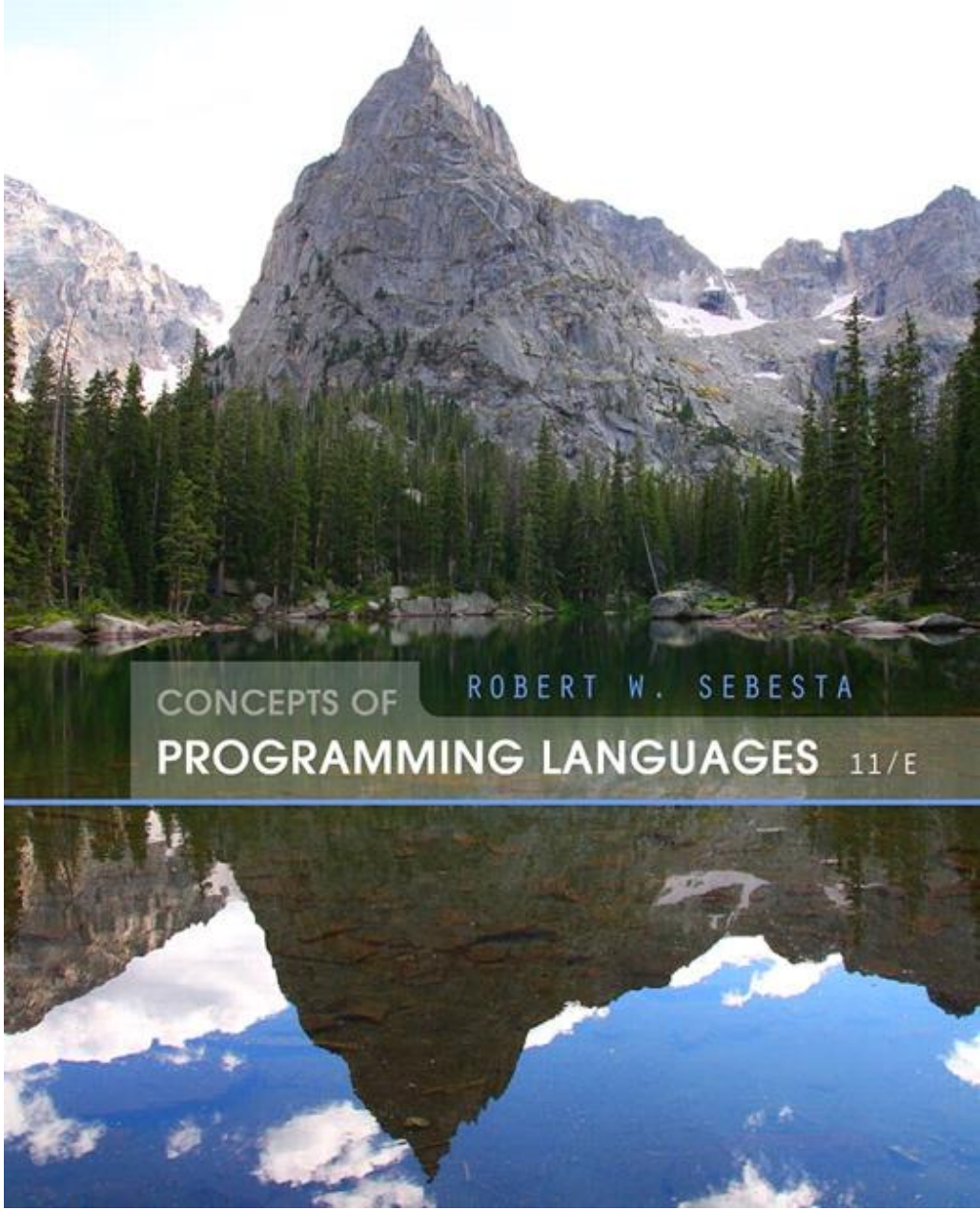
Εισαγωγή

Βασικές έννοιες

Γκόγκος Χρήστος
Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)
Πανεπιστήμιο Ιωαννίνων



ISBN 0-321-49362-1

Θεματικές Ενότητες – Κεφάλαιο 1

- Για ποιους λόγους να μελετήσει κανείς τις βασικές έννοιες γλωσσών προγραμματισμού;
- Πεδία εφαρμογής γλωσσών προγραμματισμού
- Κριτήρια αποτίμησης γλωσσών
- Παράγοντες που επηρεάζουν τη σχεδίαση των γλωσσών προγραμματισμού
- Κατηγορίες γλωσσών
- Συμβιβασμοί που γίνονται κατά τη σχεδίαση γλωσσών προγραμματισμού
- Μέθοδοι υλοποίησης γλωσσών προγραμματισμού
- Περιβάλλοντα ανάπτυξης προγραμμάτων

Για ποιους λόγους να μελετήσει κανείς τις βασικές έννοιες γλωσσών προγραμματισμού;

- Αυξημένη ικανότητα να εκφράζει ιδέες
- Βελτιωμένο υπόβαθρο για επιλογή, κατά περίπτωση, κατάλληλων γλωσσών
- Αυξημένη ικανότητα να μαθαίνει νέες γλώσσες
- Καλύτερη κατανόηση της σημασίας της υλοποίησης
- Καλύτερη χρήση γλωσσών που ήδη γνωρίζει
- Αναβάθμιση γνώσεων στην επιστήμη υπολογιστών

Πεδία προγραμματισμού

- Επιστημονικές εφαρμογές
  - Μεγάλο πλήθος υπολογισμών με χρήση αριθμών κινητής υποδιαστολής – χρήση διατάξεων (πινάκων)
  - Fortran
- Επιχειρηματικές εφαρμογές
  - Παραγωγή αναφορών, χρήση δεκαδικών αριθμών και χαρακτήρων
  - COBOL
- Τεχνητή Νοημοσύνη (A.I.)
  - Χειρισμός συμβόλων αντί για αριθμούς – χρήση συνδεδεμένων λιστών
  - LISP
- Προγραμματισμός συστημάτων
  - Απαίτηση για υψηλή απόδοση λόγω συνεχούς χρήσης
  - C
- Λογισμικό για το Web
  - Διάφορες επιλογές και συνδυασμοί γλωσσών: markup (π.χ., HTML), σεναρίων (π.χ., PHP), γενικού σκοπού (π.χ., Java)

Κριτήρια αποτίμησης γλωσσών

- **Αναγνωσιμότητα**: η ευκολία με την οποία μπορούν να αναγνωστούν τα προγράμματα και να κατανοηθούν
- **Ευκολία γραφής**: η ευκολία με την οποία η γλώσσα μπορεί να χρησιμοποιηθεί για τη δημιουργία προγραμμάτων
- **Αξιοπιστία**: συμμόρφωση με τις προδιαγραφές (δλδ, κατά πόσο η γλώσσα λειτουργεί όπως περιγράφεται στις προδιαγραφές της)
- **Κόστος**: αφορά το συνολικό κόστος που σχετίζεται με την ανάπτυξη εφαρμογών στη γλώσσα

Κτιτήρια αποτίμησης: Αναγνωσιμότητα

- Συνολική απλότητα
  - Ένα διαχειρίσιμο σύνολο χαρακτηριστικών και προγραμματιστικών κατασκευών
  - Ελάχιστη πολλαπλότητα χαρακτηριστικών
  - Ελάχιστη υπερφόρτωση τελεστών
- Ορθογωνικότητα (Orthogonality)
  - Ένα σχετικά μικρό σύνολο από βασικές προγραμματιστικές κατασκευές μπορούν να συνδυαστούν σε ένα σχετικά μικρό αριθμό τρόπων
  - Οποιοσδήποτε συνδυασμός είναι έγκυρος
- Τύποι δεδομένων
  - Κατά πόσο υπάρχουν επαρκείς προκαθορισμένοι τύποι δεδομένων
- Θέματα σχετικά με τη σύνταξη
  - Μορφές αναγνωριστικών: ευέλικτη σύνταξη
  - Ειδικές λέξεις και μέθοδοι για τη δημιουργία σύνθετων εντολών
  - Μορφή και νόημα: αυτό-περιγραφόμενες προγραμματιστικές κατασκευές, λέξεις κλειδιά με εύκολα κατανοητό νόημα

Κριτήρια αποτίμησης: Απλότητα Γραφής

- Απλότητα και ορθογωνικότητα
  - Λίγες προγραμματιστικές κατασκευές, μικρός αριθμός πρωτογενών στοιχείων, μικρό σύνολο κανόνων για το συνδυασμό τους
- Υποστήριξη αφαίρεσης
  - Η ικανότητα να ορίζονται και να χρησιμοποιούνται σύνθετες δομές ή λειτουργίες με τέτοιο τρόπο που να επιτρέπει να αγνοηθούν οι λεπτομέρειες
- Εκφραστικότητα
  - Ένα σύνολο από σχετικά εύχρηστους τρόπους ορισμού λειτουργιών
  - Ισχύς και πλήθος τελεστών και προκαθορισμένων συναρτήσεων

Κριτήρια αποτίμησης: Αξιοπιστία

- Έλεγχος τύπων (type checking)
  - Έλεγχος για σφάλματα τύπων
- Χειρισμός εξαιρέσεων
  - Παρεμπόδιση σφαλμάτων χρόνου εκτέλεσης και λήψη διορθωτικών μέτρων
- Ψευδώνυμα (aliasing)
  - Παρουσία δύο ή περισσότερων διακριτών τρόπων αναφοράς στην ίδια θέση μνήμης
- Αναγνωσιμότητα και γραφή
  - Μια γλώσσα που δεν υποστηρίζει «φυσικούς» τρόπους έκφρασης ενός αλγορίθμου θα απαιτήσει «μη φυσικές» προσεγγίσεις, και συνεπώς θα μειωθεί η αναγνωσιμότητα

