1η Διάλεξη στις Δομές Δεδομένων

Γιάννης Λιβιεράτος

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Μαθηματικών

5 Μαρτίου 2024

Περιεχόμενα

Γλώσσες

1. Γλώσσα Μηχανής (Machine Language)
2. Συμβολική Γλώσσα (Assembly Language)
3. Γλώσσες Προγραμματισμού (Programming Languages)
4. Ψευδογλώσσες
5. Φυσική Γλώσσα

Γλώσσες

1. Γλώσσα Μηχανής (Machine Language)
   - Εντολές σε δυαδικό σύστημα: 000000110101
2. Συμβολική Γλώσσα (Assembly Language)
3. Γλώσσες Προγραμματισμού (Programming Languages)
4. Ψευδογλώσσες
5. Φυσική Γλώσσα

Γλώσσες

1. Γλώσσα Μηχανής (Machine Language)
   - Εντολές σε δυαδικό σύστημα: 000000110101
2. Συμβολική Γλώσσα (Assembly Language)
   - Εντολές με αντιστοιχία στην γλώσσα μηχανής: STO 53
3. Γλώσσες Προγραμματισμού (Programming Languages)
4. Ψευδογλώσσες
5. Φυσική Γλώσσα

Γλώσσες

1. Γλώσσα Μηχανής (Machine Language)
   - Εντολές σε δυαδικό σύστημα: 000000110101
2. Συμβολική Γλώσσα (Assembly Language)
   - Εντολές με αντιστοιχία στην γλώσσα μηχανής: STO 53
3. Γλώσσες Προγραμματισμού (Programming Languages)
   - $int * p = \%x$
4. Ψευδογλώσσες
5. Φυσική Γλώσσα

Γλώσσες

1. Γλώσσα Μηχανής (Machine Language)
   - Εντολές σε δυαδικό σύστημα: 000000110101
2. Συμβολική Γλώσσα (Assembly Language)
   - Εντολές με αντιστοιχία στην γλώσσα μηχανής: STO 53
3. Γλώσσες Προγραμματισμού (Programming Languages)
   - $int * p = \%x$
4. Ψευδογλώσσες
   - $x \leftarrow f(y + z)$
5. Φυσική Γλώσσα

Γλώσσες

1. Γλώσσα Μηχανής (Machine Language)
   - Εντολές σε δυαδικό σύστημα: 000000110101
2. Συμβολική Γλώσσα (Assembly Language)
   - Εντολές με αντιστοιχία στην γλώσσα μηχανής: STO 53
3. Γλώσσες Προγραμματισμού (Programming Languages)
   - $int * p = \%x$
4. Ψευδογλώσσες
   - $x \leftarrow f(y + z)$
5. Φυσική Γλώσσα
   - Αποθήκευσε το μέγεθος του πίνακα εισόδου στην μεταβλητή x

Ψευδογλώσσα

- Είσοδος/έξοδος: $\textsc{Alg}(arg_1, arg_2, \ldots, arg_n)$, $\textsc{Return}$
- Εκχώρηση τιμής: $\leftarrow$, $=$
- Σχόλια: $\%$, $\#$, $//$
- Μεταβλητές, σταθερές: γράμματα και λέξεις του ελληνικού ή αγγλικού αλφαβήτου
- Αριθμητικοί τελεστές: $+, -, \cdot, *, /, \div,$ $\mod$
- Τελεστές σύγκρισης: $<, >, \leq, \geq, =, ==$
- Λογικοί τελεστές: $\wedge$, **and**, $\vee$, **or**, $\neg$, **not**, **!**
- Έλεγχος ροής: **if – else if / elif – else**, **while**, **for**

Πίνακες

$$A = [1, 5, -2, 5, 6, 6, 3, 1, 7],\ \textsc{length}(A) = 9,\ A[7] = 3$$

$$B = \begin{bmatrix} 5 & -7 & 1 \\ 6 & 2 & -3 \\ 0 & -1 & -7 \\ 0 & 1 & 0, \end{bmatrix},\ \textsc{size}(B, 1) = 4,\ \textsc{size}(B, 2) = 3, B[2, 3] = -3.$$

Αναζήτηση στοιχείου σε ταξινομημένη λίστα

BinSearch(list,key) # list μονοδιάστατος πίνακας σε αύξουσα # σειρά, key το ζητούμενο στοιχείο, $\text{length}(list) = 2^n$, $n \in \mathbb{N}$

```
1:  if length(list) = 1 and ¬list[1] = key then
2:      return notFound
3:  else if list[1] = key then
4:      return found
5:  else
6:      mid = length(list)/2
7:      if list[mid] = key then
8:          return found
9:      else if list[mid] < key then
10:         BinSearch(list[mid + 1 : length(list)], key)
11:     else
12:         BinSearch(list[1 : list(mid)], key)
13:     end if
14: end if
```

