Κατσούπης Ευάγγελος
2017030077

$1^{η}$ Υπολογισμός Αθροισμάτων

1. $\sum_{1\leq k\leq n} k^2 = \frac{n(n+1)(2n+1)}{6}$ (αποδείχθηκε στο μάθημα).

2. $\sum_{1\leq k\leq n} k^3 = \frac{n^2(n+1)^2}{4}$ (αποδείχθηκε στο μάθημα).

3. $\sum_{1\leq k\leq n} 2^{k/3} = \sum_{1\leq k\leq n} \left(\sqrt[3]{2}\right)^k$ έστω $\sqrt[3]{2} = a$ τότε το άθροισμα είναι της μορφής : $\sum_{1\leq k\leq n} a^k$.
   Είναι γνωστό ότι το άθροισμα $\sum_{0\leq k\leq n} a^k = \frac{a^{n+1}-1}{a-1}$.
   Αν από τον παραπάνω όρο αφαιρέσουμε το $a^0 = 1$,
   φτάνουμε στο συμπέρασμα ότι $\sum_{1\leq k\leq n} a^k = \frac{a^{n+1}-1}{a-1} - 1 = \frac{a^{n+1}-a}{a-1}$.
   Τότε $\sum_{1\leq k\leq n} 2^{k/3} = \frac{\sqrt[3]{2}^{n+1} - \sqrt[3]{2}}{\sqrt[3]{2}-1}$

4. $\sum_{1\leq k\leq n} \frac{1}{k(k+1)}\frac{1}{k(k+1)}$ θα μετατρέψω τον όρο του αθροίσματος σε άθροισμα από γνωστά ζεύγη.
   Έχω $\frac{1}{k(k+1)} = \frac{A}{k} + \frac{B}{k+1} \overset{\cdot(k(k+1))}{\leftrightarrow} 1 = k(A+B) + A$
   Θα πρέπει $A = 1$ και άρα $B = -1$.
   Έτσι το παραπάνω αθροισμα γίνεται: $\sum_{1\leq k\leq n} \left(\frac{1}{k} - \frac{1}{k+1}\right) = 1 - \cancel{\frac{1}{2}} + \cancel{\frac{1}{2}} - \cdots + \cancel{\frac{1}{n}} - \frac{1}{n+1} = 1 - \frac{1}{n+1} = \frac{n}{n+1}$

5. $\sum_{1\leq k\leq n} k(k+1) = \sum_{1\leq k\leq n} k^2 + k = \sum_{1\leq k\leq n} k^2 + \sum_{1\leq k\leq n} k$. Γνωστά αθροίσματα με:
   $\sum_{1\leq k\leq n} k^2 + \sum_{1\leq k\leq n} k = \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} = \cdots = \frac{n(n+1)(n+2)}{3}$

6. $\sum_{1\leq k\leq n} \log(k) = \log(1) + \log(2) + \cdots + \log(n) \overset{\log(ab)=\log(a)+\log(b)}{=} \log(n!)$ (Stirling's formula)

7. $\sum_{1\leq i\leq n}\sum_{i<j\leq n} (j-i) = \sum_{1\leq j\leq n}\sum_{1\leq i\leq j} (j-i)$ (αν $j = i$ τότε θα προσθέτουμε 0, άρα εναλλαγή όρων).
   Για το εσωτερικό άθροισμα:

$$\sum_{1\leq i\leq j} (j-i) = \sum_{1\leq i\leq j} j - \sum_{1\leq i\leq j} i = (j\cdot j) - \frac{j(j+1)}{2} = \frac{j^2-j}{2}$$

   Με βάση την παραπάνω απλουστευση το συνολικό άθροισμα ορίζεται ώς:

$$\sum_{1\leq j\leq n} \frac{j^2-j}{2} = (\frac{1}{2}) \sum_{1\leq j\leq n} j^2 - \sum_{1\leq j\leq n} j = \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}\right) = \cdots = \frac{n(n^2-1)}{6}$$

8. $\sum_{1\le k\le n} k\cdot k!$ με $k\cdot k! = ((k+1)-1)k! = (k+1)k! - k! = (k+1)! - k!$.
Έτσι το άθροισμα γίνεται:

$$\sum_{1\le k\le n} (k+1)! - k! = \cancel{2!} - 1! + \cancel{3!} - \cancel{2!} + \cdots + (n+1)! - \cancel{n!} = (n+1)! - 1$$

9. $\sum_{(1\le k\le n)}\sum_{(k\le j\le n)}\sum_{(1\le i\le n-j)} 1 = \sum_{(1\le k\le n)}\sum_{(k\le j\le n)} (n-j)$