Κριτήρια αποτίμησης: Κόστος

- Εκπαίδευση προγραμματιστών στη χρήση της γλώσσας
- Συγγραφή προγραμμάτων (εγγύτητα προς συγκεκριμένες εφαρμογές)
- Μεταγλώττιση προγραμμάτων
- Εκτέλεση προγραμμάτων
- Σύστημα υλοποίησης της γλώσσας: διαθεσιμότητα ελεύθερων μεταγλωττιστών
- Αξιοπιστία: η χαμηλή αξιοπιστία οδηγεί σε υψηλά κόστη
- Συντήρηση προγραμμάτων

Κριτήρια αποτίμησης: Άλλα

- Φορητότητα
  - Η ευκολία με την οποία προγράμματα μπορούν να μεταφερθούν από μια υλοποίηση σε μια άλλη
- Γενικότητα
  - Εφαρμοσιμότητα σε ευρύ φάσμα εφαρμογών
- Καλά ορισμένη
  - Η πληρότητα και η ακρίβεια του επίσημου ορισμού της γλώσσας

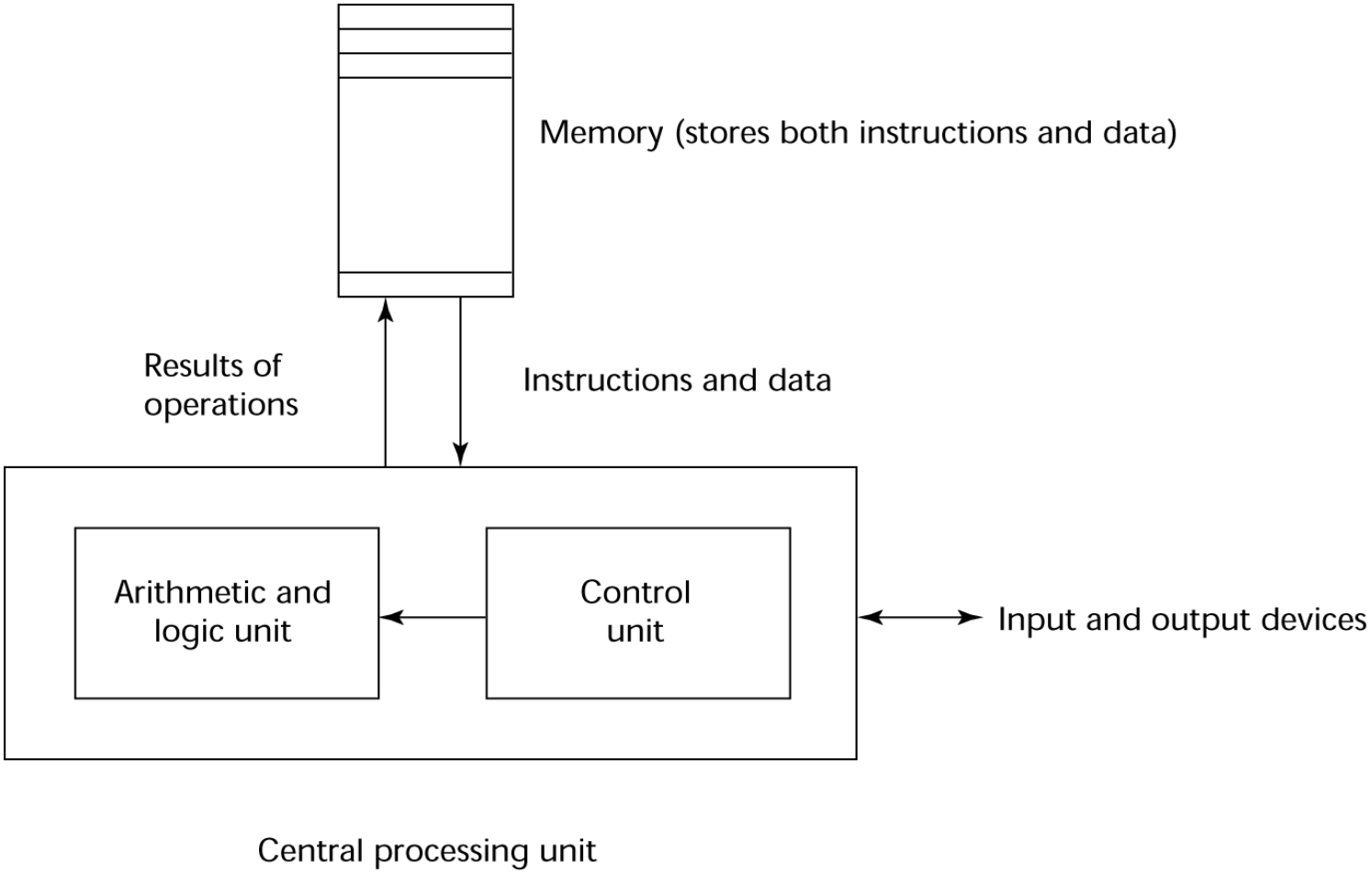
Επιρροές στη Σχεδίαση Γλωσσών

- Αρχιτεκτονική Υπολογιστών
  - Οι γλώσσες αναπτύσσονται με κέντρο την επικρατούσα αρχιτεκτονική υπολογιστών , γνωστή και αρχιτεκτονική *von Neumann.*
- Μεθοδολογίες Σχεδίασης Προγραμμάτων
  - Νέες μεθοδολογίες σχεδίασης λογισμικού (π.χ., αντικειμενοστραφής ανάπτυξη λογισμικού) έχουν οδηγήσει σε νέα προγραμματιστικά παραδείγματα και κατ' επέκταση, σε νέες γλώσσες προγραμματισμού

Επιρροή από την Αρχιτεκτονική Υπολογιστών

- Πλέον διαδεδομένη αρχιτεκτονική υπολογιστών: Von Neumann
- Οι προστακτικές γλώσσες, είναι οι πλέον επικρατέστερες λόγω των von Neumann υπολογιστών
  - Τα δεδομένα και τα προγράμματα αποθηκεύονται στη μνήμη
  - Η μνήμη βρίσκεται ξεχωριστά από τη CPU
  - Οι εντολές και τα δεδομένα διοχετεύονται από τη μνήμη στη CPU
  - Βάση προστακτικών γλωσσών
    - Οι μεταβλητές μοντελοποιούν τα κελιά μνήμης
    - Οι εντολές ανάθεσης μοντελοποιούν τη διοχέτευση
    - Οι επαναλήψεις είναι αποδοτικές

Η αρχιτεκτονική von Neumann



Η αρχιτεκτονική von Neumann

- Κύκλος Ανάκλησης-Εκτέλεσης (fetch-execute)

```
αρχικοποίησε το μετρητή προγράμματος (PC)
repeat για πάντα
  ανάκληση της εντολής στην οποία δείχνει ο PC
  μοναδιαία αύξηση του PC
  αποκωδικοποίηση της εντολής
  εκτέλεση της εντολής
end repeat
```

Επιρροή των μεθοδολογιών προγραμματισμού

- 1950s και νωρίς 1960s: Απλές εφαρμογές – κύρια ανησυχία, η αποδοτικότητα της μηχανής
- Τέλος 1960s: Η αποδοτικότητα των ανθρώπων καθίσταται σημαντική – αναζήτηση αναγνωσιμότητας και καλύτερων προγραμματιστικών κατασκευών
  - δομημένος προγραμματισμός
  - Από πάνω προς τα κάτω (top-down) σχεδίαση και βηματική εκλέπτυνση
- Τέλος 1970s: Μετάβαση από προσανατολισμό σε εργασίες σε προσανατολισμό σε δεδομένα
  - αφαίρεση δεδομένων
- Μέσα 1980s: Αντικειμενοστραφής προγραμματισμός
  - Αφαίρεση δεδομένων + κληρονομικότητα + πολυμορφισμός

Κατηγορίες γλωσσών

- Προστακτικές
  - Τα κύρια χαρακτηριστικά είναι οι μεταβλητές, οι εντολές ανάθεσης, και η επανάληψη
  - Περιέχει γλώσσες που υποστηρίζουν τον αντικειμενοστραφή προγραμματισμό
  - Περιέχει γλώσσες σεναρίων
  - Περιέχει οπτικές γλώσσες
  - Παραδείγματα: C, Java, Perl, JavaScript, Visual BASIC .NET, C++
- Συναρτησιακές
  - Ο κύριος τρόπος με τον οποίο γίνονται υπολογισμοί είναι εφαρμόζοντας συναρτήσεις σε παραμέτρους που δίνονται
  - Παραδείγματα: LISP, Scheme, ML, F#
- Λογικές
  - Βασίζονται σε κανόνες (οι κανόνες ορίζονται χωρίς καμία συγκεκριμένη σειρά)
  - Παράδειγμα: Prolog
- Υβριδικές γλώσσες markup (γλώσσες σήμανσης)/προγραμματισμού
  - Markup επεκτάσεις γλωσσών που υποστηρίζουν σε κάποιο βαθμό προγραμματισμό
  - Παραδείγματα: JSTL, XSLT

Συμβιβασμοί Σχεδίαση Γλωσσών

- Αξιοπιστία vs. Κόστος εκτέλεσης
  - Παράδειγμα: Η Java απαιτεί όλες οι αναφορές σε στοιχεία διατάξεων να ελέγχονται ότι πραγματοποιούν έγκυρη δεικτοδότηση, το οποίο οδηγεί σε αυξημένο κόστος εκτέλεσης
- Ευκολία Ανάγνωσης vs. Ευκολία Γραφής