Αναζήτηση στοιχείου σε ταξινομημένη λίστα

Κάθε κλήση της BinSearch(L, key) θέλει το πολύ:

- 5 συγκρίσεις ($2x$line 1, lines 3, 7, 9),
- 1 εκχώρηση τιμής σε μεταβλητή (line 6),
- 1 επιστροφή τιμής (lines 2 ή 4 ή 8),

Αναζήτηση στοιχείου σε ταξινομημένη λίστα

Κάθε κλήση της BinSearch(L, key) θέλει το πολύ:

- 5 συγκρίσεις ($2x$line 1, lines 3, 7, 9),
- 1 εκχώρηση τιμής σε μεταβλητή (line 6),
- 1 επιστροφή τιμής (lines 2 ή 4 ή 8),

Θα γίνουν το πολύ $\log_2(\text{Length}(L))$ κλήσεις της BinSearch(L, key), με $\text{Length}(L) \leq 2^n$.
Συνολικά: το πολύ $7 \cdot \log_2(2^n) = 7n$ βήματα, για είσοδο μεγέθους 2^n.

Κώδικας σε python

```
def BinSearch(array,key):
    if len(array) == 1 and array[0] != key:
        print(False)
        return False
    elif array[0] == key:
        print(False)
        return False
    else:
        mid = len(array)//2
        if array[mid] == key:
            print(True)
            return True
        elif array[mid] < key:
            BinSearch(array[mid+1:],key)
        else:
            BinSearch(array[:mid],key)
```

Εύρεση μεγίστου και ελαχίστου σε λίστα

MaxMin(list) # list μονοδιάστατος πίνακας

Εύρεση μεγίστου και ελαχίστου σε λίστα

MaxMin(list) # list μονοδιάστατος πίνακας

```
1:  max=min=list[1], argmax=argmin=i=1
2:  while i <= length(list) do
3:     if max<list[i] then
4:        max = list[i], argmax = i
5:     else if min > list[i] then
6:        min = list[i], argmin = i
7:     end if
8:     i = i+1
9:  end while
10: return  max, argmax, min, argmin
```

Εύρεση μεγίστου και ελαχίστου σε λίστα

Έχουμε 5 εκχωρήσεις τιμής στην 1η γραμμή του κώδικα, και 4 επιστροφές στην 10η.

Σε κάθε μία από τις n επαναλήψεις των γραμμών $2 - -9$, όπου n το μήκος της list, γίνονται το πολύ 2 συγκρίσεις (lines $3, 5$) και 3 εκχωρήσεις τιμής (lines 4 ή 6, line 8).

Συνολικά: $5n + 9$ βήματα.

Αναζήτηση στοιχείου σε δισδιάστατο πίνακα

SEARCH(array,key) # array δισδιάστατος πίνακας, key το
ζητούμενο στοιχείο.

```
1:  Found = 0
2:  for i = 1, ..., SIZE(array, 1) do
3:      for j = 1, ..., SIZE(array, 2) do
4:          if array[i, j] = key then
5:              Found = 1
6:              BREAK
7:          end if
8:      end for
9:  end for
10: if Found = 1 then
11:     return found
12: else
13:     return notFound
14: end if
```

Αναζήτηση στοιχείου σε δισδιάστατο πίνακα

Έχουμε 1 εκχώριση τιμής στην 1η γραμμή του κώδικα, μία σύγκριση (line 10) και μία επιστροφή τιμής (lines 11 ή 13).

Σε κάθε μία από τις το πολύ nm επαναλήψεις των γραμμών 2–9, όπου $n \times m$ η διάσταση του array, έχουμε 1 σύγκριση (line 4) και το πολύ μία εκχώρηση τιμής (line 5).

Συνολικά: $2nm + 3$ βήματα.

