ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Λογικός Προγραμματισμός

Ασκήσεις

Παναγιώτης Σταματόπουλος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Περιεχόμενα

1. Ασκήσεις "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2004-05

1.1 Άσκηση 1 (0.2 μονάδες)

Γράψτε ένα κατηγόρημα `cross/1` το οποίο, όταν καλείται σαν `cross(N)`, να εκτυπώνει σε σχήμα σταυρού, με διαγώνιο μήκους N, αριθμούς κατά σειρά αρχίζοντας από το 1, κυκλικά και με δεξιόστροφη φορά, πρώτα από το εξωτερικό του σταυρού και προχωρώντας σταδιακά προς το κέντρο του. Φροντίστε κατά την εκτύπωση οι στήλες να έχουν σωστή στοίχιση. Συγκεκριμένα, κάθε αριθμός να εκτυπώνεται στη δεξιά πλευρά τόσων θέσεων όσο είναι το πλήθος των ψηφίων του μεγαλύτερου αριθμού που πρόκειται να εκτυπωθεί συν ένα. Κάποια παραδείγματα ερωτήσεων θα κάνουν πιο σαφές το ζητούμενο.

```
?- cross(3).
 1   2
   5
 4   3
yes

?- cross(6).
  1              2
     5        6
        9 10
       12 11
     8        7
  4              3
yes

?- cross(11).
  1                           2
     5                     6
        9                10
          13           14
             17    18
                21
             20    19
          16           15
       12                11
     8                     7
  4                           3
yes
```

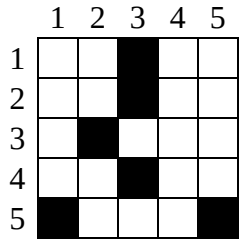
1.2 Άσκηση 2 (0.3 μονάδες)

Έστω ότι έχετε στη διάθεσή σας ένα μικροεπεξεργαστή που διαθέτει N καταχωρητές R_1, R_2, ..., R_N οι οποίοι έχουν δακτυλιοειδή διευθέτηση. Αυτό σημαίνει ότι μπορεί κάποιος να μεταφέρει τα περιεχόμενα του καταχωρητή R_i στον καταχωρητή R_{i+1}, για $1 \le i < N$, και του R_N στον R_1 με τις εντολές `move(i)`, για $1 \le i < N$, και `move(N)`, αντίστοιχα. Επίσης, μπορεί να αντιμεταθέσει τα περιεχόμενα των καταχωρητών R_i και R_j με την εντολή `swap(i,j)`, όπου i < j. Έστω, τώρα, ότι σας δίνονται τα αρχικά περιεχόμενα των N καταχωρητών, καθώς και τα επιθυμητά τελικά περιεχόμενα. Το ζητούμενο είναι να βρεθεί η μικρότερη αλληλουχία από τις εντολές `move` και `swap` που πρέπει να εκτελεσθούν για να επιτευχθεί ο ζητούμενος μετασχηματισμός. Συγκεκριμένα, ορίστε το κατηγόρημα `codegen/3`, έτσι ώστε όταν αυτό καλείται με πρώτο όρισμα τη λίστα των αρχικών περιεχομένων των καταχωρητών και με δεύτερο όρισμα τη λίστα των τελικών περιεχομένων, να επιστρέφει στο τρίτο όρισμα τη λίστα των απαραίτητων (ελάχιστων) εντολών που απαιτούνται για το μετασχηματισμό. Σημειώστε ότι είναι δυνατόν στην αναπαράσταση των περιεχομένων των καταχωρητών να έχουμε, τόσο στην αρχική όσο και στην τελική κατάσταση, το σύμβολο *, που σημαίνει, για μεν την αρχική κατάσταση "δεν ξέρω τι περιέχεται στον καταχωρητή", για δε την τελική κατάσταση "δεν με ενδιαφέρει τι περιέχεται στον καταχωρητή". Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- codegen([a,b,c,d],[a,d,a,b],L).
L = [move(2),move(1),swap(3,4),swap(2,3)]

?- codegen([a,*,c],[c,a,*],L).
L = [move(1),move(3)]

