Πεπερασμένα Αυτόματα

6ο Φροντιστήριο
Παρουσίαση : Γιάννης Πλεξουσάκης

Χαράλαμπος Ε. Τσουρακάκης

Φροντιστήριο: ΗΥ-280 · Θεωρία Υπολογισμού

6 Νοεμβρίου 2026

Ασκήσεις: πώς επιλέγουμε λέξη για άντληση

Άσκηση 1: ίσο πλήθος μηδενικών και άσων

$$C = \{w \in \{0,1\}^* : \#_0(w) = \#_1(w)\}.$$

Ερώτημα

Είναι η C κανονική; Χρησιμοποιήστε το λήμμα της άντλησης για να αιτιολογήσετε την απάντησή σας.

Ερώτημα

Ποια από τις $s = (01)^p$ και $s = 0^p 1^p$ μάς βοηθά να ελέγξουμε **όλες** τις επιτρεπτές διασπάσεις;

Η ιδέα

Θέλουμε το y να αποτελείται αναγκαστικά από **ένα μόνο είδος συμβόλου**. Το πετυχαίνουμε με ένα μεγάλο αρχικό μπλοκ.

Λύση 1: εκμεταλλευόμαστε το $|xy| \leq p$

Υποθέτουμε ότι η C είναι κανονική και έστω p το μήκος άντλησης. Επιλέγουμε

$$s = 0^p 1^p \in C.$$

Για **οποιαδήποτε** επιτρεπτή διάσπαση $s = xyz$:

$$|xy| \leq p, \quad |y| > 0 \quad \Longrightarrow \quad y = 0^t \text{ για κάποιο } 1 \leq t \leq p.$$

Αν αντλήσουμε με $i = 0$:

$$xy^0 z = xz = 0^{p-t} 1^p.$$

Πλέον $\#_0(xz) = p - t \neq p = \#_1(xz)$. Άρα $xz \notin C$: **άτοπο**.

Συγκρατούμε τα εξής:

Η λέξη $(01)^p$ δεν είναι καλή επιλογή: π.χ. $x = \varepsilon$, $y = 01$ επιτρέπει άντληση χωρίς να χαλά η ισότητα.

Άσκηση 2: η ίδια λέξη δύο φορές

$$F = \{ww : w \in \{0,1\}^*\}.$$

Ερώτημα

Να αποδειχθεί ότι η F δεν είναι κανονική.

Ερώτημα

Γιατί η 0^p0^p είναι ακατάλληλη για εύκολη αντίφαση; Τι θα συμβεί αν τοποθετήσουμε ένα 1 σε κάθε αντίγραφο;

Η ιδέα

Χρειαζόμαστε **σημάδια** που να εντοπίζουν τα όρια των δύο πανομοιότυπων μισών.

Γιατί η επιλογή $s = 0^p0^p$ δεν λειτουργεί;

Η πρώτη μας προσπάθεια

Επιλέγουμε $s = 0^p0^p = 0^{2p} \in F$.
Από το Λήμμα Άντλησης, $s = xyz$, όπου $|xy| \leq p, \qquad |y| > 0$.
Άρα $y = 0^t$, για κάποιο $1 \leq t \leq p$.

Το πρόβλημα: τι συμβαίνει αν t είναι άρτιο;

Για οποιοδήποτε $k \geq 0$, η άντληση δίνει $xy^kz = 0^{2p+(k-1)t}$.
Αν t είναι άρτιο, τότε $2p + (k-1)t$ είναι άρτιος αριθμός για κάθε $k \geq 0$.

Επομένως, $\boxed{xy^kz = \underbrace{0^{p+(k-1)t/2}}_{w'}\underbrace{0^{p+(k-1)t/2}}_{w'} \in F \quad \forall k \geq 0.}$

Δεν υπάρχει κανένα k που να δίνει αντίφαση!

Συμπέρασμα

Στις αποδείξεις με το Λήμμα Άντλησης δεν επιλέγουμε απλώς μια μεγάλη λέξη. Επιλέγουμε μια λέξη με δομή που επιτρέπει να αποδείξουμε ότι κάθε επιτρεπτή διάσπαση αποτυγχάνει για κάποια άντληση.

Λύση 2: δύο αντίγραφα, δύο σημάδια

Ας είναι p το μήκος άντλησης. Διαλέγουμε

$$s = (0^p 1)(0^p 1) \in F.$$

Επειδή $|xy| \leq p$ και $|y| > 0$, κάθε επιτρεπτό y έχει μορφή 0^t, $1 \leq t \leq p$. Με $i = 0$ παίρνουμε

$$xz = 0^{p-t} 1\, 0^p 1.$$

Γιατί δεν ανήκει στην F;

Η λέξη έχει **ακριβώς δύο** σύμβολα 1 και τελειώνει σε 1. Αν ήταν uu, τότε κάθε u θα είχε ακριβώς ένα 1, αναγκαστικά στο τέλος του. Τα δύο 1 θα απείχαν $|u|$ θέσεις. Όμως εδώ απέχουν $p+1$ θέσεις, ενώ

$$|u| = \frac{2p + 2 - t}{2} \neq p + 1.$$

Άτοπο: η F δεν είναι κανονική.

