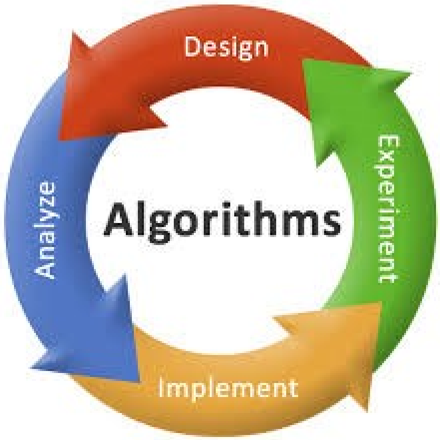
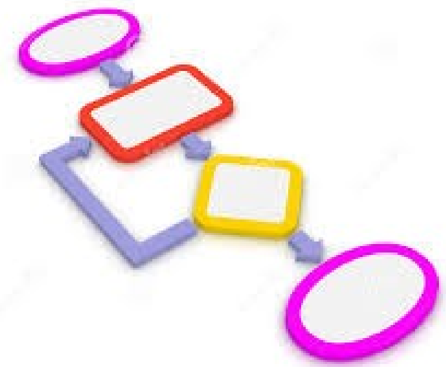




ΑΛΓΟΡΙΘΜΟΙ

Δρ. Χάρης Κουζινόπουλος

Εθνικό Κέντρο Έρευνας και Τεχνολογικής Ανάπτυξης

Πανεπιστήμιο Μακεδονίας

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Ψευδοκώδικας – [11]

Βρόχος *for...to...with step...*

for *μεταβλητή* ← *αρχική_τιμή* **to** *τελική_τιμή* **with step** *β*
        εντολές

Παράδειγμα. Υπολογισμός αθροίσματος 1+2+3+4+5

s ← 0
for x ← 1 **to** 5
    s ← s+x

Επαναλήψεις	x	s
Αρχή	-	0
1	1	1
2	2	3
3	3	6
4	4	10
5	5	15

Ψευδοκώδικας – [12]

Φωλιασμένοι βρόχοι

Παράδειγμα. Να υπολογιστεί το άθροισμα των στοιχείων του πίνακα

$$A = \begin{bmatrix} 2 & 4 \\ -2 & 1 \\ -3 & 6 \end{bmatrix}$$

```
s ← 0
for i ←1 to 3
  for j←1 to  2
    s ← s + A(i, j)
```

$s = 0$

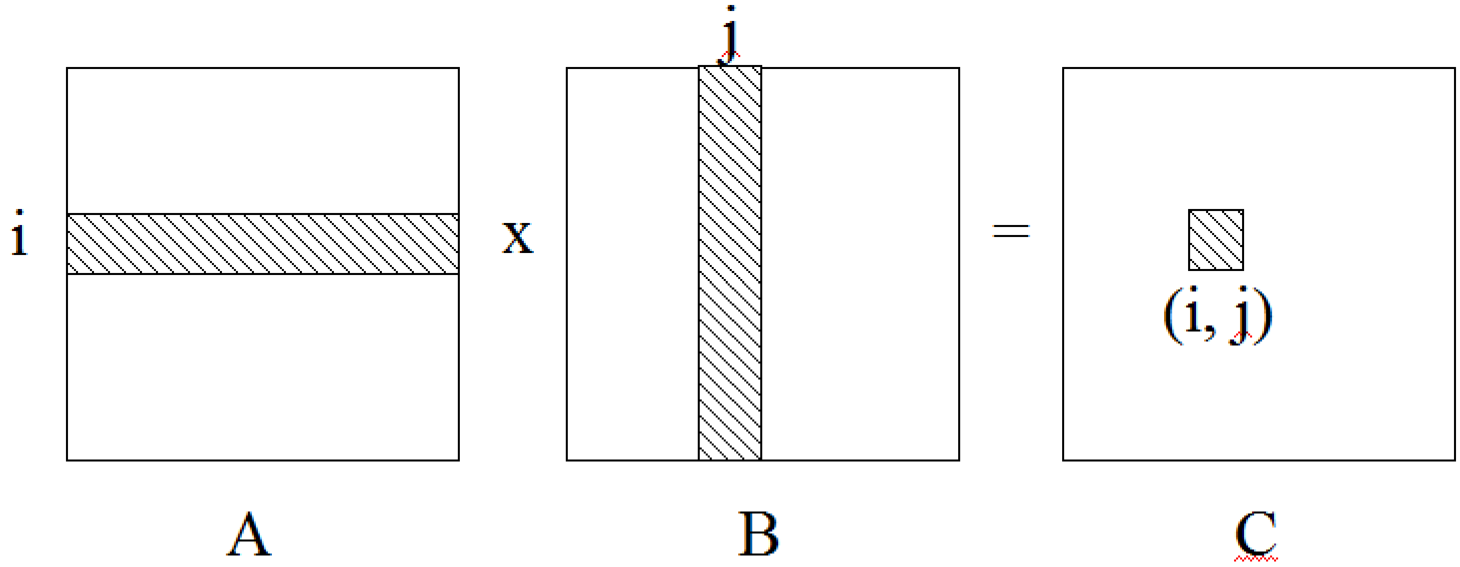
$i = 1 \ , \ j = 1 \ , \ s = 0 + A(1,1) = 2 \ , \ j = 2 \ , \ s = 2 + A(1,2) = 6$

