Department of Electrical and Computer Engineering
Technical University of Crete
Software Development Tools & Systems Programming
Instructor: A.Deligiannakis
Christos Ziskas
October 19, 2019
"First Report"

ΕΡΓΑΣΙΑ ΣΕ BASH SCRIPTS

1 ΠΡΟΓΡΑΜΜΑ REGR

1.1 ΕΙΣΑΓΩΓΗ

Η υλοποίηση του προγράμματος πραγματοποιήθηκε θεωρώντας ως αρχείο εισόδου, αριθμούς διατεταγμένους σε συγκεκριμένη διάταξη (num1:num2). Τα δεδομένα εισόδου μπορεί να είναι ακέραιοι ή δεκαδικοί με τουλάχιστον δυο δεκαδικά ψηφία. Το πρόγραμμα κατασκευάζεται για τον υπολογισμό των παραμέτρων γραμμικής παλινδρόμησης **a,b,c** που ελαχιστοποιούν το συνολικό τετραγωνικό σφάλμα **err**. Οι ζητούμενοι παράμετροι παραθέτονται παρακάτω:

$$sum\_x = \sum_{i=0}^{length-1} X[i]$$

$$sum\_y = \sum_{i=0}^{length-1} Y[i]$$

$$sum\_xy = \sum_{i=0}^{length-1} X[i] \cdot Y[i]$$

$$sum\_x2 = \sum_{i=0}^{length-1} (X[i])^2$$

$$\alpha = \frac{length \times sum\_xy - sum\_x \times sum\_y}{length \times sum\_x2 - sum\_x \times sum\_x}$$

$$b = \frac{sum\_y - \alpha \times sum\_x}{length}$$

$$err = \sum_{i=0}^{length-1} (Y[i] - (aX[i] + b))^2$$

1.2 ΥΛΟΠΟΙΗΣΗ

Το πρόγραμμα είναι σχεδιασμένο για εκτέλεση πολλαπλών αρχείων. Η απουσία κάποιου αρχείου ως παράμετρος, οδηγεί το πρόγραμμα σε ανικανότητα εκτέλεσης και ως εκ τούτου, το πρόγραμμα τερματίζει.

Αρχικά, η εκτέλεση του αρχείου ξεκινάει με την ανάγνωση του αρχείου εισόδου. Σε αυτό το σημείο η ανάγνωση αποσκοπεί στην αναγνώριση του πλήθους των γραμμών που θα χρησιμοποιηθούν. Η πληροφορία είναι απαραίτητη για την εξαγωγή των αποτελεσμάτων.

Το προγράμμα συνεχίζει με την αναγνώριση των δεδομένων. Η ανάγνωση πραγματοποιείται line by line. Σε κάθε iteration λαμβάνεται ενα στοιχείο από κάθε διάνυσμα. Για να γίνει αντιληπτό ποιό στοιχείο ανήκει σε κάθε διάνυσμα, αποκόπτεται το delimiter που διαχωρίζει τις πληροφορίες. Αποθηκεύεται σε πίνακα δυο θέσεων όπου το πρώτο κελί είναι το στοιχείο του διανύσματος $\mathbf{X}$ και το άλλο κελί το στοιχείο του διανύσματος $\mathbf{Y}$. Οι μεταβλητές των διανυσμάτων λαμβάνουν τιμή σε κάθε iteration και ανανεώνονται με το διάβασμα νέας γραμμής. Η ανανέωση αφορά την διατήρηση της προηγούμενης τιμής της μεταβλητής και την προσθήκη της νέας τιμής που διαβάστηκε ή δέχθηκε επεξεργασία. Η λογική αυτή ακολουθείται και από τις υπόλοιπες μεταβλητές.

```
sum_x=$(echo "scale=2; ${sum_x} + ${my_array[0]}" | bc)
sum_y=$(echo "scale=2; ${sum_y} + ${my_array[1]}" | bc)
sum_xy=$(echo "scale=2; ${sum_xy} + ((${my_array[0]} * ${my_array[1]}))" | bc)
sum_x2=$(echo "scale=2; ${sum_x2} + ((${my_array[0]} * ${my_array[0]}))" | bc)
```

