Η Αλγοριθμική Πολυπλοκότητα στο Γενικό Λύκειο

Αραμπατζής Γιώργος
Καθηγητής Πληροφορικής, Πειραματικό ΓΕΛ ΠΑΜΑΚ
arabatzis4@gmail.com

ΠΕΡΙΛΗΨΗ

Η μελέτη της πολυπλοκότητας των αλγορίθμων είναι ένα διδακτικό αντικείμενο το οποίο υπάρχει στο βιβλίο της Ανάπτυξης Εφαρμογών σε Προγραμματιστικό Περιβάλλον, μάθημα που εντάχθηκε από τις αρχές του 2000 στο Λύκειο, και συγκεκριμένα στην Γ' Λυκείου. Ποτέ όμως το διδακτικό αυτό αντικείμενο δεν αποτέλεσε μέρος του Αναλυτικού Προγράμματος Σπουδών (ΑΠΣ). Τα τελευταία δυο χρόνια έγιναν δυο απόπειρες να διδαχθεί. Μια στη Β' Λυκείου, κατά στο Σχολικό Έτος 2014-15, όπου και η διδασκαλία του ακυρώθηκε, και μια την φετινή χρονιά, 2015-16 στη Γ" Λυκείου, όπου και αποτελεί διδακτέα ύλη τόσο για την Ομάδα Προσανατολισμού Θετικών Επιστημών, όσο και για την Ομάδα Προσανατολισμού Σπουδών Οικονομίας και Πληροφορικής, και περιλαμβάνεται ως εξεταστέα ύλη για τις πανελλαδικές εξετάσεις της δεύτερης. Πόσο εύκολο όμως είναι για τους μαθητές να κατανοήσουν στοιχεία μιας πλευράς της Πληροφορικής όπως η Θεωρητική Πληροφορική; Θα καταλάβουν την ανάγκη κατανόησής της για κάποιον που ασχολείται με την συγγραφή αλγορίθμων σε οποιαδήποτε πραγματική γλώσσα προγραμματισμού ; Και τέλος ποιος θα μπορούσε να είναι ένας τρόπος ήπιας προσέγγισης εννοιών όπως του Συμβολισμού Ο, με τέτοιο τρόπο ώστε να γίνει εύκολα κατανοητός από τους μαθητές ;

Αυτά τα ερωτήματα είναι τόσο στο μυαλό πολλών καθηγητών πληροφορικής, αλλά και ακόμη περισσότερων μαθητών, οι οποίοι αμφισβητούν ακόμη και την σκοπιμότητα της διδασκαλίας θεωρητικών εννοιών της Πληροφορικής στο Λύκειο. Τα ερωτήματα αυτά επιχειρεί να απαντήσει η εισήγηση αυτή, να περιγράψει και να συνοψίσει με απλό τρόπο παραδείγματα και ορισμούς σχετικούς με την αλγοριθμική πολυπλοκότητα.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *Αλγοριθμική Πολυπλοκότητα Ασυμπτωτικό όριο,Συμβολισμός Ο*

ΕΙΣΑΓΩΓΗ

Η διδασκαλία των αλγορίθμων και των βασικών τους δομών άρχισε από τις αρχές της προηγούμενης δεκαετίας όταν και εντάχθηκε επίσημα το μάθημα της Ανάπτυξης Εφαρμογών σε Προγραμματιστικό Περιβάλλον, στα πανελλαδικώς εξεταζόμενα μαθήματα. Ενώ στο ΑΠΣ μέχρι τώρα διδάσκονταν οι βασικές ενότητες που αφορούσαν τις Προγραμματιστικές Δομές των αλγορίθμων σε επίπεδο εντολών, και απλών δομών, όπως οι στατικοί πίνακες, καθώς και κάποιες ιστορικές και γενικές έννοιες (Κεφάλαιο 6), ποτέ δεν έγινε προσπάθεια να ενταχθεί στο ΑΠΣ το κεφάλαιο 5 από το Βιβλίο Μαθητή, το οποίο αφορούσε στοιχεία Θεωρίας Υπολογιστικής Πολυπλοκότητας (Βακάλη, κ.α, 2000).

Το πρώτο ερώτημα - και εύλογα - που θα ήταν στο μυαλό κάθε μαθητή, θα ήταν γιατί θα έπρεπε να ασχοληθούν με ένα τόσο θεωρητικό αντικείμενο, την στιγμή που οι περισσότεροι μαθητές έχουν κατανοήσει πλήρως τις αλγοριθμικές έννοιες και έχουν την δυνατότητα να γράφουν σωστά και ολοκληρωμένα προγράμματα, όπως απαιτεί το μάθημα ;

Η απάντηση σε αυτό το ερώτημα έγινε γνωστή μόνο σε μαθητές που συμμετείχαν σε Διαγωνισμούς Πληροφορικής, Βαλκανιάδες και Ολυμπιάδες όπου προσέγγισαν την ομορφιά της τέχνης του Προγραμματισμού σε μεγαλύτερο βάθος. Όπως άλλωστε είπε και ο D. Knuth "*Programming is an Art*". Στους διαγωνισμούς αυτούς δεν ήταν αρκετή η συγγραφή ενός προγράμματος απλά που να ικανοποιεί κάποιο σετ δεδομένων εισόδου (Στιγμιότυπο ή Instance). Πολλοί μαθητές έλυναν το πρόβλημα προγραμματιστικά, έπαιρναν σωστά αποτελέσματα, αλλά το πρόγραμμα που υπέβαλαν σαν λύση στο πρόβλημα του διαγωνισμού, δεν γίνονταν αποδεκτό από τις μηχανές ελέγχου του διαγωνισμού τους λεγόμενους Profilers.