$i = 2 \ , \ j = 1 \ , \ s = 6 + A(2,1) = 4 \ , \ j = 2 \ , \ s = 4 + A(2,2) = 5$

$i = 3 \ , \ j = 1 \ , \ s = 5 + A(3,1) = 2 \ , \ j = 2 \ , \ s = 2 + A(2,3) = 8$

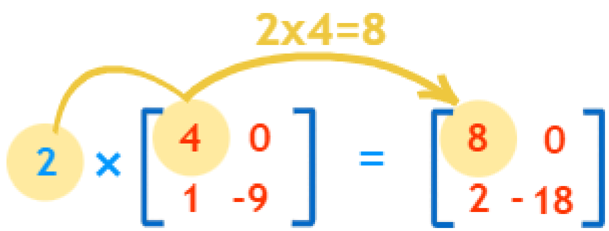
Ψευδοκώδικας – [13]

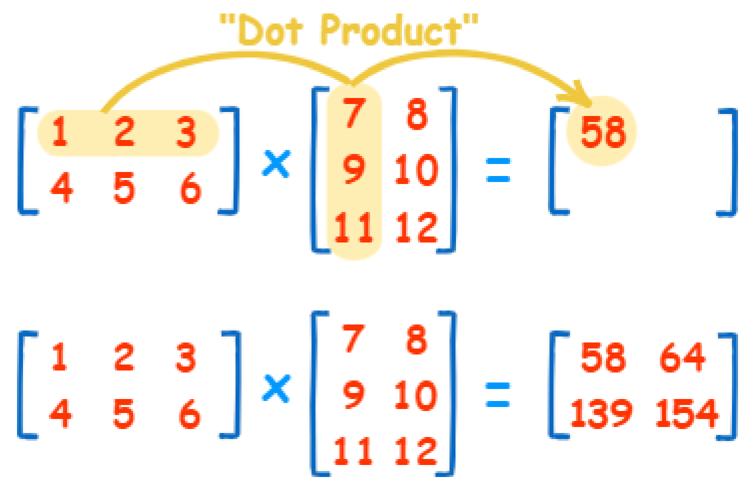
Το γινόμενο δυο πινάκων $A_{mxk}B_{kxn} = C_{mxn}$ υπολογίζεται από τα mxn εσωτερικά γινόμενα



Ψευδοκώδικας – [13]

Το γινόμενο δυο πινάκων $A_{mxk}B_{kxn} = C_{mxn}$ υπολογίζεται από τα mxn εσωτερικά γινόμενα

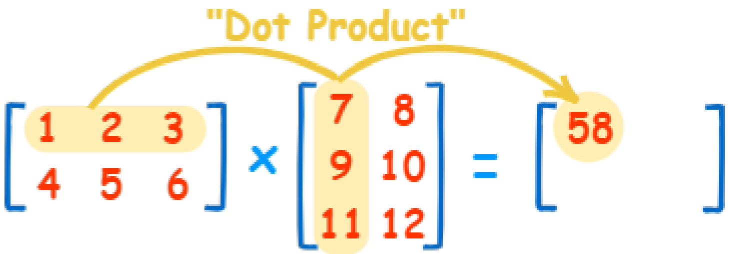




Ψευδοκώδικας – [13]