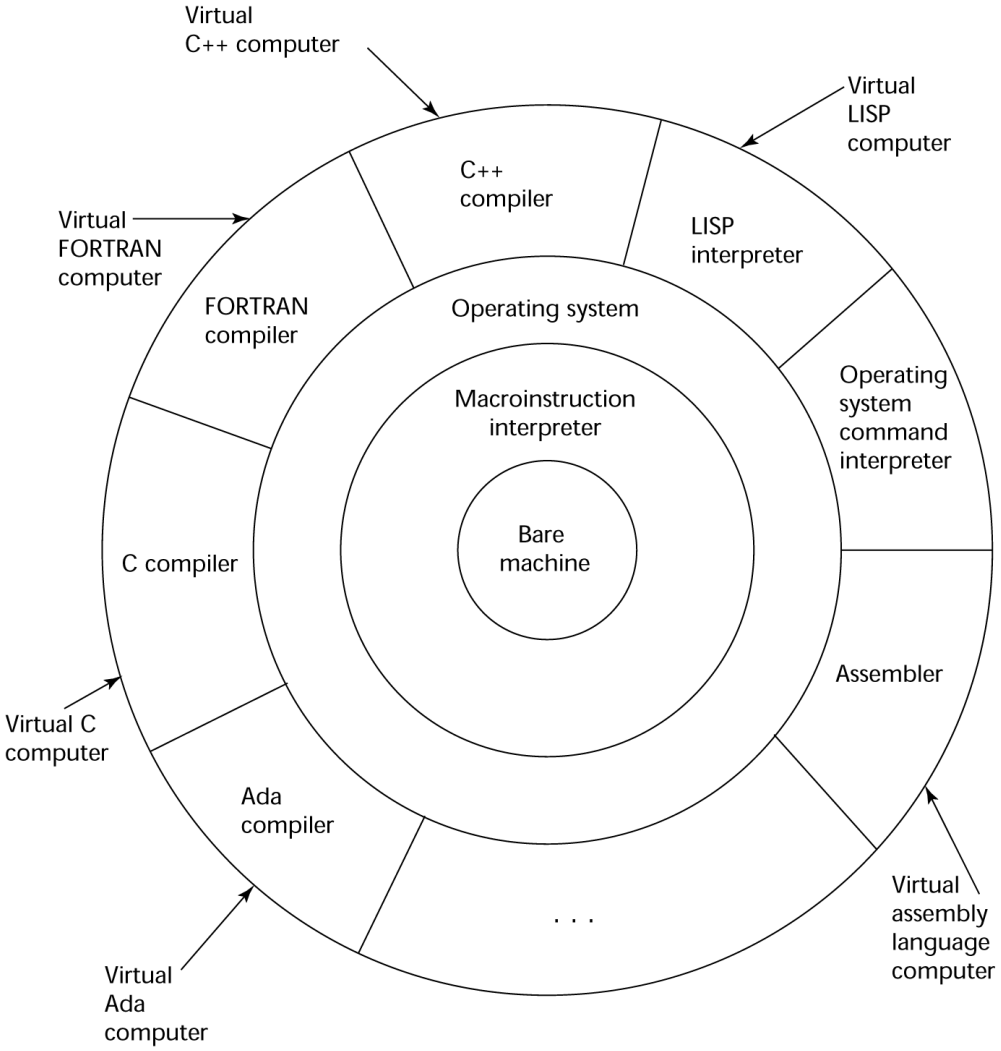
  Παράδειγμα: Η γλώσσα APL παρέχει πολλούς ισχυρούς τελεστές (και ένα μεγάλο αριθμό νέων συμβόλων), που επιτρέπουν σύνθετους υπολογισμούς να γραφούν σε συνοπτικά προγράμματα, αλλά χάνει σε ευκολία γραφής
- Γραφή (ευελιξία) vs. Αξιοπιστία
  - Παράδειγμα: Οι δείκτες C++ είναι ισχυροί και πολύ ευέλικτοι, αλλά τα προγράμματα που τους χρησιμοποιούν δεν είναι αξιόπιστα

Μέθοδοι υλοποίησης

- Μεταγλώττιση
  - Τα προγράμματα μεταφράζονται σε γλώσσα μηχανής – συμπεριλαμβάνει και τα JIT συστήματα
  - Χρήση: Μεγάλες εμπορικές εφαρμογές
- Καθαρή διερμηνεία
  - Τα προγράμματα διερμηνεύονται από ένα άλλο πρόγραμμα, τον διερμηνευτή
  - Χρήση: Μικρά προγράμματα ή όταν η αποδοτικότητα είναι δευτερεύουσας σημασίας
- Υβριδικά συστήματα υλοποίησης
  - Πρόκειται για συμβιβασμό ανάμεσα σε μεταγλωττιστές και σε καθαρούς διερμηνευτές
  - Χρήση: Μικρά και μεσαίου μεγέθους συστήματα ή όταν η αποδοτικότητα είναι δευτερεύουσας σημασίας

Οργάνωση σε Επίπεδα

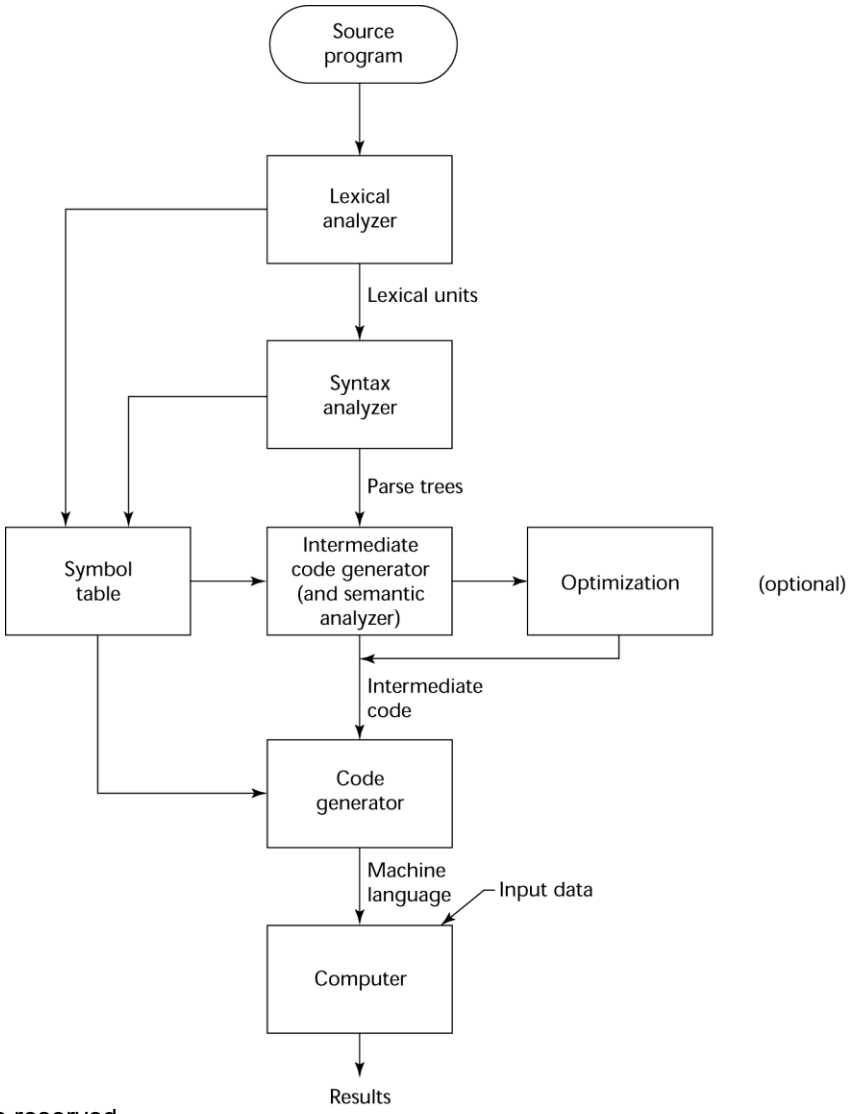
Το λειτουργικό σύστημα και η υλοποίηση της γλώσσας βρίσκονται σε επίπεδα πάνω από το υλικό του υπολογιστή



Μεταγλώττιση

- Μεταφράζει προγράμματα υψηλού-επιπέδου (πηγαία γλώσσα) σε κώδικα μηχανής (γλώσσα μηχανής)
- Αργή μετάφραση, γρήγορη εκτέλεση
- Η διαδικασία της μεταγλώττισης έχει πολλές φάσεις:
  - Λεκτική ανάλυση: μετασχηματίζει τους χαρακτήρες του πηγαίου προγράμματος σε λεκτικές μονάδες
  - Συντακτική ανάλυση: μετασχηματίζει τις λεκτικές μονάδες σε συντακτικά δένδρα (parse trees) που αναπαριστούν τη συντακτική δομή του προγράμματος
  - Σημασιολογική ανάλυση: δημιουργία ενδιάμεσου κώδικα
  - Γεννήτρια κώδικα: παράγει τον κώδικα μηχανής

Η διαδικασία μεταγλώττισης



Επιπλέον ορολογία μεταγλώττισης

- **Φόρτωση module** (executable image): ο κώδικας χρήστη και ο κώδικας συστήματος μαζί
- **Σύνδεση και φόρτωση**: η διαδικασία συλλογής προγραμματιστικών μονάδων συστήματος και η σύνδεσή τους σε ένα πρόγραμμα χρήστη