Η διαδικασία του Profiling, περιλαμβάνει δυναμική ανάλυση του κώδικα μετρώντας τις απαιτήσεις του σε χώρο μνήμης, χρόνο, πολυπλοκότητα εντολών, χρήση συγκεκριμένων εντολών και συχνότητα κλήσης υποπρογραμμάτων. Εκεί λοιπόν οι μαθητές καταλάβαιναν ότι χρειάζεται κάτι παραπάνω για να γίνει αποδεκτό το πρόγραμμά τους. Και αυτό ήταν η δημιουργία ενός γρηγορότερου και αποδοτικότερου προγράμματος το οποίο να μπορεί να ανταπεξέρχεται σε μεγάλη ποικιλία στιγμιότυπων, τόσο ποιοτικά όσο και ποσοτικά. Άρα το πρόγραμμά τους, έπρεπε να βελτιώσει την λειτουργία του στην χειρότερη περίπτωση δεδομένων.

Μήπως η μελέτη της συμπεριφοράς των αλγορίθμων λοιπόν δεν είναι ένα ασήμαντο θέμα, αλλά από τα θεμελιώδη, μιας και αποτελεί σημαντικό παράγοντα στη αποδοχή ενός αποδοτικού αλγορίθμου; Ειδικά όταν ο αλγόριθμος πρόκειται να χρησιμοποιηθεί σε πραγματικές συνθήκες, όπως σε μια εφαρμογή, σε μια επιχείρηση, τράπεζα κλπ, όπου οι πόροι και το υλικό δεν είναι άπειρα, ο χώρος και ο χρόνος μεταφράζονται σε χρήμα και αποτελεσματικότητα, και η εφαρμογή με προβληματικό χρόνο εκτέλεσης μπορεί να επηρεάσει την λειτουργικότητα μιας επιχείρησης ή μιας υπηρεσίας ;

ΟΡΙΣΜΟΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Στην παράγραφο 5.3 του βιβλίου μαθητή (Βακάλη, κ.α, 2000), αναφέρεται ο ορισμός της πολυπλοκότητας των αλγορίθμων, για το άνω φράγμα της πολυπλοκότητας ενός αλγορίθμου, Σχήμα 1.

ΟΡΙΣΜΟΣ

Αν η πολυπλοκότητα ενός αλγορίθμου είναι f(n), τότε λέγεται ότι είναι τάξης O(g(n)), αν υπάρχουν δύο θετικοί ακέραιοι c και n_0, έτσι ώστε για κάθε $n \geq n_0$ να ισχύει:

$$|f(n)| \leq c\,|g(n)|$$

Σχήμα 1: Ορισμός της Πολυπλοκότητας στο Βιβλίο Μαθητή ΑΕΠΠ της Γ΄ Λυκείου

Για να μπορέσει ο μαθητής να καταλάβει τον παραπάνω ορισμό καλό θα ήταν πρώτα να διδαχθεί ένα πλήρες αναλυτικό παράδειγμα. Επίσης θα ήταν απαραίτητο να συμπεριλάβουμε και μια οπτική απεικόνιση των συναρτήσεων f(x) και g(x), για να γίνει περισσότερο κατανοητός από τον μαθητή. Πριν προχωρήσουμε στο παράδειγμα ας δούμε κάποιους ορισμούς που σχετίζονται με το θέμα που μελετάμε.

Η O(g(n)), συμβολίζει το πάνω ασυμπτωτικό όριο της f(x). Υπάρχουν και άλλοι δύο συμβολισμοί οι οποίοι είναι

- Ω(g(n)) συμβολίζει το κάτω ασυμπτωτικό όριο πολυπλοκότητας.
- Θ(f(n)) συμβολίζει την μέσο ασυμπτωτικό όριο πολυπλοκότητας, και το οποίο βρίσκεται πιο κοντά στην f(x).

ΚΑΤΗΓΟΡΙΕΣ ΤΟΥ ΣΥΜΒΟΛΙΣΜΟΥ O

Οι περισσότεροι αλγόριθμοι που μας ενδιαφέρουν πρακτικά τους κατατάσσουμε στις εξής κατηγορίες ανάλογα με τον χρόνο εκτέλεσής τους (Dasgupta et al, 2009). Κάποιες από τις πιο συνηθισμένες κατηγορίες είναι οι εξής :

1. **O(1). Σταθερή Πολυπλοκότητα.**
   Στην κατηγορία αυτή ανήκουν οι εντολές της ακολουθιακής δομής, όπως οι εντολές εισόδου, εξόδου ή εκχώρησης.
   Επίσης στην κατηγορία αυτή ανήκουν οι εντολές επανάληψης (Για ..., Όσο ..., Αρχή επανάληψης ...), οι οποίες τελειώνουν σε συγκεκριμένο αριθμό επαναλήψεων (Για i από 1 μέχρι 100), γιατί θεωρούνται πεπερασμένος αριθμός εντολών, δεν υπάρχει δηλαδή τάση της επανάληψης να τείνει στο άπειρο. Π.χ

