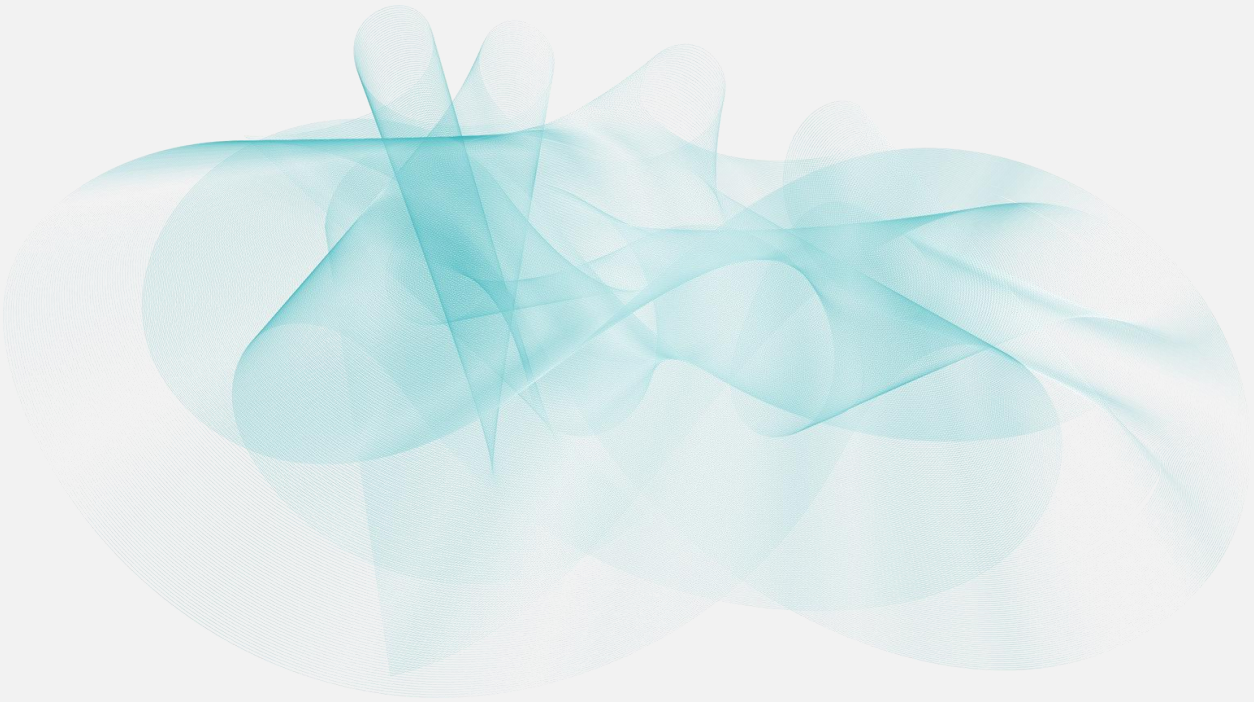
ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ

Αναφορά 1ης άσκησης



Βιριράκης Γεώργιος (2016030035)
Γαλάνης Μιχάλης (2016030036)

ΕΙΣΑΓΩΓΗ

Στη πρώτη άσκηση κληθήκαμε να ασχοληθούμε με τη **Bash Shell** (γλώσσα εντολών κέλυφους του linux) κατασκευάζοντας δύο προγράμματα με σκοπό την εξοικείωσή μας με αυτή. Αυτά περιείχαν ανάγνωση αρχείων δίσκου, υπολογισμό αριθμητικών πράξεων, εκτύπωση αποτελεσμάτων και άλλες βασικές εντολές.

Η εργασία έχει υλοποιηθεί πλήρως από κοινού.