Άσκηση 3: μήκη που είναι τέλεια τετράγωνα

$$D = \{1^{n^2} : n \geq 0\}.$$

Ερώτημα

Να δείξετε ότι η D δεν είναι κανονική, παρόλο που χρησιμοποιεί μόνο ένα σύμβολο.

Ερώτημα

Ποια είναι η απόσταση ανάμεσα στα διαδοχικά τετράγωνα p^2 και $(p+1)^2$;

Η ιδέα

Με το λήμμα, το μήκος του y είναι το πολύ p. Προσπαθούμε να πέσουμε **αυστηρά ανάμεσα** σε δύο διαδοχικά αποδεκτά μήκη.

Λύση 3: το κενό ανάμεσα στα τετράγωνα

Έστω μήκος άντλησης $p \geq 1$. Επιλέγουμε $s = 1^{p^2} \in D$. Σε κάθε νόμιμη διάσπαση $s = xyz$ ισχύει

$$y = 1^t, \qquad 1 \leq t \leq p.$$

Αν βάλουμε $i = 2$, το μήκος αυξάνεται κατά t:

$$|xy^2z| = p^2 + t.$$

Όμως

$$\underbrace{p^2}_{\text{τετράγωνο}} < p^2 + t \leq p^2 + p < \underbrace{(p+1)^2}_{\text{επόμενο τετράγωνο}}.$$

Άρα το $p^2 + t$ **δεν** είναι τέλειο τετράγωνο: αντίφαση.

Συγκρατούμε τα εξής:

Σε μονοσυμβολικές γλώσσες, η άντληση μεταβάλλει μόνο το **μήκος**.

Άσκηση Myhill-Nerode

2

Φροντιστήριο: Πέντε κλάσεις, δυαδικοί αριθμοί διαιρετοί με 5

Έστω $\Sigma = \{0, 1\}$ και

$$L = \{w \in \Sigma^* : \mathrm{val}_2(w) \equiv 0 \pmod 5\}.$$

Η $\mathrm{val}_2(w)$ είναι η αριθμητική τιμή της δυαδικής λέξης w, με $\mathrm{val}_2(\varepsilon) = 0$.

Παράδειγμα:

$$101_2 = 5 \in 5\mathbb{Z}, \qquad 110_2 = 6 \notin 5\mathbb{Z}.$$

Αν έχουμε διαβάσει τη λέξη w και στη συνέχεια διαβάσουμε ένα bit b, τότε

$$\mathrm{val}_2(wb) = 2\,\mathrm{val}_2(w) + b.$$

Επομένως, αν r είναι το τρέχον υπόλοιπο,

$$\boxed{r' = (2r + b) \bmod 5.}$$

Ερώτημα

Χρειάζεται να θυμόμαστε ολόκληρο τον δυαδικό αριθμό που διαβάσαμε;

Πέντε κλάσεις — θυμόμαστε μόνο το υπόλοιπο

Για $r \in \{0, 1, 2, 3, 4\}$ ορίζουμε

$$C_r = \{w \in \Sigma^* : \mathrm{val}_2(w) \equiv r \pmod 5\}.$$

C_0	ε	0	101	1010
C_1	1	001	110	1011
C_2	10	010	111	1100
C_3	11	011	1000	1101
C_4	100	0100	1001	1110

Συγκρατούμε τα εξής:

Λέξεις με το ίδιο υπόλοιπο έχουν ακριβώς τις ίδιες αποδεκτές συνέχειες. Άρα υπάρχουν **το πολύ 5 κλάσεις Nerode**.

Γιατί οι πέντε κλάσεις είναι διαφορετικές;

Αν μια λέξη x έχει υπόλοιπο r και προσθέσουμε μια συνέχεια z μήκους k, τότε

$$\mathrm{val}_2(xz) \equiv 2^k r + \mathrm{val}_2(z) \pmod 5.$$

Κεντρική ιδέα: Για κάθε υπόλοιπο r μπορούμε να βρούμε μια συνέχεια που οδηγεί *μόνο αυτό* το υπόλοιπο σε αποδοχή.

