[](http://www.teilar.gr/)

**Αλγόριθμοι και Στοιχεία Πολυπλοκότητας.**

**Ενότητα 4:** Αναδρομικοί Αλγόριθμοι.

Διδάσκων: Ηλίας Κ Σάββας, Αναπληρωτής Καθηγητής.

Τμήμα Μηχανικών Πληροφορικής, Τεχνολογικής Εκπαίδευσης.

**Άδειες χρήσης.**

* Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons (C C). **Αναφορά δημιουργού (B Y), Μη εμπορική χρήση (N C), Μη τροποποίηση (N D), 3.0, Μη εισαγόμενο.**
* Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

[](http://creativecommons.org/licenses/by-nc-nd/3.0/deed.el)

**Χρηματοδότηση.**

* Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
* 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.

[](http://www.edulll.gr/)

# Σκοποί ενότητας.

Ο αναγνώστης να μπορεί να:

1) χρησιμοποιεί την τεχνική της αναδρομής,

2) επιλύει προβλήματα εφαρμόζοντας αναδρομή.

# ΠΕΡΙΕΧΌΜΕΝΑ ΕΝΌΤΗΤΑΣ.

[Σκοποί ενότητας. 2](#_Toc368170668)

[ΠΕΡΙΕΧΌΜΕΝΑ ΕΝΌΤΗΤΑΣ. 3](#_Toc368170669)

[ΑΛΓΌΡΙΘΜΟΙ. 3](#_Toc368170670)

[ΠΡΟΓΡΆΜΜΑΤΑ. 3](#_Toc368170671)

[ΣΧΉΜΑΤΑ. 3](#_Toc368170672)

[ΠΊΝΑΚΕΣ. 3](#_Toc368170673)

[4. ΑΝΑΔΡΟΜΙΚΟΊ ΑΛΓΌΡΙΘΜΟΙ. 4](#_Toc368170674)

[4.1 Οι Πύργοι του Hanoi. 4](#_Toc368170675)

# ΑΛΓΌΡΙΘΜΟΙ.

|  |  |
| --- | --- |
| ***Αλγόριθμος 1****: Μη αναδρομικός αλγόριθμος επίλυσης του προβλήματος «Πύργοι του Hanoi».* | 8 |

# ΠΡΟΓΡΆΜΜΑΤΑ.

|  |  |
| --- | --- |
| ***Πρόγραμμα 1****: Πρόγραμμα υλοποίησης προβλήματος «Πύργοι του Hanoi».* | 6 |

# ΣΧΉΜΑΤΑ.

|  |  |
| --- | --- |
| ***Σχήμα 4.1*** *Πύργοι του Hanoi.* | 4 |

# ΠΊΝΑΚΕΣ.

|  |  |
| --- | --- |
| ***Πίνακας 4.1.*** *Χρόνος που απαιτείται για ορισμένες κινήσεις.* | 6 |

# 4. ΑΝΑΔΡΟΜΙΚΟΊ ΑΛΓΌΡΙΘΜΟΙ.

*Αναδρομικός* λέγεται ο αλγόριθμος ο οποίος επιλύει ένα πρόβλημα, επιλύνοντας ένα ή περισσότερα μικρότερα κομμάτια του ίδιου προβλήματος. Για να υλοποιήσουν αναδρομικούς αλγόριθμους, πολλές γλώσσες προγραμματισμού χρησιμοποιούν αναδρομικές συναρτήσεις, (συναρτήσεις οι οποίες καλούν τους εαυτούς τους). Οι αναδρομικές συναρτήσεις στη C για παράδειγμα, μοιάζουν με τον μαθηματικό ορισμό των συναρτήσεων. Ίσως ένας από τους πιο τυπικούς αναδρομικούς αλγόριθμους, είναι αυτός που υπολογίζει το *N*! χρησιμοποιώντας τον αναδρομικό τύπο . Η αντίστοιχη συνάρτηση σε C είναι η ακόλουθη:

Η παρακάτω αναδρομική συνάρτηση υπολογίζει την συνάρτηση N! με τον πρότυπο ορισμό της αναδρομικότητας. Επιστρέφει σωστό αποτέλεσμα για θετικό N, και σχετικά μικρό ώστε να μπορεί να αναπαρασταθεί σαν int.

int paragontiko(int N)

{

if (N == 0) return 1;