?- codegen([a,b,c],[a,a,*],L).
L = [move(1)]
```

1.3 Άσκηση 3 (0.4 μονάδες)

Ένα σταυρόλεξο τετραγώνου σχήματος μπορεί να ορισθεί από τη διάστασή του και τις συντεταγμένες των μαύρων θέσεων. Για παράδειγμα, το σταυρόλεξο



μπορεί να περιγραφεί από τα γεγονότα Prolog:

```
dimension(5).

black(1,3).
black(2,3).
black(3,2).
black(4,3).
black(5,1).
black(5,5).
```

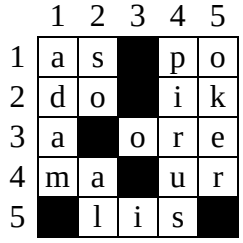
Δεδομένου και ενός γεγονότος με κατηγόρημα `words/1` μέσω του οποίου δίνονται οι λέξεις που πρέπει να τοποθετηθούν στο σταυρόλεξο, ορίστε ένα κατηγόρημα `crossword/1` που να επιστρέφει τη λίστα των διαθεσίμων λέξεων με τη σειρά που αυτές μπορούν να τοποθετηθούν στο σταυρόλεξο, πρώτα οι οριζόντιες και μετά οι κάθετες. Για παράδειγμα, αν δίνεται και το

```
words([do,ore,ma,lis,ur,as,po,so,pirus,oker,al,adam,ik]).
```

η σωστή απάντηση στο προηγούμενο σταυρόλεξο, δηλαδή αυτή που πρέπει να επιστρέψει το `crossword/1`, είναι η

```
[as,po,do,ik,ore,ma,ur,lis,adam,so,al,pirus,oker]
```

που αντιπροσωπεύει την εξής λύση:



Αν τυχόν υπάρχουν περισσότερες της μίας λύσεις, πρέπει το `crossword/1` να τις υπολογίζει όλες μέσω οπισθοδρόμησης. Φυσικά, αν δεν υπάρχει λύση, πρέπει να αποτυγχάνει. Θεωρείται γνωστό, επίσης, ότι στα σταυρόλεξα δεν λογίζονται σαν λέξεις αυτές του ενός γράμματος. Δοκιμάστε το πρόγραμμά σας και σε μεγαλύτερα σταυρόλεξα, όπως:

```
dimension(20).

black(1,1).   black(1,3).   black(1,5).   black(1,7).
black(1,9).   black(1,16).  black(1,20).  black(2,10).
black(2,12).  black(2,14).  black(3,1).   black(3,3).
black(3,5).   black(3,7).   black(3,9).   black(3,19).
black(4,10).  black(4,12).  black(5,1).   black(5,3).
black(5,5).   black(5,16).  black(6,6).   black(6,8).
black(6,10).  black(6,11).  black(6,12).  black(6,18).
black(7,1).   black(7,3).   black(7,5).   black(7,12).
black(7,14).  black(7,15).  black(7,16).  black(7,17).
black(7,18).  black(7,20).  black(8,4).   black(8,9).
black(8,11).  black(8,20).  black(9,2).   black(9,7).
black(9,13).  black(9,15).  black(9,19).  black(10,2).
black(10,4).  black(10,6).  black(10,10). black(10,11).
black(10,19). black(11,8).  black(11,12). black(11,17).
black(12,2).  black(12,4).  black(12,6).  black(12,14).
black(12,16). black(13,6).  black(13,8).  black(13,14).
black(14,2).  black(14,4).  black(14,9).  black(14,15).
black(14,16). black(14,19). black(15,5).  black(15,6).
black(15,13). black(15,15). black(15,20). black(16,1).
black(16,5).  black(16,12). black(16,17). black(16,18).
black(16,20). black(17,6).  black(17,12). black(17,20).
black(18,2).  black(18,3).  black(18,4).  black(18,6).
black(18,7).  black(18,8).  black(18,11). black(18,17).
black(19,8).  black(19,14). black(19,15). black(20,2).
black(20,4).  black(20,6).  black(20,16).

