ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Α' Ομάδα Ασκήσεων "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2020-21

Άσκηση 1[1]

Υλοποιήστε σε Prolog ένα κατηγόρημα `diags(Matrix,DiagsDown,DiagsUp)`, το οποίο να παίρνει σαν όρισμα ένα πίνακα `Matrix`, στη μορφή μίας λίστας λιστών (οι εσωτερικές λίστες είναι οι γραμμές του πίνακα), και να επιστρέφει στα `DiagsDown` και `DiagsUp` τις λίστες των στοιχείων των κατιουσών και των ανιουσών διαγωνίων του πίνακα, αντίστοιχα, με σειρά στα στοιχεία τους από επάνω προς τα κάτω.. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- diags([[a,b,c,d],[e,f,g,h],[i,j,k,l]],DiagsDown,DiagsUp).
DiagsDown = [[i],[e,j],[a,f,k],[b,g,l],[c,h],[d]]
DiagsUp = [[a],[b,e],[c,f,i],[d,g,j],[h,k],[l]]
```

Παραδοτέο για την άσκηση είναι **<u>ένα</u> πηγαίο αρχείο Prolog** με όνομα **diags.pl**.

Άσκηση 2

Τα νευρωνικά δίκτυα Hopfield λειτουργούν ως αυτο-συσχετιστικές μνήμες, υπό την έννοια ότι σε ένα τέτοιο δίκτυο με N αλληλοτροφοδοτούμενους κόμβους-νευρώνες, οι οποίοι μπορεί να βρίσκονται σε μία από δύο καταστάσεις ο καθένας, τις $+1$ ή -1, μπορούμε να αποθηκεύσουμε ένα πλήθος από N-διάστατα διανύσματα με δυαδικές τιμές ($+1$ ή -1) στα στοιχεία τους. Στη συνέχεια, όταν το δίκτυο τροφοδοτηθεί με ένα N-διάστατο διάνυσμα, μετά από κάποιες μεταπτώσεις μεταξύ διαδοχικών καταστάσεων, θα ισορροπήσει σε μία κατάσταση που αντιστοιχεί σε κάποιο από τα αποθηκευμένα διανύσματα και, μάλιστα, σε εκείνο που είναι πλησιέστερο στο διάνυσμα που δόθηκε στο δίκτυο.

Μία βασική διαδικασία σε ένα δίκτυο Hopfield είναι η εκπαίδευσή του, δηλαδή η αποθήκευση σε αυτό κάποιου αριθμού N-διάστατων διανυσμάτων. Η αποθήκευση συνίσταται στον προσδιορισμό του δισδιάστατου $N \times N$ πίνακα των βαρών του δικτύου. Κάθε βάρος του πίνακα αντιστοιχεί σε μία σύνδεση μεταξύ δύο κόμβων του δικτύου και ποσοτικοποιεί την ισχύ της σύνδεσης αυτής. Αν θέλουμε να αποθηκεύσουμε σε ένα δίκτυο Hopfield M σε πλήθος N-διάστατα διανύσματα, τότε ο πίνακας W των βαρών του υπολογίζεται από τη σχέση

$$W = \sum_{i=1}^{M} (Y_i^T \cdot Y_i) - M \cdot I_N$$

[1] **Bonus 100%**, αν δεν χρησιμοποιηθούν ενσωματωμένα κατηγορήματα αριθμητικής.

όπου Y_i είναι ένα από τα $1 \times N$ διανύσματα-γραμμές προς αποθήκευση, Y_i^T είναι το $N \times 1$ διάνυσμα-στήλη, που είναι το ανάστροφο του Y_i, και I_N είναι ο δισδιάστατος $N \times N$ ταυτοτικός πίνακας, που έχει, δηλαδή, όλα τα στοιχεία του ίσα με 0, εκτός από αυτά της κύριας διαγωνίου του, που είναι ίσα με 1.

Για παράδειγμα, έστω ότι θέλουμε να αποθηκεύσουμε σε ένα δίκτυο Hopfield τεσσάρων νευρώνων ($N = 4$) τα τρία ($M = 3$) διανύσματα $(+1, -1, -1, +1)$, $(-1, -1, +1, -1)$ και $(+1, +1, +1, +1)$. Τότε:

$$W = \begin{pmatrix} +1 \\ -1 \\ -1 \\ +1 \end{pmatrix} \cdot (+1 \quad -1 \quad -1 \quad +1) + \begin{pmatrix} -1 \\ -1 \\ +1 \\ -1 \end{pmatrix} \cdot (-1 \quad -1 \quad +1 \quad -1) + \begin{pmatrix} +1 \\ +1 \\ +1 \\ +1 \end{pmatrix} \cdot (+1 \quad +1 \quad +1 \quad +1) - 3 \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Δηλαδή:

$$W = \begin{pmatrix} +1 & -1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ -1 & +1 & +1 & -1 \\ +1 & -1 & -1 & +1 \end{pmatrix} + \begin{pmatrix} +1 & +1 & -1 & +1 \\ +1 & +1 & -1 & +1 \\ -1 & -1 & +1 & -1 \\ +1 & +1 & -1 & +1 \end{pmatrix} + \begin{pmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \end{pmatrix} - \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

Και τελικά:

$$W = \begin{pmatrix} 0 & 1 & -1 & 3 \\ 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & -1 \\ 3 & 1 & -1 & 0 \end{pmatrix}$$

Ορίστε ένα κατηγόρημα `hopfield/2`, το οποίο, όταν καλείται με πρώτο όρισμα μία λίστα διανυσμάτων προς αποθήκευση σε ένα δίκτυο Hopfield, όπου κάθε διάνυσμα είναι και αυτό μία λίστα από τα στοιχεία του, να επιστρέφει στο δεύτερο όρισμα τον πίνακα των βαρών του δικτύου, σαν μία λίστα από τις γραμμές του, όπου κάθε μία είναι μία λίστα των στοιχείων της. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- hopfield([[+1,+1,-1],
             [-1,-1,-1]],W).
W = [[0,2,0],[2,0,0],[0,0,0]]

?- hopfield([[+1,-1,-1,+1],
              [-1,-1,+1,-1],
              [+1,+1,+1,+1]],W).
W = [[0,1,-1,3],[1,0,1,1],[-1,1,0,-1],[3,1,-1,0]]

?- hopfield([[+1,+1,+1,+1,-1,+1,+1,+1,+1],
              [-1,+1,-1,-1,+1,-1,-1,+1,-1],
              [+1,-1,+1,+1,+1,+1,-1,-1,+1]],W).
W = [[0,-1,3,3,-1,3,1,-1,3],[-1,0,-1,-1,-1,-1,1,3,-1],
     [3,-1,0,3,-1,3,1,-1,3],[3,-1,3,0,-1,3,1,-1,3],
     [-1,-1,-1,-1,0,-1,-3,-1,-1],[3,-1,3,3,-1,0,1,-1,3],
     [1,1,1,1,-3,1,0,1,1],[-1,3,-1,-1,-1,-1,1,0,-1],
     [3,-1,3,3,-1,3,1,-1,0]]
```

