Ο Γρήγορος Πολλαπλασιασμός του Karatsuba

Selected Topics in Algorithms

Ο Αλγόριθμος του Πολλαπλασιασμού

Ο πολλαπλασιασμός (multiplication) είναι μία από τις τέσσερις στοιχειώδεις πράξεις της αριθμητικής στην οποία συμμετέχουν δύο αριθμοί, στη συγκεκριμένη περίπτωση ακέραιοι, ο πολλαπλασιαστέος (multiplicand) και ο πολλαπλασιαστής (multiplier). Ο πολλαπλασιασμός αποτελεί την πράξη κατά την οποία ο πολλαπλασιαστέος προστίθεται στον εαυτό του τόσες φόρες, όσες ο αριθμός του πολλαπλασιαστή. Λόγω της αντιμεταθετικής ιδιότητας ισχύει και το αντίστροφο.

$$a \times b = \underbrace{b + \cdots + b}_{a} = \underbrace{a + \cdots + a}_{b}$$

Ο Αλγόριθμος του Πολλαπλασιασμού

ΠΑΡΑΔΕΙΓΜΑ

Έστω οι αριθμοί a=1453 και b=1821, όπου ο a είναι ο πολλαπλασιαστέος και b είναι ο πολλαπλασιαστής. Η πράξη του πολλαπλασιασμού μεταξύ τους, όπως είναι γνωστή, κάθε ψηφίο του πολλαπλασιαστέου b πολλαπλασιάζεται με κάθε ψηφίο του πολλαπλασιαστή a. Στη συγκεκριμένη περίπτωση το πλήθος των ψηφίων των δύο αριθμών είναι n=4. Συνεπώς εκτελούνται 4x4=16 πολλαπλασιασμοί του ενός ψηφίου και στη συνέχεια 9 προσθέσεις, χωρίς τις προσθέσεις των κρατούμενων, που αν λειφθούν υπ’ όψιν, τότε θα εκτελεσθούν και άλλες 8 προσθέσεις στη συγκεκριμένη περίπτωση.

Ο Αλγόριθμος του Πολλαπλασιασμού

- Για αριθμούς με μεγάλο πλήθος ψηφίων, ο αλγόριθμος του πολλαπλασιασμού χρειάζεται να εκτελέσει n^2 στοιχειώδεις πολλαπλασιασμούς μαζί με ένα πλήθος προσθέσεων kn, όπου k ένας σταθερός αριθμός.

- Ο χρόνος εκτέλεσης του αλγόριθμου είναι T(n)=n^2+kn, όμως για n αρκετά μεγάλο θα ισχύει n ^2 >> kn.

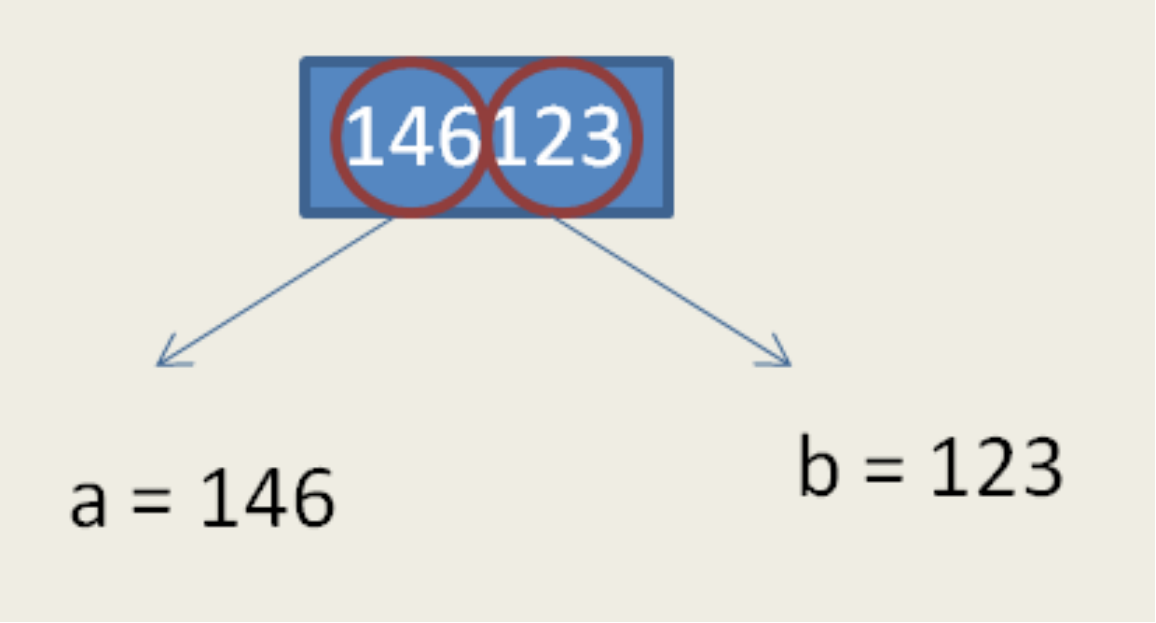
Συνεπώς, η πολυπλοκότητά του είναι O(n^2), όπου n είναι το πλήθος των ψηφίων των δύο αριθμών που μετέχουν στην πράξη. Σε περίπτωση που οι δύο αριθμοί έχουν διαφορετικό πλήθος ψηφίων, έστω n και m με n>m, όπου n,m αριθμοί με μεγάλο πλήθος ψηφίων, τότε η πολυπλοκότητα θα είναι O(nm).

