ΕΡΓΑΣΙΑ 1

Μαρία Σχοινάκη: 3210191

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF INFORMATICS

ΑΛΓΟΡΙΘΜΟΙ

Διδάσκουσα : Αλκμήνη Σγουρίτσα

1η Σειρά Ασκήσεων - Εαρινό Εξάμηνο 2023

Ονοματεπώνυμο: ***Μαρία Σχοινάκη***

Αριθμός Μητρώου: ***3210191***

Email: ***p3210191@aueb.gr***

Ασκήσεις

Άσκηση 1

1) $\sqrt{\log_9 n}$
2) $\log\sqrt{n}$
3) $2^{\log n/2}$
4) $\log(4^n)$
5) $\log(n^n)$
6) $(\log(2^n))^2$
7) $8^{\log n}$
8) $n^{\log n}$
9) 2^n
10) $2^{n\log n}$
11) $n!$

Μαρία Σχοινάκη: 3210191

Άσκηση 2

a) $n \cdot 2^n = O(3^n)$

Δηλαδή πρέπει να αποδείξουμε ότι για κάθε n: **$n \cdot 2^n \leq 3^n$**
Επαγωγική βάση: Για n=1 → $1 \cdot 2^1 \leq 3^1$ → $2 \leq 3$, που ισχύει, για n=2 → $2 \cdot 2^2 \leq 3^2$ → $8 \leq 9$, που ισχύει
Επαγωγική υπόθεση: Έστω ότι ισχύει για κάποιο $k > 2$ → $k \cdot 2^k \leq 3^k$
Επαγωγικό βήμα: Πρέπει να δείξουμε πως $(k+1) \cdot 2^{k+1} \leq 3^{k+1}$

Απόδειξη
$(k+1) \cdot 2^{k+1} \leq 3^{k+1}$ → $(k+1) \cdot 2^k \cdot 2 \leq 3^{k+1}$ → $k \cdot 2^k \cdot 2 + 2^k \cdot 2 \leq 3^{k+1}$ **(1)**
Από την επαγωγική υπόθεση ισχύει ότι: $k \cdot 2^k \leq 3^k$ → $2 \cdot k \cdot 2^k \leq 3^k \cdot 2$ **(2)**

Αφαιρώ τις **(1), (2)** κατά μέλη:
$k \cdot 2^k \cdot 2 + 2^k \cdot 2 \leq 3^{k+1}$
$2 \cdot k \cdot 2^k \leq 3^k \cdot 2$ –
$2 \cdot 2^k \leq 3^{k+1} - 3^k \cdot 2$
→ $2 \cdot 2^k \leq 3^k \cdot 3 - 3^k \cdot 2$
→ $2 \cdot 2^k \leq 3^k (3 - 2)$
→ $2 \cdot 2^k \leq 3^k$
→ $(\frac{3}{2})^k \geq 2$ το οποίο ισχύει για κάθε $k > 2$

b) $n^2 \cdot 2^n + 10 = O(3^n)$

Δηλαδή πρέπει να αποδείξουμε ότι για κάθε n: **$n^2 \cdot 2^n + 10 \leq 3^n$**
Επαγωγική βάση: Για n=1 → $1^2 \cdot 2^1 + 10 \leq c*3^1$ → $12 \leq c*3$, που ισχύει για κάθε c >= 4
για n=2 → $2^2 \cdot 2^2 + 10 \leq c*3^2$ → $16 \leq c*9$, που ισχύει για κάθε c >= 4
Επαγωγική υπόθεση: Έστω ότι ισχύει για κάποιο $k > 2$ → $k^2 \cdot 2^k + 10 \leq 3^k$
Επαγωγικό βήμα: Πρέπει να δείξουμε πως $(k+1)^2 \cdot 2^{k+1} + 10 \leq 3^{k+1}$

