Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Αγρονόμων Τοπογράφων Μηχανικών

Προγραμματιστικές Τεχνικές

Βασίλειος Βεσκούκης
Δρ. Ηλεκτρολόγος Μηχανικός &
Μηχανικός Υπολογιστών ΕΜΠ
v.vescoukis@cs.ntua.gr

Ρωμύλος Κορακίτης
Αστροφυσικός
Αναπλ. Καθηγητής ΕΜΠ
romylos@survey.ntua.gr

Αναδρομικοί αλγόριθμοι

Αλγόριθμοι αναζήτησης

Αναδρομή

Αναδρομή είναι η ικανότητα μιας συνάρτησης να καλεί τον εαυτό της.

- **Γιατί η αναδρομή είναι χρήσιμη;**

  Η νέα κλήση της συνάρτησης αναφέρεται σε μια απλούστερη κατάσταση.
- **Πότε τερματίζεται η αναδρομή;**

  Όταν ο υπολογισμός που απαιτείται είναι τόσο απλός που γίνεται με ανάθεση τιμής (συνήθως κάποια αρχική τιμή)

- **Μια αναδρομική συνάρτηση έχει δύο τμήματα:**

1. Μια κλήση του εαυτού της με απλούστερες παραμέτρους

2. Ένα τμήμα ελέγχου, που τερματίζει την συνάρτηση στην αρχική τιμή.

Αναδρομή - Παράδειγμα 1 : Παραγοντικό

Πρόβλημα:

Να σχεδιαστεί και να γραφεί μια αναδρομική συνάρτηση που θα υπολογίζει το παραγοντικό ενός ακεραίου αριθμού.

Δεδομένα εισόδου: ο ακέραιος n

Δεδομένα εξόδου: η τιμή του n!

Τύπος υπολογισμού : $n! = 1 \cdot 2 \cdot 3 \ \ldots \ \cdot n$

Αναδρομικός τύπος : $n! = n \cdot (n-1)!$

Απλούστερη συνθήκη : $0! = 1$

ΑΛΓΟΡΙΘΜΟΣ

```
Factorial(n)
if (n=0) return 1
else return n*Factorial(n-1)
```

Αναδρομή - Παράδειγμα 2 : Μέγιστος κοινός διαιρέτης

Πρόβλημα:

Να σχεδιαστεί και να γραφεί μια αναδρομική συνάρτηση που θα υπολογίζει τον μέγιστο κοινό διαιρέτη δύο ακεραίων m και n.

ΑΛΓΟΡΙΘΜΟΣ 1 (Ευκλείδιος) (m > n)

```
gcd(m,n)
if (n=0) return m
else return gcd(n,m MOD n)
```

ΑΛΓΟΡΙΘΜΟΣ 2 (Dijkstra)

```
gcd(m,n)
if (m=n) return m
else if (m>n) return gcd(m-n,n)
else return gcd(m,n-m)
```

Αναδρομή - Παράδειγμα 3 : Ταξινόμηση Quicksort

Ο πίνακας A[N] περιέχει αταξινόμητα δεδομένα και επιζητούμε ***αύξουσα*** ταξινόμηση.

Η λειτουργία του αλγορίθμου αυτού στηρίζεται στην ***αναδρομή***: ο πίνακας χωρίζεται σε τμήματα και ο αλγόριθμος καλεί τον εαυτό του στα αταξινόμητα, μικρότερα τμήματα.

Τα βασικά βήματα του αλγορίθμου είναι:

1. Αρχικά επιλέγεται ως οδηγός ένα τυχαίο στοιχείο του πίνακα (πχ το πρώτο) και μετρώνται πόσα στοιχεία του A είναι μικρότερα από αυτό. Έστω X ο αριθμός αυτός.

