Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ανάλυση και Σχεδίαση Αλγορίθμων
Εργασία 1 – Ασκήσεις Β

Ομάδα Υλοποίησης:
Χανός Στέφανος – 1573
Συγκούνας Ιωάννης – 1556

Άσκηση 1:

Αρχικά ξεκινήσαμε στην main και ζητήσαμε από τον χρήστη να εισάγει έναν μη αρνητικό αριθμό. Κάναμε τον κατάλληλο έλεγχο ώστε αυτός ο αριθμός ναι είναι όντως θετικός και στην συνέχεια προχωρήσαμε στο κάλεσμα των δυο συναρτήσεων μας.

Στην αρχή καλούμε την findDigit2 η οποία είναι η ΜΗ-αναδρομική μας συνάρτηση. Μέσα σε αυτή ορίζουμε την μεταβλητή sum στην οποία αποθηκεύουμε το άθροισμα των ψηφίων του αριθμού που εισάγει ο χρήστης. Όσο ο αριθμός είναι διάφορος του 0 τότε στην μεταβλητή μας προστίθεται η προηγούμενη τιμή της με το υπόλοιπο της διαίρεσης του αριθμού μας με το 10 και τέλος διαιρούμε τον αριθμό μας με το δέκα κάθε φορά.

Στην συνέχεια καλούμε την findDigit1 η οποία είναι η αναδρομική μας συνάρτηση. Ξανά ορίζουμε την μεταβλητή sum για τον ίδιο ακριβώς σκοπό και ορίζουμε την συνθήκη τερματισμού μας. Όσο λοιπόν δεν ισχύει αυτή η συνθήκη στην μεταβλητή μας προστίθεται η προηγούμενη τιμή της με το υπόλοιπο της διαίρεσης του αριθμού μας με το 10 και ξανακαλείται η

συνάρτηση μας με καινούργιο όρισμα τον αριθμό που είχαμε διαιρεμένο κατά 10 κάθε φορά.

```
Please enter a non negative number
-5
You have entered a negative number , please enter a positive one
4562

The number you chose is: [4562]
The sum of the number's digits (WITHOUT recursion) is: [17]
The sum of the number's digits (WITH recursion) is: [17]
```

Η ΜΗ-αναδρομική μας συνάρτηση (findDigit2) έχει πολυπλοκότητα $log\ (n)$, καθώς το βήμα μέσα στην δομή επανάληψης μας είναι διαιρετικό.

Η αναδρομική μας συνάρτηση (fingDigit1) έχει πολυπλοκότητα $log\ (n)$, γιατί ο αλγόριθμος μας θα κληθεί log(n) φορές όσα είναι και τα ψηφιά του αριθμού μας.

Άσκηση 2:

Αρχικά ξεκινάμε στη main όπου ζητάμε από τον χρήστη να μας δώσει το πλήθος των διαθέσιμων κομματιών. Κάναμε τον κατάλληλο έλεγχο ώστε αυτός ο αριθμός ναι είναι θετικός και στην συνέχεια προχωρήσαμε στο κάλεσμα των δυο συναρτήσεων μας.

Στην αρχή καλούμε την puzzle2 η οποία είναι η ΜΗ-αναδρομική μας συνάρτηση. Μέσα σε αυτή έχουμε μια δομή επανάληψης (while) η οποία θα εκτελείται όσο ο αριθμός των διαθέσιμων κομματιών είναι μεγαλύτερος του μηδενός. Αυξάνουμε κατά 1 το Levels και έχουμε έναν έλεγχο στον οποίο αν το Levels είναι ίσο με 1 αφαιρούμε μόνο ένα κομμάτι , αλλιώς με βάση την εκφώνηση καταλήξαμε ότι τα κομμάτια που πρέπει να αφαιρούνται σε όλα τα υπόλοιπα επίπεδα είναι 8*(levels-1) γι’ αυτό και αν δεν μπει στο if ο κώδικας ο τύπος για την αφαίρεση των κομματιών είναι Pieces=Pieces-(8*(Levels-1)). Τέλος μόλις τελειώσουν τα διαθέσιμα κομμάτια η συνάρτηση επιστρέφει τον αριθμό των Levels που χρειάστηκαν.