Απόδειξη
$(k+1)^2 \cdot 2^{k+1} + 10 \leq 3^{k+1}$ → $(k+1)^2 \cdot 2^k \cdot 2 + 10 \leq 3^{k+1}$ → $k^2 \cdot 2^k \cdot 2 + 4 \cdot k \cdot 2^k + 2^k \cdot 2 + 10 \leq 3^{k+1}$
→ $k^2 \cdot 2^k \cdot 2 + 4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^{k+1} - 10$ → $k^2 \cdot 2^k \cdot 2 + 4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^{k+1}$ **(1),**
αφού αν x <= y -10 τότε και x<= y
Από την επαγωγική υπόθεση ισχύει ότι: $k^2 \cdot 2^k + 10 \leq 3^k$ → $2 \cdot k^2 \cdot 2^k + 20 \leq 3^k \cdot 2$
→ $2 \cdot k^2 \cdot 2^k \leq 3^k \cdot 2 - 20$ → $2 \cdot k^2 \cdot 2^k \leq 3^k \cdot 2$ **(2)**
αφού αν x <= y -20 τότε και x<= y

Αφαιρώ τις **(1), (2)** κατά μέλη:
$k^2 \cdot 2^k \cdot 2 + 4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^{k+1}$
$2 \cdot k^2 \cdot 2^k \leq 3^k \cdot 2$ –
$4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^{k+1} - 3^k \cdot 2$
→ $4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^k \cdot 3 - 3^k \cdot 2$
→ $4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^k (3 - 2)$
→ $4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^k$ **(3)**

Στο προηγούμενο ερώτημα αποδείξαμε ότι $k \cdot 2^k \leq 3^k$ → $4 \cdot k \cdot 2^k \leq 4 \cdot 3^k$ **(4)**

Μαρία Σχοινάκη: 3210191

Αφαιρώ τις **(3)**, **(4)** κατά μέλη:
$4 \cdot k \cdot 2^k \leq 4 \cdot 3^k$
$4 \cdot k \cdot 2^k + 2^k \cdot 2 \leq 3^k \quad -$
$- 2^k \cdot 2 \leq 4 \cdot 3^k - 3^k$
$\rightarrow - 2^k \cdot 2 \leq 3 \cdot 3^k$
$\rightarrow - 2^k \cdot 2 \leq 3 \cdot 3^k$
$\rightarrow - 2^{k+1} \leq 3^{k+1}$
Το οποίο ισχύει για κάθε $k > 2$

Άσκηση 3

a)

Για τον υπολογισμό του χρόνου από το Master Theorem έχουμε
• a = 2 καθώς μελετάμε 2 υποπροβλήματα
• b = 2 καθώς κόβουμε τον αρχικό πρόβλημα στην μέση άρα κάθε υποπρόβλημα έχει το μισό μέγεθος
• d = 1 καθώς θέλουμε O(n) για τα prints
Οπότε γράφουμε την αναδρομική σχέση: **T(n) = 2T(n/2) + O(n)** και έχουμε logb (a) = $\log_2(2) = 1 = d$ άρα τελικά από τη 2η περίπτωση του Master Theorem, δηλαδή
T(n) = O(nlogn) = Θ(nlogn)
Ας προσπαθήσουμε να ελέγξουμε την απάντηση μας. Τρέχοντας τον κώδικα σε python πήραμε τα εξής αποτελέσματα.

n	#prints
1	1
2	4
4	12
8	32
16	80

Με μια εύκολη παρατήρηση βλέπουμε ότι για n είσοδο έχουμε (logn + 1)n αποτελέσματα, δηλαδή nlogn + n το οποίο είναι O(nlogn) + O(n) = O(nlogn) = Θ(nlogn)

b)

Για τον υπολογισμό του χρόνου από το Master Theorem έχουμε
• a = 2 καθώς μελετάμε 2 υποπροβλήματα
• b = 2 καθώς κόβουμε τον αρχικό πρόβλημα στην μέση άρα κάθε υποπρόβλημα έχει το μισό μέγεθος
• d = 0 καθώς θέλουμε O(1) για τα prints, καθώς εκτυπώνουμε μόνο 1 αριθμό κάθε φορά, το 1