Το γινόμενο δυο πινάκων $A_{mxk}B_{kxn} = C_{mxn}$ υπολογίζεται από τα mxn εσωτερικά γινόμενα

```
for (int i = 0; i < m; ++i)
   for (int j = 0; j < n; ++j)
      for (int r = 0; r < k; ++r)
         C[i][j] += A[i][r] * B[r][j];
```



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Ψευδοκώδικας – [16]

Παράδειγμα. Υπολογισμός όρων ακολουθίας Fibonacci (το πρόβλημα τέθηκε το 1202 μ.χ. «***Ας υποθέσουμε ότι έχετε ένα ζευγάρι κουνελιών το οποίο κάθε μήνα γεννά ένα νέο ζευγάρι το οποίο με τη σειρά του είναι έτοιμο για αναπαραγωγή από το δεύτερο μήνα. Πόσα ζευγάρια κουνελιών θα υπάρχουν σε ένα χρόνο?***»

$$f(n) = f(n-1) + f(n-2), \quad n \geq 2, \quad f(0) = 0, \quad f(1) = 1$$

n	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
f(n)	**0**	**1**	**1**	**2**	**3**	**5**	**8**	**13**	**21**

Ψευδοκώδικας – [17]

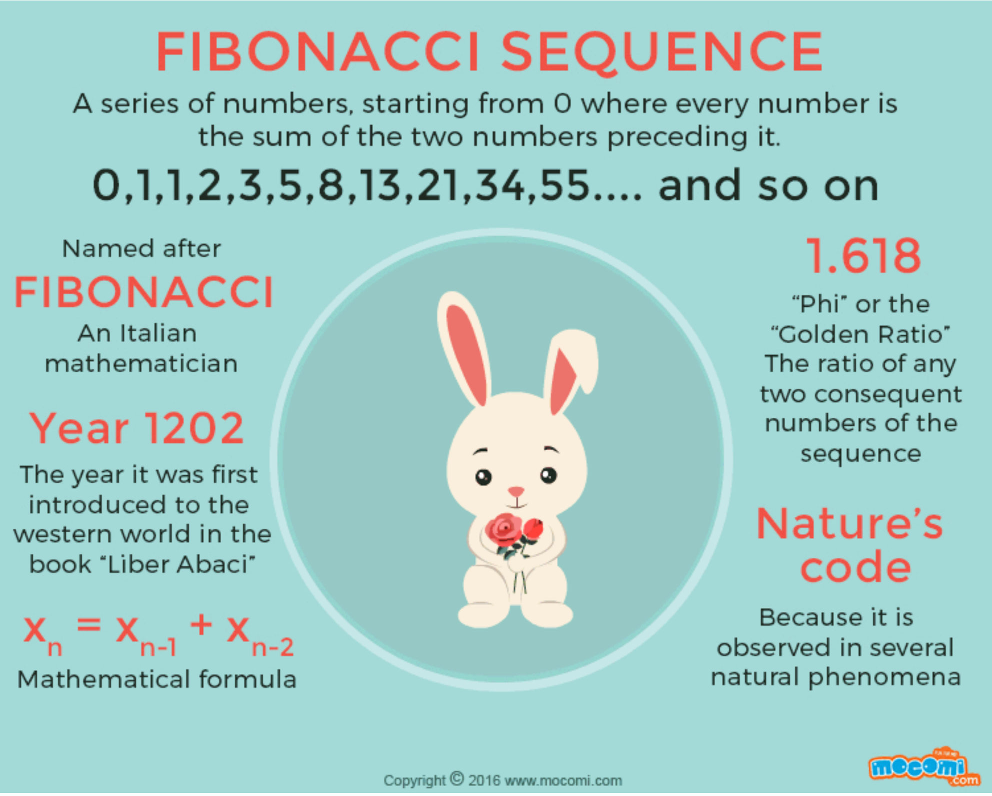
Οι αριθμοί της ακολουθίας Fibonacci αυξάνουν το ίδιο γρήγορα με τις δυνάμεις του 2. Ισχύει,

$$\mathbf{f(n) \approx 2^{0.681n}}$$

Π.χ.

f(100) = 3.5422E+20

$2^{0.681 \times 100}$ = 3.1633E+20



Ψευδοκώδικας – [18]

Λύση.

