Αλγόριθμοι σε C

Μάθημα 2:

Αναζήτηση Στοιχείου σε Πίνακα

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Σημείωση: Το μάθημα αυτό απαιτεί να έχουν μελετηθεί τα αντίστοιχα μαθήματα 1..10 της γλώσσας προγραμματισμού C.

Α. Θεωρία

1. Αναζήτηση Στοιχείου σε Πίνακα

1. Ορισμός του Προβλήματος

Η **«Αναζήτηση σε Πίνακα»** είναι το υπολογιστικό πρόβλημα στο οποίο:

- **Είσοδος:** Ένας Πίνακας Στοιχείων PIN και ένα στοιχείο x
- **Έξοδος:** ΝΑΙ/ΟΧΙ ανάλογα με το αν το x υπάρχει στον πίνακα

- Μελετούμε δύο αλγόριθμους:
  - Την σειριακή αναζήτηση που δουλεύει είτε αν ο πίνακας είναι ταξινομημένος είτε όχι
    - Ιδέα: Ψάξε τον πίνακα από αριστερά προς τα δεξιά για το στοιχείο
  - Την δυαδική αναζήτηση που δουλεύει μόνο σε ταξινομημένους πίνακες
    - Ιδέα: Κοιτά το μεσαίο στοιχείο του πίνακα. Αν είναι μεγαλύτερο από το στοιχείο που ψάχνεις, επανέλαβε στο αριστερό κομμάτι του πίνακα. Αν είναι μικρότερο από το στοιχείο που ψάχνεις, επανέλαβε στο δεξί κομμάτι του πίνακα.

Α. Θεωρία
2. Σειριακή Αναζήτηση

- Ο κώδικας κάνει μία διαπέραση του πίνακα συγκρίνοντας το τρέχον στοιχείο με το προς αναζήτηση στοιχείο. Εφόσον το βρει ενημερώνει την κατάλληλη μεταβλητή και τερματίζει.

```
found=0;

for (i=0; i<N; i++)
{
   if (x==PIN[i])
   {
      found=1;
      break;
   }
}
```

Άσκηση 1: «Τρέξτε» ένα παράδειγμα εκτέλεσης με το χέρι, αν PIN=[6,4,2,7,3] , x=7.

Άσκηση 2: Αφού μελετήσετε το πρόγραμμα linear_search.c κατασκευάσετε συνάρτηση που παίρνει ως όρισμα έναν πίνακα ακεραίων και ένα στοιχείο x και επιστρέφει 0/1 ανάλογα με το αν το στοιχείο υπάρχει στον πίνακα.

Α. Θεωρία

3. Δυαδική Αναζήτηση

1. Με Επανάληψη

- Η **Δυαδική Αναζήτηση** (Binary Search) είναι ένας από τους κλασικότερους αλγόριθμους στην Πληροφορική.
  - Έχει μία σημαντική προϋπόθεση: Ότι ο πίνακας είναι ήδη ταξινομημένος!

- Το σκεπτικό της Δυαδικής Αναζήτησης του στοιχείου x σε έναν ταξινομημένο σε αύξουσα σειρά πίνακα:
  - Αν το μεσαίο στοιχείο είναι το x, το στοιχείο βρέθηκε!
  - Αν το x είναι μικρότερο από το μεσαίο στοιχείο τότε ψάχνουμε στο κομμάτι του πίνακα από την αρχή μέχρι το μεσαίο στοιχείο
  - Αν το x είναι μεγαλύτερο από το μεσαίο στοιχείο τότε ψάχνουμε στο κομμάτι του πίνακα από το μεσαίο στοιχείο μέχρι το τέλος

Α. Θεωρία

3. Δυαδική Αναζήτηση

1. Με Επανάληψη

- Το σώμα του αλγορίθμου φαίνεται στον ακόλουθο κώδικα (ο πίνακας έχει δηλωθεί ως int pinakas[N] και οι μεταβλητές start,finish, middle είναι ακέραιες μεταβλητές):

```
start=0;
finish=N-1;
found=FALSE;

while(start<=finish)
{
   middle=(start+finish)/2;

   if (x==pinakas[middle])
   {
      found=TRUE;
      break;
   }
   else if (x<pinakas[middle])
      finish=middle-1;
   else // x>pinakas[middle]
      start=middle+1;
}
```

