Αναζήτηση

Δυαδική - Binary

Δυαδική αναζήτηση

- Αναζήτηση ενός στοιχείου σε **ταξινομημένη** λίστα
- Αναζήτηση σε ένα ταξινομημένο σύνολο στοιχείων
  - Τηλεφωνικό κατάλογο
  - Λεξικό

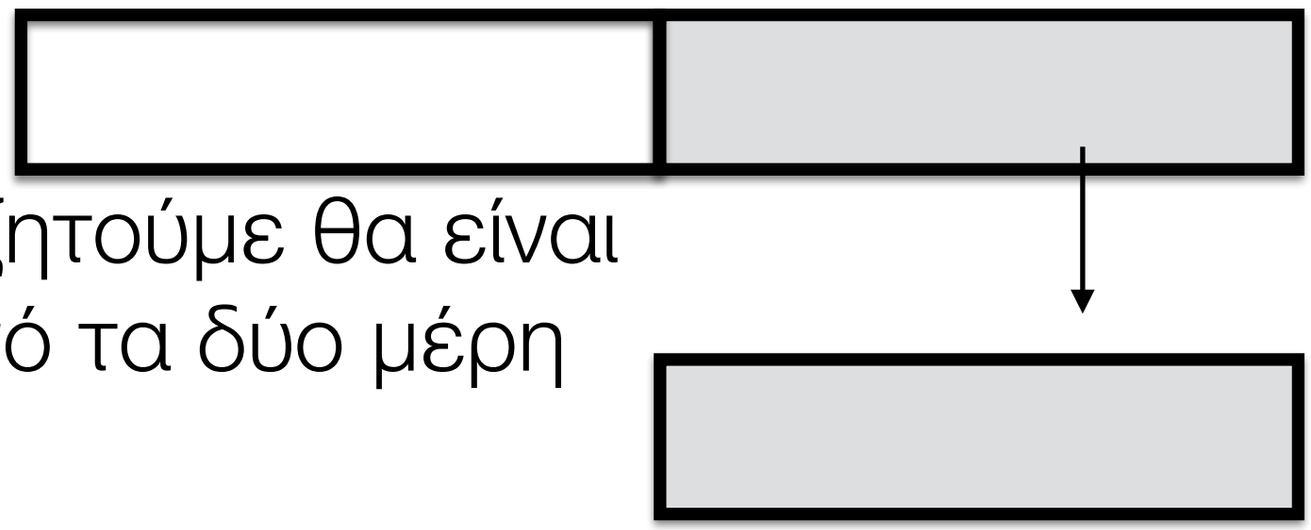
Δυαδική αναζήτηση

- “Ανοίγοντας” το λεξικό σε κάποια σελίδα γνωρίζουμε αν πρέπει να συνεχίσουμε την αναζήτηση πριν ή μετά από τη σελίδα που βρισκόμαστε

- συνεχίζουμε επιλέγοντας μια νέα σελίδα και επειδή το λεξικό είναι ταξινομημένο μπορούμε να αποφασίσουμε αν η λέξη που ψάχνουμε είναι πριν τη νέα σελίδα ή μετά.