Υλοποίηση για n = 5

Αλγόριθμος : fib1

Είσοδος : n

Έξοδος: f

```
f ← 0
i ← 1
for k ← 1 to n
    f ← f+i
    i ← f-i
```

k	f	i
-	0	1
1	1	0
2	1	1
3	2	1
4	3	2
5	5	3

Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 02-fib1.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ:
n – Ακέραιος αριθμός (τύπου int) ο οποίος δηλώνει τον όρο της ακολουθίας fibonacci που πρέπει να υπολογιστεί.

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ:
f – Ο n-οστός όρος (τύπου int) της ακολουθίας fibonacci. Το αποτέλεσμα εμφανίζεται στην οθόνη.

Filename: [02-fib1.c](02-fib1.c)

Δυναμικός προγραμματισμός

- Έξυπνοι αλγόριθμοι ωμής βίας
- Πολύ ισχυρή τεχνική!
- Η κύρια ιδέα είναι:
    - να επιλύσουμε πολλά μικρότερα προβλήματα
    - να καταγράψουμε τις λύσεις σε ένα πίνακα ώστε το κάθε **υποπρόβλημα** να επιλυθεί μόνο μια φορά
    - να επαναχρησιμοποιήσουμε τον πίνακα για τη λύση του προβλήματος

Δυναμικός προγραμματισμός

Αλγόριθμος ακολουθίας Fibonacci (εύρεση του n-οστού αριθμού)

- Κάθε αριθμός είναι το άθροισμα των δυο προηγούμενων:
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
- Σχετίζεται με τη Χρυσή αναλογία
  - *f(0) = 0*
  - *f(1) = 1*
  - *f(n) = f(n-1) + f(n-2)*

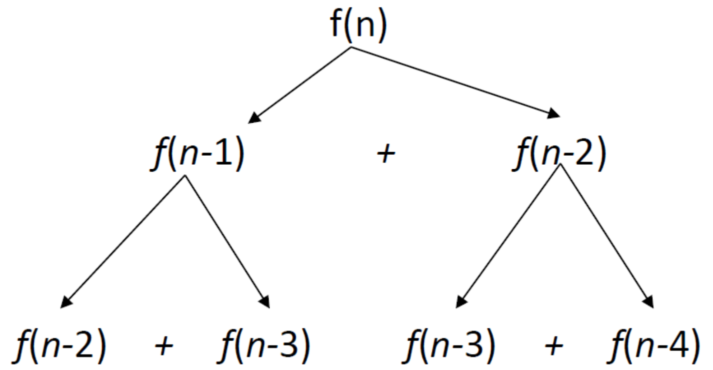
fib(n):
if n <= 2: f = 1
else f = fib(n - 1) + fib(n – 2)
return f

- Πολυπλοκότητα;

Δυναμικός προγραμματισμός
Αλγόριθμος ακολουθίας Fibonacci

- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
- Σχετίζεται με τη Χρυσή αναλογία
  - $f(0) = 0$
  - $f(1) = 1$
  - $f(n) = f(n-1) + f(n-2)$

fib(n):
if n <= 2: f = 1
else f = fib(n - 1) + fib(n – 2)
return f



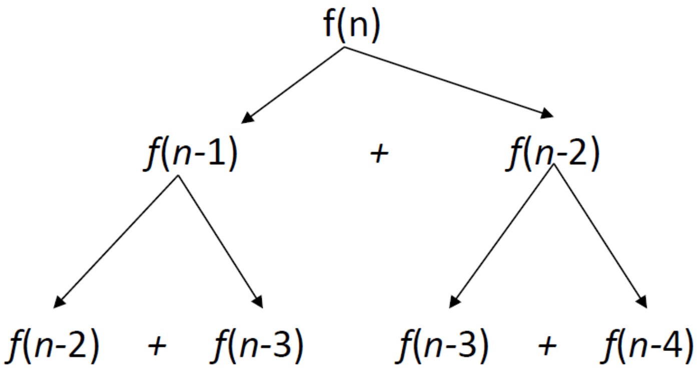
- Πολυπλοκότητα;