Όμοια με το άθροισμα 7 θα γίνει εναλλαγή ορίων:

$$\sum_{(1\le k\le n)}\sum_{(k\le j\le n)} (n-j) = \sum_{(1\le j\le n)}\sum_{(1\le k\le j)} (n-j) = \sum_{(1\le j\le n)} \left(n\cdot j - j^2\right) =$$
$$\sum_{(1\le j\le n)} j - \sum_{(1\le j\le n)} j^2 = \frac{n^2(n+1)}{2} - \frac{n(n+1)(2n+1)}{6} =$$
$$\frac{n^2(n+1)}{2} - \frac{n(n+1)(2n-1)}{6} = \cdots = \frac{n(n^2-1)}{6}$$

10. $\sum_{1\le k\le n} k^2\cdot 2^k$ Θα αναπτυχθεί ο όρος k ($k^2 = k\cdot k$) σε άθροισμα:

$$\sum_{1\le k\le n}\left(\left(\sum_{1\le i\le k} k\right)\cdot 2^k\right) = \sum_{1\le k\le n}\sum_{1\le i\le k} (k\cdot 2^k) = \sum_{1\le i\le n}\sum_{i\le k\le n} (k\cdot 2^k)$$

Θα ασχοληθώ με το εσωτερικό άθροισμα:

$$\sum_{i\le k\le n} (k\cdot 2^k) = \sum_{1\le k\le n} (k\cdot 2^k) - \sum_{1\le k\le i-1} (k\cdot 2^k)$$

Για κάθε άθροισμα έχω:

$$\sum_{1\le k\le n} (k\cdot 2^k) = \cdots = (n-1)2^{n+1} + 2 \quad \text{(σημειώσεις Σαμολαδά)}$$

$$\sum_{1\le k\le i-1} (k\cdot 2^k) \overset{i-1=a}{=} \sum_{1\le k\le a} (k\cdot 2^k) = (a-1)2^{a+1} + 2 = (i-2)2^i + 2$$

Άρα έχω το άθροισμα:

$$\sum_{1\le i\le n} \left(\ [(n-1)2^{n+1}+2] - [(x-2)2^x + 2]\ \right) = \sum_{1\le i\le n} (n-1)2^{n+1} - \sum_{1\le i\le n} (i-2)2^i$$

θα βρώ τα επιμέρους αθροίσματα:

$$\sum_{1\le i\le n} (n-1)2^{n+1} = n(n-1)2^{n+1}$$

$$\sum_{1\le i\le n} (i-2)2^i = \sum_{1\le i\le n} i2^i - 2^{i+1} = 2\underbrace{-2^2 + 2\cdot 2^2}_{2^2}\underbrace{-2^3 + 3\cdot 2^3}_{2\cdot 2^3}\underbrace{-2^4 + 4\cdot 2^4}_{3\cdot 2^4} - \cdots$$

Άρα το παραπάνω άθροισμα μπορεί να γραφεί ώς:

$$2 + \sum_{1\le k\le n-1} (k2^{k+1}) - 2^{n+1} = \sum_{1\le k\le n} (k2^{k+1}) - n2^{n+1} - 2^{n+1} + 2 =$$
$$2\sum_{1\le k\le n} (k2^k) - (n+1)2^{n+1} + 2 = 2\left((n-1)2^{n+1} + 2\right) - (n+1)2^{n+1} + 2$$

Τότε:

$$n(n-1)2^{n+1} - 2\left((n-1)2^{n+1} + 2\right) + (n+1)2^{n+1} - 2 = 2^{n+1}\left(n(n-2)+3\right) - 6$$

Επομένως, το άθροισμα είναι της μορφής:

$$\sum_{1 \le k \le n} k^2 \cdot 2^k = 2^{n+1}\left(n(n-2)+3\right) - 6$$

2^{η} $\quad \log\left(log^* n\right) \quad ? \quad \log^*\left(logn\right)$

Γνωρίζω πώς $log^* n = i$ με $i \geq 0$. Άρα:

$$log^* n = i \overset{i \geq 0}{\longleftrightarrow} \log\left(\log^* n\right) = \log i$$

Επιπλέον, το $log^* n$ μπορεί να αναπτυχθεί ως $log^* n = 1 + \log^a st\left(\log n\right)$ οδηγώντας την παραπάνω σχέση σε:

$$1 + \log^*\left(\log n\right) = i \leftrightarrow \log^*\left(\log n\right) = i - 1$$

Αρκεί, να συγκριθούν οι ισοδύναμοι όροι των παραπάνω σχέσεων:

$$\left(\log i\right) \ ? \ \left(i - 1\right)$$

Όμως $i - 1 \geq log(i) \ \forall \ i \geq 0$.