   _ Διάβασε Cnt
   _ Pnt <-- Pnt +1

_ Για i από 1 μέχρι 200
*α <-- i * 2*
Τέλος_επανάληψης

2. **O(logn). Λογαριθμική Πολυπλοκότητα.**
   Στην κατηγορία αυτή ανήκουν τμήματα αλγορίθμων που υλοποιούν την Δυαδική Αναζήτηση ή προσπελάσουν Δυαδικά Δέντρα. Μιας και η Δυαδική Αναζήτηση είναι στην ύλη της Γ' Λυκείου (Γενικές Οδηγίες, 2015), θα ήταν χρήσιμο να αναλύσουμε την περίπτωση της, όπως κάναμε παρακάτω.
   Επίσης στην κατηγορία αυτή ανήκουν τμήματα αλγορίθμων που περιέχουν εντολές Δομής Επανάληψης (Για ..., Όσο ..., Αρχή επανάληψης ...) όπου όμως το βήμα δεν αυξάνει κάθε φορά με πρόσθεση ή αφαίρεση σταθερού μεγέθους (κατά 1 ή άλλο σταθερό αριθμό) αλλά με γινόμενο ή πηλίκο αριθμου (πχ i/2). Π.χ.
   _ Για i από n μέχρι 1 με_βήμα i / 2
3. **O(n). Γραμμική Πολυπλοκότητα.**
   Στην κατηγορία αυτή ανήκουν τμήματα αλγορίθμων που υλοποιούν την Δομή Επανάληψης. Όπως για παράδειγμα η επανάληψη **Για ...,** η οποία εκτελείται **n** φορές με σταθερό βήμα. Π.χ.
   _ Για i από 1 μέχρι n ...
4. **O($n^κ$). Τετραγωνική, Κυβική Πολυπλοκότητα.**
   Στην κατηγορία αυτή ανήκουν τμήματα αλγορίθμων που υλοποιούν την Δομή Επανάληψης με εμφώλευση. Αν δηλαδή έχουμε δύο εμφωλευμένες επαναλήψεις έχουμε πολυπλοκότητα O(n^2) ή Τετραγωνική Πολυπλοκότητα . Π.χ.
   _ Για i από 1 μέχρι n
   Για j από 1 μέχρι n
   Εντολή (ες)
   Τέλος_επανάληψης
   Τέλος_επανάληψης
   Εδώ να υπογραμμίσουμε ότι η μεταβλητή k, η οποία είναι ο εκθέτης του n, διαμορφώνεται ανάλογα με το βάθος της εμφώλευσης. Αν δηλαδή έχω τρεις εμφωλευμένες επαναλήψεις, τότε η πολυπλοκότητα είναι O(n^3), δηλαδή έχω Κυβική Πολυπλοκότητα κλπ.
5. **O(2^n). Εκθετική Πολυπλοκότητα.**
   Στην κατηγορία αυτή ανήκουν αλγόριθμοι με τεράστιους χρόνους εκτέλεσης και συνήθως η ύπαρξή τους είναι μόνο στο χαρτί. Δεν υλοποιούνται στην πραγματικότητα λόγω του μεγάλου χρόνου εκτέλεσής τους. Αποτελούν όμως περιπτώσεις για μελέτη.

Συνοψίζοντας, η ταξινόμηση των κατηγοριών από την μικρότερη προς την μεγαλύτερη πολυπλοκότητα είναι :

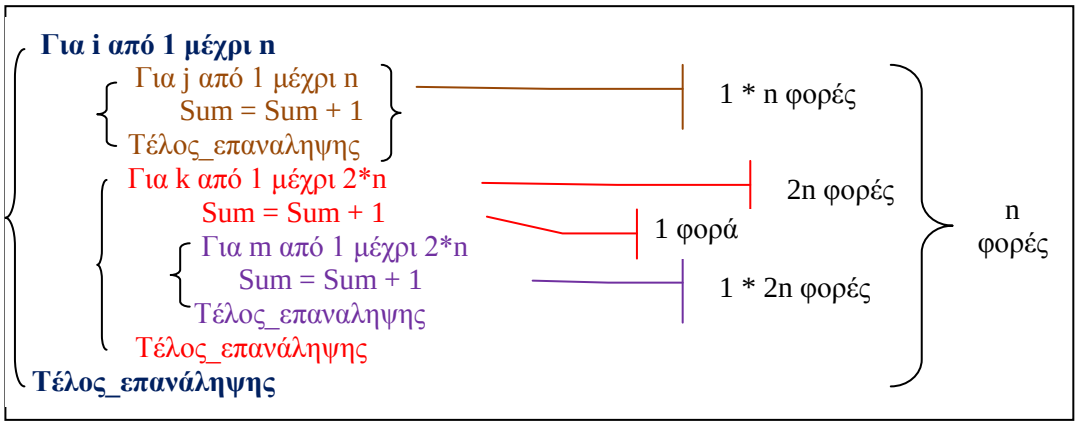
- O(1) Σταθερή
- O(loglogn) Διπλή Λογαριθμική
- O(logn) Λογαριθμική
- O($(logn)^c$), c>1 Πολυλογαριθμική
- O(n^c) ,0<c<1 Δεκαδική δύναμη
- O(n) Γραμμική
- O(nlogn) Εν λογκ εν
- O(n^2) Τετραγωνική
- O(n^c) Πολυωνυμική ή Αλγεβρική

ΠΑΡΑΔΕΙΓΜΑ ΜΕΛΕΤΗΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Ας περιγράψουμε ένα τμήμα αλγορίθμου (Σχήμα 2), το οποίο περιέχει κάποιες από τις παραπάνω μορφές. Η πλέον συνηθισμένη πρακτική είναι η μέτρηση του αριθμού των βασικών πράξεων που θα πρέπει να εκτελέσει ο αλγόριθμος στη χειρότερη περίπτωση, Για παράδειγμα μια βασική πράξη θα μπορούσε να είναι :

- ανάθεση τιμής
- σύγκριση μεταξύ δύο μεταβλητών
- οποιαδήποτε αριθμητική πράξη μεταξύ δύο μεταβλητών.

Για να βρούμε την χειρότερη περίπτωση του αλγορίθμου, θα χρησιμοποιήσουμε τιμές ως είσοδο στον αλγόριθμο, που θα τον οδηγούν στην εκτέλεση του μέγιστου αριθμού βασικών πράξεων (Βακάλη κ.α, 2000).