Figure 1: Ενημέρωση μεταβλητών

Κατά τη διάρκεια ανάγνωσης του αρχείου πραγματοποιείται έλεγχος για την ύπαρξη σταθερού διανύσματος X. Αποθηκεύεται το πρώτο στοιχείο του διανύσματος και πραγματοποιείται έλεγχος ομοιότητας με κάθε ένα από τα υπόλοιπα στοιχεία του, που θα διαβαστούν στα επόμενα iteration. Διατηρείται μεταβλητή(***flag***) που διαφοροποιείται σε περίπτωση που υπάρχει στοιχείο διαφορετικό από το αρχικό στοιχείο. Μεταβολή της τιμής της μεταβλητής οδηγεί στο συμπέρασμα ότι το διάνυσμα **X** δεν είναι σταθερό(υπάρχει τουλάχιστον ένα στοιχείο του διανύσματος που είναι διαφορετικό από τα υπόλοιπα).

```
if [ "$count" -eq 0 ]   #First iteration we save the first
then
    firstword=${my_array[0]}
fi

if [ "$count" -ne 0 ]
then                #Check if vector X is a constant vector
    if  (( $(echo "$firstword != ${my_array[0]}" | bc ) ))
    then
        flag=1 #means no constant vectors  #flag=1
    fi
fi
#echo count="${count}"

let "count=count+1"
```

Figure 2: Έλεγχος για σταθερό διάνυσμα **X**

Με την περάτωση της ανάγνωσης του αρχείου, οι μεταβλητές των αθροισμάτων έχουν λάβει την επιθυμητή τιμή τους. Η εύρεση των παραμέτρων **α** & **b** έγινε εφικτή. Καθίστανται γνωστές οι τιμές τους.

```
temp1=$(echo "scale=2; ((${line_no} * ${sum_xy}))  - ((${sum_x} * ${sum_y}))" | bc)
temp2=$(echo "scale=2; ((${line_no} * ${sum_x2}))  - ((${sum_x} * ${sum_x}))" | bc)
a=$(echo "scale=2; (${temp1} / ${temp2})/1" | bc | sed 's!\.0*$!!')

temp1=$(echo "scale=2; ${sum_y}  - ((${a} * ${sum_x}))" | bc)
b=$(echo "scale=2; (${temp1} / ${line_no})/1" | bc | sed 's!\.0*$!!')

while read LINE
do
    if [ "${LINE}" != "$prev" ]
    then
        my_array=($(echo ${LINE} | tr ":" "\n" ))
        temp1=$(echo "scale=2;  ((${a}  * ${my_array[0]})) + ${b}" | bc)
        temp1_2=$(echo "scale=2;  ${my_array[1]} - ${temp1}" | bc)
        temp2=$(echo "scale=2;  ${temp1_2} * ${temp1_2}" | bc)
        err=$(echo "scale=2; ${err} + ${temp2}" | bc)
    fi
done < 55.txt
```

Figure 3: Απόδοση τιμών για μη σταθερό διάνυσμα **X**

Σε περίπτωση σταθερού διανύσματος X, το πρόγραμμα εκτελεί τους υπολογισμούς με εισόδους κατακόρυφα διανύσματα, οπότε οι τύποι προσαρμόζονται ώστε να ικανοποιούν τις προδιαγραφές.

```
a=-1        #Different computations for constant vector
b=$(echo "scale=2; ${my_array[0]}" | bc | sed 's!\.0*$!!' )

while read LINE
do
    if [ "${LINE}" != "$prev" ]
    then
        my_array=($(echo ${LINE} | tr ":" "\n" ))
        temp1=$(echo "scale=2;  ((${a}  * ${my_array[0]})) + ${b}" | bc)
        temp1_2=$(echo "scale=2;  ${temp1} * ${temp1}" | bc)
        temp2=$(echo "scale=2;  ${my_array[1]} - ${temp1_2}" | bc)
        err=$(echo "scale=2; ${temp2} - (${my_array[1]} - ((((${a} * ${my_array[0]})) + ${b})))" | bc)
        #echo ${my_array[0]}
        #echo ${my_array[1]}
    fi
```

Figure 4: Απόδοση τιμών για σταθερό διάνυσμα

Η τελική εύρεση του σφάλματος απαιτεί εκ νέου ανάγνωση του αρχείου. Οι τιμές των διανυσμάτων δεν έχουν αποθηκευτεί σε κάποια διάταξη ώστε να είναι εύκολα προσπελάσιμες. Για κάθε αρχείο εισόδου εκτυπώνονται οι ζητούμενες πληροφορίες εξόδου.

