Κωνσταντίνος Μπουρλέσης - 43862

Πεπερασμένο Ντετερμινιστικό Αυτόματο

Ακολουθεί η επεξήγηση του κώδικα ενός ΠΝΑ υλοποιημένο με γλώσσα προγραμματισμού C, σε περιβάλλον Linux και το compile έγινε με την χρήση του gcc compiler.

Εντολή για compile: gcc deterministic_auto.c -o deterministic_auto
Εντολή για εκτέλεση: ./deterministic_auto (path αρχείου)

Λόγω των διαφορετικών line endings μεταξύ linux και windows τα dfa αρχεία θα πρέπει να αποθηκεύονται με βάση το linux line ending, σε διαφορετική περίπτωση το πρόγραμμα δεν θα έχει τα αναμενόμενα αποτελέσματα.

Το πρόγραμμα πρέπει να διαβάζει ένα αρχείο και με βάση αυτό να “δημιουργεί” ένα ΠΝΑ.

Στις γραμμές 15-18 πραγματοποιείται το άνοιγμα του αρχείου με τη χρήση της open. Η open διαβάζει το όνομα του αρχείου από τον πίνακα argv, ο οποίος περιέρχει όλα τα ορίσματα της εντολής εκτέλεσης, στη θέση ένα.

Περιεχόμενα argv[]:

argv[0] --> ./deterministic_auto
argv[1] --> (path αρχείου)

Γραμμές 20-23: Ακολουθεί ένας έλεγχος για να δούμε αν το αρχείο είναι κενό. Αν είναι, τότε τυπώνεται στην τυπική έξοδο κατάλληλο μήνυμα και το πρόγραμμα τερματίζει.

Αφού ανοίξει το αρχείο ψάχνει να βρει πόσα σύμβολα έχει το αλφάβητο και πόσες τελικές καταστάσεις, με τη βοήθεια της lseek συνάρτησης, έτσι ώστε να φτιάξει τους πίνακες στους οποίους θα αποθηκευτούν. Αυτό γίνεται στις γραμμές 27-50. Στις επόμενες δύο γραμμές γίνεται η δέσμευση μνήμης για τους πίνακες με την malloc.

* Το αρχείο που θα διαβάσει το πρόγραμμα ΔΕΝ πρέπει να περιέχει ούτε λιγότερα ούτε περισσότερα κενά και χαρακτήρες αλλαγής γραμμής από τα ήδη υπάρχοντα γιατί έτσι θα διαβάσει λάθος τα δεδομένα. *

Στη συνέχεια επιστρέφουμε στην αρχή του αρχείου με την lseek και αποθηκεύουμε τα δεδομένα στους αντίστοιχους πίνακες και μεταβλητές. Η διαχώριση των δεδομένων γίνεται με βάση τον χαρακτήρα της αλλαγής γραμμής (\n). Αποθηκεύει τα δεδομένα στον ίδιο πίνακα, απορρίπτοντας τα κενά, μέχρι να συναντήσει το \n. Αυτό σημαίνει ότι πρόκειται να διαβάσει καινούρια δεδομένα και έτσι αλλάζει πίνακα. Γραμμές 59 – 97.

Γραμμή 138: Δημιουργία πίνακα μεταβάσεων.

Προχωράμε στο αρχείο στην γραμμή από την οποία ξεκινάνε οι μεταβάσεις και τις αποθηκεύουμε στον πίνακα transitions. Γραμμές 142 – 164.

Τελικά το πρόγραμμα είναι έτοιμο να δεχτεί εισόδους από τον χρήστη και να αναγνωρίσει αν οι λέξεις που θα του δοθούν ανήκουν ή όχι στο ΠΝΑ που υλοποιεί.

Γραμμές 171-237 (υλοποίηση ΠΝΑ):

Όλος ο κώδικας του ΠΝΑ τρέχει μέσα σε μια while loop η οποία ελέγχει αν η απάντηση του χρήστη, στην ερώτηση αν θέλει να δώσει κι άλλη λέξη, είναι όχι. Έχουμε αρχικοποιήσει την απάντηση με “ναι” έτσι ώστε την πρώτη φορά να μπει στην επανάληψη και να μην τερματίσει το πρόγραμμα πριν αρχίσει.

Οι μεταβλητές input, cur_state_accepted αρχικοποιούνται με μηδέν (0) και η cur_state με την τιμή της αρχικής κατάστασης, κάθε φορά που έρχεται νέα είσοδος/λέξη.
Η input είναι δείκτης στον πίνακα usr_input ο οποίος περιέχει την λέξη που έδωσε ο χρήστης και η cur_state_accepted χρησιμοποιείται για να δούμε αν μια κατάσταση είναι αποδεκτή ως κατάσταση τερματισμού. Η cur_state μας δείχνει την τρέχουσα κατάσταση.

