ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Τεχνολογίες Υλοποίησης Αλγορίθμων

Χρήστος Ζαρολιάγκης

Καθηγητής
Τμήμα Μηχ/κων Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών
email: zaro@ceid.upatras.gr

Ενότητα 4

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο **«Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών»** έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

- Σχεδιασμός Πειραμάτων
- Μετρολογία
- Αξιολόγηση Απόδοσης

Πείραμα - Βασικό κριτήριο

- Πρέπει να απαντά σε συγκεκριμένες ερωτήσεις
- Να ακολουθεί την «επιστημονική μέθοδο» (Popper)
  - Προσδιορισμός ερωτημάτων
  - Συλλογή δεδομένων
  - Εξασφάλιση επανάληψης (αναπαραγωγής) και σπουδαιότητας πειράματος

Διαδικαστικοί Κανόνες

- Προσδιορισμός ξεκάθαρων στόχων:
  - Ποια ερωτήματα θέτονται ;
  - Τι ακριβώς θέλουμε να ελέγξουμε ;
- Συλλογή δεδομένων εισόδου
  - Όχι μετατροπές έως ότου συλλεχθούν όλα τα δεδομένα, προς αποφυγή δημιουργίας πολωτικών φαινομένων
- Ανάλυση (πειραματικών) δεδομένων για απάντηση αρχικών στόχων
  - Θεωρήστε σε *επόμενη* φάση πώς ένας νέος κύκλος πειραμάτων μπορεί να αυξήσει την κατανόησή σας

Σύνολα δεδομένων

- Τυχαία στιγμιότυπα $\Rightarrow$ μέση συμπεριφορά
- Συνθετικά στιγμιότυπα $\Rightarrow$ συγκεκριμένη συμπεριφορά (π.χ., ΠΧΠ)
- Πραγματικά στιγμιότυπα $\Rightarrow$ πρακτική συμπεριφορά
  - Πολύ χρήσιμα, αλλά συχνά δύσκολο να βρεθούν
  - Σύνολα δεδομένων που δημιουργήθηκαν από πραγματικά δεδομένα μέσω τυχαίων μεταθέσεων ή μέσω πιθανοτικών εκτιμήσεων μπορεί να αποδειχθούν χρήσιμα στη φάση δοκιμής

Ενδεχόμενοι κίνδυνοι

- Επιλογή του υπολογιστικού περιβάλλοντος (λανθάνουσα μνήμη, διευθυνσιοδότηση, μετακίνηση δεδομένων)
- Επιλογή γλώσσας προγραμματισμού (χειρισμός καταχωρητών, ενσωματωμένοι τύποι) και μεταγλωττιστή (ποιότητα βελτιστοποίησης και παραγωγής κώδικα)
- Ποιότητα προγραμματισμού (ικανότητα και συνέπεια προγρ/στών, χρήση βιβλιοθηκών)
- Συλλογή ή παραγωγή στιγμιοτύπων δεδομένων (ποικιλία τύπων και μεγάλα μεγέθη για διασφάλιση σπουδαιότητας)
- Μέθοδος ανάλυσης πειραματικών δεδομένων (ελαχιστοποίηση επίπτωσης από την επιλογή υπολογιστικού περιβάλλοντος)

Τυπικά λάθη

- Μη ενδιαφέρουσα πειραματική μελέτη
  - σύγκριση γλωσσών προγ/σμού ή (ανόμοιων) υπολογιστικών περιβαλλόντων
  - σύγκριση αλγορίθμων με τελείως διαφορετική συμπεριφορά
- Κακός σχεδιασμός
  - Πειράματα μέχρι κάποιο χρόνο ή χώρο χωρίς επαλήθευση ασυμπτωτικής πολυπλοκότητας
  - Χρήση πολύ λίγων στιγμιοτύπων εισόδου
  - Χρήση κώδικα χωρίς βελτιστοποίηση
  - Χρήση έτοιμου κώδικα χωρίς τεκμηρίωση
  - Άγνοια βιβλιοθηκών και τυποποιημένων δεδομένων δοκιμής (benchmarks)
- Κακή ανάλυση ή παρουσίαση δεδομένων
  - Απόρριψη αποτελεσμάτων χωρίς συστηματική διερεύνηση/επεξήγηση
  - Παρουσίαση όλων των αποτελεσμάτων χωρίς ανάλυση/δομή
  - Σύγκριση με όχι καλώς ορισμένα πρότυπα (π.χ. με την ρουτίνα ταξινόμησης του συστήματος)

Μετρολογία

- Τι μετράμε ;
  - Πώς το μετράμε ;
  - Πώς εξασφαλίζουμε ότι οι μετρήσεις δεν αλληλεπιδρούν με τα πειράματα ;

Καθολική Συμβουλή

Πάντοτε να κυττάτε πέρα από τις προφανείς μετρήσεις !