Σχήμα 2: Παράδειγμα Πολυπλοκότητας

Συνολικός αριθμός επαναλήψεων : $\mathbf{n\,(1 * n + 2n\,(1 + (1 * 2n)))}$

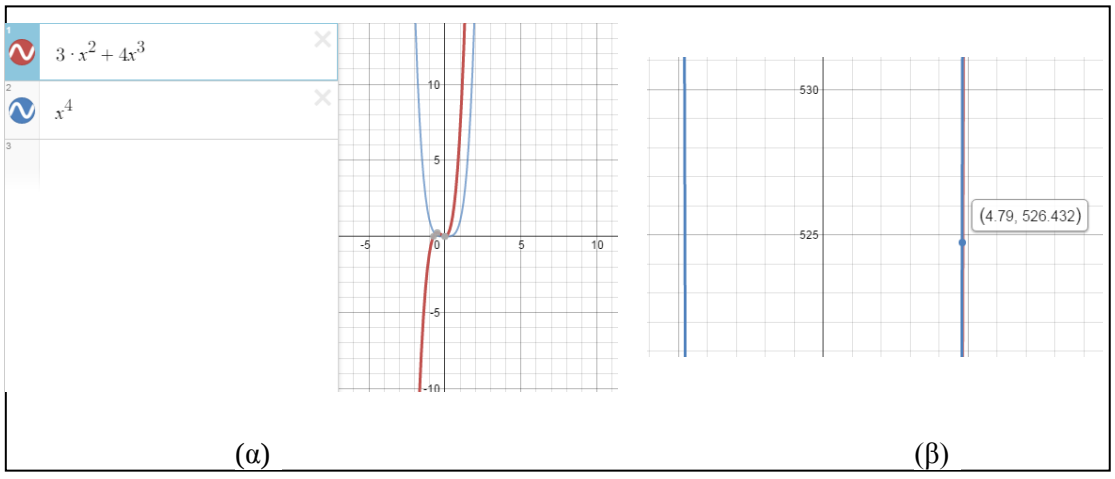
$= \mathbf{n\,(\,n + 2n\,(1 + 2n)}$

$= \mathbf{n^2 + 2n^3 + 4n^3 = 3n^2 + 4n^3}$

Άρα η συνάρτηση πολυπλοκότητας του τμήματος αλγορίθμου είναι η $f(n) = 3n^2+4n^3$. Αρκεί λοιπόν να βρούμε μια συνάρτηση g(n) η οποία μετά από κάποιο σημείο θα "υπερβαίνει" την f(n). Άρα, $\mathbf{(fn) = 3n^2 + 4n^3 <= 3n^3 + 4n^3 <= 7n^3}$ **.** Η συνάρτηση $\mathbf{g(n)=7n^3}$, σίγουρα θα είναι ένα επάνω όριο της f.

Και επειδή στο συμβολισμό Ο, οι σταθεροί συντελεστές δεν συνεισφέρουν στην τάση της συνάρτησης όσο αυτή τείνει στο άπειρο, η τελική μορφή της **g(n)** μπορεί να θεωρηθεί η $\mathbf{g(n)= n^3}$**.** Ο μεγαλύτερος, (εδώ και μοναδικός), όρος του πολυωνύμου είναι τρίτου βαθμού, λέμε λοιπόν, ότι η χειρότερη περίπτωση πολυπλοκότητας της f, είναι $O(n^3)$ (Φωτάκης & Σπυράκης, 2001).

Πέρα από τα μαθηματικά, καλό θα ήταν ο μαθητής να έχει και μια οπτική αντιπαραβολή των δύο αυτών συναρτήσεων για να μπορεί να φανταστεί τι ακριβώς είναι το πάνω ασυμπτωτικό όριο της πολυπλοκότητας του αλγοριθμικού τμήματος. Τον συνιστούμε λοιπόν να συνδεθεί σε έναν ιστοχώρο για online δωρεάν γραφικές αναπαραστάσεις συναρτήσεων όπως το Desmos graph calculator. (Σχήμα 3)



Σχήμα 3: Γραφικές παραστάσεις των f(n), g(n)

Στο Σχήμα 3, βλέπουμε ότι αρχικά η f(n), ανεβαίνει ταχύτερα της g(n) (Σχήμα 3, α). Είναι δηλαδή πιο κοντά στον άξονα των Y, και φαίνεται ότι η g(n) δεν αποτελεί ασυμπτωτικό πάνω όριο. Αυτό οφείλεται στο γεγονός ότι για μικρές τιμές του x, ο όρος $\mathbf{3n^2}$, συνεισφέρει σημαντικά στην διαμόρφωση της f(x), υπερισχύοντας της g(x).

Αν όμως οι τιμές του x υπερβούν κάποια συγκεκριμένη τιμή, θα διαπιστώσουμε ότι η g(n), εμφανίζεται εσωτερικά της f(n), δείχνοντας ταχύτερη άνοδο, γιατί η **g(n)** υπερισχύει. Για κάθε n λοιπόν μεγαλύτερο του σημείου $X_0 = 4,79$, που η g(n) καλύπτει την f(x), είναι το σημείο πέραν του οποίου η g(n) είναι ασυμπτωτικό πάνω όριο της f(n) (Σχήμα 3, β).

Οι μαθητές αν επιχειρούσαν να δουν σε αντιπαραβολή την γραφική των δυο συναρτήσεων, θα μπορούσαν πολύ εύκολα να κατανοήσουν τον ορισμό της πολυπλοκότητας, που δείχνει τόσο θεωρητικός και αφηρημένος.

ΥΠΟΛΟΓΙΣΜΟΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ ΕΠΑΝΑΛΗΨΕΩΝ

Η πιο βασική εντολή που απαιτεί κάποια προσοχή, όσο αφορά τον υπολογισμό της πολυπλοκότητας είναι η επανάληψη. Οι απλές εντολές, όπως για παράδειγμα οι εντολές ανάθεσης, είναι εύκολο να σκεφτούμε ότι είναι βαθμού O(1). Ας δούμε κάποιες βασικές παραδοχές για τον υπολογισμό της πολυπλοκότητας στις επαναλήψεις.