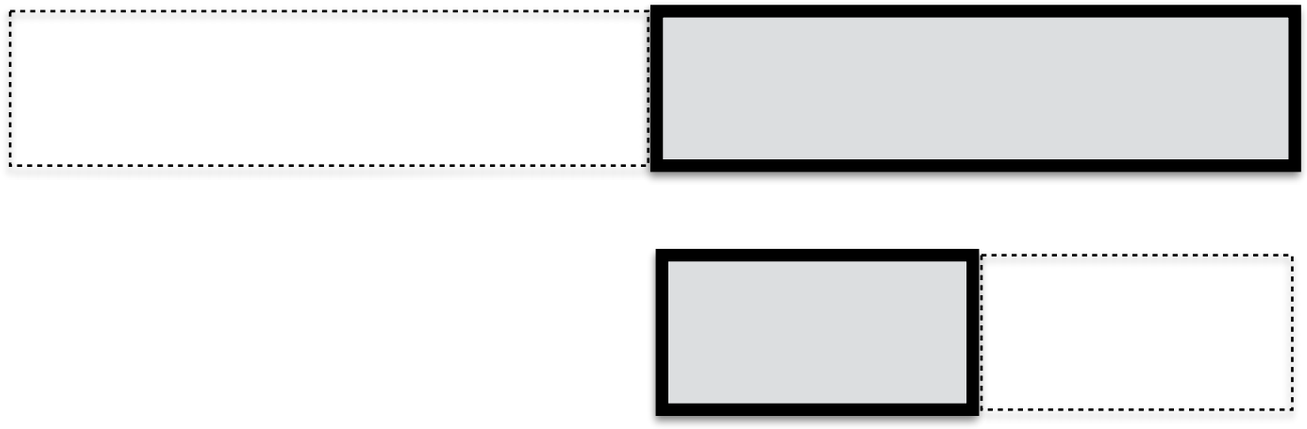
Δυαδική αναζήτηση

- Σπάμε το σύνολο που γίνεται η αναζήτηση σε δύο ίσα μέρη



- Αυτό που ζητούμε θα είναι σε ένα από τα δύο μέρη

Δυαδική αναζήτηση



- Στη συνέχεια διαιρούμε το νέο μέρος σε δύο ίσα μέρη και συνεχίζουμε με αυτό το τρόπο μέχρι μέχρι να βρούμε το ζητούμενο στοιχείο.

- Αυτή η αναζήτηση ονομάζεται **δυαδική** αναζήτηση

Δυαδική αναζήτηση

- Η αναζήτηση ολοκληρώνεται σε συγκεκριμένα βήματα
- Σε ένα βήμα ψάχνουμε λίστα με 2 στοιχεία
- Σε δύο βήματα ψάχνουμε λίστα με 4 στοιχεία
- Σε 3 βήματα ψάχνουμε λίστα με 8 στοιχεία

Δυαδική αναζήτηση

- Γενικότερα σε N βήματα ψάχνουμε 2^N αριθμούς
- Αντίστροφα αν έχουμε N αριθμούς χρειαζόμαστε $\log_2 N$ βήματα

Αριθμοί	2	8	100	1000	10000	100000
Βήματα	1	3	7	10	14	17

Δυαδική Αναζήτηση

Αλγόριθμος

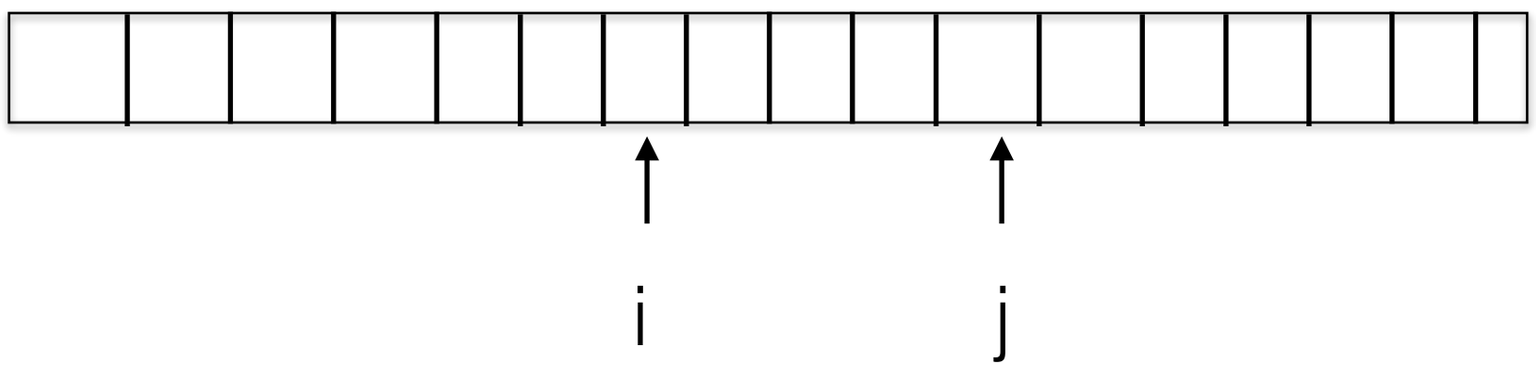
Σε κάθε βήματα της αναζήτησης έχουμε 3 ποσότητες που πρέπει να καθορίσουμε

- τα στοιχεία που είναι μεγαλύτερα από αυτό που ψάχνουμε
- τα στοιχεία που είναι μικρότερα
- και τα στοιχεία που δεν έχουμε ψάξει ακόμα