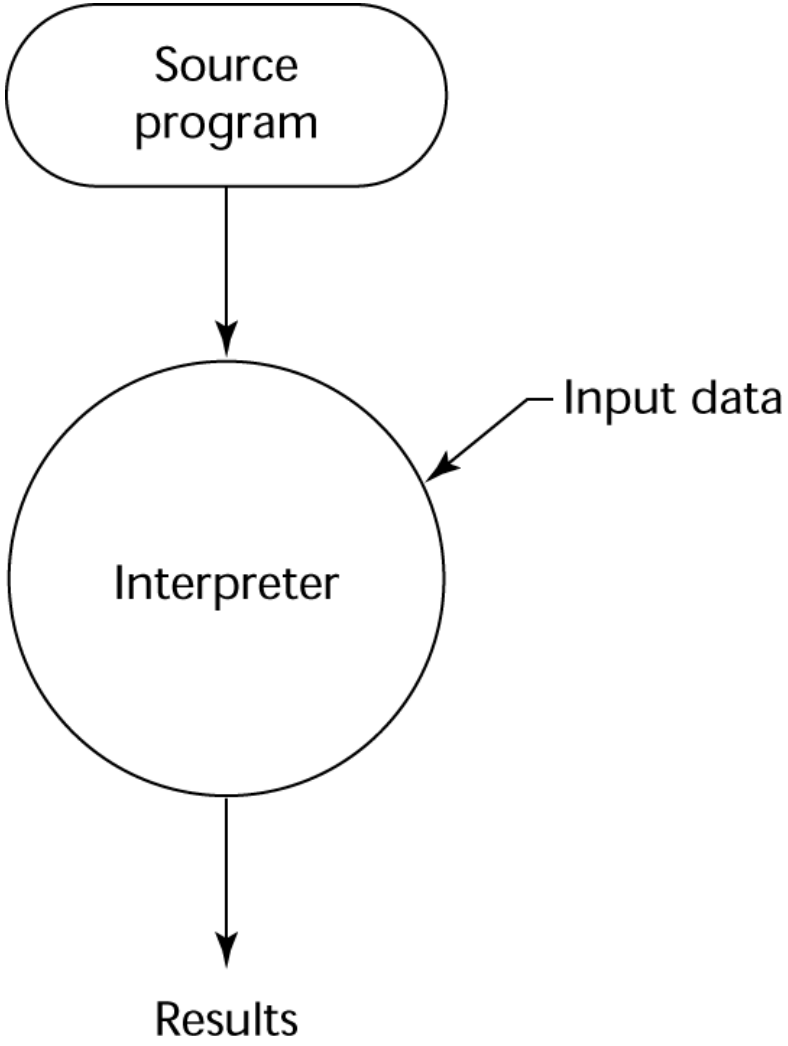
Von Neumann Bottleneck

- Η ταχύτητα σύνδεσης ανάμεσα στη μνήμη του υπολογιστή και τον επεξεργαστή του καθορίζει την ταχύτητα του υπολογιστή
- Οι εντολές του προγράμματος μπορούν συχνά να εκτελεστούν πολύ ταχύτερα από την ταχύτητα της σύνδεσης – συνεπώς η ταχύτητα της σύνδεσης αποτελεί σημείο συμφόρησης (*bottleneck)*
- Είναι γνωστό ως *von Neumann bottleneck* – αποτελεί τον κύριο περιοριστικό παράγοντα για την ταχύτητα των υπολογιστών

Καθαρή Διερμηνεία

- Δεν γίνεται μετάφραση
- Ευκολότερη υλοποίηση προγραμμάτων (τα σφάλματα χρόνου εκτέλεσης μπορούν εύκολα και άμεσα να εντοπιστούν)
- Βραδύτερη εκτέλεση (10 έως100 φορές βραδύτερα από τα μεταγλωττισμένα προγράμματα)
- Συχνά απαιτεί περισσότερο χώρο
- Πλέον συναντάται σπάνια για τις παραδοσιακές γλώσσες υψηλού επιπέδου
- Σημαντική επιστροφή της καθαρής διερμηνείας σε ορισμένες Web scripting γλώσσες (π.χ., JavaScript, PHP)

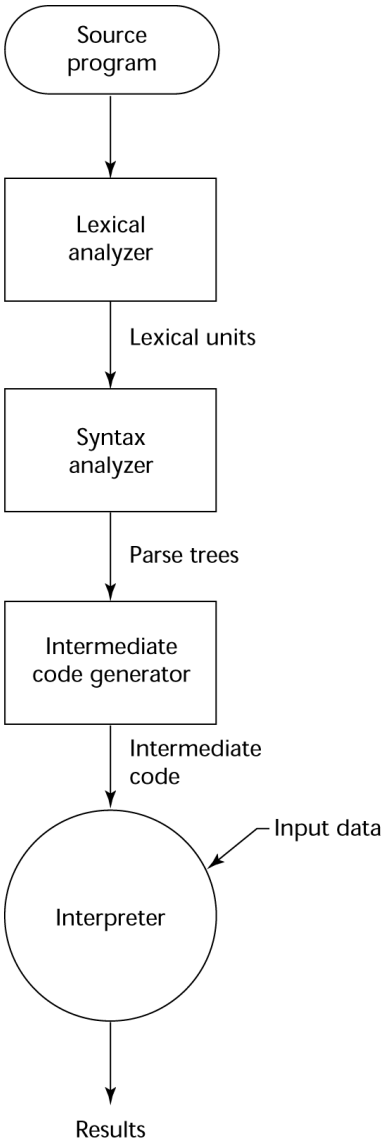
Διαδικασία καθαρής διερμηνείας



Συστήματα Υβριδικής Υλοποίησης

- Συμβιβασμός ανάμεσα σε μεταγλωττιστές και καθαρούς διερμηνευτές
- Ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου μεταφράζεται σε ενδιάμεση γλώσσα που επιτρέπει τη γρήγορη διερμηνεία
- Ταχύτερο από την καθαρή διερμηνεία
- Παραδείγματα
  - Τα προγράμματα σε Perl μεταγλωττίζονται μερικώς για τον εντοπισμό λαθών πριν τη διερμηνεία
  - Οι αρχικές υλοποιήσεις της Java ήταν υβριδικές – η ενδιάμεση μορφή (byte code) παρέχει φορητότητα σε οποιαδήποτε μηχανή διαθέτει διερμηνευτή byte code και ένα σύστημα χρόνου εκτέλεσης (και τα δύο μαζί ονομάζονται *Java Virtual Machine*)

Διαδικασία Υβριδικής Υλοποίησης



Συστήματα Υλοποίησης Just-in-Time

- Αρχικά μεταφράζουν προγράμματα σε μια ενδιάμεση γλώσσα
- Στη συνέχεια, η ενδιάμεση γλώσσα στην οποία βρίσκονται τα υποπρογράμματα μεταγλωττίζεται σε γλώσσα μηχανής όταν αυτά καλούνται
- Η έκδοση του μεταγλωττισμένου κώδικα μηχανής διατηρείται για μεταγενέστερες κλήσεις για λόγους ταχύτητας εκτέλεσης
- Τα JIT συστήματα χρησιμοποιούνται ευρέως στα προγράμματα Java
- Οι γλώσσες .NET χρησιμοποιούνται σε ένα JIT σύστημα
- Στην ουσία, τα JIT συστήματα είναι μεταγλωττιστές με καθυστέρηση

Προεπεξεργαστές

- Οι μακροεντολές προεπεξεργαστή συχνά χρησιμοποιούνται για να καθορίσουν κώδικα από άλλο αρχείο που πρόκειται να συμπεριληφθεί στον πηγαίο κώδικα
- Ένας προεπεξεργαστής επεξεργάζεται ένα πρόγραμμα αμέσως πριν μεταγλωττιστεί και επεκτείνει τις μακροεντολές προεπεξεργαστή
- Γνωστό παράδειγμα προεπεξεργαστή: Ο προεπεξεργαστής της C
  - επεκτείνει τις μακροεντολές `#include`, `#define`, και άλλες