- Χρησιμοποιούμε πάντα ακέραια διαίρεση.
- Αν ο αριθμός των επαναλήψεων της επανάληψης είναι n τότε :
  - Η εντολή αρχικοποίησης της μεταβλητής εκτελείται μια φορά
  - Η συνθήκη της επανάληψης εκτελείται n + 1 φορές
  - Κάθε εντολή στο εσωτερικό της επανάληψης εκτελείται n φορές
  - Η εντολή που ενημερώνει τον δείκτη της επανάληψης εκτελείται n φορές.
- Μια επανάληψη λέγεται **γραμμική** αν ο δείκτης αυξάνεται είτε με πρόσθεση είτε με αφαίρεση κάποιας τιμής
- Μια επανάληψη λέγεται **ανεξάρτητη** αν ο δείκτης είναι ανεξάρτητος από κάποια εξωτερική επανάληψη, και δεν επηρεάζεται από αυτή.

Περίπτωση $1^η$: Σταθερή επανάληψη

Στην περίπτωση αυτή ο βαθμός πολυπλοκότητας (Β.Π.) είναι O(1), επειδή οι επαναλήψεις είναι πεπερασμένες (Πίνακας 1). Όταν ο αριθμός είναι ιδιαίτερα μεγάλος μπορεί να προσομοιάσει Β.Π. O(n).

i ← 0
Όσο i < 200 επανάλαβε
  Εμφάνισε "Γεια σου κόσμε !"
  i ← i + 1
Τέλος_επανάληψης
	Εντολή αρχικοποίησης : 1 φορά
	Η συνθήκη τερματισμού : 201 φορές
	Εντολή Εμφάνισης : 200 φορές
	Εντολή ανανέωσης βήματος : 2*200 φορές
	Σύνολο 802 φορές

Πίνακας 1: Πολυπλοκότητα Σταθερής Επανάληψης

Περίπτωση $2^η$: Γραμμική επανάληψη με ανοιχτό Όριο

Στην επανάληψη αυτή δεν συμπεριλαμβάνεται το όριο τερματισμού, και ο Β.Π. είναι O(n).

Για μη αρνητικούς κ και n όπου n >= k	

i ← k
Όσο i < n επανάλαβε
  Εμφάνισε "Γεια σου κόσμε !"
  i ← i + 1
Τέλος_επανάληψης
	Εντολή αρχικοποίησης : 1 φορά
	Η συνθήκη τερματισμού : (n-k) + 1
	Εντολή Εμφάνισης : n-k
	Εντολή ανανέωσης βήματος : n-k

Πίνακας 2: Πολυπλοκότητα Επανάληψης με Ανοιχτό Όριο

Εδώ, έχουμε 1 + (n-k) + 1 + (n-k) + (n-k) = 3(n-k) + 2 = 3n -3k + 2 <= 3n. Συνεπώς ο Β.Π. είναι O(n). Πίνακας 2.

Περίπτωση 3η : Γραμμική επανάληψη με κλειστό Όριο
Εδώ το όριο τερματισμού συμπεριλαμβάνεται στην επανάληψη. Στην περίπτωση αυτή εμπίπτει η επανάληψη **Για …** ο Β.Π. είναι επίσης O(n).

Για μη αρνητικούς κ και n όπου n >= k	
i ← k Όσο i < = n επανάλαβε Εμφάνισε "Γεια σου κόσμε !" i ← i + 1 Τέλος_επανάληψης	Εντολή αρχικοποίησης : 1 φορά Η συνθήκη τερματισμού : (n-k) + 2 Εντολή Εμφάνισης : (n-k) + 1 Εντολή ανανέωσης βήματος : (n-k) + 1

Πίνακας 3: Πολυπλοκότητα Επανάληψης με Κλειστό Όριο

Στην περίπτωση 2, έχουμε 1 + (n-k) + 2 + (n-k) + 1 + (n-k) + 1 = 3(n-k) + 4 = 3n -3k + 4 <= 4n. Συνεπώς ο Β.Π. είναι O(n). Πίνακας 3.

Περίπτωση 4η : Επανάληψη Λογαριθμικής Πολυπλοκότητας
Εδώ ο Β.Π. είναι O(logn). Χωρίς απώλεια της γενικότητας, θεωρούμε ότι το n είναι πολλαπλάσιο του 2, η αλλιώς $\mathbf{n = 2^k}$. Στην περίπτωση αυτή για το πλήθος των εντολών, έχουμε λόγω της προηγούμενης παραδοχής μας για το n, 1 +1 + 1 + . . . +1 = k + 1 = $\log_2 n$ + 1, και ο Β.Π. είναι O($\log_2 n$). Πίνακας 4.

Για μη αρνητικούς κ και n όπου n >= k		
i ← n Όσο i >= 1 επανάλαβε Εμφάνισε "Γεια σου κόσμε !" i ← i / 2 Τέλος_επανάληψης		
Επανάληψη	Τιμή i στην αρχή κάθε επανάληψης	Φορές που η εντολή ΓΡΑΨΕ εκτελείται σε κάθε επανάληψη
1	n	1
2	$n / 2^1$	1
3	$n / 2^2$	1
…	….	…
k + 1	$n / 2^k$	1
k + 2	$n / 2^{(k+1)}$	0 (Τέλος επανάληψης)

Πίνακας 4: Λογαριθμική Πολυπλοκότητα Επανάληψης

Περίπτωση 5η : Εμφωλευμένες Ανεξάρτητες Επαναλήψεις
Συχνά οι επαναλήψεις είναι εμφωλευμένες, χωρίς να εμπλέκεται ο ένας δείκτης με τον άλλο (ανεξάρτητες). Στο συγκεκριμένο παράδειγμα, ο Β.Π. είναι O($\log_2 n$). Πίνακας 5.
Ο αριθμός των βασικών εντολών είναι :
1 + ($\log_2 n$ + 2) + ($\log_2 n$ + 1) [2+3+(n+2) + (n+1)(3 + 2)] = 6n$\log_2 n$+6n+13$\log_2 n$+15.
Ή αλλιώς, Συνολικό κόστος = Αρχικοποίηση και χρόνος εντολών επιλογής της έξω επανάληψης + Αριθμός εξωτερικών επαναλήψεων * [Κόστος ενημέρωσης της εξωτερικής επανάληψης + κόστος της εσωτερικής επανάληψης + κόστος των άλλων εντολών της εξωτερικής επανάληψης). Σχήμα 4.