Κώδικας σε C

```
#include <stdio.h>
int main(){
 int key, array[2][3];
 scanf("%d",&key);
 for(int i=0;i<2;i++)
  for(int j=0;j<3;j++)
   scanf("%d",&array[i][j]);
 int found = 0;
 for(int i=0;i<2;i++)
  for(int j=0;j<3;j++)
   if(array[i][j]==key)
    found = 1;
printf("%d\n",found);
return 0;
}
```

Περιεχόμενα

Κριτήρια απόδοσης

- Ταχύτητα εκτέλεσης (χρόνος)
- Χρήση μνήμης (χώρος)
- Πλήθος επεξεργαστών
- Κατανάλωση ρεύματος

Κριτήρια απόδοσης

- Ταχύτητα εκτέλεσης (χρόνος)
- Χρήση μνήμης (χώρος)
- Πλήθος επεξεργαστών
- Κατανάλωση ρεύματος

Κριτήρια απόδοσης

- Ταχύτητα εκτέλεσης (χρόνος)
- Χρήση μνήμης (χώρος)
- Πλήθος επεξεργαστών
- Κατανάλωση ρεύματος

★ Γενικές παραδοχές:

- Κάθε είσοδος έχει μια κατάλληλη κωδικοποίηση (σε σειρά δυαδικών ψηφίων).
- Το μέγεθος της εισόδου, έστω n, είναι το μήκος της κωδικοποίησης.
- Στην περίπτωση των πινάκων, το μέγεθός τους είναι το πλήθος των θέσεών τους.
- Μας ενδιαφέρει η χρονική πολυπλοκότητα για $n \to \infty$.

Χρονική πολυπλοκότητα χειρότερης περίπτωσης

$$T(n) := \max\{t(I) \in \mathbb{N} \mid I \in D_n\},\ D_n \text{ σύνολο εισόδων μήκους } n$$

Χρονική πολυπλοκότητα χειρότερης περίπτωσης

$T(n) := \max\{t(I) \in \mathbb{N} \mid I \in D_n\}$, D_n σύνολο εισόδων μήκους n

- Αναζήτηση στοιχείου σε ταξινομημένη λίστα: $B(n) = 7 \cdot \log_2(n)$
- Εύρεση μεγίστου και ελαχίστου σε λίστα: $M(n) = 5n + 9$
- Αναζήτηση στοιχείου σε $n \times m$ πίνακα: $S(nm) = 2nm + 3$

Ποιος είναι πιο αποδοτικός αλγόριθμος;

Χρονική πολυπλοκότητα χειρότερης περίπτωσης

$T(n) := \max\{t(I) \in \mathbb{N} \mid I \in D_n\}$, D_n σύνολο εισόδων μήκους n

- Αναζήτηση στοιχείου σε ταξινομημένη λίστα: $B(n) = 7 \cdot \log_2(n)$
- Εύρεση μεγίστου και ελαχίστου σε λίστα: $M(n) = 5n + 9$
- Αναζήτηση στοιχείου σε $n \times m$ πίνακα: $S(nm) = 2nm + 3$

Ποιος είναι πιο αποδοτικός αλγόριθμος;

Έστω δισδιάστατος πίνακας μεγέθους $\sqrt{n} \times \sqrt{n}$. Τότε $S(n) = 2n + 3$.

Χρονική πολυπλοκότητα χειρότερης περίπτωσης

$T(n) := \max\{t(I) \in \mathbb{N} \mid I \in D_n\}$, D_n σύνολο εισόδων μήκους n

- Αναζήτηση στοιχείου σε ταξινομημένη λίστα: $B(n) = 7 \cdot \log_2(n)$
- Εύρεση μεγίστου και ελαχίστου σε λίστα: $M(n) = 5n + 9$
- Αναζήτηση στοιχείου σε $n \times m$ πίνακα: $S(nm) = 2nm + 3$

Ποιος είναι πιο αποδοτικός αλγόριθμος;

Έστω δισδιάστατος πίνακας μεγέθους $\sqrt{n} \times \sqrt{n}$. Τότε $S(n) = 2n + 3$.

$$\lim_{n\to\infty} \frac{B(n)}{M(n)} = \lim_{n\to\infty} \frac{7\log_2(n)}{5n+9} =$$

$$\lim_{n\to\infty} \frac{M(n)}{S(n)} = \lim_{n\to\infty} \frac{5n+9}{2n+3} = \frac{5}{2}$$

Ασυμπτωτικός Συμβολισμός

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$. $f(n) = O(g(n))$ αν και μόνον αν

$$\exists c \in \mathbb{R}_{\geq 0}, \exists n_0 \in \mathbb{N} : f(n) \leq c \cdot g(n), \forall n \geq n_0.$$

Ασυμπτωτικός Συμβολισμός

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$. $f(n) = O(g(n))$ αν και μόνον αν

$$\exists c \in \mathbb{R}_{\geq 0}, \exists n_0 \in \mathbb{N} : f(n) \leq c \cdot g(n), \forall n \geq n_0.$$

Παραδείγματα:

1. Αν $f(n) \leq g(n)$, για κάθε $n \in \mathbb{N}$, τότε $f(n) = O(g(n))$:

Ασυμπτωτικός Συμβολισμός

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$. $f(n) = O(g(n))$ αν και μόνον αν

$$\exists c \in \mathbb{R}_{\geq 0}, \exists n_0 \in \mathbb{N} : f(n) \leq c \cdot g(n), \forall n \geq n_0.$$

Παραδείγματα:

1. Αν $f(n) \leq g(n)$, για κάθε $n \in \mathbb{N}$, τότε $f(n) = O(g(n))$: $c = 1$, $n_0 = 0$.
2. $n + 1 = O(n)$:

Ασυμπτωτικός Συμβολισμός

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$. $f(n) = O(g(n))$ αν και μόνον αν

$$\exists c \in \mathbb{R}_{\geq 0}, \exists n_0 \in \mathbb{N} : f(n) \leq c \cdot g(n), \forall n \geq n_0.$$

Παραδείγματα:

1. Αν $f(n) \leq g(n)$, για κάθε $n \in \mathbb{N}$, τότε $f(n) = O(g(n))$: $c = 1$, $n_0 = 0$.
2. $n + 1 = O(n)$: $c = 2, n_0 = 1$.
3. $7 \log_2(n) = O(5n + 9)$:

Ασυμπτωτικός Συμβολισμός

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$. $f(n) = O(g(n))$ αν και μόνον αν

$$\exists c \in \mathbb{R}_{\geq 0}, \exists n_0 \in \mathbb{N} : f(n) \leq c \cdot g(n), \forall n \geq n_0.$$

Παραδείγματα:

1. Αν $f(n) \leq g(n)$, για κάθε $n \in \mathbb{N}$, τότε $f(n) = O(g(n))$: $c = 1$, $n_0 = 0$.
2. $n + 1 = O(n)$: $c = 2, n_0 = 1$.
3. $7 \log_2(n) = O(5n + 9)$: Έστω $c > 0$ με:

$$7 \log_2(n) \leq c(5n + 9)$$

$$c \geq \frac{7 \log_2(n)}{(5n + 9)} \to \frac{7}{5n \log_1 0(2)} \to 0$$

Παραδείγματα

4. $\frac{n^2}{2} + 4 \neq O(5n + 9)$:

Παραδείγματα

4. $\frac{n^2}{2} + 4 \neq O(5n + 9)$: Έστω $c > 0$ με:

$$\frac{n^2}{2} + 4 \leq c(5n + 9)$$

$$c \geq \frac{\frac{n^2}{2} + 4}{(5n + 9)} \to \frac{n}{5} \to +\infty$$

5. $\log_a(n) = O(\log_b(n))$, για κάθε a, b:

Παραδείγματα

4. $\frac{n^2}{2} + 4 \neq O(5n + 9)$:Εστω $c > 0$ με:

$$\frac{n^2}{2} + 4 \leq c(5n + 9)$$

$$c \geq \frac{\frac{n^2}{2} + 4}{(5n + 9)} \to \frac{n}{5} \to +\infty$$

5. $\log_a(n) = O(\log_b(n))$, για κάθε a, b:

$$\log_a(n) = \frac{\log_b(n)}{\log_a(b)} = c \cdot \log_b(n), \ c = \log_a(b)^{-1}.$$

6. $16n^{37} + 24n^{19} - 8n^5 + 23n^2 + 18 = O(n^{37})$:

Παραδείγματα

4. $\frac{n^2}{2} + 4 \neq O(5n+9)$:Εστω $c > 0$ με:

$$\frac{n^2}{2} + 4 \leq c(5n+9)$$

$$c \geq \frac{\frac{n^2}{2} + 4}{(5n+9)} \to \frac{n}{5} \to +\infty$$

5. $\log_a(n) = O(\log_b(n))$, για κάθε a, b:

$$\log_a(n) = \frac{\log_b(n)}{\log_a(b)} = c \cdot \log_b(n), \ c = \log_a(b)^{-1}.$$

6. $16n^{37} + 24n^{19} - 8n^5 + 23n^2 + 18 = O(n^{37})$:

$$16n^{37} + 24n^{19} - 8n^5 + 23n^2 + 18 \leq$$
$$16n^{37} + 24n^{37} + 23n^{37} + 18n^{37} =$$
$$81n^{37}.$$

$c = 81$, $n_0 = 1$.

Βασικές Τάξεις Μεγέθους

- Σταθερός χρόνος: $O(1)$
- Πολυλογαριθμικός χρόνος: $O(\log^k(n))$
- Γραμμικός χρόνος: $O(n)$
- Πολυωνυμικός χρόνος: $O(n^k)$
- Εκθετικός χρόνος: $O(2^{n^k})$