Προγραμματιστικά περιβάλλοντα

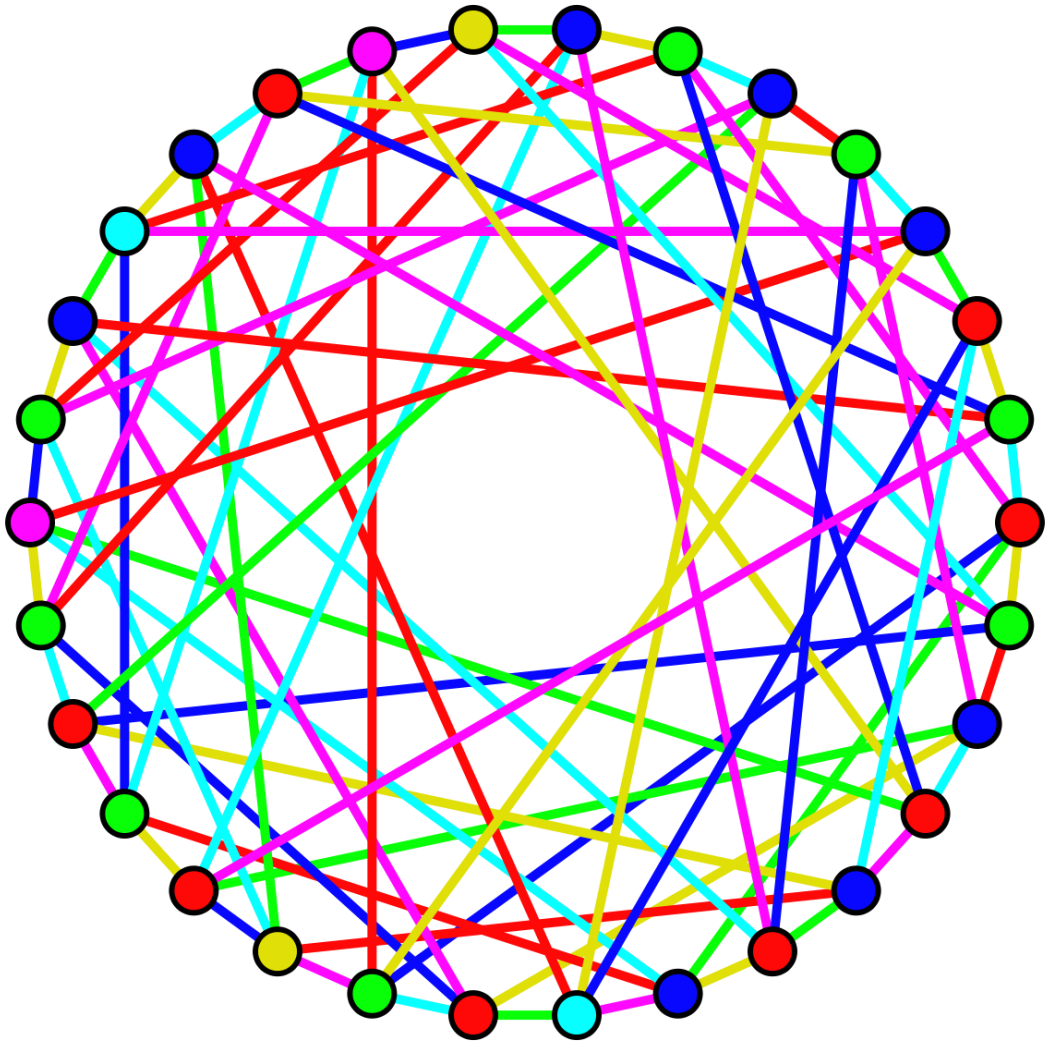
- Μια συλλογή εργαλείων που χρησιμοποιούνται για ανάπτυξη λογισμικού
- UNIX
  - Παλαιότερο λειτουργικό σύστημα και συλλογή εργαλείων
  - Σήμερα χρησιμοποιούνται συχνά μέσω γραφικών διεπαφών GUI (π.χ., CDE, KDE, or GNOME) που εκτελείται πάνω από το UNIX
- Microsoft Visual Studio.NET
  - Ένα μεγάλο, σύνθετο οπτικό περιβάλλον ανάπτυξης εφαρμογών
  - Χρησιμοποιείται για τη δημιουργία Web και non-Web εφαρμογών σε οποιαδήποτε .NET γλώσσα
- NetBeans
  - Παρόμοιο με το Visual Studio .NET, αλλά για εφαρμογές σε Java

Σύνοψη

- Η μελέτη των γλωσσών προγραμματισμού είναι σημαντική για πολλούς λόγους:
  - Ενίσχυση χρήσης διαφορετικών προγραμματιστικών κατασκευών
  - Επιτρέπει την έξυπνη επιλογή της κατάλληλης γλώσσας προγραμματισμού κατά περίπτωση
  - Καθιστά την εκμάθηση νέων γλωσσών ευκολότερη
- Τα πλέον σημαντικά κριτήρια αξιολόγησης γλωσσών προγραμματισμού είναι:
  - Αναγνωσιμότητα, ευκολία γραφής, αξιοπιστία, κόστος
- Οι κύριες επιρροές που δέχθηκε η σχεδίαση γλωσσών ήταν από την αρχιτεκτονική υπολογιστών και από τις μεθοδολογίες ανάπτυξης λογισμικού
- Οι κύριες μέθοδοι υλοποίησης γλωσσών προγραμματισμού είναι: μεταγλώττιση, καθαρή διερμηνεία, και υβριδική υλοποίηση

ΧΡΩΜΑΤΙΣΜΟΣ ΓΡΑΦΩΝ

ΤΜΗΜΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΟΙΚΙΝΩΝΙΩΝ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ



ΟΝ/ΜΟ: Νάστος Βασίλειος

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Χρήστος Γκόγκος

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ 2
1.Εισαγωγή στα Np Προβλήματα 2
1.1.P vs NP 3
1.2.NP COMPLETE Προβλήματα 3
2.Περιγραφή του προβλήματος χρωματισμού γράφων. 4
3.Προσεγγίσεις Επίλυσης 4
3.1.Δεδομένα Προβλήματος(Toronto DataSet) 4
3.2 ΠΙΝΑΚΑΣ ΣΤΑΤΙΣΤΙΚΩΝ ΣΤΟΙΧΕΙΩΝ ΠΡΟΒΛΗΜΑΤΩΝ 4
4.Αλγόριθμος First_Fit 5
5.Αλγόριθμος DSatur 6
5.1.Λειτουργία Αλγορίθμου 7
6.Αλγόριθμος RLF(Recursive Largest First) 7
7.Αλγόριθμος BACKTRACKING DSATUR 9
8.ΚΑΤΑΣΚΕΥΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΑ ΠΑΡΑΠΑΝΩ ΠΡΟΒΛΗΜΑΤΑ 10
9.ΑΝΑΦΟΡΕΣ 11

ΠΕΡΙΛΗΨΗ

Το πρόβλημα χρωματισμού γράφων τυπικά αποτελεί ομαδοποίηση δεδομένων με βάσης τον τρόπο σύνδεσης τους.Ο χρωματισμός γράφων λύνει αρκετά πρακτικά προβλήματα όπως τον χρονοπρογραμματισμό αθλητικών γεγονότων, τον προγραμματισμό εξετάσεων και την ανάθεση καταχωρητών στους μεταγλωττιστές. Ο χρωματισμός γράφων αποτελεί ένα Np-Hard πρόβλημα κάτι που σημαίνει ότι μπορεί να επιβεβαιωθεί σε πολυωνυμικό χρόνο. Στην παρούσα εργασία θα πραγματοποιηθεί μια εισαγωγή στα P και Np προβλήματα, και θα επιλυθεί ένα πρόβλημα ομαδοποίησης και χρονοπρογραμματισμού διαγωνισμάτων φοιτητών, με βάση ένα σύνολο δεδομένων. Θα εξεταστούν οι λύσεις που παράγουν 4 γνωστοί αλγόριθμοι χρωματισμού γράφων,ο greedy αλγόριθμος ,ο αλγόριθμος DSatur,ο αλγόριθμος RLF και ο αλγόριθμος DSatur με την χρήση οπισθοδρόμησης. Επίσης θα αναπτυχθεί εφαρμογή που θα χρησιμοποιείται για την αναπαράσταση των αποτελεσμάτων των τεσσάρων αλγορίθμων.

1.Εισαγωγή στα Np Προβλήματα.

Τα Np προβλήματα αφορούν προβλήματα υλοποίησης σε μη πολυωνυμικό χρόνο. Α προβλήματα Np-Complete είναι στην ουσία, τα δυσκολότερα προβλήματα της κλάσης

NP,τα οποία αφορούν προβλήματα χωρίς γνωστό αποδοτικό πολυωνυμικό αλγόριθμο. Ένα πρόβλημα Np-Complete είναι ένα πρόβλημα στο οποίο μετασχηματίζεται πολυωνυμικά κάθε άλλο πρόβλημα της κλάσης NP.Αν γνωρίζουμε τον αλγόριθμο για ένα πρόβλημα Np-Complete, μπορούμε να επιλύσουμε κάθε άλλο πρόβλημα της NP.Γνωστό NP_Complete πρόβλημα είναι το πρόβλημα ελέγχου ικανοποιησιμότητας λογικών εκφράσεων. Τα NP-Hard προβλήματα αφορούν προβλήματα που η λύση τους δεν υλοποιείται σε πολυωνυμικό χρόνο, ο βαθμός δυσκολίας είναι ίδιος με τα NP-Προβλήματα, ωστόσο τα Np-Hard προβλήματα δεν πρέπει να είναι NP.Γνωστό NP-Hard πρόβλημα είναι το πρόβλημα του πλανόδιου πωλητή. Γενικά κάθε πρόβλημα NP μπορεί να επιβεβαιωθεί σε πολυωνυμικό χρόνο από έναν αλγόριθμο που ενδιάμεσα από τις διαθέσιμες επιλογές θα επιλέγει πάντα την σωστή(«Lucky algorithm»).

1.1.P vs NP

Το πρόβλημα στην απλή του διατύπωση θέτει το ερώτημα αν μπορεί ένα πρόβλημα να επιλυθεί τόσο γρήγορα από τον υπολογιστή όσο γρήγορα μπορεί να επιβεβαιωθεί η ύπαρξη της λύσης του. Το πρόβλημα P vs NP είναι ένα ανοικτό πρόβλημα στην επιστήμη των υπολογιστών.P είναι η γενική κλάση των ερωτημάτων των προβλημάτων που υπάρχει αλγόριθμος ο οποίος επιλύει το πρόβλημα σε πολυωνυμικό χρόνο. Αντίστοιχα με τον όρο NP αναφερόμαστε στην κλάση των προβλημάτων για τα οποία η λύση δεν μπορεί να βρεθεί σε πολυωνυμικό χρόνο,ωστόσο το πρόβλημα μπορεί να επιβεβαιωθεί σε πολυωνυμικό χρόνο, δηλαδή η απάντηση μπορεί να επιβεβαιωθεί γρήγορα. (tutorialspoint, 2020)

1.2.NP COMPLETE Προβλήματα

Ας υποθέσουμε ότι έχουμε 2 προβλήματα απόφασης L1 και L2,και έναν αλγόριθμο A2 που λύνει το πρόβλημα L1.Για να μπορέσουμε να επιτύχουμε μείωση(Reduction) πρέπει να μετασχηματίσουμε τα προβλήματα L1 και L2 ώστε ο αλγόριθμος A2 να αποτελεί μέρος του αλγόριθμου A1 που θα χρησιμοποιείται για την επίλυση του προβλήματος L1.Για να μπορέσω να αναφερθώ σε ένα πρόβλημα(L) ως NP-COMPLETE πρέπει :

- Το πρόβλημα να ανήκει στην κλάση Np.
- Κάθε πρόβλημα Np πρέπει να μπορεί να αναλώνεται στο πρόβλημα L(Reduction).