2. Το οδηγό στοιχείο ανταλλάσσει την θέση του με το στοιχείο A[X]. Ο αρχικός πίνακας A έχει έτσι χωριστεί σε 3 τμήματα:

- Κάτω υποπίνακας, με στοιχεία A[0] ως A[X-1]
- Το οδηγό στοιχείο A[X], που βρίσκεται στην σωστή τελική του θέση
- Πάνω υποπίνακας, με στοιχεία A[X+1] ως A[N-1]

Αναδρομή - Παράδειγμα 3 : Ταξινόμηση Quicksort

3. Επειδή υπάρχει ένας αριθμός στοιχείων του κάτω υποπίνακα που πρέπει να μεταφερθούν στον πάνω υποπίνακα και αντίστροφα, γίνεται αυτή η εναλλαγή των στοιχείων με την βοήθεια δύο δεικτών που διατρέχουν τους υποπίνακες με αντίστροφη φορά.

4. Τώρα, όλα τα στοιχεία του κάτω υποπίνακα είναι μικρότερα από το οδηγό στοιχείο και, αντίστοιχα, όλα τα στοιχεία του πάνω υποπίνακα είναι μεγαλύτερα από αυτό. Ο αλγόριθμος καλεί τον εαυτό του, χωριστά στον κάτω και στον πάνω υποπίνακα.

5. Ο αλγόριθμος τερματίζεται όταν και οι δύο υποπίνακες της τελευταίας αναδρομής είναι κενοί, οπότε ανασυγκροτείται ο αρχικός πίνακας με διαδοχικές ενώσεις των υποπινάκων, που είναι σωστά ταξινομημένοι.

Υπάρχουν διάφορες παραλλαγές στην υλοποίηση του αλγόριθμου QuickSort. Στον ακόλουθο κώδικα γίνονται πρώτα οι αναδρομικές κλήσεις για τους κάτω υποπίνακες και μετά για τους άνω.

Αναδρομή - Παράδειγμα 3 : Ταξινόμηση Quicksort

***ΚΩΔΙΚΑΣ* C++**

```
void quicksort(float arr[],int low, int high) {
    int i, j;
    float temp;
    while (high>low) {
        i=low;
        j=high;
        temp=arr[low];
        while (i<j) {
            while (arr[j]>temp)
                j=j-1;
            arr[i]=arr[j];
            while ((i<j) && (arr[i]<=temp))
                i=i+1;
            arr[j]=arr[i]; }
        arr[i]=temp;
        quicksort(arr,low,i-1);
        low=i+1; }
}
```

Αναζήτηση (search)

Μια συνήθης διαδικασία στα στοιχεία ενός πίνακα είναι η αναζήτηση.

Αναζήτηση (search) είναι η έρευνα των στοιχείων ενός πίνακα με σκοπό την εύρεση της θέσης (= δείκτη στοιχείου) στην οποία βρίσκεται κάποια συγκεκριμένη τιμή – κλειδί (search key) ίδιου τύπου. Η αναζήτηση μπορεί να είναι επιτυχής ή ανεπιτυχής.

Ο απλούστερος αλγόριθμος αναζήτησης είναι η γραμμική αναζήτηση (linear or sequential search). Εξετάζουμε, διαδοχικά, όλα τα στοιχεία του πίνακα, συγκρίνοντάς τα με το κλειδί, μέχρις ότου βρεθεί το στοιχείο που έχει την τιμή – κλειδί ή δεν βρεθεί τέτοιο στοιχείο (ανεπιτυχής αναζήτηση).

Στην βασική της μορφή, η γραμμική αναζήτηση ΔΕΝ απαιτεί ταξινομημένο πίνακα, όμως η επίδοσή της μπορεί να βελτιωθεί αν ο πίνακας είναι ταξινομημένος.

Γραμμική αναζήτηση (Linear search)

Αναζητούμε την τιμή key, αν υπάρχει, στα στοιχεία του πίνακα A[N]

Στην απλούστερη μορφή, συγκρίνουμε διαδοχικά τα στοιχεία του πίνακα με την τιμή key μέχρις ότου εξαντληθούν τα στοιχεία ή βρεθεί στοιχείο ίσο με το key, οπότε ο αλγόριθμος τερματίζεται.