Για i από 1 μέχρι n με_βήμα i * 2 k ← n sum ← 0 Όσο k > 0 επανάλαβε sum ← sum + (i+k) k ← k - 1 Τέλος_επανάληψης print sum Τέλος_επανάληψης	Ο αριθμός επαναλήψεων της εξωτερικής επανάληψης είναι $\log_2 n + 1$. Για κάθε εξωτερική επανάληψη, οι εσωτερικές εντολές θα επαναληφθούν n+1 φορές.

Πίνακας 5: Πολυπλοκότητα Εμφωλευμένης Ανεξάρτητης Επανάληψης

Περίπτωση 6η : Εμφωλευμένες Εξαρτημένες Επαναλήψεις
Εδώ οι επαναλήψεις είναι εμφωλευμένες, και ο εσωτερικός δείκτης εξαρτάται από τον εξωτερικό. Ο Β.Π. είναι $O(n^2)$.

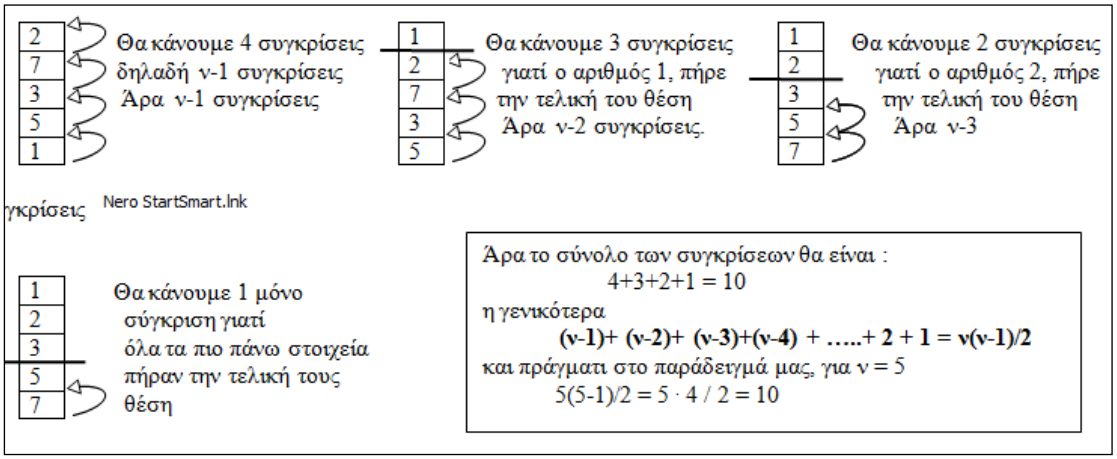
Έστω ότι το n είναι πολλαπλάσιο του 2 ή αλλιώς **n = 2^k.**	
i ← 1 Για i από 1 μέχρι n sum ← 0 Για κ από 1 μέχρι i sum ← sum + (i + k) Τέλος_επανάληψης print (sum) Τέλος_επανάληψης	Οι επαναλήψεις της εξωτερικής επανάληψης είναι **n**. Για όλες τις εξωτερικές επαναλήψεις οι εσωτερικές θα είναι : 1+2+3+4+ …..+(n+1) + 0= n((n+1) + 2)/2 $n^2/2+3n/2$

Πίνακας 6: Πολυπλοκότητα Εμφωλευμένης Ανεξάρτητης Επανάληψης

Άρα συνολικά η $f(n) = 1 + (n+1) + n(2+1+1+1)+ [n^2/2+3n/2]+5n(n+1)/2 = 3n^2 + 10n + 2$,
Ή αλλιώς το σύνολο του κόστους των εντολών θα είναι Συνολικό_Κόστος = Κόστος της εσωτερικής επανάληψης + Κόστος των άλλων εντολών της εξωτερικής επανάληψης + αρχικοποίηση, ενημέρωση μεταβλητών, και των εντολών επιλογής των δύο επαναλήψεων.

ΤΑΞΙΝΟΜΗΣΗ ΕΥΘΕΙΑΣ ΑΝΤΑΛΛΑΓΗΣ

Μετά τα παραδείγματα απλών γενικών τμημάτων αλγορίθμων, και αφού οι μαθητές κατανοήσουν τις βασικές έννοιες του συμβολισμού Ο, και της πολυπλοκότητας, συνήθης είναι η επόμενη ερώτησή τους.



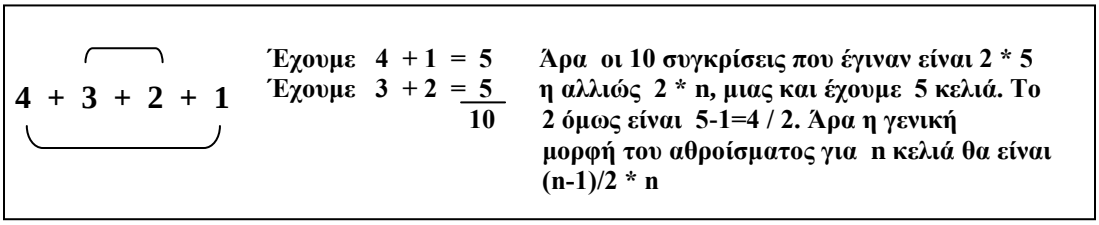
Σχήμα 4: Ταξινόμηση Ευθείας Ανταλλαγής

Και αυτή είναι, πως εξηγείται, η πολύ γνωστή ταξινόμηση σε αυτούς, αυτή της φυσαλίδας είναι **O(n^2),** και η δυαδική αναζήτηση -που φέτος έμαθαν- είναι **log(n)**.

Η απάντηση στο ερώτημα αυτό είναι χρήσιμη γιατί φέρνει τον μαθητή πιο κοντά στην κατανόηση της πολυπλοκότητας, μιας και μαθαίνει να την ανακαλύπτει σε κάτι γνώριμο σε αυτόν, όπως η φυσαλίδα και η δυαδική αναζήτηση (Σχήμα 4).