Έτσι διαπιστώνεται ότι η $\log^*\left(logn\right)$ είναι ασυμπτωματικά μεγαλύτερη από την $\log\left(log^* n\right)$

3^{η} Ασυμπτωματικές λύσεις αναδρομών

1. $T(n) = 4T(n/8) + \Omega(n^{2/3})$

   Έστω $T'(n) = 4T(n/8) + n^{2/3}$

   με $a = 4$ & $b = 8 \Rightarrow n^{\log_b a} = n^{\log_8 4} = n^{2/3}$

   Άρα συγκρίνω $n^{2/3}$ & $f(n) = n^{2/3}$

   Case 2: με $f(n) = \Theta\left(n^{2/3} \cdot \log^k(n)\right) = \Theta\left(n^{2/3}\right), \ k = 0$

   Άρα:

   $$T'(n) = \Theta\left(n^{2/3} \cdot \lg n\right) \Rightarrow T(n) = \Omega\left(n^{2/3} \cdot \lg n\right)$$

2. $T(n) = T(\lfloor 3n/5 \rfloor) + \Theta(n^{1/(e-1)})$

   Έστω $T'(n) = T(3n/5) + n^{1/(e-1)}$

   με $a = 1$ & $b = 5/3 \Rightarrow n^{\log_b a} = n^{\log_{5/3} 1} = n^0 = 1$

   Άρα συγκρίνω n^0 & $f(n) = n^{1/(e-1)} \simeq n^{0.58}$

   $$n^{1/(e-1)} > n^{0+i}, \ \forall \ i \in \left(0, \frac{1}{e-1}\right)$$

Θα ελέγξω την $af(n/b)$ (και έστω $1/(e-1) = x$):

$$af(n/b) = \frac{n^x}{8^x}) = 1/8^x n^x$$

Λόγου του ότι $1/8^x < 0.3$ μπορούμε να πούμε ότι για $c = 0.3 < 1$:

$$1/8^x n^x < cf(n)$$

Case 3: με $f(n) = \Omega(n^i)$ για $i \in \left(0, \frac{1}{e-1}\right)$

Άρα:

$$T'(n) = T(n) = \Theta\left(f(n)\right) = \Theta\left(n^{1/(e-1)}\right)$$

3. $T(n) = 8T(n/16) + O(n^{3/5})$

   Έστω $T'(n) = 8T(n/16) + n^{3/5}$

   με $a = 8$ & $b = 16 \Rightarrow n^{\log_b a} = n^{\log_{16} 8} = n^{3/4}$

   Άρα συγκρίνω $n^{3/4}$ & $f(n) = n^{3/5}$

   $$f(n) < n^{3/4} \Rightarrow f(n) < n^{3/4-i}, \forall\ i \in (0, 3/20)$$

   Case 1: με $f(n) = O(n^{3/4-i})$ για $i \in (0, 3/20)$

   Άρα:

   $$T'(n) = \Theta\left(n^{3/4}\right) \Rightarrow T(n) = O\left(n^{3/4}\right)$$

4. $T(n) = 32T(\lceil n/2 \rceil) + \Theta(n^5 log^4 n)$

   Έστω $T'(n) = 32T(n/2) + n^5 log^4 n$

   με $a = 32$ & $b = 2 \Rightarrow n^{\log_b a} = n^{\log_2 32} = n^5$

   Άρα συγκρίνω n^5 & $f(n) = n^5 log^4 n$

   Case 2: με $f(n) = \Theta(n^5 \log^k n)$ για $k = 4$

   Άρα:

   $$T'(n) = \Theta\left(n^5 \log^5 n\right) \Rightarrow T(n) = O\left(n^5 \log^5 n\right)$$

5. $T(n) = 8T(n/2) + \Theta(n^2 log^3 n)$

   Έστω $T'(n) = 8T(n/2) + n^2 log^3 n$

   με $a = 8$ & $b = 2 \Rightarrow n^{\log_b a} = n^{\log_2 8} = n^3$

   Άρα συγκρίνω n^3 & $f(n) = n^2 log^3 n$

   Case 1: με $f(n) = O(n^{3-i})$ για $i = 0.1$

   Άρα:

   $$T'(n) = T(n) = \Theta\left(n^3\right)$$

6. $T(n) = 4T(n/2) + \Omega(n^2\sqrt{n})$
   Έστω $T'(n) = 4T(n/2) + n^2\sqrt{n}$

