ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΘΕΜΕΛΙΩΔΗ ΘΕΜΑΤΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

(FOCS)

(2020-2021)

1η Σειρά Ασκήσεων

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

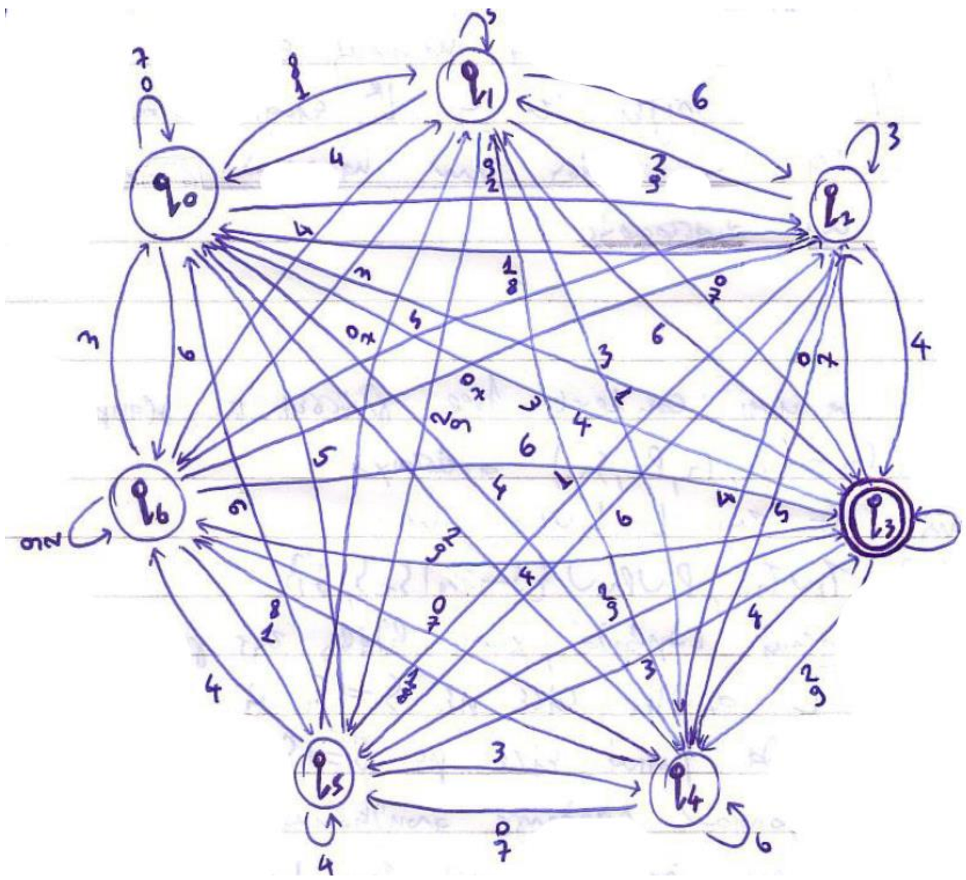
Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr

$1^{η}$ Άσκηση

Πεπερασμένο αυτόματο με αλφάβητο {0, …, 9} που να διαβάζει τα ψηφία ενός ακεραίου αριθμού n και να αποδέχεται αν n mod 7 = 3.

Στο αυτόματο αυτό, η κάθε κατάσταση καθορίζει το υπόλοιπο της διαίρεσης ενός αριθμού n με το 7. Πιο συγκεκριμένα, η q_0 αντιστοιχεί στο υπόλοιπο ίσο με 0, η q_1 αντιστοιχεί σε υπόλοιπο ίσο με 1 κλπ. Για την υλοποίηση του αυτόματου, έγινε χρήση του γεγονότος ότι, όταν το αυτόματο διαβάζει ένα δεκαδικό ψηφίο α, τότε n = 10n + a. Έτσι λοιπόν, η ενασχόληση αφορά μόνο διψήφιους αριθμούς, αφού με τη μετάβαση από το q_0 σε μια δεύτερη κατάσταση και από τις δεύτερες σε τρίτες καταστάσεις, συμπληρώνεται πλήρως το αυτόματο, ενώ δεν απαιτείται να εξεταστούν τα υπόλοιπα των διαιρέσεων τριψήφιων και μεγαλύτερων αριθμών με το 7. Οπότε:



$2^{η}$ Άσκηση

Δίνονται οι γλώσσες:

L1 = {w ∈{a, b, c}∗ | η w περιέχει την συμβολοσειρά 'bac' και όχι τη συμβολοσειρά 'cb'}.

