ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Λογικός Προγραμματισμός

Ασκήσεις

Παναγιώτης Σταματόπουλος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Περιεχόμενα

1. Β΄ Ομάδα Ασκήσεων "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2010-11

Οι ασκήσεις της ομάδας αυτής πρέπει να αντιμετωπισθούν με τη βοήθεια της τεχνολογίας του **προγραμματισμού με περιορισμούς**. Ένα σύστημα λογικού προγραμματισμού που υποστηρίζει την τεχνολογία αυτή είναι η **ECL^{i}PS^e**. Μπορείτε να χρησιμοποιήσετε είτε την παλιότερη βιβλιοθήκη **`fd`** είτε την νεώτερη **`ic`**. Η βιβλιοθήκη **`fd`** τεκμηριώνεται στο κεφάλαιο 2 του "Obsolete Libraries Manual" και η **`ic`** στα κεφάλαια 3 και 4 του "Constraint Library Manual". Αν χρησιμοποιήσετε την **`ic`**, θα σας χρειαστεί και η βιβλιοθήκη **`branch_and_bound`**, ιδιαιτέρως το κατηγόρημα `bb_min/3`, το οποίο τεκμηριώνεται, όπως και όλα τα κατηγορήματα που παρέχει η **ECL^{i}PS^e**, στο "Alphabetical Predicate Index".

1.1 Άσκηση 4

Στις αυτοκινητοβιομηχανίες, τα αυτοκίνητα κατασκευάζονται σε μία γραμμή συναρμολόγησης (assembly line). Σ' ένα δεδομένο χρονικό διάστημα, π.χ. μία ημέρα, πρέπει να κατασκευασθεί ένας συγκεκριμένος αριθμός αυτοκινήτων από κάποιο μοντέλο που παράγει η εταιρεία, τα οποία δεν είναι κατ' ανάγκη απολύτως όμοια στον εξοπλισμό τους. Κάποια αυτοκίνητα μπορεί να διαθέτουν κλιματισμό, κάποια άλλα όχι. Κάποια μπορεί να έχουν ηλιοροφή, άλλα όχι. Γενικώς, υπάρχει ένα σύνολο από επιλογές (κλιματισμός, ηλιοροφή, ηχοσύστημα, δερμάτινα καθίσματα, ζάντες αλουμινίου, κλπ.), που, με βάση τις παραγγελίες προς την εταιρεία, άλλα αυτοκίνητα πρέπει να διαθέτουν κάποια επιλογή, ενώ άλλα όχι. Δύο αυτοκίνητα που απαιτούν τις ίδιες ακριβώς επιλογές λέμε ότι ανήκουν στην ίδια σύνθεση (configuration).

Για κάθε πιθανή επιλογή στον εξοπλισμό του μοντέλου, υπάρχει μία ομάδα εργασίας που δουλεύει σε κάποια περιοχή της γραμμής συναρμολόγησης. Όμως, για να μπορεί η κάθε ομάδα εργασίας να φέρει σε πέρας το έργο της, πρέπει, όσον αφορά την επιλογή με την οποία ασχολείται, να μην υπάρχουν ποτέ σε κάθε Μ συνεχόμενα αυτοκίνητα στη γραμμή συναρμολόγησης περισσότερα από Κ που να απαιτούν την επιλογή. Τα Μ και Κ είναι συγκεκριμένα για κάθε δυνατή επιλογή. Η απαίτηση αυτή ονομάζεται περιορισμός χωρητικότητας (capacity constraint).

Το πρόβλημα της δρομολόγησης παραγωγής αυτοκινήτων (car sequencing) συνίσταται στον καθορισμό της σειράς με την οποία πρέπει να παραχθούν Ν αυτοκίνητα σε μία γραμμή συναρμολόγησης, έχοντας επίσης δεδομένες τις δυνατές επιλογές, τους περιορισμούς χωρητικότητάς τους και τα πλήθη των αυτοκινήτων που αντιστοιχούν σε επιθυμητές συνθέσεις (συνδυασμοί επιλογών).