   με $a = 4$ & $b = 2 \Rightarrow n^{\log_b a} = n^{\log_2 8} = n^2$

   Άρα συγκρίνω n^2 & $f(n) = n^2\sqrt{n}$

   Θα ελέγξω την $af(n/b)$:

$$af(n/b) = 4(\frac{n^2}{4}\sqrt{\frac{n}{2}}) = \frac{1}{\sqrt{2}}n^2\sqrt{n} = \frac{1}{\sqrt{2}}f(n),\ (\frac{1}{\sqrt{2}} < 1)$$

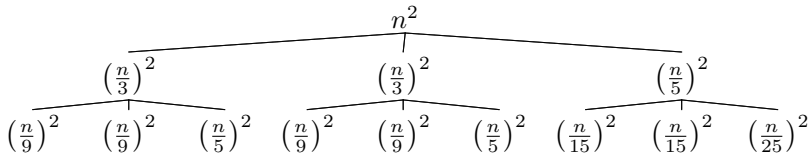
   Case 3: με $f(n) = \Omega(n^{2+i})$ για $i = 0.1$

   Άρα:

$$T'(n) = \Theta\left(n^2\right) \Rightarrow T(n) = \Omega(n^2)$$

7. $T(n) = 2T(n/3) + T(n/5) + O(n^2)$

   Έστω $T'(n) = 2T(n/3) + T(n/5) + n^2$. Θα γίνει προσέγγιση με χρήση δένδρου αναδρομής (παρουσίαση μέρους αυτού)

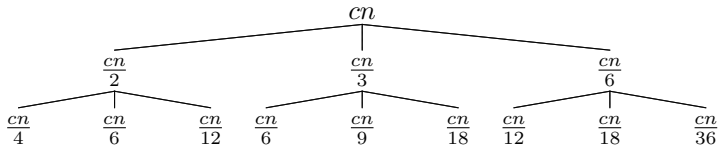


   Παρατηρώντας τα κόστη ανά επίπεδο, έχουμε n^2 κόστος για το επίπεδο 1 και $\frac{59}{225}n^2$ για το επίπεδο 2. Σε αυτό το σημείο παρατηρείται πώς $\frac{59}{225} < 1$. Επομένως, όσους όρους και να αθροίσουμε, το άθροισμα θα συγκλίνει σε μια σταθερά καθώς μιλάμε για άθροισμα γεωμετρικής σειράς. Άρα :

$$c \cdot n^2 \leq T'(n) \leq k \cdot n^2 \Rightarrow$$
$$T'(n) = \Omega(n^2)\ \ \&\ \ T'(n) = O(n^2)$$
$$T'(n) = \Theta(n^2)$$
$$\underline{T(n) = O(n^2)}$$

8. $T(n) = T(n/2) + T(n/3) + T(n/6) + \Theta(n)$

   Έστω $T'(n) = T(n/2) + T(n/3) + T(n/6) + cn$. Θα γίνει προσέγγιση με χρήση δένδρου αναδρομής (παρουσίαση μέρους αυτού)



   Όμοια, παρατηρώντας τα κόστη ανά επίπεδο, παρατηρείται ότι τα κάθε επίπεδο έχει κόστος cn. Επομένως, για να βρω την πολυπλοκότητα αρκεί να πολλαπλασιάσω το μέγιστο και το ελάχιστο βάθος με το κόστος καθώς το συνολικό κόστος θα βρίσκεται κάπου ενδιάμεσα, άρα:

$$\log_6(n) \cdot cn \leq T'(n) \leq \log_2(n) \cdot cn$$

   Θα γίνει εναλλαγή βάσης για τον $\log_6(n)$ ώς:

$$\log_6(n) = \frac{\log_2(n)}{\log_2(6)}$$

Άρα:

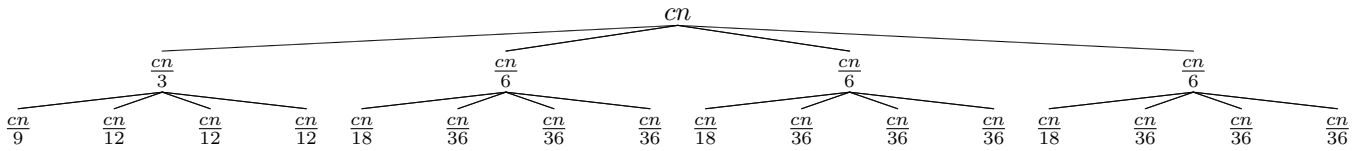
$$\frac{\log_2(n)}{\log_2(6)} \cdot cn \leq T'(n) \leq \log_2(n) \cdot cn$$

$$T'(n) = \Omega\left((\log_2(n) \cdot cn) \cdot \underbrace{\frac{1}{\log_2(6)}}_{c_1} \right) \quad \& \quad T'(n) = O(cn \cdot \log_2(n))$$

$$\underline{T'(n) = T(n) = \Theta(n \cdot \lg(n))}$$

9. $T(n) = T(n/3) + 3T(n/6) + \Theta(n)$

   Έστω $T'(n) = T(n/3) + 3T(n/6) + cn$. Θα γίνει προσέγγιση με χρήση δένδρου αναδρομής (παρουσίαση μέρους αυτού):