Μαρία Σχοινάκη: 3210191

Οπότε γράφουμε την αναδρομική σχέση: **T(n) = 2T(n/2) + O(1)** και έχουμε logb (a) = $\log_2$ (2) = 1 > d άρα τελικά από τη 3η περίπτωση του Master Theorem, δηλαδή
T(n) = O(n^1) = Θ(n^1)
Ας προσπαθήσουμε να ελέγξουμε την απάντηση μας. Τρέχοντας τον κώδικα σε python πήραμε τα εξής αποτελέσματα.

n	#prints 1
1	1
2	3
4	7
8	15
16	31

Με μια εύκολη παρατήρηση βλέπουμε ότι για n είσοδο έχουμε 2n - 1 αποτελέσματα, το οποίο είναι O(n) = Θ(n)

Άσκηση 4

a)

Έστω λοιπόν το n άρτιος. Για να είναι το x πλειοψηφικό στοιχείο του πίνακα A πρέπει #x > n/2(π.χ αν n = 6, #x>=4). Γενικά, ασχέτως αν το x είναι πλειοψηφικό, στην καλύτερη περίπτωση να περάσουν n/2 στοιχεία στον πίνακα B και αυτό μόνο αν ανά 2 τα στοιχεία είναι ίδια(π.χ [x, x, y, y, z, z]). Τώρα δεδομένου ότι το x είναι πλειοψηφικό στον A, αυτό σημαίνει ότι θα έχουμε **υποχρεωτικά** τουλάχιστον 1 ζευγαράκι x, x. Εφόσον έχουμε n/2 ζευγαράκια και n/2 + 1 τουλάχιστον x, τότε ακόμα και κάθε ζευγαράκι να είχε 1 x τότε υποχρεωτικά θα υπάρξει 1 ζευγαράκι που θα έχει 2 x. Έστω λοιπόν ότι έχουμε μόνο ένα ζευγαράκι που έχει 2 x . θα μας μείνουν n/2 – 1 x και n/2 -1 άλλα στοιχεία διάφορα του x. Αναγκαστικά θα πρέπει κάθε x να πάει με κάθε μη x και στον επόμενο γύρο δεν θα περάσει κανένα στοιχείο, παρά μόνο το x από το ζευγαράκι x, x. Αν τώρα έχουμε τόσα ζευγαράκια όσα μπορούμε με 2x επειδή έχουμε n/2 + 1 x και n/2 – 1 διάφορα του x, τα ζευγαράκια με τα x θα είναι περισσότερα από αυτά με κάτι άλλο εκτός του x , άρα πάλι στον πίνακα B θα περάσουν περισσότερα x, άρα το x θα είναι πλειοψηφικό. Αποδεικνύοντας την καλύτερη περίπτωση(να υπάρχει μόνο ένα ζευγαράκι x, x) και την χειρότερη περίπτωση(να υπάρχουν περισσότερα ζευγαράκια x, x), αποδεικνύουμε ότι ισχύει για κάθε συνδυασμό. **Ομοίως** μπορούμε να αποδείξουμε και αν ο πίνακας παρουσιάζει περισσότερα x, δηλαδή n/2 + 2, n/2 + 3, …. n.(η περίπτωση που αποδείξαμε ήταν η χειρότερη)

Μαρία Σχοινάκη: 3210191

Παράδειγμα για n = 6

Το x είναι πλειοψηφικό άρα πρέπει **x ≥ 4**
Χωρίς βλάβη της γενικότητας ας υποθέσουμε ότι όλα τα στοιχεία που δεν είναι x είναι y(*θα μπορούσαν να ναι όλα διαφορετικά π.χ y, z, k*)

1) x = 6 [x,x,x,x,x,x] →[x,x,x]
2) x = 5 [x,x,x,x,x,y]
[x,x,x,x,y,x]
[x,x,x,y,x,x] →[x,x]
[x,x,y,x,x,x]
[x,y,x,x,x,x]
[y,x,x,x,x,x]

