Readme File

Εργασία 1 Προγραμματισμός Συστήματος 2013 – 2014

(Γλώσσα : C)

Περιεχόμενα :

---

1. Makefile - Εκτέλεση

Το Makefile δημιουργεί το εκτελέσιμο με όνομα exe . Κάνει separate compilation. Έχει clean .Το Makefile αυτό είναι σχεδόν ίδιο με αυτό από τις διαφάνειες του μαθήματος .

Το εκτελέσιμο είτε με ένα όρισμα τρέχει με default το path στο $(HOME)
ή με -d ακολουθούμενο από valid path.Δεν έχω προσθέσει έλεγχο για την εγκειρότητα του αρχείου καθώς όπως το σκέφτηκα αυτό θα δεύσμευε την ονομασία του αρχείου(.dict πχ).

---

2. Δομές – Πίνακες

- Κάθε ακμή του δέντρου έχει την εξής μορφή :

```
struct Edge{
        char s ;
        TrieNode * trieNode_ptr;
};
```

Στο πρώτο όρισμα έχουμε το γράμμα το οποίο οδηγεί στον αντίστοιχο κόμβο. Το δεύτερο όρισμα είναι ο εν λόγω κόμβος .

- Ο "πατέρας" έχει την εξής μορφή :

```
struct Father{
        Edge * edge ;
        TrieNode * trieNode_ptr ;
};
```

Είναι ένα struct το οποίο δείχνει στην ακμή από την οποία προήλθε ο κόμβος που τον έχει ώστε να μπορούμε να πάρουμε το γράμμα της ακμής που οδηγεί σε αυτόν.Έχει και δείκτη στον κόμβο από τον οποίο προήλθε για να μπορέσουμε παρακάτω με προς τα πίσω αναζήτηση όλων των κόμβων "πατεράδων" να πάρουμε τη λέξη .

- Ο κάθε κόμβος του δέντρου έχει την μορφή :

```
struct TrieNode{
        Edge edges[27];
        TrieNode * frequentNodes[N];
        Father father ;
        int freq_no ;
};
```

Είναι κοινός για φύλα και εσωτερικούς κόμβους και έχει τα παραπάνω τους πιο συχνούς κόμβους και τη συχνότητα αν πρόκειται για φύλο.Οι ακμές είναι στατικά σε πίνακα των 27 θέσεων όπου το @ είναι στην τελευταία .Με αυτό χάνω σε μνήμη όμως κερδίζω σε χρόνο αναζήτησης καθώς αρκεί ένα word[0]-'a' για να πάω στον σωστό κόμβο σε σχέση με μία λίστα .Όπου word[0] το τρέχων γράμμα μιας λέξης.

- Τέλος το :

```
struct MotherTreeNode{
        TrieNode * first ;
        //might need extra info
};
```

Όπου τελικά δεν έβαλα εχτρα πληροφορία απλά δείχνει στη ρίζα του δέντρου.

- Χρησιμοποιώ και ένα stringMatrix στο οποίο κρατάω τα strings των πιο συχνών λέξεων.

3.Σύντομη περιγραφή κώδικα :

Το πρόγραμμα παίρνει από το dictionary τις λέξεις και τις περνάει στο δέντρο .Στη συνέχεια κάθε χαρακτήρα που πληκτρολογεί ο χρήστης μπαίνει στο master_buffer το οποίο ειναι στατικό και 4096 bytes .Επίσης κάθε λέξη μπαίνει στο buffer και στη συνέχεια ακολουθείται η κάθε περίπτωση (isalpha(next) κ.ο.κ.) .

BACKSPACE:Θα πρέπει να αναφέρω ότι το backspace δουλεύει αν δεν αλλάξει μια λέξη(space) αλλά δεν δουλεύει αν αλλάξει η λέξη διότι αδειάζει το buffer.

TAB: 1.το TAB εκτυπώνει στο stderr.

2.τα αποτελέσματα του TAB είναι ταξινομημένα χάρη στη συνάρτηση :
int orderStringMatrix(char ** stringMatrix).

Το πρόγραμμα προσθέτει στο τέλος του dictionary καινούριες λέξεις.Τέλος εκτυπώνει κεφαλαία όπως στο παράδειγμα εκτέλεσης .