ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2015-16
3η Ατομική Άσκηση: Σύγκριση Λέξεων με χρήση ΑΤΔ ΔΔΑ + AVL
Προθεσμία Παράδοσης 29/5/2016

Ένας φιλόλογος θέλει να μελετήσει αν στην σύγχρονη χρήση λέξεων της αγγλικής γλώσσας υπάρχουν λέξεις που χρησιμοποίησε ο Shakespeare στα έργα του και με ποια συχνότητα εμφανίζονται. Από το http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists βρήκε ~50.000 σύγχρονες αγγλικές λέξεις που έχουν ακουστεί στην τηλεόραση και στον κινηματογράφο. Επίσης βρήκε και το έργο του Shakespeare Ρωμαίος και Ιουλιέτα, RomeoAndJuliet.txt με το οποίο θα κάνει την σύγκριση των λέξεων. Για να βοηθήσετε τον φιλόλογο στο έργο του ζητείται να αναπτύξτε τον ΑΤΔ Words και ένα πρόγραμμα πελάτη, που να υλοποιούν την σύγκριση. Η υλοποίηση χρησιμοποιεί ΔΔΑ για την οργάνωση των λέξεων της σύγχρονης Αγγλικής Γλώσσας στο οποίο θα αναζητούνται οι λέξεις του έργου Ρωμαίος και Ιουλιέττα. Αν η λέξη βρεθεί αυξάνεται ο μετρητής συχνότητας της λέξης. Οι μετρητές αποθηκεύονται σε πίνακα. Κάθε λέξη στο ΔΔΑ αντιστοιχίζεται σε μια θέση στον πίνακα.

Ενδιαφέρει επίσης η αποδοτικότητα του προγράμματος. Για αυτό τον λόγο χρειάζεται να κρατήσετε στοιχεία για τους χρόνους εκτέλεσης, ώστε να μελετήσετε την αποδοτικότητα του προγράμματος σας όσον αφορά α) τους αλγορίθμους απλού ΔΔΑ και AVL και β) την οργάνωση των δεδομένων (λέξεων) εισόδου (τυχαία, ταξινομημένη αλφαβητικά και βάσει συχνότητας). Για την μέτρηση της απόδοσης σας δίδονται προγράμματα επίδειξης μετρήσεων χρόνων εκτέλεσης προγραμμάτων. Είναι απαραίτητο όλες οι μετρήσεις να γίνουν στον ίδιο υπολογιστή.

Ζητείται να υλοποιήσετε τον ΑΤΔ Words με δυο τρόπους χρησιμοποιώντας τις διεπαφές ήδη ανεπτυγμένων υλοποιήσεων απλού ΔΔΑ και AVL. Ως οδηγό της υλοποίησης μπορείτε να δείτε την τεχνική της υλοποίησης ουράς με χρήση του ΑΤΔ Λίστα στις διαφάνειες με τις εφαρμογές τις λίστας (7.8.3 των πρότυπων υλοποιήσεων). Οι δυο υλοποιήσεις BST και AVL έχουν παρόμοια διεπαφή. Οι διαφορές είναι α) ότι οι πράξεις του απλού BST έχουν το πρόθεμα Tree ενώ του AVL το AVLTree και β) οι πράξεις εισαγωγής και διαγραφής σε AVL χρειάζονται μια *int παράμετρο επιπλέον (δεν χρειάζεται αρχική τιμή). Οι δύο υλοποιημένες πράξεις στο Word.c είναι οδηγός υλοποίησης για τις υπόλοιπες.

Τα δεδομένα εισόδου λέξεων σας δίδονται σε τρεις διατάξεις 1) βάσει συχνότητας, στο αρχείο wordsByFrequency.txt, 2) σε αλφαβητικη σειρά, στο wordsByABC.txt και 3) σε τυχαία σειρά, στο wordsByRandom.txt. Σημειώνουμε ότι για απλούστευση της σύγκρισης οι λέξεις έχουν μόνο πεζούς χαρακτήρες. Δεν πρέπει να αλλάξετε τα ονόματα των αρχείων δεδομένων, γιατί το πρόγραμμά σας θα ελεχθεί με αυτά τα αρχεία. Τα αρχεία εισόδου χρειάζονται μετατροπή `dos2unix` για `linux-gcc`.

Για κάθε έναν από τους 6 συνδυασμούς υλοποίησης (απλό ΔΔΑ και AVL) και εισόδου δεδομένων (συχνότητα, αλφαβητικό, τυχαίο) να μετρήσετε 1) τον χρόνο εισαγωγής όλων λέξεων στο ΔΔΑ καθώς και μετά από 1024, 2048, 4096,..., $1024*2^i$, $0<=i<=6$ λέξεις, 2) τον χρόνο αναζήτησης όλων των λέξεων από το Ρωμαίος και Ιουλιέττα και 3) τον χρόνο της ενδοδιατεταγμένης διαδρομής για την τελική εμφάνιση των κοινών λέξεων και την συχνότητά τους. Να κρατήσετε τα αποτελέσματα σε 6 διαφορετικά αρχεία με κατάλληλο όνομα.