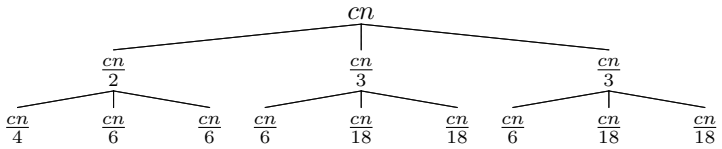


   Όμοια, παρατηρείται πώς το επίπεδο 1 έχει κόστος cn το επίπεδο 2 έχει κόστος $\frac{5}{6}cn$ και το επίπεδο 3 με κόστος $\left(\frac{5}{6}\right)^2 cn$. Άρα αθροίζουμε όρους μικρότερους του 1 (γεωμετρική σειρά), που το άθροισμά του συγκλίνει σε σταθερά. Άρα η πολυπλοκότητα θα ορίζεται ώς:

$$k_1 \cdot cn \leq T'(n) \leq k_2 \cdot cn \Rightarrow$$
$$T'(n) = \Omega(n) \quad \& \quad T'(n) = O(n)$$
$$\underline{T'(n) = T(n) = \Theta(n)}$$

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

    Έστω $T'(n) = T(n/2) + 2T(n/3) + cn$. Θα γίνει προσέγγιση με χρήση δένδρου αναδρομής (παρουσίαση μέρους αυτού):



    Όμοια, παρατηρείται πώς το επίπεδο 1 έχει κόστος cn το επίπεδο 2 έχει κόστος $\frac{7}{6}cn$ και το επίπεδο 3 με κόστος $\left(\frac{7}{6}^2\right)^2 cn$. Σε αυτή την περίπτωση, θα έχουμε 2 διαφορετικά βάθη με κόστος ανά επίπεδο $\left(\frac{7}{6}\right)^i cn$ (όπου i το επίπεδο). Άρα, η πολυπλοκότητα θα κυμαίνεται μεταξύ του μέγιστου και ελάχιστου αθροίσματος των κοστών ανά επίπεδο. Επομένως:

$$\sum_{i=1}^{h_{min}} \left(\frac{7}{6}\right)^i cn \leq T'(n) \leq \sum_{i=1}^{h_{max}} \left(\frac{7}{6}\right)^i cn \Rightarrow cn \sum_{i=1}^{h_{min}} \left(\frac{7}{6}\right)^i \leq T'(n) \leq cn \sum_{i=1}^{h_{max}} \left(\frac{7}{6}\right)^i \Rightarrow$$
$$cn \sum_{i=1}^{\log_3 n} \left(\frac{7}{6}\right)^i \leq T'(n) \leq cn \sum_{i=1}^{\lg n} \left(\frac{7}{6}\right)^i$$

Για το κάθε άθροισμα:

$$cn\sum_{i=1}^{\lg n}\left(\frac{7}{6}\right)^i = \frac{cn\left((7/6)^{lgn+1}-1\right)}{1/6} \overset{a^{log_c b}=b^{log_c a}}{\Longrightarrow} 6cn\cdot(n^{0.22}-1) = 6cn^{1.22}-6cn \Rightarrow$$

$$cn\sum_{i=1}^{\log_3 n}\left(\frac{7}{6}\right)^i = \cdots = 6cn^{1.14}-6cn$$

Άρα

$$6cn^{1.14}-6cn \le T'(n) = T(n) \le 6cn^{1.22}-6cn \Rightarrow$$

4^{η} Karatsuba's like Algorithm

Έστω οι αριθμοί X, Y με μήκους n bits το καθένα που θα 'σπάσουν' σε 3 block ψηφίων :

- **Διαίρει**

$$X = X_L\cdot 2^{2n/3} + X_M\cdot 2^{n/3} + X_R$$
$$Y = Y_L\cdot 2^{2n/3} + Y_M\cdot 2^{n/3} + Y_R$$

Κάθε block πολλαπλασιάζεται με την αντίστοιχη δύναμη του 2 ώστε να φτάσει την τάξη που ανήκει. Στο παράδειγμα, όπου κάθε block έχει μέγεθος $n/3$, θα πολλαπλασιαστεί το αριστερό μέρος του αριθμού, X_L με το $2^{2n/3}$ μιας και χρειάζεται $n/3$ αριστερές μετατοπίσεις για να φτάσει την τάξη του X_M και άλλες $n/3$ αριστερές μετατοπίσεις για να φτάσει την τάξη του X_R. Όμοια και για το X_M, που πολλαπλασιάζεται με το $2^{n/3}$, αλλά και για τον αριθμό Y.