Η προϋπόθεση για να θεωρηθεί ένα πρόβλημα(L) σαν NP-Complete είναι, ότι κάθε πρόβλημα NP να μπορεί να χρησιμοποιήσει το για την επιβεβαίωση του σε πολυωνιμικό χρόνο.Το πρώτο NP-COMPLETE πρόβλημα που διατυπώθηκε ήταν το SAT .Η γνώση γύρω από τα Np προβλήματα και συγκεκριμένα τα Np-Complete είναι σημαντική γιατί για ένα πρόβλημα μπορούμε να αποδείξουμε ότι η λύση σε πολυωνυμικό χρόνο η λύση δεν μπορεί να είναι εφικτή.[1]

[1] https://www.geeksforgeeks.org/np-completeness-set-1/

2.Περιγραφή του προβλήματος χρωματισμού γράφων.

Το πρόβλημα χρωματισμού γραφήματος τυπικά ορίζεται ως εξής. Δεδομένου ενός μη κατευθυνόμενου απλού γραφήματος G = (V, E) με ένα σύνολο κορυφών V και ένα σύνολο ακμών E, ζητείται η ανάθεση σε κάθε κορυφή v ∈ V ενός ακεραίου c(v) ∈ {1, 2, ..., k} έτσι ώστε το k να ελαχιστοποιείται και να ισχύει ότι c(v) ≠c(u) ∀{v, u} ∈ E. Το πρόβλημα συναντάται σε μεγάλο αριθμό πρακτικών εφαρμογών όπως ο χρονοπρογραμματισμός εκπαιδευτικών ιδρυμάτων (educational timetabling), ο χρονοπρογραμματισμός αθλητικών γεγονότων (sports scheduling), η ανάθεση συχνοτήτων (frequency assignment), η ανάθεση καταχωρητών στους μεταγλωττιστές (compiler register allocation) και άλλα. Πολλοί αλγόριθμοι χρωματισμού γραφημάτων έχουν προταθεί τα τελευταία 50 έτη. Στην παρούσα εργασία θα εξεταστούν τέσσερις αλγόριθμοι που ανήκουν στις λεγόμενες κατασκευαστικές τεχνικές (constructive techniques). Οι κατασκευαστικές τεχνικές δημιουργούν λύσεις βήμα προς βήμα, αναθέτοντας στη σειρά, σε κάθε κορυφή, ένα χρώμα, πιθανά εφαρμόζοντας οπισθοχώρηση κατά τη διαδικασία. Οι αλγόριθμοι που θα εξεταστούν είναι ο αλγόριθμος first fit, ο αλγόριθμος DSATUR, ο αλγόριθμος Recursive Largest First και ο αλγόριθμος backtracking DSATUR. Τα δεδομένα θα χωριστούν σε χρωματικές τάξεις. Μια χρωματική τάξη αποτελείται από κορυφές που έχουν χρωματιστεί με το ίδιο χρώμα. Οι αλγόριθμοι χρωματισμού προσπαθούν να επιτύχουν χρωματισμό γράφων με όσο το δυνατόν λιγότερο αριθμό χρωμάτων.

3.Προσεγγίσεις Επίλυσης

3.1.Δεδομένα Προβλήματος(Toronto DataSet)

Τα δεδομένα του προβλήματος βρίσκονται στον σύνδεσμο: https://github.com/chgogos/datasets/blob/main/UETT/toronto.zip .Το αρχείο περιέχει 13 αρχεία δεδομένων, όπου σε κάθε γραμμή αναπαρίσταται και ένας φοιτητής ενώ σε κάθε στήλη διαχωρισμένοι με το κενό βρίσκονται οι κωδικοί εξέτασης που έχει εγγραφεί ο κάθε φοιτητής. Κάθε αρχείο παρέχει και ένα μοναδικό αριθμό απο κορυφές, δηλαδή το σύνολο των μοναδικών διανυσμάτων που συμμετέχουν οι εγγεγραμμένοι φοιτητές.Σε κάθε αρχείο υπάρχει και ένας μοναδικός αριθμός διαγωνισμάτων. Σκοπός είναι ο χρωματισμός των κορυφών(δηλαδή η ομαδοποίηση των διαγωνισμάτων),με όσο το δυνατόν λιγότερα χρώματα.

3.2 ΠΙΝΑΚΑΣ ΣΤΑΤΙΣΤΙΚΩΝ ΣΤΟΙΧΕΙΩΝ ΠΡΟΒΛΗΜΑΤΩΝ

Στην παρακάτω εικόνα εμφανίζονται τα στατιστικά στοιχεία που προκύπτουν σε κάθε πρόβλημα. Τα ζητούμενα χαρακτηριστικά είναι ο αριθμός των κορυφών(V),η πυκνότητα των συγκρούσεων(Density),η διάμεσος κορυφή(Med),ο μικρότερος βαθμός (Min),ο μεγαλύτερο βαθμός(Max),ο μέσος βαθμός μεταξύ των κορυφών(Mean),και ο συντελεστής διακύμανσης(Cv).Τα παρακάτω δεδομένα συγκεντρώθηκαν και από τα 13 αρχεία του προβλήματος και παράχθηκαν οι ακόλουθες τιμές.

	File	Vertices	Density	Min	Median	Max	Mean	Cv
1	kfu-...	461	0.055	0.000	18	247	25.566	120.117
2	hec-...	81	0.415	9.000	32	62	33.654	36.553
3	uta-...	622	0.125	1.000	65	303	77.971	73.730
4	sta-f-83.stu	139	0.143	7.000	16	61	19.871	67.608
5	ute-...	184	0.084	2.000	13	58	15.543	69.324
6	tre-s-92.stu	261	0.180	0.000	45	145	46.981	59.733
7	ear-f-83.stu	190	0.266	4.000	45	134	50.453	56.261
8	lse-f-91.stu	381	0.062	0.000	16	134	23.785	93.278
9	rye-...	486	0.075	0.000	24	274	36.510	111.876
10	yor-...	181	0.287	7.000	51	117	52.000	35.325
11	car-f-92.stu	543	0.138	0.000	63	381	74.788	75.415
12	car-...	682	0.128	0.000	77	472	87.431	70.962
13	pur-...	2419	0.029	0.000	47	857	71.320	129.506

Εικόνα 1.Στατιστικά δεδομένα αρχείων

4.Αλγόριθμος First_Fit

Ο αλγόριθμος first -fit είναι ένας greedy(άπληστος) αλγόριθμος χρωματισμού κορυφών. Για κάθε κορυφή του γραφήματος μας ο αλγόριθμος ακολουθεί τα εξής βήματα:

- Καταγραφή χρωμάτων γειτονικών κορυφών, εφόσον έχουν περάσει την διαδικασία χρωματισμού, σαν μη διαθέσιμα.
- Εύρεση του πρώτου διαθέσιμου χρώματος.
- Χρωματισμός κορυφής με το διαθέσιμο χρώμα.
- Επαναφορά διαθεσιμότητας χρωμάτων σε κατάσταση διαθέσιμα, για να χρησιμοποίηση τους στις επόμενες κορυφές.

Ο αλγόριθμος first-fit έχει σαν κύριο χαρακτηριστικό ότι χρωματίζει τις κορυφές με την σειρά ξεκινώντας από την πρώτη και χωρίς να ενσωματώνει σε κάποιο στάδιο του χαρακτηριστικά των κορυφών όπως ο βαθμός μίας κορυφής ή το βάρος της κορυφής. Το χαρακτηριστικό αυτό των καθιστά σαν έναν άπληστο αλγόριθμο. Ο αλγόριθμος απλώς θα αναζητήσει το πρώτο διαθέσιμο χρώμα με το οποία θα χρωματίσει μία κορυφή. Το αποτέλεσμα που θα παράξει θα ομαδοποιήσει τα δεδομένα του προβλήματος μας παράγοντας ωστόσο έναν μεγάλο αριθμό ομάδων(διαθέσιμων χρωμάτων). Στην παρακάτω εικόνα φαίνονται τα αποτελέσματα που θα παραχθούν με βάση τα δεδομένα του προβλήματος μας από το αρχείο δεδομένων Toronto αν για κάθε γράφημα που παράγουν τα αρχεία εκτελεστεί και παράξει αποτελέσματα ο αλγόριθμος first-fit.Ο αλγόριθμος first fit έχει πολυπλοκότητα O(V+E).Αυτό σημαίνει ότι ο χρόνος εκτέλεσης του εξαρτάται από το μέγεθος του γραφήματος μας και Από τον αριθμό διασυνδέσεων που προκύπτουν μεταξύ των κόμβων.