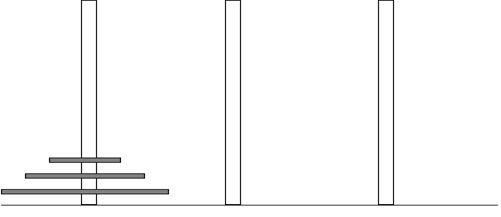
return N \* paragontiko(N -1);

}

Η χρήση της αναδρομής επιτρέπει την περιγραφή δύσκολων αλγορίθμων με ένα συμπαγή τρόπο, χωρίς να χάνεται η αποτελεσματικότητα του αλγόριθμου, (αν και μερικές φορές οι αναδρομικοί αλγόριθμοι παρουσιάζουν χειρότερη πολυπλοκότητα από τους μη αναδρομικούς).

## 4.1 Οι Πύργοι του Hanoi.

Ένα από τα πιο κλασικά παραδείγματα χρήσης αναδρομικών συναρτήσεων είναι οι «***Πύργοι του Hanoi***». Η περιγραφή του προβλήματος είναι η ακόλουθη: Υπάρχουν τρείς κολόνες (*Σχήμα 4.1*) και σε μία από αυτές είναι περασμένοι Ν ομόκεντροι δίσκοι διαφορετικών διαμέτρων μειούμενοι από κάτω προς τα πάνω.



***Σχήμα 4.1*** *Πύργοι του Hanoi.*

Το πρόβλημα έγκειται στην μεταφορά αυτών των δίσκων στη γειτονική κολόνα, αλλά με τους παρακάτω περιορισμούς:

1) Επιτρέπεται η μεταφορά ενός δίσκου κάθε φορά, και

2) Απαγορεύετε να βρεθούν δύο διαδοχικοί δίσκοι στη ίδια κολόνα με τον μικρότερο κάτω από τον μεγαλύτερο.

Ένα θρύλος λέει πως εάν ένα σύνολο μοναχών, καταφέρει να μεταφέρει 40 χρυσούς δίσκους σε ένα ναό, χρησιμοποιώντας τρείς αδαμάντινες κολόνες και στην διάρκεια της ζωής τους, τότε ο κόσμος θα «τελειώσει».

Η ιδέα επίλυσης αυτού του προβλήματος είναι η εξής: Για να μετακινηθούν οι N δίσκοι στη δεξιά κολόνα, (έστω 1 ο δίσκος της κορυφής, 2 ο επόμενος, οπότε Ν ο τελευταίος και πιο μεγάλος δίσκος), αρκεί να μετακινηθούν οι Ν-1 πιο πάνω δίσκοι στην αριστερή κολόνα, στη συνέχεια ο δίσκος Ν να μετακινηθεί στην δεξιά κολόνα, και τέλος, η στοίβα των Ν-1 δίσκων να μετακινηθεί πάνω από τον δίσκο Ν.

Η πολυπλοκότητα του αλγόριθμου υπολογίζεται σχετικά εύκολα, λόγω της αναδρομικής δομής του:

**Πρόταση:** *Ο αλγόριθμος επίλυσης του προβλήματος «Πύργοι του Hanoi» παράγει λύση η οποία δημιουργεί  κινήσεις.*

Απόδειξη: Εφόσον πρέπει να μετακινηθεί η στοίβα των Ν-1 δίσκων μία θέση δεξιά, και άλλη μία φορά επάνω από τον δίσκο Ν, τότε χρειάζεται δύο φορές το πλήθος των κινήσεων εάν είχαμε μόνο μια στοίβα Ν-1 δίσκων και επιπλέον, μία κίνηση για τον δίσκο Ν. Οπότε εάν απαιτούνται κινήσεις για μια στοίβα Κ δίσκων, τότε για να μετακινηθούν Ν δίσκοι χρειάζονται:

 κινήσεις για και .

Οπότε με επαγωγή έχουμε:

,

έστω ότι ισχύει για , δηλαδή , για τότε,

.

Οπότε και η πολυπλοκότητα της μεθόδου είναι .

Για να γίνει πιο αντιληπτή η πολυπλοκότητα του παραπάνω αλγόριθμου, ας δούμε το εξής: Εάν οι μοναχοί χρειαζόταν ένα δευτερόλεπτο για να μετακινήσουν ένα δίσκο από μια κολόνα σε μια άλλη, θα τους έπαιρνε 348 αιώνες για να τελειώσουν (με βάση τον *Πίνακα 1* που ακολουθεί).

(), με την προϋπόθεση ότι δεν έκαναν κανένα λάθος.