Εδώ παρατηρούμε ότι στο "πρώτο πέρασμα" του πίνακα κάνουμε 4 συγκρίσεις. Στο δεύτερο 3 συγκρίσεις, στο τρίτο 2 συγκρίσεις και στο τέταρτο 1 σύγκριση. Άρα ο συνολικός αριθμός βασικών πράξεων (συγκρίσεις) είναι 10. Ποια όμως είναι η συνάρτηση που θα μου δώσει τον αριθμό συγκρίσεων για την χειρότερη περίπτωση πλήθους δεδομένων, δηλαδή n ;

Στο Σχήμα 5, δείχνοντας στον μαθητή πως διαμορφώνεται το άθροισμα για n=5, δηλαδή για 5 κελιά, τον οδηγούμε με απλές πράξεις να οδηγηθεί στο ότι η γενική μορφή της συνάρτησης είναι η **n(n-1)/2**. Άρα η **f(n)=n(n-1)/2** στη περίπτωση της φυσαλίδας. Και αν η $f(n)=n^2/2 - n/2$, σίγουρα η $g(n) = n^2$, θα είναι το πάνω ασυμπτωτικό όριο της f(n), μιας και ο όρος n^2 είναι ο ισχυρότερος.



Σχήμα 5: Υπολογισμός του αθροίσματος

Εδώ θα πρέπει να αναφέρουμε ότι η βασική πράξη που σχετίζεται με τον χρόνο εκτέλεσης του αλγορίθμου είναι η σύγκριση των κελιών.

Θα μπορούσαμε να καταλήξουμε στο ίδιο συμπέρασμα, συνθέτοντας την f(n) του αλγόριθμου και από τις εντολές και τις βασικές πράξεις που περιέχει (Σχήμα 6).

```
Αλγόριθμος Φυσαλίδα
  Δεδομένα    // Table, n //              Table = Μη ταξινομημένος πίνακας,
  Για i από 2 μέχρι N                    n     = το μέγεθος του πίνακα
        Για j από N μέχρι i με_βήμα -1
              Αν T [ j -1 ] > T [ j ] τότε
                    προσωρινή_μεταβλητή ← T [ j -1 ]
                    T [ j -1 ]  ← T [ j ]
                    T [ j ]     ←  προσωρινή_μεταβλητή
              Τέλος_αν
        Τέλος_επανάληψης
  Τέλος_επανάληψης
Τέλος Φυσαλίδα
```

Σχήμα 6: Ταξινόμηση Ευθείας Ανταλλαγής (Φυσαλίδα)

Στη ταξινόμηση η εξωτερική επανάληψη θα εκτελεστεί n-1 φορές, ενώ η εσωτερική επανάληψη θα εκτελεστεί (n-1) + (n-2) + (n-3) + ... + 3 + 2 + 1 φορές, το άθροισμα των οποίων είναι n(n-1)/2 (Σχήμα 4).

Συνεπώς αν $f(x) = n^2/2 - n/2$, τότε παραλείποντας τους σταθερούς όρους, $g(n)=n^2$ και άρα ο βαθμός πολυπλοκότητας της είναι $O(n^2)$.

ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ

Ενώ στον υπολογισμό των παραπάνω πολυπλοκοτήτων οι μαθητές θα μπορούσαν να έχουν μια δεκτικότητα, μιας και οι μαθηματικές έννοιες είναι πιο οικείες σε αυτούς, είναι λίγο δύσκολο να καταλάβουν την λογαριθμική πολυπλοκότητα.

Η δυαδική αναζήτηση ίσως είναι το ιδανικό παράδειγμα, μιας και συμπεριλαμβάνεται στην διδακτέα ύλη τους (Γενικές Οδηγίες, 2015), διδάσκεται στην αίθουσα, και ο υπολογισμός της πολυπλοκότητάς του, θα τους βοηθήσει να την κατανοήσουν καλύτερα.

Στο Σχήμα 7, βλέπουμε στην $1^η$ στήλη το μέγεθος εισόδου, που είναι ο αριθμός των κελιών ή το πλήθος των δεδομένων στα οποία κάνουμε την αναζήτηση. Στην $2^η$ στήλη είναι ο αριθμός των διαβασμάτων που κάνουμε στον πίνακα (βασική πράξη), και στη, $3^η$ στήλη είναι μια σχηματική απεικόνιση του πίνακα.

Αριθμός Κελιών	**Αριθμός Διαβασμάτων**	**Πίνακας**
1	**1**	
3	**2**	
7	**3**	
15	**4**	
31	**5**	…
$\mathbf{2^{t-1}}$	**t**	…

$n = 2^{t-1}$
$\log_2(n) = \log_2(2^{t-1})$
log(n) = t – 1 ή = t μιας και η μονάδα δεν υπολογίζεται

Σχήμα 7: Υπολογισμός Λογαριθμικού χρόνου Δυαδικής Αναζήτησης

Αν έχουμε ένα κελί, κάνουμε μια ανάγνωση. Άρα το κόστος είναι 1. Αν ο πίνακας έχει τρια κελιά τα διαβάσματα που κάνουμε είναι ένα κεντρικό και ένα είτε αριστερά είτε δεξιά. Άρα συνολικά 2 διαβάσματα. Αν έχουμε επτά κελιά, έχουμε ένα κεντρικό διάβασμα και μένουν τρία κελιά δεξιά και τρία αριστερά, όπου κάνουμε άλλα δυο διαβάσματα – όπως κάναμε στην προηγούμενη περίπτωση – σύνολο 3 διαβάσματα. Συνεχίζοντας με αυτό το μοτίβο, αν έχουμε **t** διαβάσματα τότε τα κελιά που θα έχω θα είναι $\mathbf{2^{t-1}}$. Θέλοντας να εκφράσουμε τον αριθμό των κελιών ($\mathbf{2^{t-1}}$) για **n** κελιά, όταν δηλαδή $n=\mathbf{2^{t-1}}$, και λύνοντας ως προς **t**, οδηγούμαστε στον αριθμό των διαβασμάτων που απαιτούνται (Σχήμα 7).