Ορίστε ένα κατηγόρημα `carseq/1`, το οποίο όταν καλείται σαν `carseq(S)`, να επιστρέφει στο `S` τη σειρά με την οποία πρέπει να τοποθετηθούν αυτοκίνητα των διαφόρων πιθανών συνθέσεων σε μία γραμμή συναρμολόγησης. Το `S` που επιστρέφεται πρέπει να είναι μία λίστα, κάθε στοιχείο της οποίας είναι ο αύξων αριθμός της σύνθεσης στην οποία ανήκει το αντίστοιχο αυτοκίνητο στη γραμμή συναρμολόγησης. Τα δεδομένα του προβλήματος δίνονται μέσω των κατηγορημάτων `classes/1` και `options/1`. Ένα παράδειγμα δεδομένων είναι το εξής:

```
classes([1,1,2,2,2,2]).
options([2/1/[1,0,0,0,1,1],
         3/2/[0,0,1,1,0,1],
         3/1/[1,0,0,0,1,0],
         5/2/[1,1,0,1,0,0],
         5/1/[0,0,1,0,0,0]]).
```

Τα παραπάνω δεδομένα περιγράφουν την εξής κατάσταση: Υπάρχουν 6 πιθανές συνθέσεις, όσα είναι τα στοιχεία της λίστας που είναι όρισμα στο κατηγόρημα `classes/1`. Η πρώτη σύνθεση περιλαμβάνει 1 αυτοκίνητο, η δεύτερη επίσης 1, η τρίτη 2 αυτοκίνητα, κοκ. Συνολικά πρέπει να κατασκευασθούν 10 αυτοκίνητα, όσο είναι το άθροισμα των στοιχείων της λίστας που δίνεται μέσω του `classes/1`. Υπάρχουν 5 πιθανές επιλογές, όσα είναι τα στοιχεία της λίστας που είναι όρισμα στο κατηγόρημα

`options/1`. Κάθε επιλογή ορίζεται μέσω μίας τριάδας `M/K/O`, όπου τα `M` και `K` εκφράζουν τον περιορισμό χωρητικότητας της επιλογής (το πολύ K αυτοκίνητα σε κάθε M συνεχόμενα στη γραμμή συναρμολόγησης πρέπει να απαιτούν την επιλογή) και το O είναι μία λίστα από 1 και 0, μήκους όσο το πλήθος των πιθανών συνθέσεων, που εκφράζει το αν η εν λόγω επιλογή περιλαμβάνεται στην αντίστοιχη σύνθεση (τιμή 1) ή όχι (τιμή 0). Για παράδειγμα, το πρώτο στοιχείο της λίστας που δίνεται μέσω του options/1 αντιστοιχεί σε μία επιλογή (π.χ. κλιματισμός) και εκφράζει ότι πρέπει να υπάρχει το πολύ 1 αυτοκίνητο σε κάθε 2 συνεχόμενα στη γραμμή συναρμολόγησης που να απαιτεί την επιλογή και ότι η εν λόγω επιλογή περιλαμβάνεται στην 1η, την 5η και την 6η σύνθεση. Ομοίως και για τις άλλες επιλογές.

Τα γεγονότα `classes/1` και `options/1` που δίνονται παραπάνω μπορείτε να τα βρείτε στο αρχείο http://www.di.uoa.gr/~takis/carseq_data1.pl. Τα αποτελέσματα εκτέλεσης για τα δεδομένα αυτά είναι:

```
?- carseq(S).
S = [1, 2, 6, 3, 5, 4, 4, 5, 3, 6]   ---> ;

S = [1, 3, 6, 2, 5, 4, 3, 5, 4, 6]   ---> ;

S = [1, 3, 6, 2, 6, 4, 5, 3, 4, 5]   ---> ;

S = [5, 4, 3, 5, 4, 6, 2, 6, 3, 1]   ---> ;

S = [6, 3, 5, 4, 4, 5, 3, 6, 2, 1]   ---> ;

S = [6, 4, 5, 3, 4, 5, 2, 6, 3, 1]   ---> ;

no
```

Το πρώτο από τα αποτελέσματα (`[1, 2, 6, 3, 5, 4, 4, 5, 3, 6]`) εκφράζει ότι η αλυσίδα συναρμολόγησης πρέπει να αποτελείται κατά σειρά από ένα αυτοκίνητο της 1ης σύνθεσης, ένα της 2ης, ένα της 6ης, ένα της 3ης, κοκ. Όπως παρατηρείτε, για τα δεδομένα αυτά, υπάρχουν έξι δυνατές λύσεις.