Παραδοτέο για την άσκηση είναι **<u>ένα</u> πηγαίο αρχείο Prolog** με όνομα **hopfield.pl**.

Άσκηση 3

Θεωρήστε ότι σε ένα χώρο διασκέδασης υπάρχουν συγκεκριμένα ηλεκτρονικά παιγνίδια, στα οποία μπορείτε να παίξετε με κάποια δεδομένη σειρά. Αρχίζετε από το πρώτο, συνεχίζετε στο δεύτερο, και ούτω καθεξής, μέχρι το τελευταίο. Κάθε παιγνίδι μπορείτε να το παίξετε περισσότερες από μία φορά, όλες όμως θα είναι συνεχόμενες. Για να παίξετε μία φορά ένα παγνίδι, πρέπει να πληρώσετε μία μάρκα. Οι διαθέσιμες μάρκες σας βρίσκονται σε ένα κουτί, που έχει χωρητικότητα T μάρκες, το οποίο αρχικά είναι γεμάτο. Αφού ολοκληρώσετε όλες τις φορές παιξίματος ενός παιγνιδιού, και πριν ξεκινήσετε το επόμενο, σας δίνονται δώρο K μάρκες, ή λιγότερες, και πάντως, όχι περισσότερες από όσες μπορούν να χωρέσουν στο κουτί σας. Κάθε φορά που παίζετε το παιγνίδι i, η ευχαρίστησή σας είναι P_i. Η ευχαρίστηση ενός παιγνιδιού μπορεί να είναι και αρνητική, που σημαίνει ότι το παιγνίδι δεν σας άρεσε, ή μηδενική, που σημαίνει ότι το παιγνίδι σας είναι αδιάφορο. Το ερώτημα είναι, δεδομένων των T, K και P_i, πόσες φορές πρέπει να παίξετε το κάθε παιγνίδι, ώστε στο τέλος να έχετε τη μέγιστη συνολική ευχαρίστηση. Κάθε παιγνίδι πρέπει να το παίξετε τουλάχιστον μία φορά, ώστε να μπορείτε να διαπιστώσετε πόσο σας αρέσει.