Όπως θα δειχθεί και παρακάτω, ο αλγόριθμος Karatsuba για τον πολλαπλασιασμό δύο ακεραίων αριθμών αποτελεί έναν ταχύτερο και σαφώς πιο αποδοτικό αλγόριθμο από αυτόν του απλού πολλαπλασιασμού. Η μελέτη του αλγόριθμου και η υλοποίησή του θα γίνει με σκοπό τη βελτίωση της απόδοσης, τόσο σε επίπεδο κόστους υλοποίησης, όσο και σε ταχύτητα.

Ο αλγόριθμος Karatsuba πρόκειται για έναν αλγόριθμο «διαίρει και βασίλευε» (Divide and Conquer), δηλαδή αποτελεί έναν αλγόριθμο ο οποίος αντιμετωπίζει το πρόβλημα του πολλαπλασιασμού διαιρώντας αρχικά τις ακολουθίες ψηφίων ενός αριθμού σε υπακολουθίες, στη συνέχεια εκτελεί τη πράξη του πολλαπλασιασμού σε κάθε μία από αυτές και τέλος ενώνει τις λύσεις των υποπροβλημάτων σε μία τελική.

Μία πρώτη προσέγγιση

Ας υποθέσω έναν ακέραιο αριθμό με n=6 ψηφία και έστω ότι αυτός είναι ο x = 146123. Αυτόν τον αριθμό μπορώ να τον σπάσω σε δύο μέρη:



Μία πρώτη προσέγγιση

Ξέρω ότι:

146123 = 146000 + 123 = 146 * 10^3 + 123

Και η γενική του μορφή είναι:

$$x = a * 10^{n/2} + b$$

Μία πρώτη προσέγγιση

Έστω τώρα και ένας αριθμός y = 352120, τον οποίο επίσης χωρίζω σε δύο μέρη c = 352 και d = 120 και έχω τις εξισώσεις:

$$x = a * 10^{n/2} + b$$

$$y = c * 10^{n/2} + d$$

Μία πρώτη προσέγγιση

Το γινόμενο των δύο αυτών αριθμών θα είναι:

$$x * y = (a * 10^{n/2} + b)(c * 10^{n/2} + d)$$
$$= (a * 10^{n/2})(c * 10^{n/2}) + ad * 10^{n/2} + bc * 10^{n/2} + bd$$
$$= ac * 10^{2(n/2)} + (ad + bc) * 10^{n/2} + bd$$

Μία πρώτη προσέγγιση

Η πολυπλοκότητα του αλγόριθμου, λαμβάνοντας υπ’ όψην αυτή τη διαμέριση δίνεται απο την εξής αναδρομική εξίσωση:

$$T(n) = 4T(n/2) + O(n).$$

Η εξίσωση αυτή περιγράφει το χρόνο εκτέλεσης του αλγόριθμου για τη συγκεκριμένη διαμέριση, δηλάδη τέσσερα υποπροβλήματα μεγέθους n/2, τα οποία το κάθε ένα έχει να επιλύσει μία διαδικασία πολυπλοκότητας O(n). Λύνοντας αυτήν την εξίσωση προκύπτει το συμπέρασμα ότι η πολυπλοκότητα του αλγόριθμου είναι T(n) = O(n^2).