L2 = {w ∈{0, 1}∗ | στη w κάθε '0' που εμφανίζεται ακολουθείται από τουλάχιστον δύο '1'}

L3 = {w ∈{a, b}∗ | w αρχίζει με 'a' και είναι περιττού μήκους ή αρχίζει με 'b' και είναι άρτιου μήκους }

(i) *Κατασκευάστε DFA με όσο το δυνατόν λιγότερες καταστάσεις για τις γλώσσες L1 και L2: Υπόδειξη: όπου βοηθάει, σχεδιάστε και συνδυάστε αυτόματα για απλούστερες γλώσσες.*

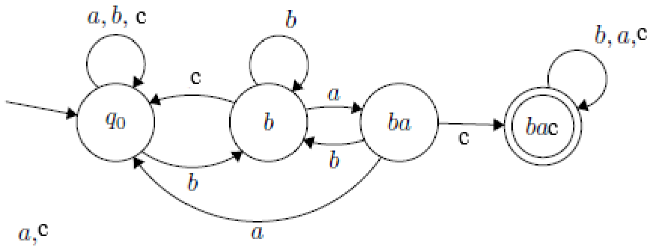
(ii) *Δώστε κανονική παράσταση για τις γλώσσες L1 και L2.*

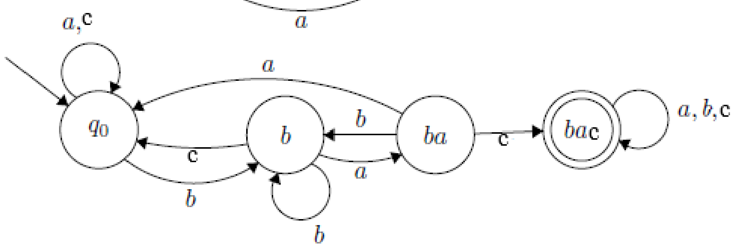
(iii) *Κατασκευάστε NFA για την γλώσσα L3 και βρείτε το ισοδύναμο DFA. Είναι το ελάχιστο; Αποδείξτε. Αν όχι, βρείτε το.*

a) Η κατασκευή των DFA παρουσιάζεται παρακάτω:

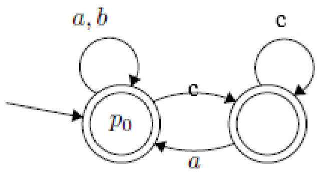
i) Για την γλώσσα L1:

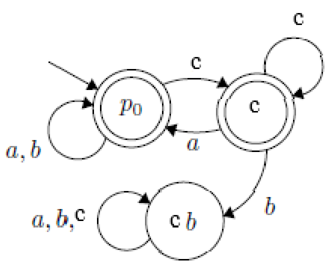
Η γλώσσα γράφεται σαν τομή των L1 := {w ∈ {a, b, c}∗ | η w περιέχει την συμβολοσειρά 'bac'} & L2 := { w ∈ {a, b, c}∗ | η w δεν περιέχει την συμβολοσειρά 'cb' }. Τα δύο αυτόματα M1, M2 που παράγουν τις γλώσσες είναι τα:





και





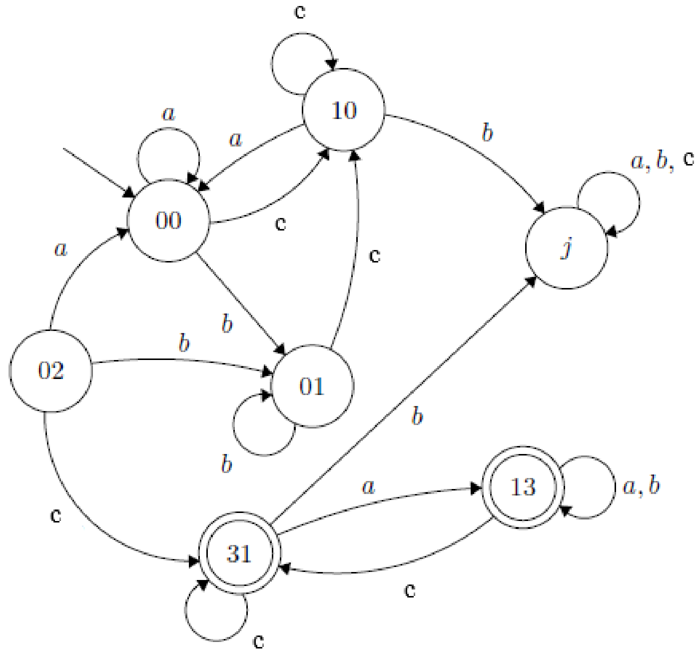
Άρα, η τομή θα παρθεί από το γινόμενο των DFA με αποδεκτές καταστάσεις τις p0q3, p1q3.