Ελέγχουμε αν η απάντηση του χρήστη είναι Y ή y που σημαίνει “ναι” και αν είναι τότε προχωράμε στην είσοδο δεδομένων στο πρόγραμμα, σε κάθε άλλη περίπτωση ρωτάμε πάλι τον χρήστη αν θέλει να εισάγει καινούρια λέξη.

Εφόσον περάσουμε τον έλεγχο αυτό και πάρουμε την είσοδο χρησιμοποιούμε μια for loop η οποία θα τρέξει τόσες φορές όσες είναι οι πιθανές μεταβάσεις επί το μήκος της λέξης που μας έδωσε ο χρήστης. Για παράδειγμα αν έχουμε 2 σύμβολα και 3 καταστάσεις σημαίνει ότι έχουμε 2*3 = 6 πιθανές μεταβάσεις. Με μια είσοδο 10100 αυτό σημαίνει ότι πρέπει να τρέξει, έξι φορές για κάθε αλφαριθμητικό της λέξης, 6*5 = 30 φορές σύνολο.

Ο δείκτης input μας λέει για ποιο αλφαριθμητικό της λέξης τρέχει η επανάληψη μας αυτή τη στιγμή, αν το input+1 λοιπόν είναι μεγαλύτερο από το μήκος της εισόδου τότε σημαίνει ότι έχει τρέξει για όλη τη λέξη και σπάει η επανάληψη.

Ξέροντας ότι πρέπει να τρέξει ο ίδιος πίνακας από την αρχή για κάθε ένα αλφαριθμητικό και πως θα τρέξει KxΣ (σύνολο καταστάσεων x σύνολο συμβόλων) φορές συμπεραίνουμε ότι πρέπει να αρχικοποιούμε τους δείκτες του ανά KxΣ επαναλήψεις. Δηλαδή για κάθε διαφορετικό στοιχείο είναι σαν να τρέχουμε τον πίνακα από την αρχή. Έτσι γλιτώνουμε την εμφωλευμένη λούπα η οποία αυξάνει την πολυπλοκότητα του προγράμματος. Αυτό το πετυχαίνουμε υπολογίζοντας το υπόλοιπο (mod) του μετρητή (i) της for loop με το 6. Κάθε φορά που το αποτέλεσμα είναι μηδέν (0) σημαίνει ότι έχει τρέξει 6 φορές, οπότε για νέο αλφαριθμητικό αρχικοποίησε τους δείκτες.

Τα state / symbol / next_state είναι δείκτες του πίνακα transitions και το κάθε ένα δείχνει:

state = καταστάσεις (0 , 1 , 2)
symbol = σύμβολα του αλφάβητου (0 , 1)
next_state = κατάσταση στην οποία πηγαίνουμε με τον συνδυασμό του state + symbol (ΑΝ υπάρχει στον πίνακα των μεταβάσεων).

Οι καταστάσεις, τα σύμβολα και οι επόμενες καταστάσεις είναι αποθηκευμένες στον πίνακα transitions ανά 3 θέσεις. Γι αυτό σε κάθε αναζήτηση αν δεν βρούμε αυτό που ψάχνουμε κατευθείαν τους αυξάνουμε κατά 3.
Αρχικοποιούμε τους δείκτες έτσι ώστε να δείχνουν ο κάθε ένας την κατηγορία του.

Το word_accepted είναι ένα flag που αρχικοποιείται με 0 και γίνεται 1 μόνο όταν βρεθεί αποδεκτός συνδυασμός κατάστασης και εισόδου. Έτσι αν όλα τα στοιχεία της εισόδου έχουν κάνει το flag 1 τότε όλη η λέξη είναι αποδεκτή από το αλφάβητο του αυτόματου, αλλιώς παραμένει 0 και τότε τυπώνεται μήνυμα ότι η λέξη δεν είναι αποδεκτή.

Στην περίπτωση που είναι, ελέγχουμε με μια for loop αν η κατάσταση στην οποία βρίσκεται το αυτόματο υπάρχει μέσα στον πίνακα των τελικών καταστάσεων και αν ναι τότε θέτουμε το flag cur_state_accepted ίσο με 1 για να τυπώσει μήνυμα ότι η κατάσταση είναι αποδεκτή. Σε διαφορετική περίπτωση το flag αυτό είναι 0 και τυπώνει μήνυμα ότι η λέξη είναι αποδεκτή από το αλφάβητο αλλά η τελική κατάσταση είναι μη αποδεκτή.