- **Κυρίευε**

$$X\cdot Y = \left(X_L\cdot 2^{2n/3} + X_M\cdot 2^{n/3} + X_R\right)\cdot\left(Y_L\cdot 2^{2n/3} + Y_M\cdot 2^{n/3} + Y_R\right) =$$
$$X_LY_L\cdot 2^{4n/3} + (X_LY_M + X_MY_L)\,2^n + (X_LY_R + X_MY_M + X_RY_L)\,2^{2n/3} + (X_RY_M + X_MY_R)\,2^{n/3} + X_RY_R$$

Σύμφωνα με τις παραπάνω ομαδοποιήσεις, παρατηρώ πώς τα αθροίσματα μπορούν να αναπτυχθουν ώς:

- $X_LY_M + X_MY_L = \underbrace{(X_M+X_L)(Y_L+Y_M)}_{Z_1} - \underbrace{X_MY_M}_{Z_2} - \underbrace{X_LY_L}_{Z_3}$
- $X_LY_R + X_RY_L = \underbrace{(X_L+X_R)(Y_L+Y_R)}_{Z_4} - X_LY_L - \underbrace{X_RY_R}_{Z_5}$
- $X_MY_R + X_RY_M = \underbrace{(X_M+X_R)(Y_M+Y_R)}_{Z_6} - X_RY_R - X_MY_M$

Άρα το γινόμενο μπορεί να μετατραπεί ώς:

$$X\cdot Y = Z_3\cdot 2^{4n/3} + (Z_1 - Z_2 - Z_3)\,2^n + (Z_4 - Z_3 - Z_5 + Z_2)\,2^{2n/3} + (Z_6 - Z_5 - Z_2)\,2^{n/3} + Z_5$$

Παρατηρείται πως πρέπει να γίνουν συνολικά 9 αναδρομές, με είσοδο $n/3$ bits για τον υπολογισμό των παραπάνω γινομένων. Όμως, οι αναδρομές για τον υπολογισμό των Z_1, Z_4, Z_6 είναι οι εξής:

- $Z_1 : X_M Y_L + X_M Y_M + X_L Y_L + X_L Y_M$
- $Z_2 : X_L Y_L + X_M Y_R + X_R Y_L + X_R Y_R$
- $Z_2 : X_M Y_M + X_M Y_M + X_R Y_M + X_R Y_R$

Όμως οι όροι $X_M Y_M$, $X_L Y_L$ και $X_R Y_R$ έχουν ίδη υπολογιστεί αναδρομικά. Άρα οι αναδρομές, για είσοδο $n/3$ bits, που πρέπει να γίνουν πέφτουν στις 6. Επιπλέον αξίζει να σημειωθεί πως στο τέλος του αλγορίθμου θα προστίθενται οι τελικοί όροι με πολυπλοκότητα $\Theta(1)$. Αυτό θα γίνεται για όλα τα n στοιχεία και άρα θα ξεκινάμε από πολυπλοκότητα $n \cdot \Theta(1) = \Theta(n)$

Άρα η πολυπλοκότητα του αλγορίθμου μπορεί να οριστεί ώς:

$$T(n) = 6T(\frac{n}{3}) + \Theta(n)$$

Έστω $T'(n) = 6T(n/3) + n$

με $a = 6$ & $b = 3 \Rightarrow n^{\log_b a} = n^{\log_3 6} \simeq n^{1.63}$

Άρα συγκρίνω $n^{1.63}$ & $f(n) = n$

Case 1: με $f(n) = O(n^{1.63-i})$ για $i \in (0,\ 0.63)$

Άρα:

$$T'(n) = T(n) = \Theta\left((n^{1.63}\right)$$

- **Πόρισμα**
  Διαπιστώνεται πώς με αυτόν τον αλγόριθμο πετυχαίνουμε πολυπλοκότητα μικρότερη της $\Theta(n^2)$ αλλά μεγαλύτερη της $\Theta(n^{log_2 3})$ που είναι η πολυπλοκότητα του γνήσιου αλγορίθμου