Ο πίνακας για την απλοποίηση του νέου DFA είναι:

$Q_1 \times Q_2 \mid \Sigma$	a	b	c	
00	00	01	10	
01	02	01	10	
02	00	01	13	
03	03	03	13	
10	00	21	10	
11	02	21	10	AXP.
12	00	21	13	AXP.
13	03	23	13	
20	20	21	20	
21	22	21	20	
22	20	21	23	
23	23	23	23	

01	X2								
02	X1	X1							
03	X0	X0	X0						
10	X2	X2	X1	X0					
13	X0	X0	X0	X1	X0				
20	X3	X2	X1	X0	X4	X0			
21	X3	X2	X1	X0	X4	X0			
22	X3	X2	X1	X0	X4	X0			
23	X3	X2	X1	X0	X4	X0			
	00	01	02	**03**	10	**13**	20	21	22

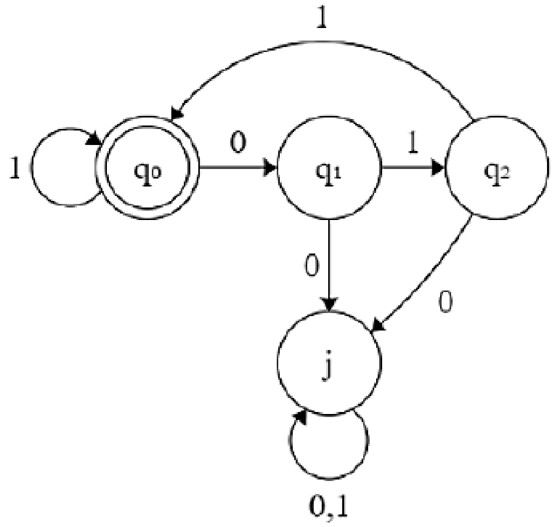
$p_2q_0 \equiv p_2q_1 \equiv p_2q_2 \equiv p_2q_3 \equiv j$:



ii) Κανονική έκφραση: L1 = (a + cc*a + bb*cc*a + bb*a(bb*a)*(a + bb*cc*a))*(bb*a(bb*a)*cc* + bb*a(bb*a)*cc*a(a + b + cc*a)*(ε + cc*))

b) i) Για την γλώσσα L2:

Κατασκευάζεται απευθείας το DFA (εξ’ ορισμού τα DFA είναι και NFA):



state/input	0	1
q_0	q_1	q_0
q_1	j	q_2
q_2	j	q_0
j	j	j

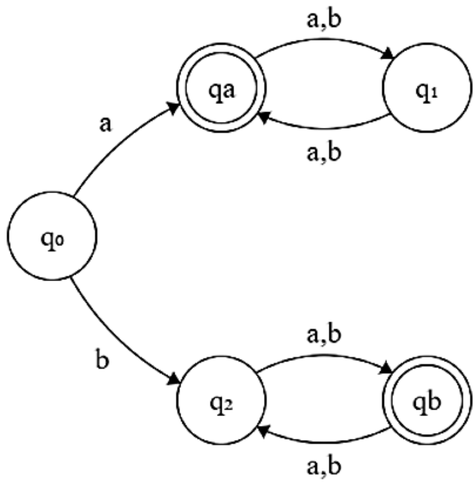
Έλεγχος για το αν είναι ελάχιστο:

q_0				
q_1	X			
q_2	X	X1		
j	X	X2	X1	
	q_0	q_1	q_2	j

Όλες οι καταστάσεις είναι διακρίσιμες άρα είναι ελάχιστο. Συγκεκριμένα, η $q0$ είναι η μόνη τελική άρα διακρίνεται από τις υπόλοιπες, η $q2$ διακρίνεται από την j και την $q1$ επειδή με είσοδο 1 οδηγεί σε τελική κατάσταση, και η $q1$ διακρίνεται από τη j γιατί με είσοδο 1 οδηγούν αντίστοιχα στις $q2$, j που τις έχουμε ήδη διακρίνει με X1.

ii) Κανονική έκφραση: L2 = (011 + 1)*

c) iii) Κατασκευάζεται απευθείας DFA για την L3:

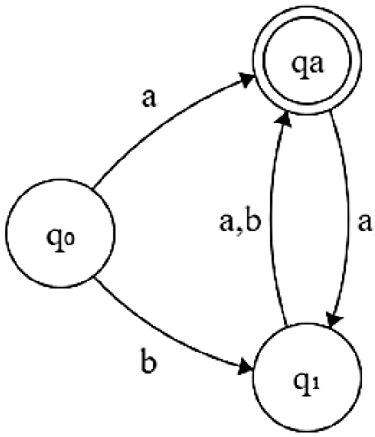


state/input	a	b
q0	qa	q2
qa	q1	q1
q1	qa	qa
q2	qb	qb
qb	q2	q2

Έλεγχος για το αν είναι ελάχιστο:

q0					
qa	X				
q1	X1	X			
q2	X1	X			
qb	X		X	X	
	q0	**qa**	**q1**	**q2**	**qb**

Όπως φαίνεται, μπορούμε να συγχωνεύσουμε τις qa, qb σε μία κατάσταση αποδοχής και τις q1, q2. Το ελάχιστο DFA της L3:



<u>3η Άσκηση</u>

Είναι κανονικές οι παρακάτω γλώσσες; Αν μια γλώσσα δεν είναι κανονική, να το αποδείξετε χρησιμοποιώντας είτε το Λήμμα Άντλησης είτε κάποια ιδιότητα κλειστότητας. Αν μια γλώσσα είναι κανονική, να το αιτιολογήσετε κατάλληλα.

α) $\{0^n1^m : n,m \geq 1, n \neq m\}$.

β) $\{0^n1+0^m : n,m \geq 1, n = m\}$.

γ) $\{w \in \{0, 1\}*$*: η* w *δεν είναι παλινδρομική*$\}$

δ) $\{ww : w \in \{0, 1\}*$ *το μήκος της* w *είναι* $\leq 10^{100}\}$.

a) Θεωρούμε την γλώσσα L1 με πλήθος πεπερασμένων καταστάσεων n- με z=(0n)(1m). Θέτουμε σπάσιμο $u=0^{j1}$ $v=0^{j2}$ $w=[0^{n-j1-j21}m]$, με j1,j2 > 0. Τότε, από το Λήμμα Άντλησης έχουμε ότι:
I) | z | = n+m>n
II) | v | = j2 > 0, αφού ορίσαμε j2 > 0
III) | uv | = j1+j2<=n
IV) Για $u(v^i)z$, έχουμε: $\{0^{j1}\}\{0(^{i*j2})\}\{0(^{n-j1-j2})1^m\}=\{0^{[n+(i-1)j1]}\}\{1^m\}$
Θέτοντας i=2, έχουμε: $\{0^{n+j2}\}\{1^m\}$, που δεν ανήκει στην γλώσσα L1, αφού για j2>0, είναι n+j2>n
Συνεπώς, από το (IV) βήμα, καταλήγουμε σε άτοπο και η γλώσσα L1 δεν είναι κανονική.

b) -

c) Αρχικά, αποδεικνύουμε ότι οι παλινδρομικές γλώσσες δεν είναι κανονικές, μέσω του Pumping Lemma. Συγκεκριμένα:
Θεωρούμε την γλώσσα L3-με πλήθος πεπερασμένων καταστάσεων n- με z=(a^n)b(a^n).
Θέτουμε σπάσιμο $u=a^{j1}$ $v=a^{j2}$ $w=[a^{n-j1-j2}][b][a^n]$,. Τότε, από το Λήμμα Άντλησης έχουμε ότι:
I) | z | = 2n+1>=n
II) | v | = j2 > 0, αφού ορίσαμε j2 >0
III) | uv | = j1+j2 <= n
IV) Για uviz, έχουμε: $zi=\{a^{j1}\}\{a(^{i*j2})\}\{a^{n-j1-j2}ba^n\}=\{a^{n+(i-1)j2}ba^n\}$. Για i=2, προκύπτει z2={a^[n+j2]ba^n}. Το παλίνδρομο z2^R της εν λόγω συμβολοσειράς z2 είναι $\{a^nba^{n+j2}\}$.
Προφανώς, $z2<> z2^R$. Συνεπώς, από το (IV) βήμα καταλήγουμε σε άτοπο και η γλώσσα L3 δεν είναι κανονική.
Ωστόσο, από τις ιδιότητες κλειστότητας, γνωρίζουμε ότι η κανονικότητα της αντίθετης γλώσσας L' είναι ίδια με αυτήν της L.
Εν προκειμένω, αφού η γλώσσα L3 δεν είναι κανονική, τότε δεν είναι κανονική και η L3'={wε{0,1}* : η w δεν είναι παλινδρομική}

d) Γνωρίζουμε ότι μία κανονική γλώσσα διαθέτει πεπερασμένο μήκος. Εν προκειμένω, το μήκος της γλώσσα L4 είναι <= (10^100), που αποτελεί πεπερασμένο μήκος. Συνεπώς, η γλώσσα L4 είναι κανονική.