	FILE	VERTICES	COLORS	ALGORITHM
1	car-f-92.stu	543	40	FIRST FIT
2	car-s-91.stu	682	43	FIRST FIT
3	ear-f-83.stu	190	28	FIRST FIT
4	hec-s-92.stu	81	22	FIRST FIT
5	kfu-s-93.stu	461	24	FIRST FIT
6	lse-f-91.stu	381	21	FIRST FIT
7	pur-s-93.stu	2419	47	FIRST FIT
8	rye-s-93.stu	486	30	FIRST FIT
9	sta-f-83.stu	139	13	FIRST FIT
10	tre-s-92.stu	261	28	FIRST FIT
11	uta-s-92.stu	622	42	FIRST FIT
12	ute-s-92.stu	184	12	FIRST FIT
13	yor-f-83.stu	181	26	FIRST FIT

ΕΙΚΟΝΑ 2.ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΛΓΟΡΙΘΜΟΥ FIRST_FIT([2])

5.Αλγόριθμος DSatur

Ο αλγόριθμος DSATUR,είναι ένας ευρετικός αλγόριθμος χρωματισμού γράφων. Ο αλγόριθμος επιλέγει την κορυφή με τον μεγαλύτερο βαθμό και την χρωματίζει με το πρώτο χρώμα. Στην συνέχεια επιλέγεται η κορυφή με τον μεγαλύτερο βαθμό κορεσμού. Αν δύο κορυφές έχουν τον ίδιο βαθμό κορεσμού επιλέγεται η κορυφή με το μεγαλύτερο βαθμό. Παραπάνω ισότητες διαχωρίζονται με τυχαίο τρόπο. Έπειτα με χρήση επανάληψης ελέγχουμε από τις γειτονικές κορυφές της επιλεγμένης κορυφής ,το χρώμα που έχουν χρωματιστεί και βρίσκουμε το κατάλληλο χρώμα για την κορυφή. Αν όλες οι κορυφές έχουν χρωματιστεί ο αλγόριθμος τερματίζει αλλιώς βρίσκει την επόμενη κορυφή με τον μεγαλύτερο βαθμό κορεσμού και εκτελεί τα επόμενα βήματα ,ως ότου ολοκληρωθεί ο χρωματισμός όλων των κορυφών του γραφήματος. Η πολυπλοκότητα του αλγορίθμου DSatur στην χειρότερη περίπτωση είναι O(n^2).Παρατηρώντας το αποτελέσματα που παράγει ο DSatur συγκριτικά με τον First Fit,προκύπτει το συμπέρασμα ότι ο αλγόριθμος παράγει καλύτερα αποτελέσματα χρωματισμού από ότι ο first fit και γενικά οι greedy coloring αλγόριθμοι, κάτι που είναι λογικό από την στιγμή που αλγόριθμος αναζητά το λιγότερο χρησιμοποιημένο χρώμα και όχι το πρώτο διαθέσιμο όπως ο αλγόριθμος first fit.Πηγή (Hemert, 2006)

[2]https://github.com/vasnastos/Algorithms_and_complexity/blob/main/Alco_report_images/FIRST_FIT.png

Pseudocode [edit]

Define the degree of saturation of a vertex as the number of different colours in its neighbourhood. Given a simple, undirected graph G compromising vertex set V and edge set E:[4]

1. Generate a degree ordering of V.
2. Select a vertex of maximal degree and colour it with the first colour.
3. Consider a vertex with the highest degree of saturation. Break ties by considering that vertex with the highest degree. Further ties are broken randomly.
4. Loop through the colour classes created so far, and colour the selected vertex with the first suitable colour.
5. Unless V is all coloured, return to step 3

Εικόνα 3.Ψευδοκώδικας για υλοποίηση DSATUR

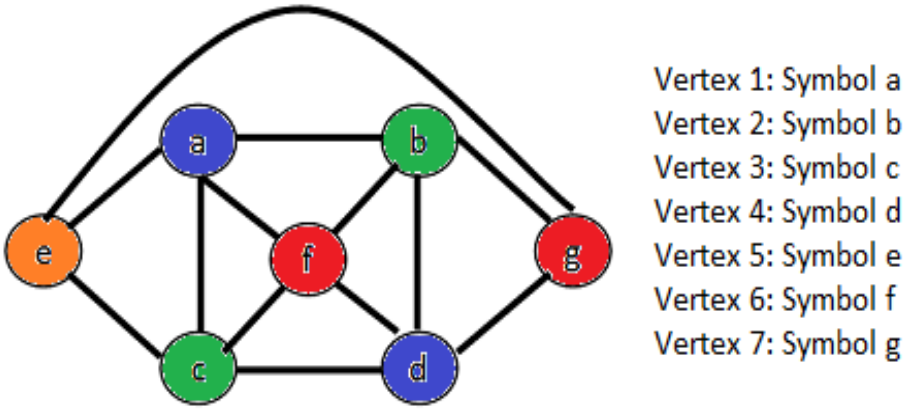
5.1.Λειτουργία Αλγορίθμου

Ο αλγόριθμος υπολογίζει τον βαθμό όλων των διαθέσιμων κορυφών. Στην συνέχεια επιλέγει την κορυφή με τον μεγαλύτερο βαθμό και την χρωματίζει με το πρώτο χρώμα. Έπειτα υπολογίζει τον βαθμό κορεσμού των κορυφών και επιλέγει την κορυφή με τον υψηλότερο βαθμό κορεσμού(Ο βαθμός κορεσμού προκύπτει από το πλήθος των χρωματισμένων κορυφών των γειτονικών κορυφών, της κορυφής που θέλω να χρωματίσω). Στην συνέχεια ελέγχονται οι χρωματικές κλάσεις που έχουν δημιουργηθεί με σκοπό να βρεθεί αυτή που είναι κατάλληλη για να χρωματίσει την κορυφή. Η παραπάνω διαδικασία θε πραγματοποιηθεί έως ότου χρωματιστούν όλες οι κορυφές του γραφήματός μας.

	FILE	VERTICES	COLORS	ALGORITHM
1	car-f-92.stu	543	34	DSATUR
2	car-s-91.stu	682	33	DSATUR
3	ear-f-83.stu	190	25	DSATUR
4	hec-s-92.stu	81	19	DSATUR
5	kfu-s-93.stu	461	18	DSATUR
6	lse-f-91.stu	381	16	DSATUR
7	pur-s-93.stu	2419	35	DSATUR
8	rye-s-93.stu	486	21	DSATUR
9	sta-f-83.stu	139	14	DSATUR
10	tre-s-92.stu	261	23	DSATUR
11	uta-s-92.stu	622	31	DSATUR
12	ute-s-92.stu	184	10	DSATUR
13	yor-f-83.stu	181	25	DSATUR

ΕΙΚΟΝΑ 4.ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΛΓΟΡΙΘΜΟΥ DSATUR([i])

6.Αλγόριθμος RLF(Recursive Largest First)



Εικόνα 5.Παράδειγμα εκτέλεσης Αλγορίθμου RLF (rlf icon image, n.d.)

Ο αλγόριθμος RLF, σε κάθε βήμα της διαδικασίας επιλέγει ένα κόμβο για χρωματισμό που, χρωματίζοντας όλες τις κορυφές με τα λιγότερα δυνατά χρώματα. Ο αλγόριθμος RLF υλοποιείται ως εξής: Δοθέντος ενός γράφου G με κορυφές V και ακμές E,ο αλγόριθμος αναθέτει το χρώμα 1(η 0),στην κορυφή με τον μέγιστο βαθμό, υποθετικά η u1.Το χρώμα που χρωματίζει αρχικά ο αλγόριθμος είναι i.Όταν i κόμβοι χρωματιστούν με το χρώμα ο αλγόριθμος τοποθετεί σε ένα σύνολο U1 τους κόμβους που είναι γείτονες με έναν τουλάχιστον χρωματισμένο κόμβο και δεν έχουν χρωματιστεί, και σε ένα σύνολο U2 τους κόμβους που δεν είναι γείτονες με κανέναν από τους χρωματισμένους κόμβους και επιλέγεται η κορυφή με τον ελάχιστο βαθμό από το σύνολο U1,αν δεν υπάρχει. Αν δεν υπάρχει κάποια διαθέσιμη επιλογή(σ.σ το σύνολο είναι άδειο),ο αλγόριθμος εκτελείται αναδρομικά για το υπόλοιπο τμήμα του γράφου, συμπεριλαμβανομένων και των μη χρωματισμένων κορυφών στις οποίες δεν υπήρχε γειτονικότατα με χρωματισμένες κορυφές, χρησιμοποιώντας το επόμενο διαθέσιμο χρώμα. (Leighton, 1979)

Στην παρακάτω εικόνα ακολουθεί ένα παράδειγμα εκτέλεσης του αλγορίθμου Rlf,για το αρχείο δεδομένων sta-f-83.stu.Ο αλγόριθμος επιτυγχάνει να χρωματίσει 139 κορυφές χρησιμοποιώντας δεκατρία χρώματα. Στην παρακάτω εικόνα φαίνονται τα αποτελέσματα εκτέλεσης του αλγορίθμου RLF,ο οποίος επιτυγχάνει στο μεγαλύτερο ποσοστό των αρχείων το μικρότερο αριθμό χρωμάτων σε σχέση με τους αλγόριθμους first fit και DSatur.