Άρα ο χρόνος που θα χρειαστεί, για να αναζητήσουμε ένα στοιχείο σε έναν ταξινομημένο πίνακα με δυαδικό τρόπο, για n αριθμούς, θα είναι **log(n).** Ο δε λογάριθμος έχει ως βάση το 2. Είναι δηλαδή δυαδικός λογάριθμος.

Στον παραπάνω πίνακα θα μπορούσε κάποιος να ρωτήσει γιατί χρησιμοποιούμε τον συγκεκριμένο αριθμό κελιών (1, 3, 7, 15, 31), και όχι άλλους αριθμούς. Σίγουρα οι αριθμοί των κελιών επιλέχθηκαν για να διαμορφώνουν καλύτερα τον αριθμό των διαβασμάτων, μιας και είναι περιττός αριθμός κελιών και πάντα έχω ένα "μεσαίο" κελί. Αλλά και ζυγός αριθμός κελιών να ήταν το αποτέλεσμα θα είναι το ίδιο, όπως μπορεί κανείς εύκολα να υπολογίσει.

ΣΥΜΠΕΡΑΣΜΑ

Αναμφισβήτητα, η έννοια της πολυπλοκότητας είναι για κάποιους μαθητές λίγο δύσκολη στην κατανόηση. Είναι όμως έννοια που είναι απαραίτητη και αναπόσπαστη με την συγγραφή σωστών αλγορίθμων, και δίνει στο μαθητή την αίσθηση της πρακτικής και πραγματικής υπόστασης του. Συχνά ο μαθητής γράφει έναν αλγόριθμο στο χαρτί, χωρίς να έχει την αίσθηση πως ο αλγόριθμος αυτός θα μπορούσε να χρησιμοποιηθεί σε πραγματικές συνθήκες ή αν ικανοποιεί κάποιους περιορισμούς.

Η σύλληψη της ιδέας του αλγορίθμου σίγουρα πηγάζει από μια ιδεατή βάση. Όπως η μη-ντετερμινιστική μηχανή Turing είναι ένα πολύ ισχυρό υπολογιστικό μοντέλο, αλλά είναι μη ρεαλιστικό. Έτσι και ένας αλγόριθμος όταν θα χρειαστεί να αξιοποιηθεί στη μηχανογράφηση κάποιας εταιρίας ή υπηρεσίας, η μηχανή Turing δεν θα βοηθήσει στην

πράξη, μιας και ούτε ο χρόνος ούτε οι πόροι θα είναι απεριόριστα. Ενώ η ανάλυση της υπολογιστικής πολυπλοκότητας του, θα τον οδηγήσει στην συγγραφή βέλτιστων και χρήσιμων αλγορίθμων.

ΑΝΑΦΟΡΕΣ

Βακάλη, Α., Γιαννόπουλος, Η., Ιωαννίδης, Ν., Κοίλιας, Χ., Μαλάμας, Κ., Μανωλόπουλος, Ι., Πολίτης, Π. (1999). *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον. Βιβλίο Μαθητή*. ΥΠ.Ε.Π.Θ. - Π.Ι. Αθήνα: Εκδόσεις Νέων Τεχνολογιών, σελ. 95.

Βακάλη, Α., Γιαννόπουλος, Η., Ιωαννίδης, Ν., Κοίλιας, Χ., Μαλάμας, Κ., Μανωλόπουλος, Ι., Πολίτης, Π. (1999). *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον. Τετράδιο Μαθητή*. ΥΠ.Ε.Π.Θ. - Π.Ι. Αθήνα: Εκδόσεις Νέων Τεχνολογιών, σελ 51.

Βακάλη, Α., Γιαννόπουλος, Η., Ιωαννίδης, Ν., Κοίλιας, Χ., Μαλάμας, Κ., Μανωλόπουλος, Ι., Πολίτης, Π. (1999). *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον. Βιβλίο Καθηγητή*. ΥΠ.Ε.Π.Θ. - Π.Ι. Αθήνα: Εκδόσεις Νέων Τεχνολογιών.

Βούρος Γεώργιος, (2002), *Διακριτά Μαθηματικά*, Πάτρα : Ελληνικό Ανοικτό Πανεπιστήμιο, σελ. 183.

Γενικές Οδηγίες για την Διδασκαλία του Μαθήματος "*Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον" της Γ΄τάξης Ημερήσιου Γενικού Λυκείου για το Σχολ. Έτος 2015-16, 2015*, Ινστιτούτο Εκπαιδευτικής Πολιτικής : Αθήνα

Φωτάκης Δημήτριος, Σπυράκης Παύλος, (2001), *Αλγόριθμοι και Πολυπλοκότητα*, Πάτρα : Ελληνικό Ανοικτό Πανεπιστήμιο.

Big O Notation. Ανακτήθηκε στις 17η Ιανουαρίου 2016, από τη διεύθυνση https://en.wikipedia.org/wiki/Big_O_notaion.

Complexity Analysys Part I, Ανακτήθηκε την 17η Ιανουαρίου 2016, από τη διεύθυνση http://faculty.kfupm.edu.sa/ICS/saquib/ICS202/Unit03_ComplexityAnalysis1.pdf

Cormen T, Leiserson C, Rivest R, Stein C, (2009). *Introduction to Algorithms,3rd Edition*, MIT Press.

Dasgupta Sanjoy, Papadimitriou Christos, Umesh Vazirani, (2006). *Algorithms*, Mc Grow Hill Higher Education.

Desmos Graphing Calculator, https://www.desmos.com/calculator

Herbert S. Wilf, (2002), *Algorithms and Complexity, Second Edition,* Natic Massachusetts : A. K. Peters. σελ. 23, 34.