Η τιμή του δείκτη που αναφέρεται στο ζητούμενο στοιχείο καταχωρείται σε μια ακέραια μεταβλητή , π.χ. την K, που αρχικοποιείται σε μια τιμή άκυρη ως δείκτης του πίνακα (π.χ. K = – 1). Μετά τον τερματισμό του αλγορίθμου, αν η τιμή key ΔΕΝ υπάρχει στον πίνακα (ανεπιτυχής αναζήτηση), η K διατηρεί την αρχική τιμή της. Αν η αναζήτηση είναι επιτυχής, η τιμή του K δηλώνει το πρώτο στοιχείο του πίνακα που έχει τιμή key.

Γραμμική αναζήτηση (Linear search)

ΨΕΥΔΟΚΩΔΙΚΑΣ

ΑΡΧΗ
Β1 Επιτυχία = ΨΕΥΔΗΣ
K = − 1
I = 0
Β2 **Εφ όσον** (I < N *ΚΑΙ* Επιτυχία = ΨΕΥΔΗΣ) , **εκτέλεσε**:
Β3 **Αν** (A[I] = key) , *τότε* : Επιτυχία = ΑΛΗΘΗΣ
K = I
Β4 I = I + 1
Β5 *Επιστροφή στο* Β2
ΤΕΛΟΣ

Δυαδική αναζήτηση (Binary search)

Αναζητούμε την τιμή key, αν υπάρχει, στα στοιχεία του **ταξινομημένου** πίνακα A[N]

Ο αλγόριθμος αυτός στηρίζεται στην διαδοχική αναζήτηση της τιμής – κλειδί key σε υποπίνακες, που ο καθένας έχει εύρος *μισό* από τον προηγούμενο. Όπως και στην γραμμική αναζήτηση, χρησιμοποιείται μια ακέραια μεταβλητή (π.χ. K) για την καταχώρηση της θέσης (δείκτη) του στοιχείου του πίνακα που αναζητείται.

Σε κάθε υποπίνακα γίνεται σύγκριση του key με το ***μεσαίο*** στοιχείο του υποπίνακα.

- Αν υπάρχει ταύτιση τιμών, η αναζήτηση είναι επιτυχής, καταχωρείται ο δείκτης του στοιχείου στην K και ο αλγόριθμος τερματίζει.
- Αν το μεσαίο στοιχείο έχει τιμή μεγαλύτερη από το key, αυτό σημαίνει ότι το ζητούμενο στοιχείο είναι δυνατόν να βρίσκεται στον πρώτο μισό υποπίνακα (με τιμές μικρότερες του μεσαίου στοιχείου). Αναπροσαρμόζονται τα όρια του υποπίνακα και η αναζήτηση επαναλαμβάνεται στον νέο υποπίνακα, που έχει μέγεθος μισό από τον προηγούμενο.
- Αντίστοιχα, αν το μεσαίο στοιχείο έχει τιμή μικρότερη από το key, αυτό σημαίνει ότι το ζητούμενο στοιχείο είναι δυνατόν να βρίσκεται στον δεύτερο μισό υποπίνακα (με τιμές μεγαλύτερες του μεσαίου στοιχείου). Αναπροσαρμόζονται τα όρια του υποπίνακα και η αναζήτηση επαναλαμβάνεται στον νέο υποπίνακα.

Μετά τον τερματισμό του αλγορίθμου, αν K = – 1 , τότε η τιμή key ΔΕΝ υπάρχει στον πίνακα. Αλλιώς, η τιμή του K δηλώνει το πρώτο στοιχείο του πίνακα που έχει τιμή key.