5^{η} Εύρεση μέγιστου αθροίσματος υπακολουθίας

Έστω array A, που αποτελείται από n bits, για το οποίο θέλω να βρώ το sub-array με το μέγιστο άθροισμα με πολυπλοκότητα $O(nlogn)$. Ο αλγόριθμος, θα επιστρέφει τα indexes $(i,\ j)$ για τα οποία θα μεγιστοποιείται το παρακάτω άθροισμα:

$$A[i : j] = \sum_{k=i}^{j} A[k]$$

Για την εύρεση τέτοιου αλγορίθμου, θα γίνει χρήση της τακτικής Διαίρει και Κυρίευε, ώστε να διαιρεθεί το array σε δύο subarray όσο το δυνατόν ποιο ισομεγέθεις. Με αυτόν τον τρόπο, βρίσκω το μέσο σημείο του array (έστω mid) και εξετάζονται τα subarray $A[head,\ mid]$ και $A[mid + 1,\ tail]$ (έστω head η αρχή της συμβολοσειράς και tail το τέλος αυτής).

Μπορούν να διακριθούν 3 περιπτώσεις για το που μπορεί να βρίσκεται τη μέγιστη υπακολουθία στο array A μήκους n:

- Εξ ολοκλήρου στο αριστερό subarray $A[head,\ mid]$ με $head \leq i \leq j \leq mid$
- Εξ ολοκλήρου στο δεξί subarray $A[mid + 1,\ tail]$ με $mid + 1 \leq i \leq j \leq mid$
- Μεταξύ του αριστερού και δεξιού μέρους, με $head \leq i \leq mid \leq j \leq tail$

Γνωρίζοντας τι παραπάνω περιπτώσεις θα βρεθεί μια αναδρομική λύση για τον υπολογισμό των $i,\ j$ μιας και οι δύο πρώτες περιπτώσεις είναι μικρότερα στιγμιότυπα του αρχικού προβλήματος. Για την τρίτη περίπτωση $(i \leq mid \leq j)$, παρατηρείται πώς και αυτό το subarray μπορεί να διαιρεθεί σε δύο υπακολουθίες και άρα να δημιουργηθούν τρεις

περιπτώσεις για τις οποίες έγινε αναφορά παραπάνω, από τις οποίες μα ενδιαφέρει αυτή που επιστρέφει το μέγιστο άθροισμα.

Επομένως, μπορούν να οριστούν οι συναρτήσεις:

- $findMaxCrossingSubArray(a,\ head,\ tail,\ mid)$

  Με την χρήση αυτής, βρίσκεται από μια μέγιστη υποσυστοιχία, για το αριστερό μισό και το δεξί μισό.

  Ο ψευδοκώδικας για την παραπάνω συνάρτηση:

```
1  findMaxCrossingSubArray(a, head, tail, mid){
2
3      max_left = -1
4      max_right = -1
5
6      left_sum  = - \infty
7      right_sum = - \infty
8
9      current_sum = 0
10     for i = mid downto head{
11
12         current_sum += a[i]
13
14         if ( current_sum > left_sum){
15             left_sum = current_sum  // Store sum
16             max_left = i    // Store index
17         }
18     }
19
20     current_sum = 0
21     for i = (mid + 1) to tail{
22
23         current_sum += a[i]
24
25         if ( current_sum > right_sum){
26             right_sum = current_sum
27             max_right = i
28         }
29     }
30
31     return  max_left, max_right, (left_sum + right_sum)
32 }
```

  Στις αρχικές γραμμές βλέπουμε τις αρχικοποιήσεις για τις μεταβλητές max_left και max_right, που κρατούν τα index που μας ενδιαφέρουν (αρχικοποιούνται στο -1), αλλά και τις μεταβλητές που θα κρατούν τα αντίστοιχα αθροίσματα.

  Λόγου του ότι θα συγκρίνουμε αριθμούς, για τους οποίους θέλουμε το μέγιστο άθροισμα, θα πρέπει οι πρώτες συγκρίσεις (γραμμές 14, 25) να γίνουν με το μικρότερο δυνατό αριθμό (έστω $-\infty$). Κατά την εύρεση του αριστερού subarray (γραμμές $9 \rightarrow 18$), ο μετρητής, ξεκινά από την τιμή mid και μειώνει μέχρι την τιμή $head$ και εξετάζει array της μορφής $A[i : mid]$. Κάθε στοιχείο του array προστίθεται στο προηγούμενο άθροισμα (current_sum), ώστε να είναι γνωστό πάντα το συνολικό άθροισμα. Το τελευταίο, ελέγχεται με το left_sum και στην περίπτωση που είναι μεγαλύτερο αυτού, αποθηκεύεται ως νέο left_sum όπως αποθηκεύεται και το index της τιμής αυτής (γραμμές $9 \rightarrow 18$). Με παρόμοια διαδικασία συμπεριφέρεται και για την εύρεση του δεξί subarray (γραμμές $20 \rightarrow 29$).
  Τέλος επιστρέφονται τα index max_left και max_right αλλά και το άθροισμα το οποίο δημιουργούν.