Το διάβασμα όλων των αρχείων πραγματοποιείται με την μεταβολή της μεταβλητής των παραμέτρων στο τέλος του προγράμματος, μέσω της εντολής shift, ώστε να εκτελεστούν και τα επόμενα αρχεία.

2 ΠΡΟΓΡΑΜΜΑ RESULTS

Η υλοποίηση του προγράμματος πραγματοποιήθηκε θεωρώντας ως αρχείου εισόδου τους αγώνες που παίζουν ένα σύνολο ομάδων. Τα διαδικαστικά των νικών έχουν δωθεί στην εκφώνηση.

Αρχικά πραγματοποιείται έλεγχος ύπαρξης του αρχείου εισόδου. Η ορθότητα της ύπαρξης του αρχείου δίνει το έναυσμα για να προχωρήσει το πρόγραμμα. Σε αντίθετη περίπτωση, το πρόγραμμα κατευθύνεται σε εξαναγκασμένη έξοδο. Το πρόγραμμα είναι λειτουργικό με ένα αρχείο εισόδου ανα εκτέλεση.

Το αρχείο διαβάζεται διατηρώντας τα ονόματα των ομάδων που έχουν κενά και τυπώνοντας το, σε νέο αρχείο. Το βήμα αυτό είναι προαιρετικό και αποσκοπεί σε εύκολη επεξεργασία.

Το αρχείο διαβάζεται γραμμή προς γραμμή. Αναλύεται κάθε γραμμή για απομόνωση των ονομάτων των ομάδων και των σκορ τους. Τα σκόρ είναι χρήσιμη πληροφορία, καθώς τα γκολ που πετυχαίνει μια ομάδα αποτελούν τα γκολ που δέχεται η αντίπαλός της. Ακολούθως, πραγματοποιείται ο έλεγχος του σκόρ. Σημειώνονται οι πόντοι και τα γκολ για κάθε ομάδα(υπερ-κατα). Καταγράφονται σε αρχείο πληροφορίες για τον κάθε αγώνα που συμμετείχει η κάθε ομάδα. Ένας αγώνας εξάγει πληροφορία δυο φορές, μια φορά για την νικήτρια με τις εκάστοτε πληροφοριες για τον αγώνα και μια φορά για την ηττημένη. Δεδομένου συνόλου αγώνων 6 αγώνες, θα καταγραφούν στο αρχείο 12 καταχωρήσεις. Η μορφοποίηση του προσωρινού αρχείου ακολουθεί σε παραλαγμένη μορφή την επιθημητή μορφοποίηση εξαγωγής καθώς τα δεδομένα χωρίζονται με κενα αλλά διατηρούν την σωστή διάταξη.

```
if [ "${my_array[2]}" -gt "${my_array[3]}"  ] #Recognize which team is the winner
then
    echo -e ${my_array[0]}" 3 "${my_array[2]}" "${my_array[3]} >> output.txt    #
    echo -e ${my_array[1]}" 0 "${my_array[3]}" " ${my_array[2]} >> output.txt   #
    #echo Assos
elif [ "${my_array[2]}" -lt "${my_array[3]}" ]
then
    echo -e ${my_array[0]}" 0 "${my_array[2]}" "${my_array[3]} >> output.txt
    echo -e ${my_array[1]}" 3 "${my_array[3]}" "${my_array[2]} >> output.txt
    #echo Diplo
else
    echo -e ${my_array[0]}" 1 "${my_array[2]}" "${my_array[3]} >> output.txt
    echo -e ${my_array[1]}" 1 "${my_array[3]}" "${my_array[2]} >> output.txt
    #echo Isopalia
fi
```

Figure 5: Καταγραφή πληροφοριών για τις ομάδες

Η διαδικασία γίνεται ώστε κάθε πληροφορία να χωρίζεται από άλλη με το ίδιο διαχωριστικο. Βοηθάει στην ομαδοποίηση των δεδομένων σε επόμενο στάδιο.