Δυναμικός προγραμματισμός
Αλγόριθμος ακολουθίας Fibonacci

- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
- Σχετίζεται με τη Χρυσή αναλογία
  - $f(0) = 0$
  - $f(1) = 1$
  - $f(n) = f(n-1) + f(n-2)$

fib(n):
if n <= 2: f = 1
else f = fib(n - 1) + fib(n – 2)
return f

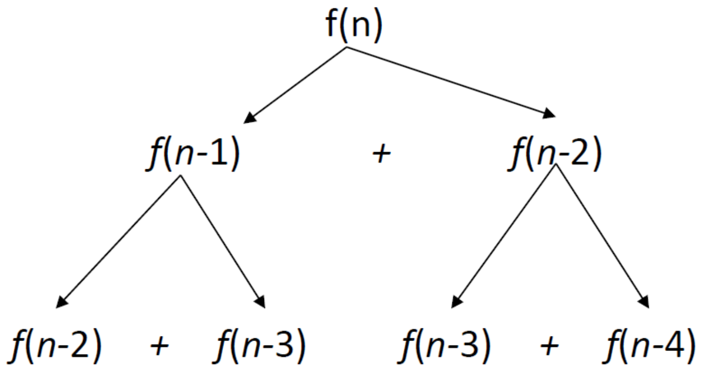
- Πολυπλοκότητα; Εκθετική



Δυναμικός προγραμματισμός
Αλγόριθμος ακολουθίας Fibonacci με πίνακα

```
fib(n):
memo = {}
If n in memo: return memo[n]
if n <= 2: f = 1
else f = fib(n - 1) + fib(n – 2)
Memo[n] = f
return f
```

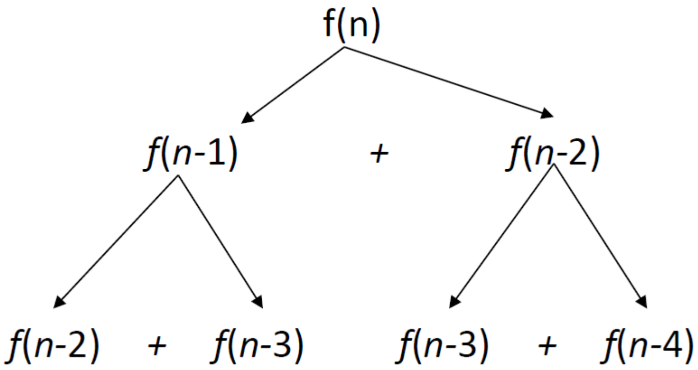
- Πολυπλοκότητα;



Δυναμικός προγραμματισμός
Αλγόριθμος ακολουθίας Fibonacci με πίνακα

```
fib(n):
memo = {}
If n in memo: return memo[n]
if n <= 2: f = 1
else f = fib(n - 1) + fib(n – 2)
Memo[n] = f
return f
```

- Πολυπλοκότητα; n^2



Ψευδοκώδικας – [19]

Παράδειγμα. Να γραφεί αλγόριθμος, ο οποίος, να υπολογίζει το ελάχιστο (min) στοιχείο ενός διανύσματος Τ, n στοιχείων, καθώς και τη θέση του (index).

Ψευδοκώδικας – [19]

Παράδειγμα. Να γραφεί αλγόριθμος, ο οποίος, να υπολογίζει το ελάχιστο (min) στοιχείο ενός διανύσματος T, n στοιχείων, καθώς και τη θέση του (index).

Λύση

Αλγόριθμος: min1
Είσοδος: T, n
Έξοδος: min, index

```
min ← T(1)
index ← 1
for i ←2 to n
    if T(i) < min
        min ← T(i)
        index ← i
```

Ψευδοκώδικας – [20]

Παράδειγμα. Να γραφεί ψευδοκώδικας για την εύρεση μικρότερου αριθμού σε ένα διάνυσμα T με n στοιχεία. Στο ψευδοκώδικα να συμπεριλάβετε και την περίπτωση στην οποία ο μικρότερος αριθμός εμφανίζεται περισσότερες από μια φορά.

Hint! Η προσπέλαση όλων των στοιχείων του T να γίνει με ένα πέρασμα (σάρωση).

Ψευδοκώδικας – [21]

Να αναπτυχτεί αλγόριθμος σε μορφή ψευδοκώδικα, ο οποίος να υπολογίζει τον δεύτερο μεγαλύτερο αριθμό σε έναν πίνακα T.