<u>$4^η$ Άσκηση</u>

(α) Έστω G : S → aS | aSbS | ε. Περιγράψτε σε φυσική γλώσσα τη γλώσσα που παράγει η G .

(β) Δώστε γραμματική για τη γλώσσα $\{a^{3n}b^{2n} \mid n >=1\}$.

(γ) Δώστε γραμματική για τη γλώσσα {we{0,1} | το w έχει περιττό μήκος και στη μέση 111}.*

a) Με συνεχείς εφαρμογές των κανόνων παίρνουμε:
$S \rightarrow aSbS \rightarrow aaSbSbaSbS \rightarrow aabbaaSbaS \rightarrow aabbaaba$
Φαίνεται ότι παράγει μια γλώσσα $\{a,b\}$ με συμβολοσειρές στις οποίες αν διαλέξουμε οποιοδήποτε σύμβολο, τότε για τα σύμβολα στα αριστερά του θα ισχύει $πλήθος\ a \geq πλήθος\ b$. Δηλαδή, αν ξεκινήσουμε να το διαβάζουμε από αριστερά προς τα αριστερά, σε κάθε στιγμή θα έχουμε περισσότερα ή ίσα a σε σχέση με τα b. Επίσης περιλαμβάνει την κενή συμβολοσειρά.

b) Για κάθε 3 a στην αρχή πρέπει να παραθέτουμε 2 b στο τέλος άρα χρειαζόμαστε έναν κανόνα $S \rightarrow aaaSbb$. Πρέπει να προσέξουμε όμως ότι η κενή συμβολοσειρά δεν ανήκει στη γλώσσα άρα χρειαζόμαστε ένα ακόμα μη τερματικό σύμβολο.
Η γραμματική για τη γλώσσα είναι: G=($\{S,A\},\{a,b\},\{S \rightarrow aaaAbb, A \rightarrow aaaAbb|\varepsilon\},\{S\}$)

c) Θέτουμε κανόνες ώστε να συμπεριλάβουμε κάθε πιθανή συμβολοσειρά: $S \rightarrow 0S0 \mid 0S1 \mid 1S0 \mid 1S1$.
Για να τερματίσει βάζουμε επίσης τον κανόνα $S \rightarrow 111$. Προφανώς το 111 θα είναι στη μέση αφού κάθε φορά που χρησιμοποιούμε τους άλλους κανόνες η συμβολοσειρά επεκτείνεται κατά 1 δεξιά και 1 αριστερά, και θα έχει περιττό μήκος 3+2×($φορές\ που\ εφαρμόζεται\ άλλος\ κανόνας$).
Η γραμματική: G=($\{S\},\{0,1\},\{S \rightarrow 0S0 \mid 0S1 \mid 1S0 \mid 1S1 \mid 111\},\{S\}$)

$5^η$ Άσκηση

Εξετάστε αν η κλάση των γλωσσών χωρίς συμφραζόμενα είναι κλειστή ως προς τις πράξεις ένωση και παράθεση.

Έστω CFL γλώσσα L. Ορίζουμε τη γλώσσα LR έτσι ώστε:

• Για κάθε κανόνα της μορφής $A \rightarrow \varepsilon$ ή $A \rightarrow a$ στην L, η L^R περιέχει τον ίδιο κανόνα.

• Για κάθε κανόνα της μορφής $A \rightarrow w1Bw2$ στην L, η L^R περιέχει τον κανόνα

$A \rightarrow w2^R B w1^R$

Έστω uv που παράγεται από την L. Τότε $(uv)^R = v^R u^R = vu$ που παράγεται από την L^R σύμφωνα με τα παραπάνω.

Επαγωγική υπόθεση: έστω n σύμβολα έτσι ώστε η συμβολοσειρά $a_1 a_2 \ldots a_n$ να παράγεται από την L και η αντίστροφη συμβολοσειρά $a_n a_{n-1} \ldots a_1$ να παράγεται από την L^R.