Δεδομένα για άλλο στιγμιότυπο του προβλήματος, μεγαλύτερου μεγέθους, βρίσκονται στο http://www.di.uoa.gr/~takis/carseq_data2.pl. Τα πρώτα αποτελέσματα γι' αυτό το στιγμιότυπο είναι:

```
?- carseq(S).
S = [1, 4, 7, 15, 3, 16, 2, 10, 8, 15, 1, 13, 2, 15, 8, 9, 2,
     17, 3, 15, 6, 10, 3, 17, 5, 9, 7, 11, 5, 17, 1, 10, 8,
     15, 5, 12, 5, 10, 7, 11, 5, 16, 3, 10, 7, 11, 5, 16, 3,
     10, 7, 11, 5, 16, 3, 15, 7, 11, 1, 17, 3, 15, 7, 9, 5,
     17, 5, 11, 6, 15, 5, 17, 7, 9, 8, 15, 7, 11, 1, 17, 8,
     17, 14, 11, 7, 16, 7, 11, 14, 17, 11, 14, 17, 11, 14, 17,
     11, 14, 17, 18]   --> ;

S = [1, 4, 7, 15, 3, 16, 2, 10, 8, 15, 1, 13, 2, 15, 8, 9, 2,
     17, 3, 15, 6, 10, 3, 17, 5, 9, 7, 11, 5, 17, 1, 10, 8,
     15, 5, 12, 5, 10, 7, 11, 5, 16, 3, 10, 7, 11, 5, 16, 3,
     10, 7, 11, 5, 16, 3, 15, 7, 11, 1, 17, 3, 15, 7, 9, 5,
     17, 5, 11, 6, 15, 5, 17, 7, 9, 8, 15, 7, 11, 1, 17, 8,
     17, 14, 11, 7, 16, 7, 11, 14, 17, 11, 14, 17, 11, 14, 17,
     11, 17, 14, 18]   --> ;

.....................................................
```

Μην αποπειραθείτε να βρείτε όλες τις λύσεις για τα προηγούμενα δεδομένα. Είναι υπερβολικά πολλές.

Στο αρχείο http://www.di.uoa.gr/~takis/carseq_data3.pl, βρίσκεται και ένα τρίτο στιγμιότυπο του προβλήματος, αρκετά πιο δύσκολο στην επίλυσή του από τα προηγούμενα.[1]

Παραδοτέο για την άσκηση είναι ένα αρχείο που περιλαμβάνει τον πηγαίο κώδικα Prolog που θα γράψετε.

1.2 Άσκηση 5

Στο συμμετρικό πρόβλημα του πλανόδιου πωλητή (symmetric traveling salesman problem), είναι δεδομένο ένα σύνολο από πόλεις και, για κάθε πιθανό ζευγάρι πόλεων C_1 και C_2, είναι γνωστό το κόστος μετάβασης από τη μία προς την άλλη, που είναι το ίδιο για οποιαδήποτε από τις δύο πιθανές κατευθύνσεις. Αυτό το κόστος μπορεί να είναι, για παράδειγμα, η απόσταση των δύο πόλεων. Το ζητούμενο του προβλήματος είναι να βρεθεί με ποια σειρά πρέπει να επισκεφθεί ο πωλητής όλες τις πόλεις, αρχίζοντας από κάποια και καταλήγοντας σ' αυτήν από την οποία ξεκίνησε, ώστε το συνολικό κόστος των μεταβάσεών του να είναι το ελάχιστο δυνατό.

Δεδομένα για το πρόβλημα, που αντιστοιχούν σε ένα δίκτυο 12 πόλεων, φαίνονται στη συνέχεια. Τα δεδομένα αυτά μπορείτε να τα πάρετε από το αρχείο http://www.di.uoa.gr/~takis/tsp_data.pl.

```
costs([[31,12,7,2,17,19,25,21,30,10,32],

       [13,11,8,29,23,14,17,4,21,11],

       [9,14,28,25,11,32,14,10,8],

       [22,11,15,8,13,7,23,25],

       [18,19,3,29,5,18,34],

       [14,30,9,22,17,11],

       [22,11,10,8,19],

       [8,24,42,33],

       [17,6,32],

       [15,28],

       [22]]).
```

