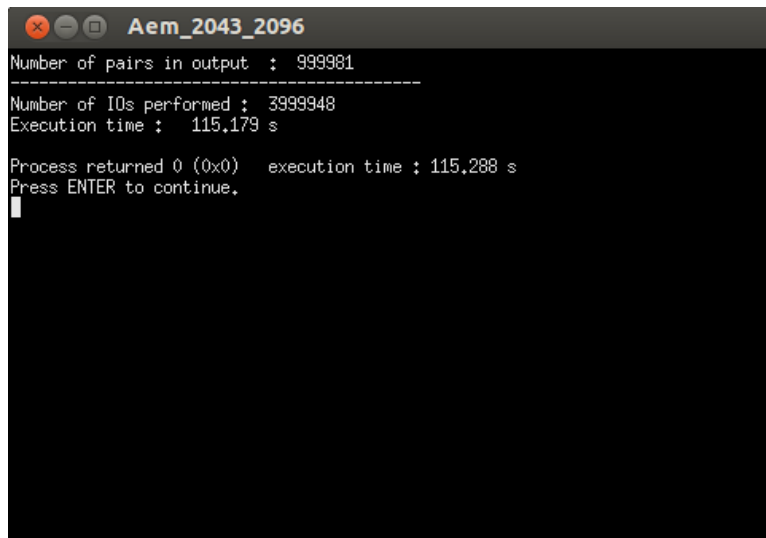
Αποτέλεσμα με 400 MB RAM και 2 GB είσοδο δεδομένων στο δίσκο για το κάθε αρχείο.



```
Aem_2043_2096
Number of pairs in output : 999983
-----
Number of IOs performed : 3999952
Execution time : 63.424 s

Process returned 0 (0x0) execution time : 63.473 s
Press ENTER to continue.
```


Αποτέλεσμα με 400 MB RAM και 4 GB είσοδο δεδομένων στο δίσκο για το κάθε αρχείο.



```
Aem_2043_2096
Number of pairs in output : 999981
-----
Number of IOs performed : 3999948
Execution time : 115,179 s

Process returned 0 (0x0) execution time : 115,288 s
Press ENTER to continue.
```

ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