Για n+1 σύμβολα παίρνουμε: η L παράγει την $(a_1 a_2 \ldots a_n) a_{n+1} = a_1 a_2 \ldots a_n a_{n+1}$ ενώ η L^R παράγει την $a_{n+1}{}^R (a_1 a_2 \ldots a_n)^R = a_{n+1} a_n a_{n-1} \ldots a_1 = (a_1 a_2 \ldots a_n a_{n+1})^R$.

Άρα και η L^R είναι CFL.

Όμοια, για κάθε vu που παράγεται από την L^R η L παράγει την $(vu)^R = uv$ και με παρόμοια επαγωγική διαδικασία προκύπτει ότι και η L είναι CFL.

Άρα οι CFL γλώσσες είναι κλειστές ως προς την αναστροφή.

$6^η$ Άσκηση (Υπολογισιμότητα)

Αποδείξτε ότι το πρόβλημα του ελέγχου αν ένα πρόγραμμα τερματίζει για όλες τις εισόδους είναι μη επιλύσιμο.

Έστω ότι υπάρχει πρόγραμμα Π που παίρνει σαν είσοδο οποιοδήποτε πρόγραμμα Π' και βρίσκει αν το Π' τερματίζει για όλες τις εισόδους. Θα δείξουμε ότι αυτό δεν γίνεται γιατί συνεπάγεται λύση στο Halting Problem, που γνωρίζουμε ότι είναι μη επιλύσιμο.

Έστω ότι διαθέτουμε πρόγραμμα Β, που δέχεται εισόδους x και θέλουμε να αποφανθούμε αν θα τερματίσει για μια συγκεκριμένη είσοδο n. Τότε τροποποιούμε το Β ως εξής και φτιάχνουμε το Β':

• Το Β' αποθηκεύει τη συμβολοσειρά n ως σταθερά στην αρχή. Καθώς κατασκευάζουμε το Β' επιλέγουμε κατά βούληση τη n αυτή.

• Όταν αρχίζει η εκτέλεση του Β' για είσοδο x, συγκρίνεται η x με τη n.

• Αν $x \neq n$, το Β' σταματά.

• Αλλιώς αν $x = n$, εκτελείται το Β για είσοδο $x = n$

Με αυτήν την τροποποίηση και επιλέγοντας την ενδιαφερόμενη τιμή του n κατά τη δημιουργία του Β', μπορούμε να δώσουμε το Β' ως είσοδο στο Π και να λύσουμε το Halting Problem. Συγκεκριμένα, αν το Β τερματίζει για είσοδο n, τότε το Β' τερματίζει για κάθε είσοδο (αφού εξ'ορισμού τερματίζει για κάθε είσοδο $x \neq n$), άρα το Π θα δώσει θετική απάντηση. Αντίθετα, αν το Β δεν τερματίζει για είσοδο n, τότε το Β' δεν τερματίζει για όλες τις εισόδους, άρα το Π θα δώσει αρνητική απάντηση.

Αφού όμως το πρόβλημα τερματισμού είναι μη επιλύσιμο πρόβλημα, συμπεραίνουμε ότι δεν μπορεί να υπάρξει τέτοιο πρόγραμμα Π, επομένως το πρόβλημα ελέγχου αν ένα πρόγραμμα τερματίζει για όλες τις εισόδους είναι μη επιλύσιμο.

7η Άσκηση (Λογική & Αλγόριθμοι)

Διατυπώστε αποδοτικό αλγόριθμο που να δέχεται σαν είσοδο οποιονδήποτε τύπο της προτασιακής λογικής σε διαζευκτική κανονική μορφή (DNF) και να αποφαίνεται αν είναι ικανοποιήσιμος. Σε περίπτωση που είναι θα πρέπει να επιστρέφει μία ανάθεση αληθοτιμών που ικανοποιεί τον τύπο.

Μία παράσταση DNF είναι αληθής αν τουλάχιστον ένας όρος της είναι αληθής. Αφού η είσοδος δίνεται σε τέτοια μορφή, αρκεί να ελέγξουμε αν ένας όρος της είναι ικανοποιήσιμος. Οι όροι δίνονται σε συζεύξεις literals ($xi1 \cap xi2 \cap \ldots \cap xin$) , άρα είναι ικανοποιήσιμοι εκτός αν περιέχουν $xi \cap \neg xi$. Επομένως ανάμεσα στις διαζεύξεις πρέπει να ελέγχουμε αν υπάρχει ταυτόχρονα xi και $\neg xi$ για κάθε μεταβλητή του όρου.