ΘΕΜΑ 1 - ΠΡΟΓΡΑΜΜΑ REGR

Επισκόπηση

Το πρώτο πρόγραμμα αποσκοπούσε στον υπολογισμό των παραμέτρων γραμμικής παλινδρόμησης $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ που ελαχιστοποιούν το συνολικό τετραγωνικό σφάλμα $\boldsymbol{err}$ της προσέγγισης $\boldsymbol{cY} = \boldsymbol{aX} + \boldsymbol{b}$. Τα $\boldsymbol{X}, \boldsymbol{Y}$ είναι διανύσματα των τιμών πρώτης και δεύτερης στήλης αντίστοιχα σε αρχεία εισόδου που δημιουργήσαμε με μορφή num1:num2 δεκαδικούς αριθμούς. Παρακάτω παραθέτουμε τις απαραίτητες σχέσεις που πρέπει να υπολογιστούν για ΜΗ σταθερά διανύσματα $\boldsymbol{X}$:

- $sum\_x = \sum_{i=0}^{length-1} X[i]$
- $sum\_y = \sum_{i=0}^{length-1} Y[i]$
- $sum\_xy = \sum_{i=0}^{length-1} X[i]Y[i]$
- $sum\_x2 = \sum_{i=0}^{length-1} (X[i])^2$
- $a = \frac{length*sum\_xy - sum\_x*sum\_y}{length*sum\_x2 - sum\_x*sum\_x}$
- $b = \frac{sum\_y - a*sum\_y}{length}$
- $err = \sum_{i=0}^{length-1} (Y[i] - (aX[i] + b))^2$

Υλοποίηση

Αρχικά δημιουργήσαμε 3 αρχεία εισόδου με τυχαίους δεκαδικούς αριθμούς όπως φαίνεται στον παρακάτω πίνακα:

input_1	input_2	input_3
2.7 : 5.6 24.942 : 59.53 19.09 : 43.21 52.88 : 95.67 2.912 : 18.241	42.23 : 42.04 65.02 : 8.124 91.213 : 418.1 12.412 : 23.55 23.066 : 12.3 18.199 : 14.123	358.12 : 33.04 358.12 : 8.566 358.12 : 95.11 358.12 : 74.63

Για την ανάγνωση των αρχείων εισόδου, χρησιμοποιούμε δομή επανάληψης for, η οποία διατρέχει τα ορίσματα του προγράμματος που ισοδυναμούν με τα αρχεία εισόδου.

```
#!/bin/bash
for j
do
    #Internal Code
done
```

Εσωτερικά, για κάθε αρχείο διαβάζουμε γραμμή – γραμμή το περιεχόμενό του και κατανέμουμε κατάλληλα τους αριθμούς στα αρχικοποιημένα διανύσματα $\boldsymbol{X}, \boldsymbol{Y}$. Σημειώνεται ότι χρησιμοποιούμε τεχνικές διάσπασης string.

```
    #Reading Input File & Filling  up  X, Y
    X=(); Y=()
    exec < "$j"
    while read line
    do
        X+=(${line%%:*})
        Y+=(${line##*:})
    done
    #-------------------------------------------------------------------
```

Σε περίπτωση σταθερού διανύσματος $\boldsymbol{X}$, δεν ισχύουν οι παραπάνω σχέσεις. Πιο συγκεκριμένα, όλα τα σημεία θα έχουν συντεταγμένες μορφής (X_0, Y_i) με $i \in (0, n)$ οπότε βρίσκονται σε μια κατακόρυφη ευθεία η οποία περιγράφεται ως $x = x_0$. Στην εξίσωση $cY = ax + b$ θέτουμε $c = 0$ για να μηδενίσουμε το Y, οπότε έχουμε $x = x_0 = -b/a$. Για $a = 1$, έχουμε $b = -X[0]$, οπότε τελικά: $\boldsymbol{a} = \mathbf{1}$, $\boldsymbol{b} = -\boldsymbol{X}[\mathbf{0}]$, $\boldsymbol{c} = \mathbf{0}$. Και προφανώς $\boldsymbol{err} = \mathbf{0}$ αφού η κατακόρυφη ευθεία περνάει από όλα τα σημεία. Θέτουμε λοιπόν μια Boolean μεταβλητή true και διατρέχουμε αρχικά το διάνυσμα $\boldsymbol{X}$. Αν διαπιστωθεί ότι τουλάχιστον ένα στοιχείο του X δεν ισούται με τη πρώτη τιμή, η Boolean γίνεται false και ο βρόγχος επανάληψης τερματίζεται. Αργότερα, ελέγχοντας τη μεταβλητή αυτή, εκτυπώνουμε τα κατάλληλα αποτελέσματα.

```
    #In case of constant vector X
    x_const=true
    let "length = ${#X[@]}";
    for ((i = 0; i < length; i++))
    do
        if [ ${X[i]} != ${X[0]} ]; then
            x_const=false
            break
        fi
    done
    if ${x_const}; then
        echo "FILE: $j, a=1 b=-${X[0]} c=0 err=0"
        continue
    fi
    #-------------------------------------------------------------------
```