Δυαδική Αναζήτηση

Αλγόριθμος

- Σε κάθε βήμα θα καθορίζουμε τα άκρα του κομματιού που δεν έχουμε ψάξει ακόμα
- `i` ο “κάτω” δείκτης - εκεί που αρχίζει το άγνωστο κομμάτι
- `j` “άνω” δείκτης - εκεί που τελειώνει το άγνωστο κομμάτι



Δυαδική αναζήτηση

- Στην αρχή της αναζήτησης `i=0` και `j=len(L)-1`
- Η αναζήτηση ολοκληρώνεται όταν το μήκος του άγνωστου μέρους είναι μηδέν
- `i=j+1` (όχι `i=j`, τότε υπάρχει ένα άγνωστο στοιχείο)

Δυαδική αναζήτηση

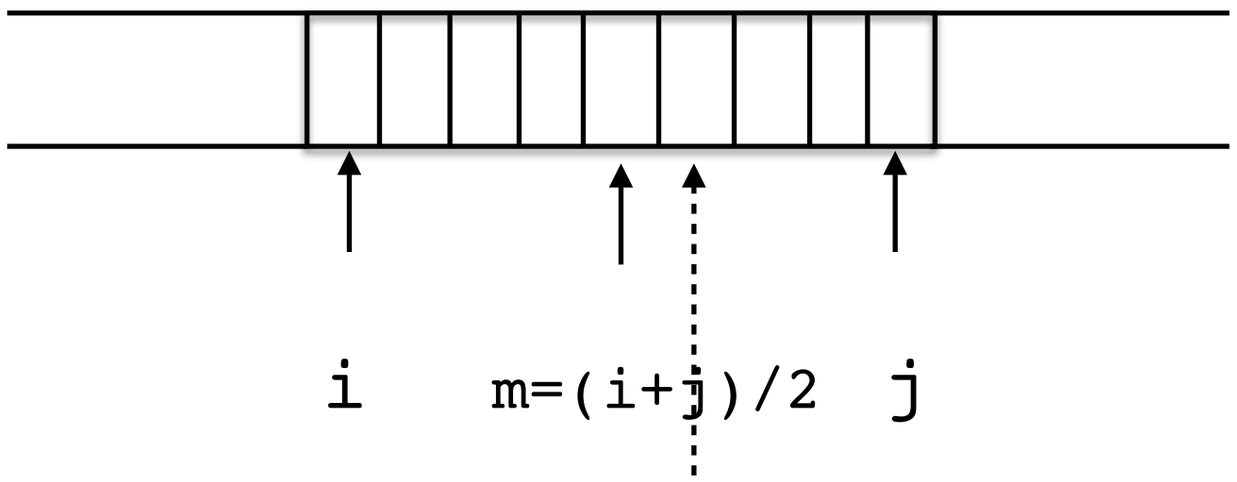
- Για το επόμενο βήμα της αναζήτησης εξετάζουμε το μεσαίο σημείο του άγνωστου μέρους, δηλαδή το στοιχείο που είναι στη θέση `(i+j)/2`

- Αν το στοιχείο στη θέση `(i+j)/2`

    - είναι μικρότερο από το στοιχείο που ψάχνουμε τότε αυξάνουμε το δείκτη `i`

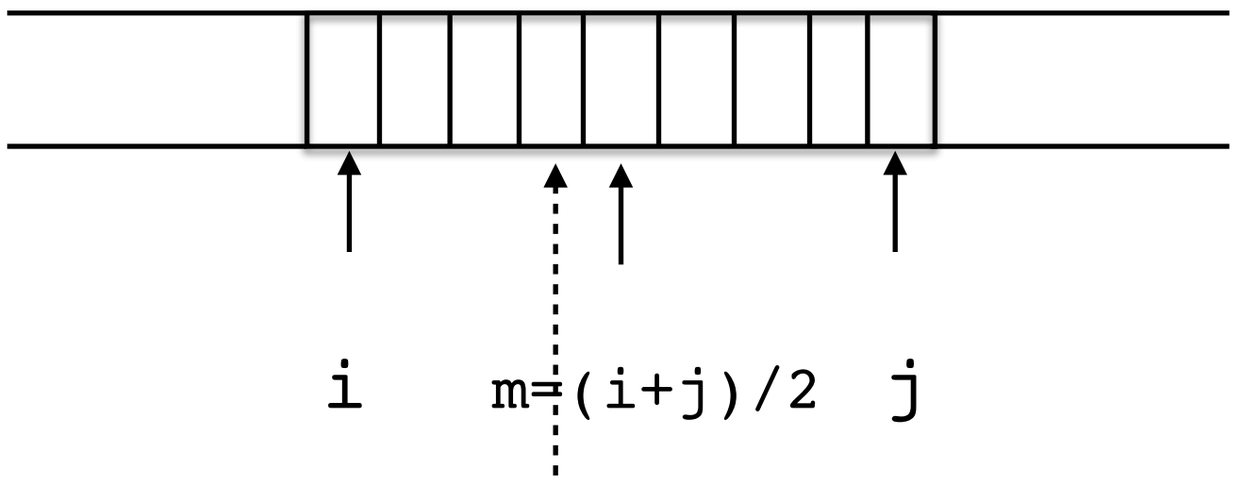
    - είναι μεγαλύτερο τότε ελαττώνουμε το δείκτη `j`