Στην περίπτωση που βρούμε όρο που να ικανοποιείται, επιστρέφουμε 'Ναι'. Η ανάθεση τιμών είναι απλή: αναθέτουμε αληθοτιμή true σε κάθε θετικό literal και false σε κάθε αρνητικό. Οι μεταβλητές που δεν εμφανίζονται στον ικανοποιήσιμο όρο δεν επηρεάζουν το αποτέλεσμα άρα τις θέτουμε αυθαίρετα. Στην περίπτωση που κανένας όρος δεν ικανοποιείται επιστρέφουμε 'Όχι'.

Ο αλγόριθμος παρατίθεται παρακάτω. Η πολυπλοκότητα του αλγορίθμου είναι $O(n^2)$ αφού στη χειρότερη περίπτωση συγκρίνουμε κάθε literal με όλα τα υπόλοιπα.

```
Begin
create data structure M (index, value)
        set global flag f=FALSE

        for every term T:
                if global flag f is FALSE then:

                        create data structure N (index, value)
                        set flag g=TRUE

                        for every literal x_i in T:
                                if index i is not in M then: store (i, no value) in M
                                if x_i is positive then:
                                        if index i is not in N then: store (i, TRUE) in N
                                        else if [value of i in N] is FALSE then: set flag g=FALSE
                                else if x_i is negative then:
                                        if index i is not in N then: store (i, FALSE) in N
                                        else if [value of i in N] is TRUE then: set flag g=FALSE
                        end loop

                        for every index i in N:
                                set [value of i in M] = [value of i in N]
                        end loop
                        if flag g=TRUE then: set global flag f=TRUE

                else if global flag f is TRUE then:
                        for every literal x_i in T:
                                if index i is not in M then: store (i, TRUE) in M
        end loop
if global flag f = TRUE then:
        output 'Yes'
        for every i in M: output [value of i in M]
else if global flag f = FALSE then:
        output 'No'

End
```

<u>$8^η$ Άσκηση (Πολυπλοκότητα: αναγωγές προς απόδειξη δυσκολίας)</u>

Έστω ένα μη κατευθυνόμενο γράφημα G = (V,E). Κλίκα λέγεται ένα υπογράφημα του G το οποίο είναι πλήρες (δηλ. όλες οι κορυφές του ενώνονται με ακμή). Ένα σύνολο κορυφών του G λέγεται ανεξάρτητο σύνολο αν κάθε δύο κορυφές του δε συνδέονται με ακμή. Θεωρήστε τα προβλήματα απόφασης: (i) Independent Set, το οποίο δεδομένης εισόδου (G, k) έχει θετική απάντηση αν και μόνο αν το γράφημα G περιέχει κάποιο ανεξάρτητο σύνολο μεγέθους τουλάχιστον k, (ii) Clique, το οποίο δεδομένης εισόδου (G, k) έχει θετική απάντηση μεγέθους τουλάχιστον k και (iii) Dense Subgraph, το οποίο δεδομένου ενός γραφήματος G και δύο θετικών ακέραιων a, b έχει θετική απάντηση αν και μόνο αν το G περιέχει ένα σύνολο κορυφών μεγέθους a έτσι ώστε να υπάρχουν τουλάχιστον b ακμές μεταξύ τους.

(α) Δείξτε ότι αν είναι γνωστό ότι το πρόβλημα Clique είναι NPπλήρες, τότε και το πρόβλημα Independent Set είναι NPπλήρες.

(β) Δείξτε επίσης ότι το πρόβλημα Dense Subgraph είναι NPπλήρες.

<u>Γενική μέθοδος</u>:

(α) Δείχνουμε ότι το ζητούμενο πρόβλημα A είναι NP.

(β) Βρίσκουμε ένα γνωστό NP-πλήρες πρόβλημα B.

(γ) Βρίσκουμε πολυωνυνικό αλγόριθμο μετάβασης από το B στο A.

(δ) Δείξαμε ότι το A είναι NP-πλήρες αφού-με δεδομένη λύση από το B-καταλήξαμε στο A σε πολυωνυμικό χρόνο

(i) Αρχικά, παρατηρούμε ότι η κλίκα με το ανεξάρτητο σύνολο αποτελούν αντίστροφα είδη υπογραφημάτων, καθώς το ανεξάρτητο σύνολο χαρακτηρίζεται από απουσία ακμών ανάμεσα σε k κορυφές, ενώ η κλίκα χαρακτηρίζεται από την ένωση όλων των k κορυφών με ακμές.