Ακολούθως, καθώς υπάρχουν για κάθε ομάδα οι καταχωρήσεις της, πραγματοποιείται ομαδοποίηση των στοιχειων κάθε ομάδας. Ως εκ τούτου, αθροίζονται οι βαθμοί και τα γκολ εις βάρος της, αλλά και τα υπέρ της. Τα ομαδοποιημένα στοιχεία εξάγωνται σε νέο προσωρινό αρχείο. Τα αποτελεσματα εξάγωνται ξανά σε νέο αρχείο στο οποίο τοποθετούνται τα tabs. Το νέο αρχείο ταξινομείται σύμφωνα με τον αριθμό βαθμών της κάθε ομάδας και έπειτα ταξινομείται σύμφωνα με το όνομα της ομάδας. Το τελικό αρχείο συμπεριλαμβάνει και αρίθμηση των ομάδων.

```
awk 'BEGIN { FS=OFS=" " }      # set separators  #Only separator is whitespace.
{
    a[$1]+=$2             # Sum second field to a hash
    b[$1]+=$3               #Sum third field to a hash
    c[$1]+=$4          #    Sum fourth field to a hash
}
END {                     # in the end
    for(i in a)           # loop all
        print i"\t",a[i]"\t",b[i]"-"c[i] # and output
}' output.txt > output2.txt
#Keep the grouped information to a new file and write information with the give

sort -k2,2rn -k1 output2.txt > output3.txt
#Sort file by points and then by name
awk '{print NR  ".\t " $s}' output3.txt > output4.txt #Numerate
tr "_" " " < output4.txt > final2.txt #We used _ so we can handle teams' names
```

Figure 6: Ομαδοποίηση πληροφοριών για τις ομάδες

Τα προσωρινά αρχεία διαγράφονται στο τέλος της εκτέλεσης.

3 ΑΠΟΤΕΛΕΣΜΑΤΑ

3.1 regr

Τα συμπεράσματα έχουν εξαχθεί μέσα από την εκτέλεση του προγραμμάτος σε πολλαπλά αρχεία εισόδου. Στα αρχεία αυτά συμπεριλαμβάνεται και ένα αρχείου που δεν υφίσταται και προκαλεί τον τερματισμό του προγράμματος.

Τα συμπεράσματα εξάγονται παρακάτω:

```
zisk@zisk-Lenovo-G510:~/Επιφάνεια εργασίας/delitelos/regrrr$ ./regr 1h 2h 3h 4h 5h 6h 8k
FILE:1h,a=.58 b=11.40 c=1 err=239.62
FILE:2h,a=.36 b=13.63 c=1 err=239.65
FILE:3h,a=.58 b=11.36 c=1 err=239.61
FILE:4h,a=-1 b=5.1 c=0 err=0
FILE:5h,a=0 b=15 c=1 err=0
FILE:6h,a=.25 b=5.05 c=1 err=10.92


8k file does not exist
Forced Exit!!
```

Figure 7: Αποτελέσματα προγράμματος #1

Τα αρχεία εισόδου έχουν μελετηθεί ώστε να ελέγχονται διάφορα ζητήματα ορθότητας του προγράμματος.

- **1h:** Διαθέτει αριθμούς με ένα δεκαδικό ψηφίο.
- **2h:** Περιέχει επιπλέον ορισμένους δεδομένα με δυο δεκαδικά ψηφία.
- **3h:** Περιλαμβάνει ακέραιους αριθμούς.
- **4h:** Περιέχει σταθερό διάνυσμα X.
- **5h:** Περιέχει το σταθερό διάνυσμα Υ.
- **6h:** Είναι όμοιο με το τρίτο.
- **8k:** Δεν υπάρχει και το πρόγραμμα τερματίζει.