Μετρολογία – Τι μετράμε

Πειραματικές μελέτες που αφορούν την ποιότητα της (προσεγγιστικής) λύσης:

- Χρόνο με δομημένο τρόπο (π.χ. αριθμός επαναλήψεων)
- Ρυθμός σύγκλισης στην τελική λύση
- Όπου είναι δυνατόν, προσδιορισμός της βέλτιστης λύσης (με ωμή βία, αν είναι αναγκαίο)

Μετρολογία – Τι μετράμε

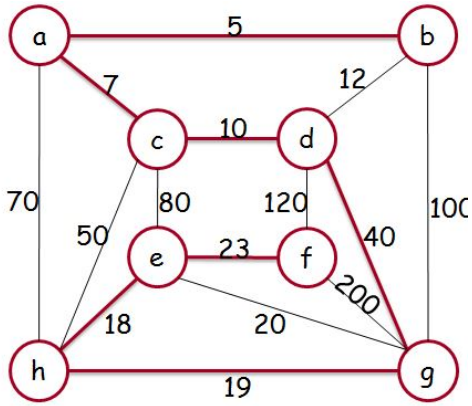
Πειραματικές συγκριτικές μελέτες που αφορούν αλγορίθμους επιλύσιμων προβλημάτων:

- *Πάντοτε* το χρόνο εκτέλεσης !
- Δομικές μετρικές (αριθμός προσπελάσεων στην μνήμη, αριθμός συγκρίσεων, αριθμός κορυφών/πλευρών που εξετάστηκαν, μετακινήσεις πλοκάδων δεδομένων, κλπ)
- Καθορισμός τιμών αναφοράς για (μελλοντική) κανονικοποίηση
- "Profiling" (που και σε ποια ποσότητα αναλώθηκε ο χρόνος ;)

Μετρολογία – Αναπαράσταση και ανάλυση δεδομένων

- Διασφάλιση αναπαραγωγής (επανάληψης) πειράματος
- Όχι απόρριψη «ανωμαλιών», αλλά διαλεύκανσή τους (όσο είναι δυνατόν)
- Χρήση κατάλληλων στατιστικών μετρικών
- Ελαχιστοποίηση επίδρασης υπολογιστικής πλατφόρμας, κώδικα, λανθάνουσας μνήμης, κλπ, με αντιπαραβολή και κανονικοποίηση
- Όχι παρουσίαση ακατέργαστων δεδομένων όταν είναι δυνατή η κανονικοποίηση
  - αποτέλεσμα επίλυσης / βέλτιστη λύση
  - χρόνος εκτέλεσης / χρόνος εκτέλεσης ρουτίνας αναφοράς

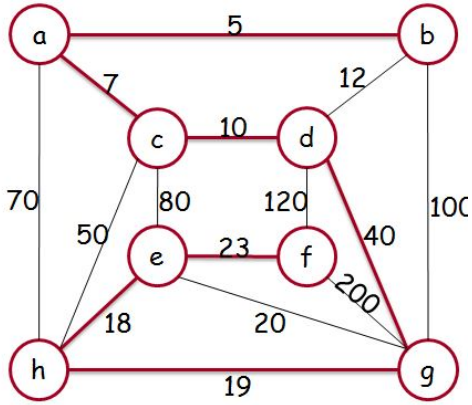
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Υλοποιημένοι Αλγόριθμοι

- (Kruskal 1956)
- (Prim 1958) (χρησιμοποιώντας δυαδικούς σωρούς και σωρούς pairing, Fibonacci, relaxed)
- (Cheriton & Tarjan, 1976)
- (Fredman & Tarjan, 1987)
- (Gabow, Galil, Spence, & Tarjan, 1986)

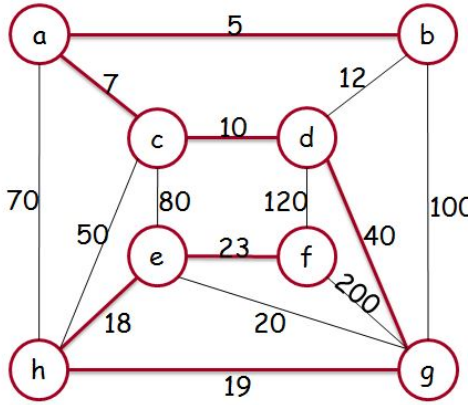
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Πειραματικό περιβάλλον και αρχικοποίηση

- Τρεις διαφορετικές υπολογιστικές πλατφόρμες
- Διαφορετικές γλώσσες προγ/σμού και μεταγλωττιστές, αλλά ένας προγραμματιστής
- Διερεύνηση ζητημάτων χαμηλού-επιπέδου (δείκτες vs αριθμοδείκτες, μετακινήσεις δεδομένων vs έμμεση διευθυνσιοδότηση, κλπ) πριν την τελική απόφαση των υλοποιήσεων