(α)Ισχύει ότι το πρόβλημα Clique είναι NP, δηλαδή ότι μία δεδομένη λύση επαληθεύεται σε πολυωνυμικό χρόνο. Αυτό ισχύει, αν αναλογισθούμε ότι η πλειονότητα των γράφων υλοποιείται με πίνακα(nxn) ή λίστα(n). Έτσι, απαιτείται πολυωνυμικός αλγόριθμος, έτσι ώστε να ελέγξουμε αν ένα δεδομένο υποσύνολο κορυφών (ή δεδομένη λύση) είναι κλίκα ή όχι. Συγκεκριμένα, για τον εν λόγω αλγόριθμο, έχουμε φράγμα O((k-1)k-Σi)=>αν το υπογράφημα κλίκα και αναγκαστικά ελεγχθούν οι ακμές ανάμεσα σε όλες τις κορυφές.

(β)Έστω ότι έχουμε έναν γράφο G(V,E), ο οποίος γνωρίζουμε ότι διαθέτει ανεξάρτητο σύνολο μεγέθους k. Μάλιστα, το πρόβλημα Indepedent Set γνωρίζουμε ότι είναι NP-Πλήρες,και κατ’επέκταση NP, οπότε δίνει θετική απάντηση για τον γράφο G σε πολυωνυμικό χρόνο.

(γ)Στην συνέχεια, δημιουργούμε τον γράφο G’, ο οποίος είναι ο αντίστροφος του G. Αν ο G γράφος περιλαμβάνει ανεξάρτητο σύνολο μεγέθους κ, τότε ο G’ γράφος περιλαμβάνει κλίκα αντίστοιχου μεγέθους. Κατ’αυτόν τον τρόπο, το Indepedent Set μετατράπηκε σε Clique.

Αξίζει να σημειωθεί ότι input του Indepedent Set είναι ο αρχικός γράφος G, και output του/input του Clique είναι ο G’. Δηλαδή, για την αλγοριθμική συνάρτηση f, ισχύει ότι: f: G → G’

Μάλιστα, ο αλγόριθμος μετατροπής f του ανεξάρτητου συνόλου σε κλίκα κοστίζει πολυωνυμικό χρόνο, μιας και απαιτούνται a*[k(k-1) – Σi (i=1 μέχρι και i=k-1)] προσθήκες ακμών (το οποίο είναι το μόνο που απαιτείται για την εν λόγω μετατροπή των a ανεξάρτητων συνόλων σε a κλίκες).

(δ)Έτσι, με δεδομένη λύση το πρόβλημα Clique λύνεται σε πολυωνυμικό χρόνο, εφόσον η διαδικασία μετάβασης σε αυτό αποτελείται από δύο επιμέρους πολυωνυμικές διαδικασίες. Συνεπώς, αν το πρόβλημα Indepedent Set είναι NP-Πλήρες τότε είναι και το Clique.

(ii)Λειτουργούμε όμοια με το (i).

(α)Το πρόβλημα Dense Subgraph είναι NP, μιας και απαιτούνται από b(ελέγχει μόνο τις ακμές) μέχρι x(ελέγχει όλες τις πιθανές ακμές) βήματα για την επαλήθευση μίας δεδομένης λύσης, δηλαδή έχουμε πολυωνυμικό κόστος.

(β)Επίσης, το πρόβλημα Clique είναι από το προηγούμενο ερώτημα NP-Πλήρες, δηλαδή απαιτείται πολυωνυμικός χρόνος για την επαλήθευση μίας λύσης.

(γ)Παρόλα αυτά, αλλάζει ο αλγόριθμος μετάβασης από Clique(a) σε Dense Subgraph(a,b). Συγκεκριμένα, ο αλγόριθμος πρέπει να μετατρέπει τις x ακμές της κλίκας του γράφου G(V,E) σε y, με x=[a(a-1)-Σi(i=1 μέχρι και i=a-1)] και b<=y<=x. Συγκεκριμένα, απαιτούνται y-x βήματα για την ολοκλήρωση του αλγορίθμου, που αντιστοιχεί σε πολυωνυμικό χρόνο.

(δ)Συνεπώς, η μετάβαση από Clique σε Dense Subgraph απαιτεί δύο πολυωνυμικές διεργασίες(έλεγχος λύσης Clique και αλγόριθμος μετατροπής. Επομένως, αφού το πρόβλημα Clique είναι NP-Πλήρες είναι και το Dense Subgraph