Α. Θεωρία

3. Δυαδική Αναζήτηση

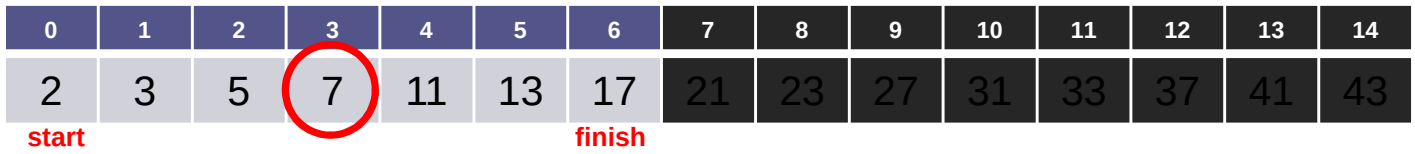
1. Με Επανάληψη

- Εκτελούμε τον αλγόριθμο ψάχνοντας το στοιχείο 11 στον πίνακα:

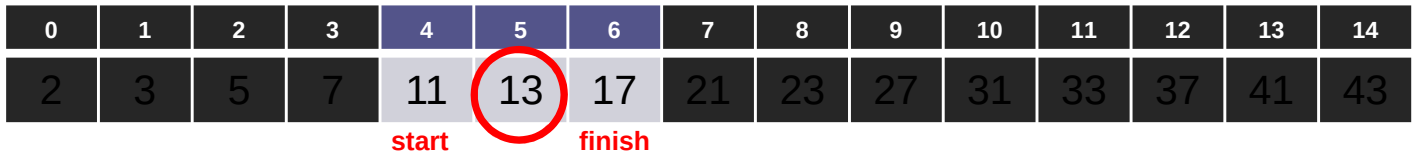
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3	5	7	11	13	17	21	23	27	31	33	37	41	43

- 1η επανάληψη: start=0,finish=14: middle=(0+14) div 2=7. x<A[middle]

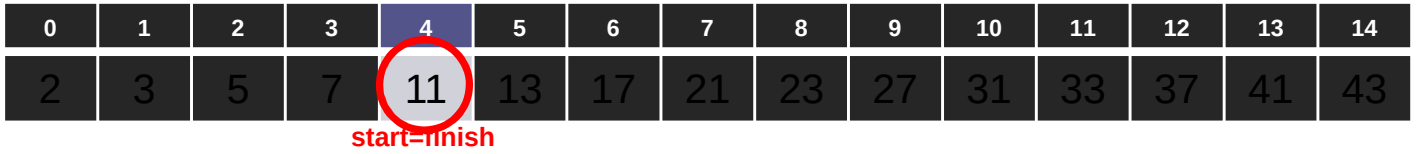
- 2η επανάληψη: start=0,finish=6: middle=(0+6) div 2=3. x>A[middle]



- 3η επανάληψη: start=4,finish=6: middle=(4+6) div 2=5. x<A[middle]



- 4η επανάληψη: start=4,finish=4: middle=(4+6) div 2=5. x==A[middle]



Α. Θεωρία

3. Δυαδική Αναζήτηση

1. Με Επανάληψη

Άσκηση: Αφού μελετήσετε το πρόγραμμα binary_search.c κατασκευάσετε συνάρτηση που παίρνει ως όρισμα έναν πίνακα ακεραίων και ένα στοιχείο x και επιστρέφει 0/1 ανάλογα με το αν το στοιχείο υπάρχει στον πίνακα.

Α. Θεωρία

3. Δυαδική Αναζήτηση

2. Με Αναδρομή

- Η ουσία του αλγορίθμου (κάνε την ίδια διαδικασία σε έναν μικρότερο πίνακα) καθοδηγεί και στο σκεπτικό της αναδρομής

- Θα δούμε εδώ για λόγους πληρότητας και την αναδρομική εκδοχή του αλγορίθμου. Το σκεπτικό είναι το ίδιο, αλλά αντί για επανάληψη κάνουμε αναδρομή.