Μία δεύτερη προσέγγιση

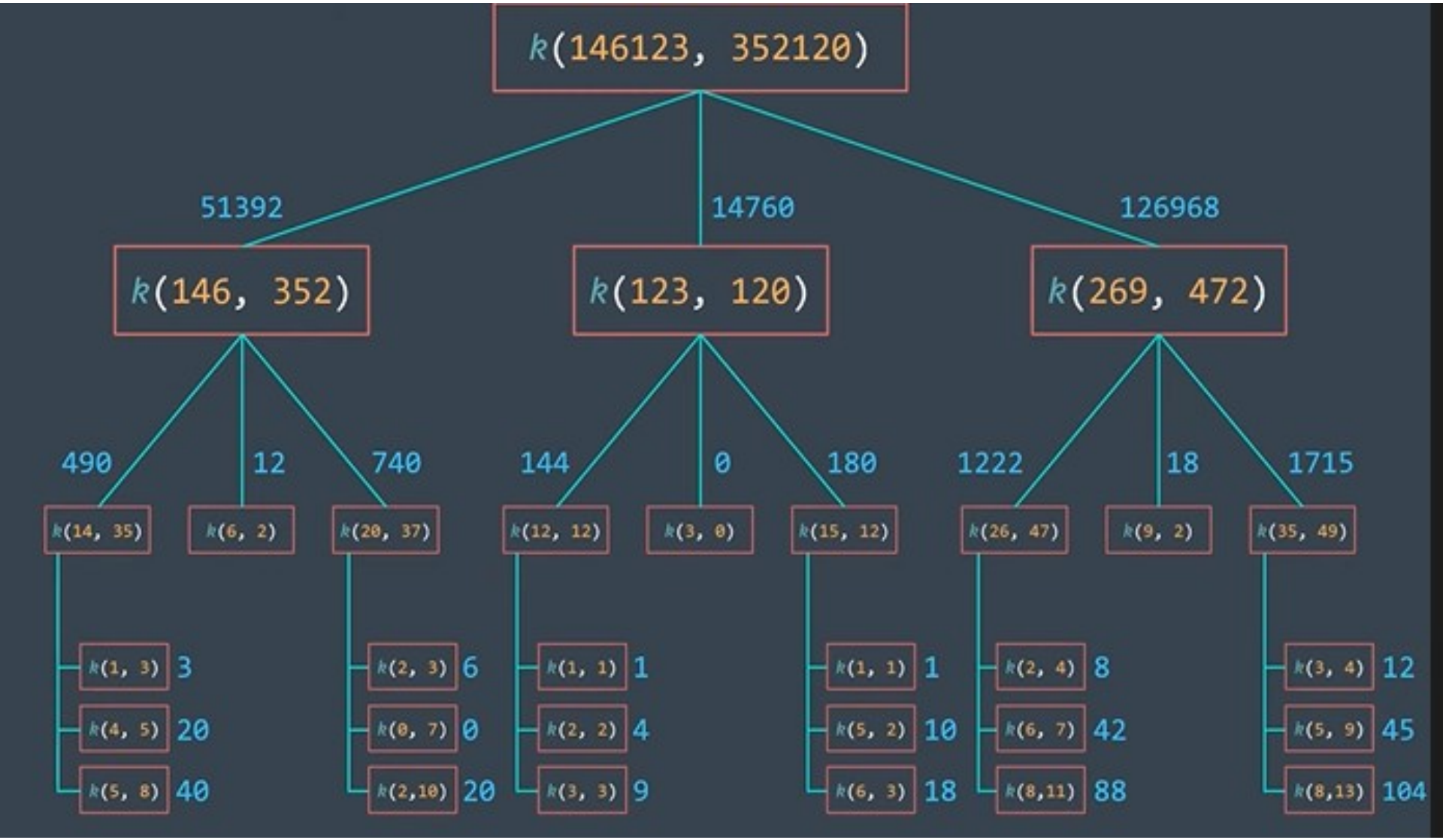
Όπως φάνηκε και πριν, η μια απλή διαμέριση του προβλήματος του πολλαπλασιασμού σε υποπροβλήματα δεν βελτιώνει την πολυπλοκότητά του