Ο πίνακας με τα δεδομένα των αρχείων είναι ο ακόλουθως:

1h		2h		3h		4h		5h		6h	
10.1	15.1	10.2	15.16	10	15	5.1	15	10	15	3	4
5.1	20.1	5.3	22.12	5	20	5.1	20	5	15	4	5
5.1	15.1	5.4	18.17	5	15	5.1	15	5	15	5	6
2.1	10.1	2.5	11.16	2	10	5.1	10	2	15	2	8
3.1	6.1	3.6	8.15	3	6	5.1	6	3	15	9	8
7.1	10.1	7.7	10.14	7	10	5.1	10	7	15		
5.1	25.1	5.8	25.12	5	25	5.1	25	5	15		

3.2 results

Τα συμπεράσματα έχουν εξαχθεί για διάφορες εκτελέσεις των προγραμμάτων. Η άριστη λειτουργικότητα του προγράμματος αποδεικνύεται από πολλαπλές εκτελέσεις του προγράμματος για διαφορετικά δεδομένα εισόδου.
Τα συμπεράσματα εξάγονται παρακάτω:

```
zisk@zisk-Lenovo-G510:~/Επιφάνεια εργασίας/delitelos/resultss$ ./results nes2.txt

1.      Russia  5       4-3
2.      Spain   5       4-3
3.      Italy   3       3-3
4.      Poland  1       3-5
```

Figure 8: Αποτελέσματα προγράμματος #2 Σενάριο #1

Στο πείραμα αυτό, ελέγχονται οι αναλύσεις των σκόρ και οι βαθμοί που συγκεντρώνουν οι ομάδες. Επιπλέον υπάρχει περίπτωση ισοβαθμίας. Παρατηρείται ότι στην περίπτωση αυτή, οι ομάδες ταξινομούνται σύμφωνα με το όνομα της ομάδας. Επομένως επιβεβαιώνεται η ορθή λειτουργικότητα.

```
zisk@zisk-Lenovo-G510:~/Επιφάνεια εργασίας/delitelos/resultss$ ./results res.txt

1.      Man City United         6       4-2
2.      Mpa     3       2-1
3.      Russia  3       2-4
4.      Spain Magiorca  3       1-0
5.      Spa     1       2-3
6.      Spain   1       1-2
```

Figure 9: Αποτελέσματα προγράμματος #2 Σενάριο #2

Το πείραμα αυτό αποτελεί όμοια περίπτωση με το προηγούμενο με την διαφορά ότι τα ονόματα των ομάδα χωρίζονται από κενά. Η ορθή λειτουργικότητα στην περίπτωση αυτή θεωρείται όταν το πρόγραμμα διατηρεί ολόκληρο το όνομα της ομάδας, χωρις να λάβει υπόψην του τα κενά στα ονόματα των ομάδων. Υπάρχει και ομάδα με πολλαπλά κενά και αντιμετωπίζεται με επιτυχία. Οι ισοβαθμίες αντιμετωπίζονται με επιτυχία και σε αυτό το πείραμα.

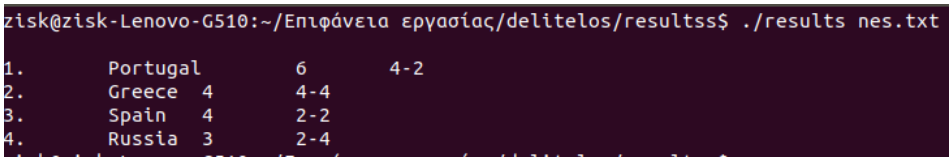

Figure 10: Αποτελέσματα προγράμματος #2 Σενάριο #3

Το πείραμα αυτό αποτελεί απλή περίπτωση. Η διάταξη εξόδου είναι η επιθυμητή. Υπάρχει περίπτωση ισοβαθμίας και αντιμετωπίζεται ορθά.
Ο πίνακας με τα δεδομένα εισόδου είναι ο εξής:

nes2.txt		res.txt		nes.txt	
Italy-Spain	1-1	Man City United-Mpa	1-2	Portugal-Greece	1-2
Russia-Poland	2-1	Spain Magiorca-Russia	1-0	Spain-Russia	1-0
Russia-Spain	1-1	Spa-Spain	1-1	Greece-Spain	1-1
Poland-Italy	1-1	Russia-Man City United	0-2	Russia-Portugal	0-2
Italy-Russia	1-1	Spa-Russia	1-2	Spain-Portugal	0-1
Spain-Poland	2-1	Spain-Man City United	0-1	Russia-Greece	2-1