Στην συνέχεια καλούμε την puzzle1 η οποία είναι η αναδρομική μας συνάρτηση. Όμοια αυξάνουμε το levels κατά 1 και αν είμαστε στο πρώτο επίπεδο αφαιρούμε το πρώτο κομμάτι από τα Pieces , αλλιώς ξανακαλούμε την puzzle1 με όρισμα όμως αυτή την φορά όχι το Pieces αλλά την νέα τιμή

η οποία προκύπτει από την αφαίρεση Pieces-(8*(Levels-1)). Τέλος μόλις τελειώσουν τα διαθέσιμα κομμάτια το πρόγραμμα μπαίνει στην συνθήκη τερματισμού και επιστρέφει τον αριθμό των επίπεδων που χρειαστήκαν.

```
Please enter the number of pieces that are available
-81
The number of pieces cant negative , please enter a positive number
0
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [0]
The levels needed for the amount of pieces you entered (WITH recursion) are: [0]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
1
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [1]
The levels needed for the amount of pieces you entered (WITH recursion) are: [1]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
2
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [2]
The levels needed for the amount of pieces you entered (WITH recursion) are: [2]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
9
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [2]
The levels needed for the amount of pieces you entered (WITH recursion) are: [2]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
9
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [2]
The levels needed for the amount of pieces you entered (WITH recursion) are: [2]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
25
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [3]
The levels needed for the amount of pieces you entered (WITH recursion) are: [3]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
49
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [4]
The levels needed for the amount of pieces you entered (WITH recursion) are: [4]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1> cd "e:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση
 και σχεδιαση Αλγοριθμων\Εργασια 1\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Please enter the number of pieces that are available
81
The levels needed for the amount of pieces you entered (WITHOUT recursion) are: [5]
The levels needed for the amount of pieces you entered (WITH recursion) are: [5]
PS E:\ΣΧΟΛΗ\Visual Studio Code\Αναλυση και σχεδιαση Αλγοριθμων\Εργασια 1>
```

Η ΜΗ-αναδρομική μας συνάρτηση (puzzle2) έχει πολυπλοκότητα n , καθώς το βήμα μέσα στην δομή επανάληψης μας είναι αφαιρετικό.

Η αναδρομική μας συνάρτησή (puzzle1) έχει πολυπλοκότητα n , γιατί η συνάρτησή ξανακαλείται n φορές (Pieces) μείον k ($8 * (Levels - 1)$) αρά τελικά n.

Άσκηση 3:

Αρχικά ξεκινάμε στη main δημιουργώντας πέντε πίνακες και γεμίζοντας τους με τυχαίες τιμές αξιοποιώντας την συνάρτηση rand(). Έπειτα ζητάμε από τον χρήστη να επιλέξει ποια συνάρτηση θέλει να χρησιμοποιήσει και κάνουμε τον κατάλληλο έλεγχο για την απάντηση του. Άμα διαλέξει την πρώτη χρονομετράμε την εύρεση της ελάχιστης διαφοράς για κάθε έναν από τους πέντε πίνακες και βρίσκουμε και έναν μέσο ορό αυτών των χρονών. Ομοίως και άμα ο χρήστης διαλέξει την δεύτερη συνάρτηση.

Για την πρώτη μας συνάρτηση , την Starter, αξιοποιήσαμε τον δοθέντα ψευδοκώδικα. Ο συγκεκριμένος αφαιρεί το πρώτο στοιχείο με όλα τα υπόλοιπα στοιχεία του πίνακα , μετά το δεύτερο και ούτω καθεξής και αποθηκεύει την μικρότερη τιμή σε μια μεταβλητή την οποία επιστρέφει κιόλας.

Η δεύτερη συνάρτηση μας , η Better , είναι η βελτιούμενη εκδοχή της πρώτης. Χρειάστηκε πρώτα να αξιοποιήσουμε μια έτοιμη υλοποίηση της συνάρτησης quickshort() βάση της οποίας ταξινομούμε τον πίνακα μας. Με αυτόν τον τρόπο πλέον η αφαίρεση γίνεται μεταξύ ενός στοιχείου του πίνακα μόνο με το αμέσως επόμενο του , καθιστώντας τον αισθητά πιο γρήγορο από τον πρώτο.