Στην προηγούμενη διαμέριση το πλήθος των υποπροβλημάτων ήταν τέσσερα. Στην πραγματικότητα όμως είναι απαραίτητα μόνο τρία υποπροβλήματα για τον υπολογισμό του τελικού γινομένου.

Τα υποπροβλήματα αυτά είναι τα εξής:

```
ac = karatsuba(a, c)
bd = karatsuba(b, d)
ad_plus_bc = karatsuba(a+b, c+d)-ac-bd
```

```python
def karatsuba(x, y):
    if x < 10 or y < 10:
        return x * y
    else:
        n = max(len(str(x)), len(str(y)))
        half = n // 2
        a = x // (10 ** (half))  # left part of x
        b = x % (10 ** (half))  # right part of x
        c = y // (10 ** (half))  # left part of y
        d = y % (10 ** (half))  # right part of y
        ac = karatsuba(a, c)
        bd = karatsuba(b, d)
        ad_plus_bc = karatsuba(a+b, c+d)-ac-bd
        return ac * (10 ** (2 * half)) + (ad_plus_bc * (10 ** half)) + bd
```


k(146123, 352120)
51392
14760
126968
k(146, 352)
k(123, 120)
k(269, 472)
490
12
740
144
0
180
1222
18
1715
k(14, 35)
k(6, 2)
k(20, 37)
k(12, 12)
k(3, 0)
k(15, 12)
k(26, 47)
k(9, 2)
k(35, 49)
k(1, 3) 3
k(4, 5) 20
k(5, 8) 40
k(2, 3) 6
k(0, 7) 0
k(2,10) 20
k(1, 1) 1
k(2, 2) 4
k(3, 3) 9
k(1, 1) 1
k(5, 2) 10
k(6, 3) 18
k(2, 4) 8
k(6, 7) 42
k(8,11) 88
k(3, 4) 12
k(5, 9) 45
k(8,13) 104

Άρα,

$$T(n) = 3T(n/2) + \Theta(n)$$

Για τον έλεγχο της πολυπλοκότητας θα χρησιμοποιήσουμε το Master Theorem :

$$c < \log_b a \rightarrow T(n) = \Theta(n^{\log_b a})$$
$$c = \log_b a: \quad d > -1 \rightarrow T(n) = \Theta(n^c \log^{d+1} n)$$
$$d = -1 \rightarrow T(n) = \Theta(n^c log log n)$$
$$d < -1 \rightarrow T(n) = \Theta(n^c)$$
$$c > \log_b a: \quad d \geq 0 \rightarrow T(n) = \Theta(n^c \log^d n)$$
$$d < 0 \rightarrow T(n) = \Theta(n^c)$$

Κοιτάζοντας την γενική σχέση:

T(n) = a T(n/b) + Θ(n^c *log^ (n))

Βλέπουμε ότι στην περίπτωση μας έχουμε

a = 3 , b = 2 , c=1 , d=0

Άρα βρισκόμαστε στη πρώτη περίπτωση του Master

Theorem και με αντικατάσταση ισχύει ότι:

T(n) = Θ (n ^1,58).

Συμπέρασμα

Είναι εμφανές ότι με αυτήν την προσέγγιση, ο αλγόριθμος του πολλαπλασιασμού έχει μία αισθητά βελτιωμένη πολυπλοκότητα, περίπου O(n^1.5), σε σχέση με αυτή του κλασσικού αλγόριθμου του πολλαπλασιασμού που έχει παρουσιαστεί μέχρι τώρα, η οποία είναι O(n^2).