Σας δίδεται ένας σκελετός του προγράμματος στον φάκελο skeletonProgram, όπου αναλύονται οι προδιαγραφές των πράξεων των ΑΤΔ Words (2 αρχεία), ΑΤΔ typos_stoixeiouDDA (2 αρχεία), και το main.c. Επίσης δίδονται η υλοποίηση του ΑΤΔ απλού ΔΔΑ (2 αρχεία στο BST), η υλοποίηση του ΑΤΔ AVL (2 αρχεία στο AVL). Να αναπτύξετε κώδικα για την υλοποίηση συναρτήσεων στα αρχεία .c. Μπορείτε να επέμβετε αν χρειαστεί και στα .h, αλλά μάλλον δεν θα χρειαστεί. Ο ΑΤΔ Words πρέπει να υλοποιηθεί με την χρήση των υλοποιήσεων απλού ΔΔΑ και AVL. Ο κώδικας των 2 υλοποιήσεων, απλού ΔΔΑ και AVL, είναι πλήρης και **δεν χρειάζεται ούτε πρέπει να τον αλλάξετε**. Ο σκελετός προγράμματος, όπως δίδεται μεταγλωττίζεται και «τρέχει», αλλά φυσικά δεν κάνει τίποτα χρήσιμο. Προτείνεται να διατηρήσετε τον σκελετό και να τον αναπτύξετε σε πλήρες λειτουργικό πρόγραμμα.

Ορισμός ΑΤΔ Words

Ο τύπος δεδομένων που προτείνεται είναι (με ολική απόκρυψη)

```
typedef struct RecWords * typosWords;

typedef struct RecWords       /* to Words apoteleitai apo
{  typos_deikti WordsRiza;          /* enan deikti se DDA */
   int SearchFrequencies[100000];   /* array of times each word is found */
   int wordCounter;                 /* counts current number of words */
   double InsertTime [10];          /* xronoi eisagvghs stoixeivn*/
   double CheckTime;                /* xronos anazhthshs */
   double DiadromhTime;             /* xronos diadromhs */
} RecWords;
```

Οι Πράξεις του ΑΤΔ Words (για προδιαγραφές λειτουργίας δείτε επίσης στο Words.c)

`typosWords dhmiourgia_Words();`	Δημιουργεί νέα δομή Words (υλοποιημένη)
`void katastrofh_Words(typosWords * Wordsptr);`	Καταστρέφει δομή Words (υλοποιημένη)
`void InsertWord(typosWords Words, char * w);`	Εισάγει w στο ΔΔΑ και αρχικοποιεί αντίστοιχο στοιχείο στον πίνακα συχνοτήτων
`void CheckWord(typosWords Words, char * w);`	Αναζητά w στο ΔΔΑ και αν το βρει αυξάνει αντίστοιχο στοιχείο στον πίνακα συχνοτήτων. Αυξάνει αντίστοιχα έναν από τους δυο μετρητές Found/NotFound.
`void ShowCommonWords(FILE *out, typosWords Words);`	Διαδρομή ΔΔΑ ώστε να εμφανίζονται οι κοινές λέξεις αλφαβητικά και η συχνότητα τους.
`void SetInsertTime(typosWords Words, float time, int pos);`	Εισάγει στη θέση pos του πίνακα `InsertTime` τον χρόνο εισαγωγής των $1024*2^{pos.}$ Στην τελική θέση τον συνολικό χρόνο.
`void SetCheckTime(typosWords Words, float time);`	Εισάγει τον χρόνο αναζήτησης στο `CheckTime`
`void SetDiadromhTime(typosWords Words, float time);`	Εισάγει τον χρόνο διαδρομής στο `DiadromhTime`
`void PrintData(FILE *out, typosWords Words);`	Εμφανίζει τους χρόνους `InsertTime, CheckTime, DiadromhTime` και μετρητές.

Ο ΑΤΔ typos_stoixeioyDDA

Το περιεχόμενο των κόμβων του ΔΔΑ αποτελείται από μια συμβολοσειρά (την λέξη) και έναν ακέραιο που θα μετράει την συχνότητα που η λέξη που αναζητήθηκε η λέξη. Ο τύπος δεδομένων του στοιχείου του ΔΔΑ που προτείνεται είναι

```
typedef struct dedomena {
        char * word;    /* a word, the key of DDA */
        int index;      /* index to an element of array of frequencies */
} TStoixeioyDDA;
```

Οι πράξεις του

`int TSDDA_writeValue(FILE *to, TStoixeioyDDA Elem);`	Εκτυπώνει το στοιχείο
`int TSDDA_iso(TStoixeioyDDA Elem1, TStoixeioyDDA Elem2);`	Συγκρίνει (==) 2 λέξεις των στοιχείων
`int TSDDA_mikrotero(TStoixeioyDDA Elem1, TStoixeioyDDA Elem2);`	Συγκρίνει (<) 2 λέξεις των στοιχείων
`int TSDDA_megalytero(TStoixeioyDDA Elem1, TStoixeioyDDA Elem2);`	Συγκρίνει (>) 2 λέξεις των στοιχείων
`void TSDDA_setValue (TStoixeioyDDA *target, TStoixeioyDDA source);`	Ανάθεση τιμών target=source
`int TSDDA_readValue (FILE *from, TStoixeioyDDA * Elem);`	Διαβάζει τιμές