Η πρώτη συνάρτηση (Starter) έχει πολυπλοκότητα n^2 , καθώς έχουμε δυο εμφωλευμένες δομές επανάληψης με προσθετικό βήμα. Οπότε κάθε δομή επανάληψης έχει πολυπλοκότητα n και αφού είναι εμφωλευμένες συνολικά θα έχουμε n^2.

Η δεύτερη συνάρτηση (Better) έχει πολυπλοκότητα n , καθώς το βήμα μέσα στην δομή επανάληψης μας είναι προσθετικό.

Για 60000 στοιχεία μέσα στους πίνακες μας οι μέτρησης των δυο συναρτήσεων μας έχουν ως εξής:

```
Would you like to use the starting function (1) or the improved one (2) ??
1
For 60000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 10.258000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 10.272000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 10.264000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 10.273000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 10.269000 seconds
The average run time of the the five tables is: 10.267200 seconds
```

```
Would you like to use the starting function (1) or the improved one (2) ??
2
For 60000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 0.010000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 0.014000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 0.015000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 0.014000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 0.014000 seconds
The average run time of the the five tables is: 0.013400 seconds
```

Για 70000 στοιχεία μέσα στους πίνακες μας οι μέτρησης των δυο συναρτήσεων μας έχουν ως εξής:

```
Would you like to use the starting function (1) or the improved one (2) ??
1
For 70000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 13.972000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 14.000000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 13.994000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 13.995000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 13.987000 seconds
The average run time of the the five tables is: 13.989600 seconds
```

```
Would you like to use the starting function (1) or the improved one (2) ??
2
For 70000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 0.012000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 0.012000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 0.011000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 0.012000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 0.012000 seconds
The average run time of the the five tables is: 0.011800 seconds
```

Για 80000 στοιχεία μέσα στους πίνακες μας οι μέτρησης των δυο συναρτήσεων μας έχουν ως εξής:

```
Would you like to use the starting function (1) or the improved one (2) ??
1
For 80000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 18.291000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 18.273000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 18.285000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 18.296000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 18.294000 seconds
The average run time of the the five tables is: 18.287800 seconds
```

```
Would you like to use the starting function (1) or the improved one (2) ??
2
For 80000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 0.016000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 0.020000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 0.018000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 0.014000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 0.014000 seconds
The average run time of the the five tables is: 0.016400 seconds
```

Για 90000 στοιχεία μέσα στους πίνακες μας οι μέτρησης των δυο συναρτήσεων μας έχουν ως εξής:

```
Would you like to use the starting function (1) or the improved one (2) ??
1
For 90000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 23.161000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 23.097000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 23.093000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 23.097000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 23.083000 seconds
The average run time of the the five tables is: 23.106200 seconds
```

```
Would you like to use the starting function (1) or the improved one (2) ??
2
For 90000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 0.018000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 0.016000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 0.022000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 0.021000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 0.022000 seconds
The average run time of the the five tables is: 0.019800 seconds
```

Για 100000 στοιχεία μέσα στους πίνακες μας οι μέτρησης των δυο συναρτήσεων μας έχουν ως εξής:

```
Would you like to use the starting function (1) or the improved one (2) ??
1
For 100000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 28.529000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 28.512000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 28.576000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 28.560000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 28.525000 seconds
The average run time of the the five tables is: 28.540400 seconds
```

```
Would you like to use the starting function (1) or the improved one (2) ??
2
For 100000 table elements
The smallest difference of two numbers in table A is: [0] , and it needed 0.017000 seconds
The smallest difference of two numbers in table B is: [0] , and it needed 0.019000 seconds
The smallest difference of two numbers in table C is: [0] , and it needed 0.019000 seconds
The smallest difference of two numbers in table D is: [0] , and it needed 0.017000 seconds
The smallest difference of two numbers in table E is: [0] , and it needed 0.018000 seconds
The average run time of the the five tables is: 0.018000 seconds
```

Ο Λόγος που η ελάχιστη διαφορά βγαίνει πάντα μηδέν είναι διότι σε πίνακες με τόσα πολλά στοιχεία μέσα τυγχάνει να υπάρχουν ιδίες τιμές , κάναμε και δοκιμές σε πίνακες με λιγότερα στοιχεία και η ελάχιστη διαφορά ήταν διαφορετική κάθε φορά

Παρατηρούμε λοιπόν πως η βελτιωμένη συνάρτηση είναι κατά πολύ μεγάλο βαθμό γρηγορότερη από την αρχική.