Δυαδική αναζήτηση (Binary search)

ΨΕΥΔΟΚΩΔΙΚΑΣ

```
ΑΡΧΗ
Β1      Επιτυχία = ΨΕΥΔΗΣ
        Κ = – 1
        Αριστερά = 0
        Δεξιά = Ν – 1
Β2      Εφ όσον (Αριστερά <= Δεξιά ΚΑΙ Επιτυχία = ΨΕΥΔΗΣ) εκτέλεσε:
                Μέσον = (Αριστερά + Δεξιά) / 2
Β3              Αν (Α[Μέσον] = key) , τότε : Επιτυχία = ΑΛΗΘΗΣ
                                             Κ = Μέσον
Β4              Αλλιώς Αν (Α[Μέσον] > key) , τότε : Δεξιά = Μέσον – 1
Β5              Αλλιώς          Αριστερά = Μέσον + 1
Β6      Επιστροφή στο Β2
ΤΕΛΟΣ
```

Δυαδική αναζήτηση (Binary search)

ΠΑΡΑΔΕΙΓΜΑ

Αναζητούμε τον αριθμό 28 στον ταξινομημένο πίνακα (N = 13):

16, 17, 22, 28, 36, 43, 49, 55, 69, 72, 73, 88, 91

key = 28

Αρχικές τιμές : Αριστερά = 0, Δεξιά = 12 (εύρος = 13)

1η επανάληψη : Μέσον = 6

A[6] = 49 > 28 → Αριστερά = 0, Δεξιά = 5 (εύρος = 6)

2η επανάληψη : Μέσον = 2

A[2] = 22 < 28 → Αριστερά = 3, Δεξιά = 5 (εύρος = 3)

3η επανάληψη : Μέσον = 4

A[4] = 36 > 28 → Αριστερά = 3, Δεξιά = 3 (εύρος = 1)

4η επανάληψη : Μέσον = 3

A[3] = 28 = 28 → Αναζήτηση επιτυχής στο στοιχείο A[3]

Σύγκριση των αλγορίθμων αναζήτησης

Ο αλγόριθμος γραμμικής αναζήτησης χρειάζεται, κατά μέσο όρο, N/2 συγκρίσεις όταν η αναζήτηση είναι επιτυχής ή N όταν είναι ανεπιτυχής (στην απλή μορφή του).

Αντίθετα, ο αλγόριθμος δυαδικής αναζήτησης εκτελεί μόνο μια σύγκριση σε κάθε υποπίνακα. Επομένως, χρειάζεται *το πολύ* Π συγκρίσεις, όπου $2^{\Pi - 1} <= N <= 2^{\Pi}$.

Είναι προφανές ότι η διαφορά είναι πολύ μεγάλη, ιδιαίτερα για μεγάλες τιμές του N. Για παράδειγμα, κατά την αναζήτηση ενός στοιχείου σε ένα πίνακα μεγέθους 10^6 μπορεί να απαιτηθούν 500 000 συγκρίσεις με τον αλγόριθμο γραμμικής αναζήτησης και μόνο 20, το πολύ, συγκρίσεις με τον αλγόριθμο δυαδικής αναζήτησης.

Παραδείγματα

1. Εύρεση του ΜΚΔ δυο ακεραίων με τον αλγόριθμο του Dijkstra.

   (pt2006_lect4_example1.cpp)

2. Ταξινόμηση πίνακα ακεραίων που έχει τυχαίους αριθμούς (κρυφούς) και αναζήτηση κάποιου αριθμού που δίνει ο χρήστης.

   (pt2006_lect4_example2.cpp)

3. Ανάγνωση του ταξινομημένου αρχείου των αποστάσεων sortlen.txt (που δημιουργήθηκε στο pt2006_lect3_example.cpp) και αναζήτηση των εγγραφών που βρίσκονται μεταξύ δύο τιμών a και b που δίνει ο χρήστης.

   (pt2006_lect4_example3.cpp)