Δυαδική Αναζήτηση



- Αν το στοιχείο στη θέση `(i+j)/2` είναι μικρότερο από εκείνο που ψάχνουμε

Δυαδική Αναζήτηση



νέο `j=(i+j)/2 -1`

- Αν το στοιχείο στη θέση `(i+j)/2` είναι μεγαλύτερο από εκείνο που ψάχνουμε

Δυαδική Αναζήτηση

Παράδειγμα

- L=[3,5,7,14,20] Ψάχνω το 5
- 1ο βήμα: `i=0, j=len(L)-1=4, m=(i+j)/2=(0+4)/2=2`
- `L[2]=7<5` (Ψευδές): μειώνω το `j`: νέο `j=(i+j)/2-1=1`
- 2ο βήμα: `i=0, j=1, m=(0+1)/2=0` (διαίρεση ακεραίων)
- L[0]=3<5 (Αληθές): αυξάνω το `i`: νέο `i=m+1=1`
- `3ο βήμα: i=1, j=1, m=1`
- `L[1]=5 (το βρήκα) επιστρέφω το m`

Δυαδική Αναζήτηση

Παράδειγμα

- L=[3,5,7,14,20] Ψάχνω το 6
- 1ο βήμα: `i=0, j=len(L)-1=4, m=(i+j)/2=(0+4)/2=2`
- `L[2]=7<6` (Ψευδές): μειώνω το `j`: νέο `j=(i+j)/2-1=1`
- 2ο βήμα: `i=0, j=1, m=(0+1)/2=0` (διαίρεση ακεραίων)
- `L[0]=3<6` (Αληθές): αυξάνω το `i`: νέο `i=m+1=1`
- 3ο βήμα: `i=1, j=1, m=1`
- `L[1]=5<6` (Αληθές): αυξάνω το `i`: νέο `i=m+1=2`

Δυαδική Αναζήτηση

Παράδειγμα

- L=[3,5,7,14,20] Ψάχνω το 6
- 1ο βήμα: `i=0, j=len(L)-1=4, m=(i+j)/2=(0+4)/2=2`
- 2ο βήμα: `i=0, j=1, m=(0+1)/2=0` (διαίρεση ακεραίων)
- 3ο βήμα: `i=1, j=1, m=1`
- `L[1]=5<6` (Αληθές): αυξάνω το `i`: νέο `i=m+1=2`
- Ο δείκτης `i` είναι μεγαλύτερος από το `j` επομένως δεν υπάρχει άγνωστο κομμάτι για να κάνω αναζήτηση (`i=2=j+1`)
- Δεν βρήκα το ζητούμενο στοιχείο

Δυαδική Αναζήτηση

```
def binary_search(v,L):
    i=0
    j=len(L)-1
    while i!=j+1:
        m=(i+j)//2
        if L[m]==v:
            return m
        elif L[m]<v:
            i=m+1
        else:
            j=m-1
    return -1
```

Έλεγχος αν υπάρχει άγνωστο μέρος

Επιστρέφω τη θέση που το βρήκα

Επιστρέφω έναν αριθμό που δηλώνει θέση που δεν υπάρχει

Δυαδική αναζήτηση

Χρονομέτρηση

- `L=[i for i in range(10**6)]`

v	10	100000	999900
binary_search()	0.02msec	0.019msec	0.017msec
L.index()	0.006msec	2.056msec	20.41msec