Η λίστα που δίνεται σαν όρισμα στο κατηγόρημα costs/1 περιλαμβάνει στοιχεία λίστες, όπου η πρώτη περιέχει τα κόστη μετάβασης από την 1η πόλη προς όλες τις επόμενές της (2η, 3η, κλπ.), η δεύτερη λίστα περιλαμβάνει τα κόστη μετάβασης από τη 2η πόλη προς όλες τις επόμενές της (3η, 4η, κλπ.), κοκ. Το τελευταίο στοιχείο (22) είναι το κόστος μετάβασης από την 11η πόλη στη 12η (ή αντίστροφα).

Ορίστε ένα κατηγόρημα tsp/1, το οποίο όταν καλείται σαν tsp(N, R, C), να επιλύει το συμμετρικό πρόβλημα του πλανόδιου πωλητή, λαμβάνοντας ως είσοδο τις N **τελευταίες** πόλεις των δεδομένων που έχουν οριστεί μέσω του κατηγορήματος costs/1 (δηλαδή τα N-1 τελευταία στοιχεία του ορίσματός του). Το κατηγόρημα να επιστρέφει στο R τη βέλτιστη σειρά επίσκεψης των N πόλεων και στο C το βέλτιστο συνολικό κόστος. Παραδείγματα εκτέλεσης είναι τα εξής:

[1] Αποτελεί πρόκληση να βρεθεί έστω και μία λύση γι' αυτό το πρόβλημα μέσω προγραμματισμού με περιορισμούς. Στην περίπτωση που κάποιος/κάποια από τους/τις φοιτητές/φοιτήτριες του μαθήματος το κατορθώσει, αυτό θα εκτιμηθεί δεόντως ☺.

```
?- tsp(1, R, C).
R = [1]
C = 0

?- tsp(2, R, C).
R = [1, 2]
C = 44

?- tsp(3, R, C).
R = [1, 2, 3]
C = 65

?- tsp(4, R, C).
R = [1, 2, 4, 3]
C = 73

?- tsp(5, R, C).
R = [1, 2, 4, 5, 3]
C = 88


?- tsp(6, R, C).
R = [1, 4, 2, 3, 5, 6]
C = 89

?- tsp(7, R, C).
R = [1, 4, 3, 5, 2, 6, 7]
C = 92

?- tsp(8, R, C).
R = [1, 4, 5, 2, 8, 7, 3, 6]
C = 76

?- tsp(9, R, C).
R = [1, 3, 9, 4, 8, 6, 5, 2, 7]
C = 78

?- tsp(10, R, C).
R = [1, 2, 6, 3, 8, 5, 9, 7, 4, 10]
C = 77

?- tsp(11, R, C).
R = [1, 4, 7, 3, 2, 11, 5, 8, 10, 6, 9]
C = 84

?- tsp(12, R, C).
R = [1, 4, 2, 10, 7, 11, 9, 6, 12, 3, 8, 5]
C = 90
```

Παραδοτέο για την άσκηση είναι ένα αρχείο που περιλαμβάνει τον πηγαίο κώδικα Prolog που θα γράψετε.

1.3 Άσκηση 6

Αντιμετωπίστε τις προηγούμενες ασκήσεις της Β΄ ομάδας (4 και 5) με τη βοήθεια της βιβλιοθήκης C++ για προγραμματισμό με περιορισμούς, τον Naxos Solver (http://www.di.uoa.gr/~pothitos/naxos/), που έχει σχεδιάσει και αναπτύξει ο υποψήφιος διδάκτωρ του Τμήματός μας **Νίκος Ποθητός** (pothitos@di.uoa.gr).

Παραδοτέο για την άσκηση είναι ένα αρχείο zip, που θα περιλαμβάνει όλη τη δουλειά σας για τα δύο προβλήματα, κατάλληλα οργανωμένη σε καταλόγους ανά πρόβλημα. Μην περιλάβετε μέσα στο αρχείο zip εκτελέσιμα ή αντικειμενικά προγράμματα.

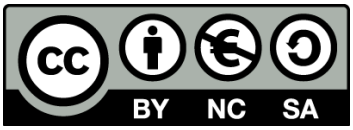
Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων

- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.