Μπορείτε να παρέμβετε στο τύπο στοιχείου, να αλλάξετε τις πράξεις, να ορίσετε δικές σας, κλπ

Το πρόγραμμα-πελάτης

Κατασκευάζει μια δομή Words και εισάγει λέξεις σε αυτή από τα αρχεία λέξεων. Κατόπιν αναζητά λέξεις από το αρχείο `RomeoAndJuliet.txt`. Τέλος εμφανίζει τις κοινές λέξεις των δύο αρχείων την συχνότητα των κοινών και τους χρόνους. Η δομή του main δίδεται, όπως επίσης και οι επικεφαλίδες των 3 συναρτήσεων που πρέπει να αναπτύξετε χρησιμοποιώντας βασικές πράξεις του ΑΤΔ Words.

`void InitialiseTree (FILE *wordlist, typosWords W);`	Διαβάζει λέξεις από wordlist και τις εισάγει στο W (InsertWord). Βρίσκει χρόνους εισαγωγής και ενημερώνει W (SetInsertTime).
`void SearchUpdateTree (FILE *wordlist, typosWords W);`	Διαβάζει λέξεις από το RomeoAndJuliet.txt και τις αναζητά στο W (CheckWord).
`void Results (FILE *out, typosWords W);`	Εμφανίζει αποτελέσματα

Παραδοτέα και Οδηγίες Παράδοσης

1. Πηγαίος κώδικας (όλα τα αρχεία σας .h και .c).
2. Τεκμηρίωση Σύντομο κείμενο (txt, word, pdf) με την εξής δομή
   - Τα στοιχεία σας: (Όνομα-Επώνυμο-ΑΜ)
   - Λειτουργικότητα: Να περιγράψετε τι κάνει το πρόγραμμά σας (μπορεί να κάνει περισσότερα ή και λιγότερα από τα ζητούμενα της άσκησης).
   - Οδηγίες Χρήσης του προγράμματος σας: π.χ. Διάταξη δεδομένων εισόδου.
   - Περιβάλλον Υλοποίησης και Δοκιμών: πχ. Αναπτύχθηκε σε Dev C++ σε περιβάλλον Windows XP, δοκιμάστηκε επίσης σε gcc και Unix.
   - Αποτελεσματα μετρήσεων σε πίνακες καθώς και τις παρατηρήσεις σας σχετικά με τους χρόνους και συγκρίσεις με αναμενόμενα αποτελέσματα.

Το αρχείο τεκμηρίωσης μαζί με τα αρχεία του προγράμματος να τα βάλετε σε έναν φάκελο (directory) **με όνομα τον ΑΜ σας**, τον οποίο θα συμπιέσετε (zip, rar) και θα ανεβάσετε στο eclass. Προσοχή ανεβάστε το στην κατάλληλη κατηγορία υλοποίησης (Dev-C++ ή gcc).

Τρόπος Αξιολόγησης

Οι ασκήσεις είναι **ατομικές** και θα ελεγχθούν για ομοιότητες χρησιμοποιώντας ειδικό σύστημα εντοπισμού ομοιοτήτων. Σε περίπτωση μεγάλης «ομοιότητας» οι εμπλεκόμενοι αποκλείονται από τα εργαστήρια και τελική εξέταση. Θα αξιολογηθούν: η λειτουργικότητα, η δομή και η τεκμηρίωση του προγράμματος. Αναλυτικά:

Λειτουργικότητα (80/100)

Πράξεις/Συναρτήσεις	**Βαθμοί**
InsertWord	5
CheckWord	10
ShowCommonWords	10
SetInsertTime	5
SetDiadromhTime, SetCheckTime	2
PrintTimes	5
Τύπος Στοιχείου (συνολικά)	10
InitialiseTree	10
SearchUpdateTree	10
Πρόγραμμα Πελάτη main	8
Υπόλοιπες ήδη υλοποιημένες (έλεγχος)	5

Δομή (10/100)

Οργάνωση σε 3 Ενότητες (.h, .c) και πρόγραμμα πελάτη	4
Απόκρυψη Υλοποίησης	4
Δομημένο Πρόγραμμα (μορφοποίηση, σχόλια, κλπ)	2

Τεκμηρίωση και Παρουσίαση (10/100)

ΠΡΟΣΟΧΗ: Για να αξιολογηθεί το πρόγραμμα σας (έστω και για την δομή του) πρέπει τουλάχιστον να μεταγλωττίζεται. Αν δεν μεταγλωττίζεται δεν παίρνει βαθμό. Πριν παραδώσετε το πρόγραμμά σας δοκιμάστε το μια τελευταία φορά και βεβαιωθείτε ότι παραδίδετε τα σωστά αρχεία.