3) x = 4 [x,x,x,x,y,y] →[x,x,y]
[x,x,x,y,x,y] →[x]
.
.
.

b)

Έστω λοιπόν το n περιττός. Για να είναι το x πλειοψηφικό στοιχείο του πίνακα A πρέπει #x > n/2(π.χ αν n = 7, #x>=4). Εφόσον είμαστε στην περίπτωση που το περισσευόμενο στοιχείο **δεν** είναι πλειοψηφικό, αυτό σημαίνει ότι αυτό το στοιχείο **δεν** είναι το x. Άρα στην ουσία έχουμε πάλι ένα άρτιο πλήθος στοιχείων εκ των οποίων το x είναι πλειοψηφικό. Η απόδειξη παρουσιάστηκε στο **a** ερώτημα.

Παράδειγμα για n = 7

Το x είναι πλειοψηφικό άρα πρέπει **x ≥ 4**
Χωρίς βλάβη της γενικότητας ας υποθέσουμε ότι όλα τα στοιχεία που δεν είναι x είναι y(*θα μπορούσαν να ναι όλα διαφορετικά π.χ y, z, k*)

1) x = 6 [x,x,x,x,x,x,y] →[x,x,x]
2) x = 5 [x,x,x,x,x,y,y]
[x,x,x,x,y,x,y]
[x,x,x,y,x,x,y] →[x,x]
[x,x,y,x,x,x,y]
[x,y,x,x,x,x,y]
[y,x,x,x,x,x,y]

3) x = 4 [x,x,x,x,y,y,y] →[x,x,y]
[x,x,x,y,x,y,y] →[x]
.
.
.

ΕΡΓΑΣΙΑ 1

Μαρία Σχοινάκη: 3210191

c)

ΚΩΔΙΚΑΣ

```
Majority Element(A)    //A[1…n]
  n = len(A)
  if n = 1 then
     return A[1]
  end if

  if n = 0 then
     return
  end if

  if n mod 2 = 1 then
      if count(A, A[n]) > n/2
            return A[n]
      end if
  end if

  B=[]
  for i = 1 to n-1 step = 2 do
     if A[i] = A[i+1]
           B += A[i]
     end if
  end for

  return Majority Element(B)
```

Στην ουσία ο αλγόριθμος ελέγχει αρχικά αν το πλήθος των στοιχείων είναι περιττό. Αν είναι τότε ελέγχουμε αν το τελευταίο στοιχείο του πίνακα είναι πλειοψηφικό. Ο έλεγχος γίνεται με την βοήθεια της count και αν το αποτέλεσμα είναι > n/2 τότε έχουμε πλειοψηφικό. Αλλιώς, έχουμε εκτελέσει σε μια for την περιγραφή που μας δίνεται, δηλαδή ελέγχουμε την ισότητα στα στοιχεία ανά 2 και αν ισχύει η ισότητα βάζουμε το στοιχείο στον πίνακα B. Με την χρήση της αναδρομής, ξανακαλούμε την συνάρτηση για τον πίνακα B και γίνεται η ίδια διαδικασία. Αν τώρα ο B είναι μονοσύνολο, επιστρέφουμε το μοναδικό του στοιχείο, ενώ αν είναι άδειος, δηλαδή αν από το προηγούμενο βήμα τα ζευγαράκια ήταν όλα διαφορετικά μεταξύ τους, τότε δεν έχουμε πλειοψηφικό στοιχείο και άρα επιστρέφει το «τίποτα».

d)

Για τον υπολογισμό του χρόνου από το Master Theorem έχουμε
• a = 1 καθώς μελετάμε 1 υποπρόβλημα
• b = 2 καθώς κόβουμε τον αρχικό πρόβλημα στην μέση στην χειρότερη περίπτωση, δηλαδή ο B να έχει n/2 στοιχεία
• d = 1 καθώς θέλουμε O(n) για την count, και O(1) για τις συγκρίσεις και τα return
Οπότε γράφουμε την αναδρομική σχέση: **T(n) = 1T(n/2) + O(n)** και έχουμε logb (a) = $\log_2(1) = 0 < d$ άρα τελικά από τη 1η περίπτωση του Master Theorem, δηλαδή
$T(n) = O(n^1) = O(n)$