***Πίνακας 4.1.*** *Χρόνος που απαιτείται για ορισμένες κινήσεις.*

|  |  |
| --- | --- |
| **Κινήσεις** | **Χρόνος** |
| 102 | 1,7 λεπτά |
| 104 | 2,8 ώρες |
| 105 | 1,1 ημέρες |
| 106 | 1,6 εβδομάδες |
| 107 | 3,8 μήνες |
| 108 | 3,1 χρόνια |
| 109 | 3,1 δεκαετίες |
| 1010 | 3,1 αιώνες |
| 1011 | ΠΟΤΕ |

**Μετατροπή δευτερολέπτων**: Η διαφορά χρόνου φαίνεται εάν μετατρέψουμε τις μονάδες σε ποιο κατανοητές χρονικές μονάδες.

Στο παρακάτω πρόγραμμα φαίνεται καθαρά ο αναδρομικός αλγόριθμος επίλυσης.

***Πρόγραμμα 1****: Πρόγραμμα υλοποίησης προβλήματος «Πύργοι του Hanoi».*

#include <stdio.h>

void hanoi(int, int);

void metakinise(int, int);

char thesi[40]; /\* H τρέχουσα θέση του κάθε δίσκου\*/,

long met = 0; /\* Πλήθος Κινήσεων \*/,

main()

{

int diskoi;

int i;

printf("\n\n Πλήθος δίσκων:");

scanf("%d", &diskoi);

for (i=1; i<=diskoi; i++)

thesi[i] = 'A';

hanoi(diskoi, 1);

}

**void hanoi(int N, int k)**

**{**

**if (N == 0) return;**

**hanoi(N-1, -k);**

**metakinise(N, k);**

**hanoi(N-1, -k);**

**}**

void metakinise(int N, int k)

{

met++;

if (k>0) thesi[N]++;

else thesi[N]--;

if (thesi[N] > 'C') thesi[N] = 'A';

if (thesi[N] < 'A') thesi[N] = 'C';

printf("\n %3d: Δίσκος %5d στη θέση %5c", met, N, thesi[N]);

}

Ο αλγόριθμος προσδιορίζει ποιός δίσκος θα μεταφερθεί και προς ποιά κατεύθυνση, (δηλαδή σε ποιά κολόνα), σε κάθε βήμα. Εάν το k είναι θετικό θα μεταφερθεί δεξιά, και εάν δεν υπάρχει πιο δεξιά κολόνα, τότε στην πρώτη από αριστερά, αλλιώς εάν το k είναι αρνητικό, θα μεταφερθεί αριστερά και πάλι εάν δεν υπάρχει πιο αριστερά στην δεξιότερη κολόνα, (στο πρόγραμμα οι κολόνες ονομάζονται, A, B, C).

Η εκτέλεση του προγράμματος για Ν = 3 παράγει τις παρακάτω κινήσεις:

hanoi(3, 1),

hanoi(2, -1),

hanoi(1, 1),

hanoi(0, -1),

metakinise(1, +1, άρα δεξιά),

hanoi(0, -1),

metakinise(2, -1, άρα αριστερά),

hanoi(1, 1),

hanoi(0, -1),

metakinise(1, +1),

hanoi(0, -1),

metakinise(3, +1),

hanoi(2, -1),

hanoi(1, 1),

hanoi(0, -1),

metakinise(1, +1),

hanoi(0, -1),

metakinise(2, -1),

hanoi(1, 1),

hanoi(0, -1),

metakinise(1, +1),

hanoi(0, -1),

Τέλος, ένας μη αναδρομικός αλγόριθμος επίλυσης του προβλήματος των πύργων του Hanoi, είναι ο ακόλουθος:

***Αλγόριθμος 1****: Μη αναδρομικός αλγόριθμος επίλυσης του προβλήματος «Πύργοι του Hanoi».*

1: Εφόσον δεν τελείωσε,

2: Μετακίνησε τον μικρότερο δίσκο μια θέση δεξιά, εάν το πλήθος των δίσκων n είναι περιττός αριθμός, (αριστερά, εάν είναι άρτιος),

3: Μετακίνησε τον μόνο δίσκο που μπορεί να μετακινηθεί (πλην του μικρότερου),

4: Τέλος Εφόσον.

5: Τέλος αλγόριθμου.

Τέλος τέταρτης ενότητας.

[](http://creativecommons.org/licenses/by-nc-nd/3.0/deed.el)[](http://www.edulll.gr/)