Σημείωση:

- Κάθε αναδρομικός αλγόριθμος μπορεί να υλοποιηθεί και με επανάληψη. Σε γενικές γραμμές μάλιστα η επανάληψη είναι πιο γρήγορη από την αναδρομή.
- Ωστόσο αν κάποιος κατανοεί το σκεπτικό του σχεδιασμού αναδρομικών αλγορίθμων, τότε έχει ως εργαλείο το σκεπτικό αυτό για την κατασκευή αλγορίθμων.
- Σε γενικές γραμμές στα μαθήματα αυτά θα βλέπουμε (και θα προτιμούμε) επαναληπτικές εκδοχές των αλγορίθμων

Α. Θεωρία

3. Δυαδική Αναζήτηση

2. Με Αναδρομή

- Το σώμα του αλγορίθμου φαίνεται στον ακόλουθο κώδικα (ο πίνακας έχει δηλωθεί ως int pinakas[N] και οι μεταβλητές start,finish, middle είναι ακέραιες μεταβλητές):

```
int binary_search_rec(int *pinakas, int start, int finish, int x)
{
   int middle;

   if (start>finish)
      return FALSE;
   else
   {
      middle=(start+finish)/2;
      if (x==pinakas[middle])
         return TRUE;
      else if (x<pinakas[middle])
         return binary_search_rec(pinakas,start,middle-1,x);
      else
         return binary_search_rec(pinakas,middle+1,finish,x);
   }
}
```

Β. Ασκήσεις

Εφαρμογή 1: Επιστροφή της θέσης στη LinearSearch

- Γράψτε μία συνάρτηση στην γλώσσα C η οποία τροποποιεί την γραμμική αναζήτηση ως εξής:
  - Αντί να επιστρέφεται 0/1 ανάλογα με το αν το στοιχείο υπάρχει στον πίνακα, να επιστρέφει
    - -1 αν το στοιχείο δεν υπάρχει στον πίνακα
    - Την θέση του στοιχείου, αν το στοιχείο υπάρχει στον πίνακα

- Γράψτε μία συνάρτηση main η οποία να αναδεικνύει την συνάρτηση αυτή (να εκτυπώνει κατάλληλο μήνυμα με την θέση του στοιχείου στον πίνακα)

Β. Ασκήσεις

Εφαρμογή 2: Επιστροφή της θέσης στη BinarySearch

- Γράψτε μία συνάρτηση στην γλώσσα C η οποία τροποποιεί την δυαδική αναζήτηση ως εξής:
  - Επιστρέφει 0/1 ανάλογα με το αν το στοιχείο υπάρχει στον πίνακα
  - Μέσω αναφοράς σε μία νέα μεταβλητή επιστρέφει:
    - -1 αν το στοιχείο δεν υπάρχει στον πίνακα
    - Την θέση του στοιχείου, αν το στοιχείο υπάρχει στον πίνακα

- Γράψτε μία συνάρτηση main η οποία να αναδεικνύει την συνάρτηση αυτή (να εκτυπώνει αν βρέθηκε και την θέση του στοιχείου στον πίνακα)

Β. Ασκήσεις

Εφαρμογή 3: Δυαδική Αναζήτηση με Φθίνουσα Σειρά

- Γράψτε μία συνάρτηση - παραλλαγη της δυαδικής αναζήτησης (χρησιμοποιήστε την επαναληπτική εκδοχή) η οποία να δουλεύει για πίνακα ο οποίο είναι ταξινομημένος σε φθίνουσα σειρά

Β. Ασκήσεις

Εφαρμογή 4: Πλήθος Επαναλήψεων των Αλγορίθμων Αναζήτησης

- Γενικά αξιολογούμε έναν αλγόριθμο μετρώντας πόσο γρήγορος είναι. Αν και υπάρχουν κατάλληλα θεωρητικά μοντέλα (υπολογιστική πολυπλοκότητα αλγορίθμου κ.α.) θα προσεγγίσουμε μία σύγκριση των δύο αλγορίθμων αναζήτησης ως εξής:
  - Κατασκευάστε μια παραλλαγή της σειριακής αναζήτησης που θα μετρά το πλήθος των επαναλήψεων (πόσες φορές τρέχει το for).
  - Κατασκευάστε μια παραλλαγή της δυαδικής αναζήτησης που θα μετρά το πλήθος των επαναλήψεων (πόσες φορές τρέχει το while).
  - Συγκρίνετε το πλήθος των επαναλήψεων και αναδείξτε τον νικητή (ο αλγόριθμος που έκανε τις λιγότερες επαναλήψεις)