words([ados,aere,aleser,alliee,apis,apivor,aria,arno,
       attire,attristee,ave,aviser,blocage,blonde,bol,ca,
       casse,caverne,client,colibri,cou,cultiver,dada,
       deite,demi,doter,eau,eclos,ecrase,ecrin,ecuri,
       eden,egri,elevation,emier,envol,eole,eolie,epte,
       ese,esope,et,etendard,etier,etoilee,etroitesse,
       europeenne,evier,evoe,feu,fraisier,gambader,gre,
       hesperides,id,ille,ils,in,ios,ir,isar,isole,lace,
       lave,le,les,levrette,lie,lieue,lippe,lo,mer,mulet,
       nato,nep,nie,noel,norme,notaire,ohe,ointe,ole,olt,
       orienter,oui,peler,pitre,puissante,rale,ratier,
       rose,savonner,soir,tavele,tirer,tresorier,trou,
       tulle,usa,usurper,utilise,va,valse,vilipe,voleter]).
```

1.4 Άσκηση 4 (0.5 μονάδες)

Μία εκδοχή του λεγόμενου job-shop προβλήματος είναι η εξής: Έχουμε να εκτελέσουμε μία σειρά από έργα `j1, j2, j3 …`, ανεξάρτητα το ένα από το άλλο, καθένα από τα οποία αποτελείται από έναν αριθμό από εργασίες `t11, t12 …, t21, t22, …, t31, t32, …` οι οποίες πρέπει να εκτελεσθούν με τη δεδομένη σειρά για κάθε έργο. Κάθε εργασία μπορεί να εκτελεσθεί σε συγκεκριμένο τύπο μηχανής και χρειάζεται δεδομένο χρόνο για να έλθει σε πέρας. Επίσης είναι γνωστό το πλήθος των μηχανών που έχουμε διαθέσιμες από κάθε τύπο. Αυτό που θέλουμε να βρούμε είναι πότε και σε ποιες μηχανές πρέπει να εκτελέσουμε τις εργασίες έτσι ώστε όλα τα έργα να

έχουν τελειώσει πριν από δεδομένη προθεσμία. Ένα συγκεκριμένο στιγμιότυπο αυτού του προβλήματος δίνεται με τα παρακάτω γεγονότα Prolog:

```
job(j1,[t11,t12]).

job(j2,[t21,t22,t23]).

job(j3,[t31]).

job(j4,[t41,t42]).


task(t11,m1,2).

task(t12,m2,6).

task(t21,m2,5).

task(t22,m1,3).

task(t23,m2,3).

task(t31,m2,4).

task(t41,m1,5).

task(t42,m2,2).


machine(m1,1).

machine(m2,2).


deadline(14).
```

Τα γεγονότα `job/3` περιγράφουν τα έργα που πρέπει να εκτελέσουμε και τις εργασίες από τις οποίες αποτελούνται, κάθε γεγονός `task/3` δίνει, για μία εργασία, σε τι τύπου μηχανή πρέπει να εκτελεσθεί και πόσες μονάδες χρόνου απαιτούνται για την εκτέλεσή της, τα γεγονότα `machine/2` μας πληροφορούν πόσες μηχανές έχουμε διαθέσιμες από κάθε τύπο και τέλος το κατηγόρημα `deadline/1` δίνει την προθεσμία (σε μονάδες χρόνου) μέσα στην οποία πρέπει να έχουν εκτελεσθεί όλα τα έργα. Ορίστε σε Prolog το κατηγόρημα `job_shop/1`, το οποίο να επιστρέφει το πρόγραμμα εκτέλεσης των εργασιών στις μηχανές. Παράδειγμα:

```
?- job_shop(S).

S = [execs(m1,[t(t11,0,2),t(t41,2,7),t(t22,7,10)]),

     execs(m2,[t(t12,2,8),t(t42,8,10),t(t23,10,13)]),

     execs(m2,[t(t21,0,5),t(t31,5,9)])]                  -> ;
```

```
S = ..................................................

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

Η παραπάνω απάντηση σημαίνει ότι στη μηχανή τύπου `m1` θα εκτελεσθούν από τη χρονική στιγμή 0 έως τη 2 η εργασία `t11`, από τη χρονική στιγμή 2 έως την 7 η εργασία `t41` και από τη χρονική στιγμή 7 έως τη 10 η εργασία `t22`. Μετά ακολουθούν τα προγράμματα εργασίας για τις δύο μηχανές τύπου `m2`.