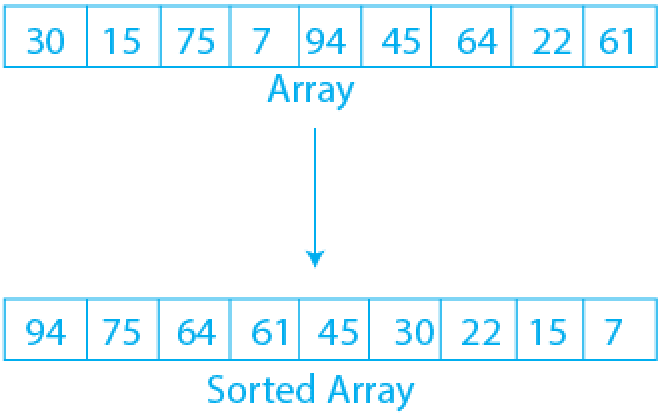
Στα δεδομένα i=0, j=n-1

Ψευδοκώδικας – [21]

Να αναπτυχτεί αλγόριθμος σε μορφή ψευδοκώδικα, ο οποίος να υπολογίζει τον δεύτερο μεγαλύτερο αριθμό σε έναν πίνακα T.

Στα δεδομένα i=0, j=n-1

Με αντίστροφη ταξινόμηση: O(nlogn)



Ψευδοκώδικας – [21]

Να αναπτυχτεί αλγόριθμος σε μορφή ψευδοκώδικα, ο οποίος να υπολογίζει τον δεύτερο μεγαλύτερο αριθμό σε έναν πίνακα T.

Με δύο μετρητές:

```
def secondLargest(arr, n):
        largest = second_largest = -2147483648
        for i in range(n):
                if (arr[i] > largest):
                        second_largest = largest
                        largest = arr[i]
                elif (arr[i] > second_largest and arr[i] != largest):
                        second_largest = arr[i]

        if (second_largest == -2147483648):
                print("There is no second largest element")
        else:
                print("The second largest element is", second_largest)
arr = [30, 15, 75, 7, 94, 45, 64, 22, 61]
n = len(arr)
secondLargest(arr, n)
```

Ψευδοκώδικας – [23]

Να εφαρμοστεί ο αλγόριθμος Find_Second_Max στον παρακάτω πίνακα T.

Έστω T=[3 1 4 6 8 10 2 9 5 7] με i=1 και j=10.

Ψευδοκώδικας – [24]

Βρόχος *while…συνθήκη*

Παράδειγμα. Υπολογισμός αθροίσματος 1+2+3+4+5

$s \leftarrow 0$
$x \leftarrow 1$
while $x \leq 5$
  $s \leftarrow s+x$
  $x \leftarrow x+1$

Επαναλήψεις	**s**	**x**	**$x \leq 5$**
Αρχή	0	1	Α
1	1	2	Α
2	3	3	Α
3	6	4	Α
4	10	5	Α
5	15	6	Ψ

Ψευδοκώδικας – [25]

Ρωσικός πολλαπλασιασμός αριθμών a, b.

πηλίκο $a_{k-1}/2$

Θέσε $a_1 = a$, $b_1 = b$.

Υπολόγισε $(a_k, b_k) = (\lfloor a_{k-1}/2 \rfloor, 2b_{k-1})$ μέχρι $a_n = 1$.

Άθροισε τα b_k για τα οποία το a_k είναι περιττός.

k →	**1**	**2**	**3**	**4**	**5**	**6**
a_k →	**45**	**22**	**11**	**5**	**2**	**1**
b_k→	**19**	**38=2·19**	**76=2·38**	**152=2·76**	**304=2·152**	**608=2·304**

$45 \times 19 = b_1 + b_3 + b_4 + b_6 =$ **855**

Ψευδοκώδικας – [26]

Παράδειγμα. Ψευδοκώδικας ρωσικού πολλαπλασιασμού.

```
Αλγόριθμος : russe
Είσοδος : a, b
Έξοδος : p

p ← 0
while a ≥ 1
   if mod(a,2) = 1
      p ← p + b
   a ← div(a,2)
   b ← 2b
```

p	a	b
0	18	6
-	9	12
12	4	24
-	2	48
-	1	96
108	0	192

Ψευδοκώδικας – [27]

Παράδειγμα. Να αναπτυχθεί αλγόριθμος, ο οποίος, δοθέντος ταξινομημένου διανύσματος T με m στοιχεία και αριθμού a, να ταξινομεί το [T a].

Λύση

Αλγόριθμος: insert
Είσοδος: T, m, a
Έξοδος: T

```
T(m+1) ← a
i ← m+1
while i > 1 and T(i-1) > T(i)
    temp ← T(i-1)
    T(i-1) ← T(i)
    T(i) ← temp
    i ← i-1
```