Άσκηση 5

a)

ΚΩΔΙΚΑΣ

```
Put(A)                          //A[1…n]
  A.sort()
  n = len(A)
  boxes = []                    //boxes είναι ένας πίνακας n πινάκων n
                                // στοιχείων
  for i = 1 to n do
    box = 20
    b = []
    k = 0

    for j = 1 to n do
      b += 0
      if A[j] < box
          if k == 0 and A[j] != 0
            b[j] = A[j]
            box -= A[j]
            A[j] = 0
            k += 1
          end if
      else
          b[j] = box
          A[j] -= box
          box = 0
      end if
    end for
    boxes += b
end for
return boxes
```

ΕΡΓΑΣΙΑ 1

Μαρία Σχοινάκη: 3210191

b)

Είσοδος αλγορίθμου: ένας πίνακας A[1..n] όπου A[i] πλήθος μπαλών χρώματος i
Έξοδος αλγορίθμου: ένας πίνακας $B[[1..n]_1..[1..n]_n]$ που περιέχει πίνακες n στοιχείων. Κάθε πίνακας του B αντιπροσωπεύει ένα κουτί και σε κάθε τέτοιο πίνακα n στοιχείων έστω C[1..n], C[i] αντιπροσωπεύει το πλήθος των μπαλών χρώματος i που βάλαμε στο τρέχον κουτί
Πρόταση P(i): Στο τέλος κάθε for loop i, ο B[i] πίνακας αντιπροσωπεύει σωστό συνδυασμό χρωμάτων για το κουτί i, δηλαδή έχει γεμίσει σωστά το κουτί i
Βάση επαγωγής: P(1) είναι αληθές γιατί ο A[1] περιέχει ένα μοναδικό στοιχείο που είναι ίσο με 20 και οι μπάλες μπαίνουν σε 1 και μοναδικό κουτί, τον πίνακα B[1]
Επαγωγική υπόθεση: P(k) είναι αληθές για κάθε 1<k≤n
Επαγωγικό βήμα: πρέπει να δείξουμε ότι P(k+1) είναι αληθές δεδομένης της επαγωγικής υπόθεσης
Από υπόθεση, P(k) αληθές, άρα έχω γεμίσει **σωστά** k από τα k + 1 κουτιά μου. Επειδή οι μπάλες είναι 20*#κουτιών και κάθε κουτί χωράει 20 μπάλες αυτό σημαίνει ότι έχω k γεμάτα κουτιά με 20 μπάλες το καθένα(*σε κάθε κουτί το πολύ 2 διαφορετικών χρωμάτων μπάλες*), και άρα από τις (k + 1)*20 μπάλες, έχουν μείνει εκτός κουτιού 20 μπάλες. Οπότε αυτές τις 20 μπάλες τις βάζουμε στο τελευταίο κουτί, το k + 1 κουτί.

c)

Η διαδικασία της τοποθέτησης μιας μπάλας σε ένα κουτί γίνεται σε ένα βήμα, δηλαδή το γέμισμα ενός κουτιού με 20 μπάλες χρειάζεται χρόνο O(1), και οι πράξεις πρόσθεσης και αφαίρεσης γίνονται επίσης σε ένα βήμα.

Έχουμε:

```
O(nlogn)                  // πολυπλοκότητα της συνάρτησης sort() της python
O(1)
for i = 1 to n do
    O(1)
    for j = 1 to n do
        O(1)
O(1)
```

Δηλαδή έχουμε O(nlogn) + O(1) + O(n) + O(n^2) + O(1) = O(n^2)
Αφού $\sum_1^n \mathbf{1} = \boldsymbol{n}$ **= O(n)**
Και $\sum_1^n \boldsymbol{n}$ **= n * n = n^2 = O(n^2)**