Μία δεύτερη εκδοχή του job-shop προβλήματος είναι αυτή στην οποία για κάθε εργασία έχουμε και ένα δεδομένο πλήθος ατόμων που πρέπει να εργασθούν στη μηχανή που θα τη φέρει σε πέρας. Δηλαδή, τώρα έχουμε το κατηγόρημα `task/4` (αντί για `task/3`), όπου το τελευταίο επιπλέον όρισμα δίνει και το πλήθος αυτό των ατόμων. Τα γεγονότα τώρα είναι:

```
task(t11,m1,2,3).

task(t12,m2,6,2).

task(t21,m2,5,2).

task(t22,m1,3,3).

task(t23,m2,3,2).

task(t31,m2,4,2).

task(t41,m1,5,4).

task(t42,m2,2,1).
```

Επίσης έχουμε δεδομένο και το προσωπικό που υπάρχει διαθέσιμο ανά πάσα χρονική στιγμή, το οποίο ορίζεται με το γεγονός `staff/1`, δηλαδή:

```
staff(6).
```

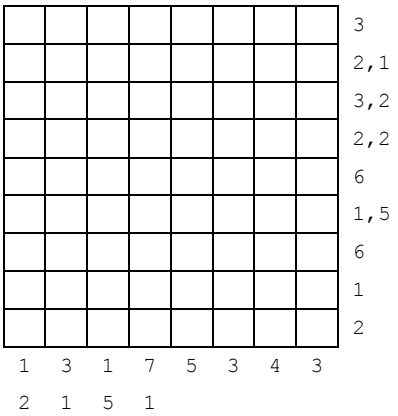
Υποτίθεται ότι κάθε εργαζόμενος μπορεί να δουλέψει σε οποιαδήποτε μηχανή για οποιοδήποτε εργασία.

Ορίστε σε Prolog το κατηγόρημα `job_shop_with_manpower/1`, το οποίο να λύνει και αυτήν την εκδοχή του `job-shop` προβλήματος. Παράδειγμα εκτέλεσης:

```
?- job_shop_with_manpower(S).

S = [execs(m1,[t(t11,0,2),t(t41,2,7),t(t22,7,10)]),

     execs(m2,[t(t42,7,9),t(t31,10,14)]),

     execs(m2,[t(t21,0,5),t(t12,5,11),t(t23,11,14)])]   -> ;


S = ..................................................

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

1.5 Άσκηση 5 (0.6 μονάδες)

Θεωρήστε ένα γρίφο στον οποίο δίνεται ένα ορθογώνιο πλέγμα και το ζητούμενο είναι να προσδιοριστεί ποια στοιχεία του πλέγματος είναι ON και ποια OFF, έχοντας σαν δεδομένα, για κάθε γραμμή και για κάθε στήλη, τα μήκη των συνεχόμενων ομάδων από στοιχεία που είναι ON. Για παράδειγμα, ακολουθεί ένα τέτοιο πλέγμα στο οποίο ζητείται στην πρώτη γραμμή να υπάρχει μία μόνο ομάδα από 3 συνεχόμενα ON στοιχεία, στη δεύτερη γραμμή, μία ομάδα από 2 συνεχόμενα ON στοιχεία και μετά, αφού παρεμβληθεί τουλάχιστον 1 OFF στοιχείο, να υπάρχει 1 ON στοιχείο μόνο του, κ.ο.κ. Επίσης στην πρώτη στήλη πρέπει να υπάρχει 1 απομονωμένο ON στοιχείο, ακολουθούμενο, αφού παρεμβληθεί τουλάχιστον ένα OFF στοιχείο, από μία ομάδα από 2 συνεχόμενα ON στοιχεία, κ.ο.κ.



Να ορισθεί ένα κατηγόρημα `nn/3` το οποίο, όταν καλείται σαν `nn(Rows, Columns, Grid)`, δίνοντας σαν Rows και Columns τις λίστες με τα μήκη των συνεχόμενων ομάδων από ON στοιχεία στις γραμμές και στις στήλες του πλέγματος, αντίστοιχα, να επιστρέφει στο Grid τον πίνακα του πλέγματος, σαν λίστα από λίστες με στοιχεία `1` και `0`. Είναι επιθυμητό, το κατηγόρημα `nn/3` να κάνει και μία ευπαρουσίαστη εκτύπωση του πλέγματος. Για παράδειγμα:

```
?- nn([[3],[2,1],[3,2],[2,2],[6],[1,5],[6],[1],[2]],
      [[1,2],[3,1],[1,5],[7,1],[5],[3],[4],[3]],
      Grid).
_###____
##_#____
_###__##
__##__##
__######
#_#####_
######__
____#___
___##___