	FILE	VERTICES	COLORS	ALGORITHM
1	car-f-92.stu	543	31	RLF
2	car-s-91.stu	682	32	RLF
3	ear-f-83.stu	190	23	RLF
4	hec-s-92.stu	81	20	RLF
5	kfu-s-93.stu	461	20	RLF
6	lse-f-91.stu	381	18	RLF
7	pur-s-93.stu	2419	32	RLF
8	rye-s-93.stu	486	24	RLF
9	sta-f-83.stu	139	13	RLF
10	tre-s-92.stu	261	23	RLF
11	uta-s-92.stu	622	33	RLF
12	ute-s-92.stu	184	10	RLF
13	yor-f-83.stu	181	23	RLF

Εικόνα 6.Αποτελέσματα αλγορίθμου RLF

7.Αλγόριθμος BACKTRACKING DSATUR

→ *Ψευδοκώδικας για backtracking.*

```
procedure bt(c) is
    if reject(P, c) then return
    if accept(P, c) then output(P, c)
    s ← first(P, c)
    while s ≠ NULL do
        bt(s)
        s ← next(P, s)
```

Ο αλγόριθμός backtracking DSatur με την χρήση οπισθοδρόμησης επιτυγχάνει αποτελέσματα, παρόμοια με του αλγορίθμου DSatur,παρέχοντας ωστόσο την δυνατότητα εύρεσης της βέλτιστης λύσης η οποία ακολουθεί τα πρότυπα τα οποία έχει ορίσει ο χρήστης. Τέτοια πρότυπα αποτελούν ο χρόνος εκτέλεσης του αλγορίθμου(μπορεί να παραχθεί συνθήκη τερματισμού ανάλογα με το πόσα δευτερόλεπτα χρειάστηκε ο αλγόριθμος για να εκτελεστεί, και ο μέγιστος αριθμός χρωμάτων που επιθυμεί να χρησιμοποιήσει ο χρήστης, υιοθετώντας ωστόσο και την επιλογή του αδιεξόδου σε περίπτωση που δεν μπορεί να παραχθεί κάποια επιθυμητή λύση, βάση του αριθμού χρωμάτων που παρείχε ο χρήστης).Η οπισθοδρόμησή μας δίνει την δυνατότητα, αναίρεση βημάτων που έχουμε πραγματοποιήσει, με σκοπό την χρησιμοποίηση όλων των συνδυασμών, ώστε να παραχθεί λύση με τον διαθέσιμο αριθμό χρωμάτων που μας παρέχεται η να ενημερωθεί ο χρήστης για την αδυναμία χρησιμοποίησης του αριθμού διαθέσιμων χρωμάτων, ώστε να δοκιμάσει μια διαφορετική προσέγγιση ομαδοποίησης των δεδομένων.Η οπισθοδρόμηση μας δίνει την δυνατότητα να επιλύσουμε τα προβλήματα αναδρομικά αφαιρώντας εκείνες τις λύσεις οι οποίες δεν ικανοποιούν τις προϋποθέσεις.

Γνωστό πρόβλημα που επιλύεται με την χρήση οπισθοδρόμησης είναι το πρόβλημα των 8 βασιλισσών. (Geeks, 2014)(Rhyd Lewis, n.d.).

Στην παρακάτω εικόνα παρουσιάζεται το αποτέλεσμα εκτέλεσης του αλγορίθμου DSatur για το αρχείο sta-f-83.stu.Τα αποτελέσματα που παράγει ο αλγόριθμος DSatur είναι παρόμοια με του ευρετικού αλγορίθμου DSatur,ωστόσο ο αλγόριθμος DSatur με την χρήση οπισθοδρόμησης μας μας βοηθάει να καταλήξουμε στο βέλτιστο αποτέλεσμα.

FileName:sta-f-83.stu
File Created at:15:24:30
Backtracking DSatur
Colors used:13

VERTEX	COLOR	VERTEX	COLOR	VERTEX	COLOR	VERTEX	COLOR	VERTEX	COLOR	VERTEX	COLOR	VERTEX	COLOR
VERTEX_1	1	VERTEX_22	3	VERTEX_43	5	VERTEX_64	6	VERTEX_85	5	VERTEX_106	8	VERTEX_127	10
VERTEX_2	1	VERTEX_23	3	VERTEX_44	5	VERTEX_65	6	VERTEX_86	7	VERTEX_107	9	VERTEX_128	10
VERTEX_3	1	VERTEX_24	3	VERTEX_45	5	VERTEX_66	6	VERTEX_87	7	VERTEX_108	9	VERTEX_129	10
VERTEX_4	2	VERTEX_25	3	VERTEX_46	5	VERTEX_67	6	VERTEX_88	7	VERTEX_109	7	VERTEX_130	12
VERTEX_5	2	VERTEX_26	3	VERTEX_47	5	VERTEX_68	11	VERTEX_89	6	VERTEX_110	7	VERTEX_131	1
VERTEX_6	2	VERTEX_27	4	VERTEX_48	3	VERTEX_69	11	VERTEX_90	7	VERTEX_111	7	VERTEX_132	3
VERTEX_7	2	VERTEX_28	1	VERTEX_49	10	VERTEX_70	11	VERTEX_91	7	VERTEX_112	9	VERTEX_133	11
VERTEX_8	2	VERTEX_29	10	VERTEX_50	4	VERTEX_71	5	VERTEX_92	7	VERTEX_113	9	VERTEX_134	8
VERTEX_9	2	VERTEX_30	3	VERTEX_51	4	VERTEX_72	1	VERTEX_93	7	VERTEX_114	9	VERTEX_135	10
VERTEX_10	2	VERTEX_31	3	VERTEX_52	4	VERTEX_73	4	VERTEX_94	7	VERTEX_115	9	VERTEX_136	12
VERTEX_11	2	VERTEX_32	3	VERTEX_53	4	VERTEX_74	4	VERTEX_95	5	VERTEX_116	9	VERTEX_137	9
VERTEX_12	2	VERTEX_33	3	VERTEX_54	4	VERTEX_75	4	VERTEX_96	5	VERTEX_117	9	VERTEX_138	11
VERTEX_13	2	VERTEX_34	3	VERTEX_55	4	VERTEX_76	4	VERTEX_97	7	VERTEX_118	9	VERTEX_139	13
VERTEX_14	2	VERTEX_35	3	VERTEX_56	4	VERTEX_77	6	VERTEX_98	8	VERTEX_119	9		
VERTEX_15	2	VERTEX_36	3	VERTEX_57	4	VERTEX_78	6	VERTEX_99	8	VERTEX_120	6		
VERTEX_16	2	VERTEX_37	3	VERTEX_58	4	VERTEX_79	6	VERTEX_100	5	VERTEX_121	10		
VERTEX_17	2	VERTEX_38	3	VERTEX_59	6	VERTEX_80	6	VERTEX_101	8	VERTEX_122	10		
VERTEX_18	3	VERTEX_39	5	VERTEX_60	6	VERTEX_81	6	VERTEX_102	8	VERTEX_123	10		
VERTEX_19	3	VERTEX_40	5	VERTEX_61	6	VERTEX_82	6	VERTEX_103	6	VERTEX_124	7		
VERTEX_20	3	VERTEX_41	5	VERTEX_62	5	VERTEX_83	6	VERTEX_104	6	VERTEX_125	10		
VERTEX_21	3	VERTEX_42	4	VERTEX_63	6	VERTEX_84	6	VERTEX_105	8	VERTEX_126	10		

Εικόνα 7.Παράδειγμα εκτέλεσης αλγορίθμου BDSatur

Συνοψίζοντας ο αλγόριθμος DSatur με χρήση οπισθοδρόμησης, συγκριτικά με τους υπόλοιπους τρεις αλγορίθμους, παράγει το βέλτιστο αποτέλεσμα, ομαδοποιώντας τα δεδομένα με τον μικρότερο αριθμό χρωμάτων. Με την χρήση της οπισθοδρόμησης παρέχετε η δυνατότητα να δοκιμαστούν όλοι οι διαθέσιμοι συνδυασμοί με σκοπό την εύρεση χρωματισμού με έναν συγκεκριμένο αριθμό χρωμάτων.

8.ΚΑΤΑΣΚΕΥΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΑ ΠΑΡΑΠΑΝΩ ΠΡΟΒΛΗΜΑΤΑ

Η εφαρμογή που κατασκευάστηκε στα πλαίσια του προβλήματος του χρονοπρογραμματισμού και του np-hard προβλήματος χρωματισμού γράφων έχει υλοποιηθεί σε γλώσσα C++ με την χρήση του framework Qt Creator (QtCompany, 2018).Περισσότερες πληροφορίες θα βρείτε στους ακόλουθους συνδέσμους. Για την υλοποίηση των αλγορίθμων χρησιμοποιήθηκαν έτοιμες δομές δεδομένων, όπως λίστες, διανύσματα,maps,ενώ χρησιμοποιήθηκαν και αρχές αντικειμενοστραφούς προγραμματισμού.

Οδηγίες εγκατάστασης της εφαρμογής μπορείτε να βρείτε στον ακόλουθο σύνδεσμο:

https://vasnastos.github.io/Algorithms_and_complexity/installation.html

Περισσότερες πληροφορίες θα βρείτε στη ακόλουθη ιστοσελίδα:

https://vasnastos.github.io/Algorithms_and_complexity/

9.ΑΝΑΦΟΡΕΣ

Company, Q., 2018. *Qt Documentition.* [Ηλεκτρονικό]
Available at: https://doc.qt.io/
[Πρόσβαση 16 1 2021].

Geeks, G. f., 2014. *Geeks for Geeks.* [Ηλεκτρονικό]
Available at: https://www.geeksforgeeks.org/backtracking-algorithms/
[Πρόσβαση 16 01 2021].

Hemert, I.΄. J.-J. I. v., 2006. *jsoftware.* [Ηλεκτρονικό]
Available at: http://www.jsoftware.us/vol1/jsw0102-03.pdf
[Πρόσβαση 8 December 2020].

Leighton, F. T., 1979. [Ηλεκτρονικό]
Available at:
https://pdfs.semanticscholar.org/128d/490e1f116b410e4fd2482b54c742eb8d4371.pdf
[Πρόσβαση 29 Νοέμβριος 2020].

Rhyd Lewis, J. T. C. M. a. J. G., χ.χ. A wide-ranging computational. Στο: s.l.:s.n.

rlf icon image, χ.χ. [Ηλεκτρονικό]
Available at:
https://www.codeproject.com/KB/recipes/graph_coloring_using_RLF/gcq_3.png
[Πρόσβαση 29 Νοέμβριος 2020].

tutorialspoint, 2020. *tutorialspoint.* [Ηλεκτρονικό]
Available at:
https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_np_hard_complete_classes.htm
[Πρόσβαση Tuesday 12 2020].