Σε περίπτωση μη σταθερού διανύσματος $\boldsymbol{X}$, υπολογίζουμε τις απαραίτητες παραμέτρους σύμφωνα με τις σχέσεις που μας δίνονται (δε παραθέτουμε τμήμα κώδικα) και εκτυπώνουμε τα ζητούμενα αποτελέσματα.

Αποτελέσματα & Συμπεράσματα

Βάσει των input αρχείων που δόθηκαν, παράγουμε τα παρακάτω αποτελέσματα:

```
mixalis@mgalanis-PC:~/Desktop/LAB21142508/regr$ ./regr input_1 input_2 input_3
FILE: input_1, a=1.69 b=9.79 c=1 err=160.06
FILE: input_2, a=4.10 b=-85.92 c=1 err=52813.75
FILE: input_3, a=1 b=-358.12 c=0 err=0
```

Παρατηρούμε ότι στο input_3 αρχείο, λόγω σταθερού διανύσματος $\boldsymbol{X}$, έχουμε τις προβλεπόμενες τιμές.

ΘΕΜΑ 2 - ΠΡΟΓΡΑΜΜΑ RESULTS

Επισκόπηση

Στο δεύτερο θέμα, μας ζητήθηκε να κατασκευάσουμε μια βαθμολογία αθλητικού πρωταθλήματος όπου οι ομάδες είναι ταξινομημένες με βάση τους βαθμούς που συγκέντρωσαν. Οι βαθμοί αυτοί «χτίζονται» από δεδομένα αγώνων που εισέρχονται από εξωτερικό αρχείο. Ακόμη, η έξοδος έχει συγκεκριμένη μορφή και στοίχιση που θα αναλυθεί παρακάτω.

Υλοποίηση

Αρχικά δημιουργήσαμε αρχείο εισόδου του οποίου τα δεδομένα αντλήσαμε από το courses για την επαλήθευση των αποτελεσμάτων. Όπως και προηγουμένως, διαβάζουμε γραμμή-γραμμή τους αγώνες και τα αποτελέσματά τους, και τα διασπάμε σε προσωρινές μεταβλητές.

```
5     #Temporary Strings for string splitting ("A-B:C-D" -> 'A-B', 'C-D')
6     temp_club_strings=(${line%%:*})
7     scores_strings=(${line##*:})
8     #-------------------------------------------------------------------
9
10    #Other Temporary Strings for further splitting ("A-B" -> 'A', 'B')
11    home_club=(${temp_club_strings%%-*})
12    away_club=(${temp_club_strings##*-})
13    home_score=(${scores_strings%%-*})
14    away_score=(${scores_strings##*-})
15    #-------------------------------------------------------------------
```

Ο βασικός μηχανισμός μας περιλαμβάνει 4 βασικούς πίνακες: **clubs** (ονόματα)**, points, goals_scored, goals_against.** Αυτοί συσχετίζονται με κοινά indexes. Η κάθε ομάδα δηλαδή αναπαρίσταται από κάποιο συγκεκριμένο index. Για παράδειγμα η i-οστή ομάδα που έχει εισαχθεί έχει όνομα clubs[i], πόντους points[i], γκόλ υπέρ goals_scored[i] και γκολ κατά goals_against[i]. Φροντίζουμε στον κώδικά μας αυτή η συσχέτιση να μη χάνεται ποτέ.