Grid = [[0,1,1,1,0,0,0,0],[1,1,0,1,0,0,0,0],[0,1,1,1,0,0,1,1],
        [0,0,1,1,0,0,1,1],[0,0,1,1,1,1,1,1],[1,0,1,1,1,1,1,0],
        [1,1,1,1,1,1,0,0],[0,0,0,0,1,0,0,0],[0,0,0,1,1,0,0,0]]
```

Τι αποτέλεσμα δίνει το πρόγραμμά σας για την παρακάτω ερώτηση;

```
?- nn([[2],[1,1],[1,1],[2],[2,1],[1,2,2],[4,1],[3]],
      [[2],[1,1],[2],[2,4],[1,1,2],[1,1,1,1],[2,2],[0]],
      Grid).
```

Για την ερώτηση;

```
?- nn([[6],[9],[12],[14],[4,14],
       [6,12],[7,3,2],[8,2],[8,11],[7,4,1,3],
       [7,4,1,2],[10,5,1,2],[9,7,1,2],[15,1,2],[12,1,2],
```

```
      [11,1,2],[14,2],[13,3],[22],[21],
      [20],[19],[2,17],[3,2,12],[3,2,7],
      [5,2],[4,1],[4,2],[7],[9]],
     [[3],[5],[1,6],[1,7],[1,8],
      [2,8],[2,7],[2,6],[1,3],[1,3],
      [2,2],[2,3],[2,3,1],[2,3,2],[1,3,1],
      [1,4,2],[1,3,1],[2,4,2],[2,6,1],[2,8,2],
      [2,10,3],[2,16],[1,14,2],[2,14,1],[2,15,2],
      [2,16,2],[2,2,9,1],[2,1,9,1],[3,1,9,3],[6,14],
      [23,2],[3,1,7,1],[2,1,7,1],[2,1,6,1],[2,2,6,1],
      [2,2,6,1],[2,3,5,1],[1,4,6],[2,10],[2,7]],
Grid).
```

1.6 Άσκηση 6[1] (0.5 μονάδες)

Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο `N` πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς `1, 2, 3, …, N`. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει `M` συνδυασμούς πτήσεων P_i ($1 \le i \le M$). Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις `1, 2, 3, …, N` και, επίσης, τα P_i δεν είναι κατ' ανάγκη ξένα μεταξύ τους. Οι συνδυασμοί αυτοί είναι έτσι κατασκευασμένοι ώστε να είναι δυνατόν λόγω κανονισμών, συμβάσεων κλπ. να πραγματοποιηθούν ο καθένας, δηλαδή όλες οι πτήσεις που περιλαμβάνει, με έναν από τους διαθέσιμους κυβερνήτες της εταιρείας. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί που καλύπτουν ακριβώς τις πτήσεις της εταιρείας, με σκοπό να ανατεθούν σε συγκεκριμένους κυβερνήτες. Δηλαδή, δεν πρέπει ούτε να μείνει πτήση χωρίς κυβερνήτη, ούτε να κάποια πτήση να έχει δύο ή περισσότερους κυβερνήτες. Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος (έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.) στην εταιρεία `Ci`, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος. Εφαρμόστε τη μέθοδό σας για εισόδους που προκύπτουν από το `get_flight_data(I,N,P,C)` του προγράμματος `~takis/lp/flight_data.pl` (από το δίκτυο των Suns του τμήματος) δίνοντας στο I έναν αύξοντα αριθμό εισόδου `(1, 2, …)` και παίρνοντας στα `N, P` και `C` το πλήθος των πτήσεων, μία λίστα συνδυασμών πτήσεων και τη λίστα κοστών των συνδυασμών αυτών, αντίστοιχα.

[1] Να αντιμετωπισθεί με τη βοήθεια της τεχνολογίας του λογικού προγραμματισμού με περιορισμούς, π.χ. στο περιβάλλον της γλώσσας ECLiPSe (με χρήση είτε της παλιότερης βιβλιοθήκης fd είτε της νεώτερης ic).

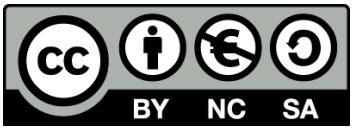
Σημειώματα

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.