  Αν το array a αποτελείται από n στοιχεία και δεδομένου ότι κάθε επανάληψη, του κάθε βρόχου απαιτεί πολυπλοκότητα $\Theta(1)$ μπορεί να μετρηθεί η συνολική πολυπλοκότητά του. Για την ανεύρεση του αριστερού μέρους, θα πραγματοποιηθούν $mid - head + 1$ επαναλήψεις, ενώ για την ανεύρεση του δεξιού μέρους θα πραγματοποιηθούν $tail - mid$ επαναλήψεις. Μπορεί να παρατηθεί, ότι οι συνολικές επαναλήψεις είναι n μιας και $\cancel{mid} - head + 1 + tail - \cancel{mid} = head + tail + 1 = n$.

  Άρα η συνάρτηση απαιτεί χρόνο $\Theta(n)$.

- $findMaxSubArray(a,\ head,\ tail)$

  Με αρχική κλίση $findMaxSubArray(a,\ 0,\ length(a))$ ο αλγόριθμος εκτελεί την μέθοδο Διαίρει και Κυρίευε επιστρέφει τα index $i,\ j$.

  Ο ψευδοκώδικας για την παραπάνω συνάρτηση:

```
1  findMaxSubArray(a, head, tail) {
2
3      if (head == tail)
4          return (head, tail, a[head])
5      else{
6          mid = (head + tail)/2
7          // Case 1
8          (left_head, left_head, left_sum) = findMaxSubArray(a, head, mid)
9          // Case 2
10         (right_head, right_head, right_sum) = findMaxSubArray(a, mid + 1, tail)
11         // Case 3
12         (cr_head, cr_head, cr_sum) = findMaxCrossingSubArray(a, head, tail, mid)
13
14         // Max subarray at left
15         if (left_sum >= right_sum + left_sum >= cr_sum)
16             return (left_head, left_tail)
17         // Max subarray at right
18         else if (right_sum >= right_sum + left_sum >= cr_sum)
19             return (right_head, right_tail)
20         // Max subarray between
21         else
22             return (cr_head, cr_tail)
23     }
24 }
```

  Στις γραμμές $(3 \to 4)$ γίνεται ο έλεγχος για το αν το array αποτελείται από ένα μόνο στοιχείο. Στην περίπτωση αυτή, που το τελικό subarray είναι το ίδιο το στοιχείο και επιστρέφονται οι δείκτες αυτής (είναι ίδιοι) αλλά και η τιμή του. Στις γραμμές $(6 \to 10)$ υλοποιείται η αναδρομικές περιπτώσεις. Γνωρίζοντας το μέσο (mid) εκτελείτε το Διαίρει και στην συνέχεια υπολογίζονται τα μέγιστα subarray αριστερά και δεξιά του μέσου, τα οποία θα αποτελούνται από τουλάχιστον ένα στοιχείο (Κυρίευε). Στην γραμμή (12) υπολογίζονται τα μέγιστα subarray εκατέρωθεν του μέσου. Στις γραμμές $(14 \to 22)$ ελέγχεται, ποια περίπτωση έχει επιστρέψει μεγαλύτερη τιμή και επιστρέφονται οι αντίστοιχοι δείκτες.

  Σε αυτή την περίπτωση (με έστω array a με n στοιχεία) παρατηρείται πώς γίνονται δυο αναδρομές (γραμμές 8 και 10) με μέγεθος εισόδου $n/2$ bits Άρα θα χρειαστούν χρόνο ίσο με $2T(\frac{n}{2})$. Αν πάρουμε υπόψιν και τον χρόνο που απαιτεί η γραμμή 12 η συνολική πολυπλοκότητα θα γίνει $2T(\frac{n}{2}) + \Theta(n)$. Οι υπόλοιπες γραμμές του αλγορίθμου προσφέρουν στην συνολική πολυπλοκότητα ένα σταθερό κόστος πράγμα το οποίο παραλείπεται.

  Έτσι φτάνουμε στην σχέση:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

  Γνωρίζοντας την πολυπλοκότητα του Merge Sort που είναι ίδια με αυτήν του παραδείγματος μπορώ να ισχυριστώ πώς:

$$T(n) = \Theta(n \log_2 n)$$

  (Με την χρήση του αλγορίθμου του Kadane θα πετυχαίνουμε πολυπλοκότητα $T(n) = \Theta(n)$)

6^{η} Percentiles