Επιλέγουμε συνέχειες μήκους 3. Επειδή $2^3 = 8 \equiv 3 \pmod 5$, θέλουμε

$$3r + \mathrm{val}_2(z) \equiv 0 \pmod 5.$$

Υπόλοιπο r	**Συνέχεια** z	**Τελικό υπόλοιπο**
0	000	$(3 \cdot 0 + 0) \bmod 5 = 0$
1	010	$(3 \cdot 1 + 2) \bmod 5 = 0$
2	100	$(3 \cdot 2 + 4) \bmod 5 = 0$
3	001	$(3 \cdot 3 + 1) \bmod 5 = 0$
4	011	$(3 \cdot 4 + 3) \bmod 5 = 0$

Κατασκευάζοντας το DFA: ο πίνακας μεταβάσεων

Κάθε κατάσταση q_r αντιστοιχεί στο υπόλοιπο r.

$$\boxed{\delta(q_r, b) = q_{(2r+b) \bmod 5}.}$$

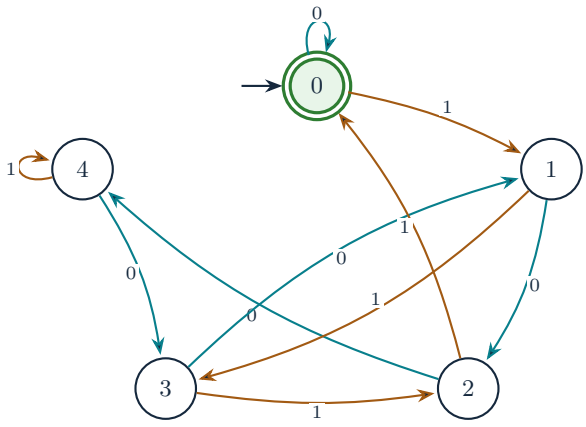
Κατάσταση	**Bit** 0	**Bit** 1
$\to *q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

Παράδειγμα: διαβάζουμε 1010

$$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_0 \xrightarrow{0} q_0.$$

Ο αριθμός $1010_2 = 10$ είναι διαιρετός με 5.

Το ελάχιστο DFA για διαιρετότητα με 5



Bit 0 **Bit** 1

Regular expressions στο Linux: το grep ως καθημερινό αυτόματο

3

Κανονικές εκφράσεις: Από τον Sipser στο grep

Ίδια ιδέα, διαφορετικός σκοπός

- Στον **Sipser**, χρησιμοποιούμε κανονικές εκφράσεις για να περιγράψουμε γλώσσες.
- Στο **Linux**, χρησιμοποιούμε το `grep` για να εντοπίσουμε γραμμές κειμένου που ταιριάζουν με ένα συγκεκριμένο μοτίβο.

Πράξη	Sipser	grep -E
Ένωση	$A \cup B$	`A|B`
Συνένωση	AB	`AB`
Επανάληψη	A^*	`A*`
Μία ή περισσότερες φορές	AA^*	`A+`
Μηδέν ή μία φορά	$\varepsilon \cup A$	`A?`
Επιλογή χαρακτήρα	$a \cup b \cup c$	`[abc]`

Ένα παράδειγμα

Η κανονική έκφραση ab στον Sipser περιγράφει τη γλώσσα $\{ab\}$.

Η εντολή `grep -E 'ab' file.txt` βρίσκει **κάθε γραμμή που περιέχει** το ab, όπως για παράδειγμα τη γραμμή xxabyy.

Τι επιπλέον μας προσφέρει το grep;

Το `grep -E` διαθέτει μια πιο εύχρηστη σύνταξη, με αρκετές συντομογραφίες που μας διευκολύνουν στην αναζήτηση κειμένου.

Μοτίβο	Τι σημαίνει
`[0-9]`	Ένα ψηφίο
`[a-z]`	Ένα πεζό λατινικό γράμμα
`.`	Ένας οποιοσδήποτε χαρακτήρας
`^abc`	Η γραμμή αρχίζει με abc
`abc$`	Η γραμμή τελειώνει σε abc
`a{3}`	Ακριβώς τρία συνεχόμενα a

Παράδειγμα

```
grep -E '^[0-9]{5}$' file.txt
```

Βρίσκει γραμμές που αποτελούνται από **ακριβώς πέντε ψηφία**, όπως το `71201`.

Ίδια εκφραστική ισχύς — με μια εξαίρεση

Τι πρέπει να θυμόμαστε;

Όλες οι συντομογραφίες που είδαμε στο `grep -E` μπορούν να εκφραστούν και με τις κανονικές εκφράσεις του Sipser.

Δεν αποκτούμε μεγαλύτερη εκφραστική ισχύ: απλώς γράφουμε πιο εύκολα τα ίδια μοτίβα.

Πέρα από τις κλασικές κανονικές εκφράσεις

Ορισμένες επεκτάσεις, όπως τα *backreferences*, επιτρέπουν την αντιστοίχιση επαναλαμβανόμενων υποσυμβολοσειρών και μπορούν να περιγράψουν μη κανονικές γλώσσες.

Η θεωρία των DFA καλύπτει τον κανονικό πυρήνα των regex.

Εργαστηριακό υλικό: Regular Expressions και grep

Από τη θεωρία στην πράξη

Παραδείγματα και ασκήσεις για την αναζήτηση κειμένου με regular expressions στο Linux.

GitHub: CS280 -- grep_basics

github.com/tsourolampis/
cs280-theory-of-computation/tree/main/grep_basics