Επιλύστε το παραπάνω πρόβλημα, ορίζοντας ένα κατηγόρημα `games/5`, το οποίο, όταν καλείται σαν `games(Ps,T,K,Gs,P)`, με `Ps` μία λίστα από τις ευχαριστήσεις των παιγνιδιών (μήκους όσο το πλήθος των παιγνιδιών), `T` τη χωρητικότητα του κουτιού με τις μάρκες και `K` το μέγιστο πλήθος κέρδους μαρκών μετά από κάθε παιγνίδι, να επιστρέφει στο `Gs` τη λίστα από τις φορές που πρέπει να παιχθεί το κάθε παιγνίδι, ώστε να έχετε τη μέγιστη ευχαρίστηση, και στο `P` αυτή τη μέγιστη ευχαρίστηση. Αν υπάρχουν περισσότερες από μία βέλτιστη λύση με την ίδια μέγιστη ευχαρίστηση, το κατηγόρημα να τις επιστρέφει όλες μέσω οπισθοδρόμησης. Κάποιες ενδεικτικές εκτελέσεις είναι οι εξής:

```
?- games([4,1,2,3],5,2,Gs,P).
Gs = [5,1,1,4]
P = 35        --> ;
no

?- games([4,-1,-2,3],5,2,Gs,P).
Gs = [5,1,1,4]
P = 29        --> ;
no

?- games([4,1,-2,3,4],3,2,Gs,P).
Gs = [3,2,1,2,3]
P = 30        --> ;
no

?- games([3,-2,4,-5,2,0,4,-1,3,4],5,2,Gs,P).
Gs = [3,1,5,1,1,1,5,1,1,4]
P = 62        --> ;
Gs = [3,1,5,1,1,1,4,1,1,5]
P = 62        --> ;
no

?- games([2,-3,4,1,-2,2,1],8,3,Gs,P).
Gs = [5,1,8,1,1,7,3]
```

```
P = 55        --> ;
no

?- games([1,2,3,4,5,0,-1,-2,-3,-4,-5],5,4,Gs,P).
Gs = [4,4,4,4,5,4,1,1,1,1,1]
P = 50        --> ;
Gs = [4,4,4,4,5,3,1,1,1,1,1]
P = 50        --> ;
Gs = [4,4,4,4,5,2,1,1,1,1,1]
P = 50        --> ;
Gs = [4,4,4,4,5,1,1,1,1,1,1]
P = 50        --> ;
no

?- games([2,4,1,-3,3,-4,2,3,5,4,5],6,2,Gs,P).
Gs = [2,6,1,1,3,1,1,1,6,1,3]
P = 85        --> ;
Gs = [2,6,1,1,3,1,1,1,5,1,4]
P = 85        --> ;
Gs = [2,6,1,1,3,1,1,1,4,1,5]
P = 85        --> ;
Gs = [2,6,1,1,3,1,1,1,3,1,6]
P = 85        --> ;
Gs = [2,6,1,1,2,1,1,2,6,1,3]
P = 85        --> ;
Gs = [2,6,1,1,2,1,1,2,5,1,4]
P = 85        --> ;
Gs = [2,6,1,1,2,1,1,2,4,1,5]
P = 85        --> ;
Gs = [2,6,1,1,2,1,1,2,3,1,6]
P = 85        --> ;
no

?- games([0],3,1,Gs,P).
Gs = [3]
P = 0        --> ;
Gs = [2]
P = 0        --> ;
Gs = [1]
P = 0        --> ;
no

?- games([-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
          -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
          -1],1,1,Gs,P).
Gs = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
      1,1,1,1,1,1,1,1]
P = -35

?- findall(Gs,games([1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,1,2,3,
                     1,2,3],5,1,Gs,_),L),length(L,N).
. . . . . .
N = 210
```

Παραδοτέο για την άσκηση είναι **<u>ένα</u> πηγαίο αρχείο Prolog** με όνομα **games.pl**.