Αρχικά εισάγουμε τα ονόματα των ομάδων στον πίνακα **clubs**. Για να επιτευχθεί αυτό, ελέγχουμε γενικώς εάν οι ομάδες προς εξέταση έχουν ήδη εισαχθεί. Τη πρώτη φορά (στη πρώτη γραμμή) εισάγονται οι δύο ομάδες κατευθείαν. Τις επόμενες φορές για τον έλεγχο που προαναφέρθηκε χρησιμοποιούμε τους δείκτες **home_club_index** και **away_club_index** που διατρέχουν τον πίνακα clubs για ήδη υπάρχουσες ομάδες. Εάν η τιμή των indexes ξεχωριστά είναι μηδενική (και διάφορου του πρώτου στοιχείου) σημαίνει ότι η ομάδα δεν έχει ακόμη εισαχθεί και εισάγεται. Σε κάθε περίπτωση οι τελικές τιμές των indexes δηλώνουν τη γενική θέση **i** της ομάδας στους πίνακες (πριν τη ταξινόμηση).

```
17      #Filling up array data. Each club is represented by a specific index
18      #This is how we connect arrays with each other
19      let home_club_index=0; let away_club_index=0
20      if [ ${#clubs[@]} -eq 0 ]; then #First two clubs are entered (all arrays are empty)
21          #Insert club data and estimating indexes
22          clubs+=(${home_club})
23          home_club_index=$[${#clubs[@]}-1]
24          clubs+=(${away_club})
25          away_club_index=$[${#clubs[@]}-1]
26      else #when arrays are not empty
27          #Checking new temp clubs with existing ones to avoid duplicate clubs
28          #So we estimate the club indexes and they will indicate if a club is new or not
29          let initial_length=${#clubs[@]}
30          for ((i=0; i < ${initial_length}; i++))
31          do
32              if [ "${clubs[i]}" == "${home_club}" ]; then
33                  home_club_index=$i
34              fi
35              if [ "${clubs[i]}" == "${away_club}" ]; then
36                  away_club_index=$i
37              fi
38          done
39          #Insert non existing clubs based on club indexes
40          if [ "${home_club_index}" == 0 ] && [ "${clubs[0]}" != "${home_club}" ]; then
41              clubs+=(${home_club})
42              home_club_index=$[${#clubs[@]}-1]
43          fi
44          if [ "${away_club_index}" == 0 ] && [ "${clubs[0]}" != "${away_club}" ]; then
45              clubs+=(${away_club})
46              away_club_index=$[${#clubs[@]}-1]
47          fi
48          #-----------------------------------------------------------------
49      fi
50      #-------------------------------------------------------------------
```

Ύστερα υπολογίζουμε τους κατάλληλους βαθμούς ανάλογα με τα αποτελέσματα των αγώνων και γεμίζουμε τις αντίστοιχες θέσεις του πίνακα points με τα index που αναλύθηκαν νωρίτερα. Αντίστοιχη διαδικασία γίνεται για τους πίνακες goals_scored & goals_against.

Έως τώρα, έχουν εισαχθεί όλες οι ομάδες και τα δεδομένα τους στους πίνακες. Για να τις ταξινομήσουμε βάσει των πόντων, χρησιμοποιούμε αλγόριθμο ταξινόμησης Bubble sort, καθώς είναι εύκολα υλοποιήσιμος και έχουμε μικρό αριθμό δεδομένων (δεν τον συνιστούμε για τεράστιο όγκο δεδομένων). Πιο συγκεκριμένα, στη διαδικασία του swap, προσθέσαμε και συνθήκη ελέγχου ονόματος για τη περίπτωση ισοβαθμίας. Σημειώνουμε ότι πραγματοποιούμε την ταξινόμηση ταυτόχρονα σε όλους τους πίνακες για να διατηρηθεί η συσχέτισή τους.

```
#Using Bubble Sort algorithm to sort the arrays using the points array.
#We chose this because it is the easiest one to implement, although we don't recommend it
#for very large input files.
for ((i = 0; i < ${#points[@]}; i++))
do
    for ((j = 1; j < $[${#points[@]} - $i]; j++))
    do
        #We swap on the following conditions:
        #- Either when points[j - 1] < points[j]
        #- OR when (points[j - 1] == points[j] BUT ALSO clubs[j] < clubs[j-1])
        if [ ${points[j-1]} -lt ${points[j]} ] || ([ ${points[j-1]} -eq ${points[j]} ] && [ ${clubs[j]} \< ${clubs[j-1]} ]); then
            #We swap all the arrays at the same time so their indexes remain connected
            tempClubs=${clubs[j-1]}; let tempPoints=${points[j-1]}; let tempGF=${goals_scored[j-1]}; let tempGA=${goals_against[j-1]}
            clubs[j-1]=${clubs[j]}; points[j-1]=${points[j]}; goals_scored[j-1]=${goals_scored[j]}; goals_against[j-1]=${goals_against[j]}
            clubs[j]=${tempClubs}; points[j]=${tempPoints}; goals_scored[j]=${tempGF}; goals_against[j]=${tempGA}
        fi
    done
done
```

Τέλος, για στοίχιση αξιοποιούμε την printf και τις παραμέτρους formatting που μας παρέχει, στις οποίες χρησιμοποιούμε μεταβλητό μέγεθος κενού χώρου το οποίο λαμβάνει υπόψιν το μεγαλύτερο όνομα ομάδας που υπάρχει. Αυτό υπολογίστηκε υλοποιώντας ένα βρόγχο επανάληψης.

```
#BONUS - Figuring out the exact space needed to align the club information properly
let maxLength=0
for ((i = 0; i < ${#clubs[@]}; i++))
do
    if [ $[${#clubs[i]}] -gt ${maxLength} ]; then
        maxLength=$[${#clubs[i]} + 8]
    fi
done
#-----------------------------------------------------------------

#Printing Information
for ((i = 0; i < ${#points[@]}; i++))
do
    #We use printf instead of echo so that we can take advantage of it's format properties for alignment purposes
    printf "%s\t%-$((${maxLength}))s\t%s\t%s\t%s\n" "$[$i +1 ]." "${clubs[i]}" "${points[i]}" "${goals_scored[i]}-${goals_against[i]}"
done
#-----------------------------------------------------------------
```

Αποτελέσματα & Συμπεράσματα

Παραθέτουμε ολόκληρο το πρωτάθλημα Αγγλίας για το έτος 2018-2019:

```
mixalis@mgalanis-PC:~/Desktop/LAB21142508/results$ ./results input_1
1.      ManchesterCity          98      95-23
2.      Liverpool               97      89-22
3.      Chelsea                 72      63-39
4.      Tottenham               71      67-39
5.      Arsenal                 70      73-51
6.      Manchester              66      65-54
7.      Wolvs                   57      47-46
8.      Everton                 54      54-46
9.      Leicster                52      51-48
10.     WestHam                 52      52-55
11.     Watford                 50      52-59
12.     Crystal                 49      51-53
13.     Bournemouth             45      56-70
14.     Newcastle               45      42-48
15.     Burnley                 40      45-68
16.     Southampton             39      45-65
17.     Brighton                36      35-60
18.     Cardiff                 34      34-69
19.     Fulham                  26      34-81
20.     Huddersfield            16      22-76
```

Βλέπουμε ότι έχουμε τα αναμενόμενα αποτελέσματα. Οι βαθμολογίες κατατάσσονται από την ομάδα με τους περισσότερους πόντους στην ομάδα με τους λιγότερους. Η στοίχιση είναι ακριβής και για ομάδες με ισοβαθμία προηγούνται οι ομάδες με μικρότερο αλφαβητικά όνομα.