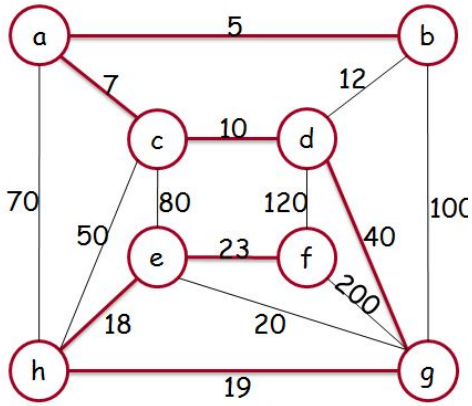
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Σύνολα δεδομένων εισόδου

- Πέντε διαφορετικές οικογένειες γραφημάτων
- Συγκεκριμένες συνθετικές οικογένειες γραφημάτων χειρότερης περίπτωσης (με συνθήκες αντιπαλότητας)
- Όλες οι οικογένειες συμπεριελάμβαναν πολύ μεγάλα γραφήματα (έως 10^6 κορυφές και άνω των 10^6 ακμών)
- Εκτέλεση τουλάχιστον 20 στιγμιοτύπων κάθε μεγέθους, ελέγχοντας την συνέπεια των αποτελεσμάτων σε ανεξάρτητες σειρές πειραμάτων
- Προληπτικά μέτρα εξ' αρχής για ελαχιστοποίηση προβλημάτων μεταφοράς δεδομένων από τη δευτερεύουσα στην κύρια μνήμη (εύκολο) καθώς και αστοχιών λανθάνουσας μνήμης (δύσκολο)

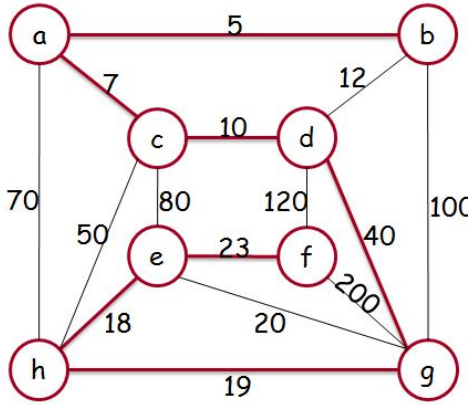
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Στόχοι πειραματικής μελέτης

- Ελαχιστοποίηση επίδρασης λανθάνουσας μνήμης και άλλων εξαρτήσεων της υπολογιστικής πλατφόρμας
- Κανονικοποίηση χρόνων εκτέλεσης σε όλες τις υπολογιστικές πλατφόρμες
- Εκτίμηση της επιρροής των όρων χαμηλής-τάξης και επαλήθευση της ασυμπτωτικής συμπεριφοράς
- Άμεση οπτική εικόνα της σχετικής αποδοτικότητας κάθε αλγορίθμου για κάθε τύπο και μέγεθος γραφήματος

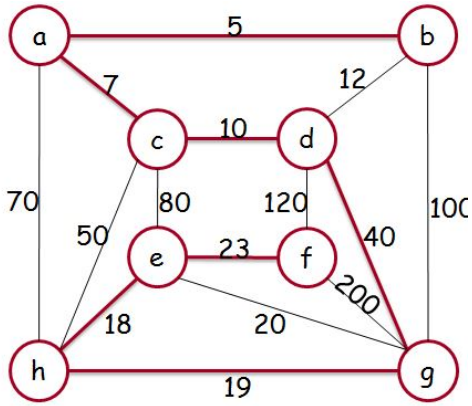
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Επίτευξη στόχων

- Απλή στρατηγική κανονικοποίησης ανά υπολογιστική πλατφόρμα:
  - Κανονικοποίηση μετρούμενων χρόνων εκτέλεσης των αλγορίθμων ΕΓΔ μέσω του χρόνου εκτέλεσης μιας απλής διαδικασίας γραμμικού χρόνου, με παρόμοιο πρότυπο προσπέλασης μνήμης
  - Διαδικασία: μέτρηση του αριθμού των ακμών του γραφήματος διατρέχοντας τις λίστες γειτονικότητας
- Παρόμοιο πρότυπο προσπέλασης μνήμης απάλειψε τις περισσότερες αστοχίες της λανθάνουσας μνήμης
- Παρόμοια εργασία σε αποαναφοροποίηση δεικτών απάλειψε τις περισσότερες εξαρτήσεις από την υπολογιστική πλατφόρμα
- Άμεση σύγκριση με το κάτω φράγμα της διαδικασίας γραμμικού χρόνου

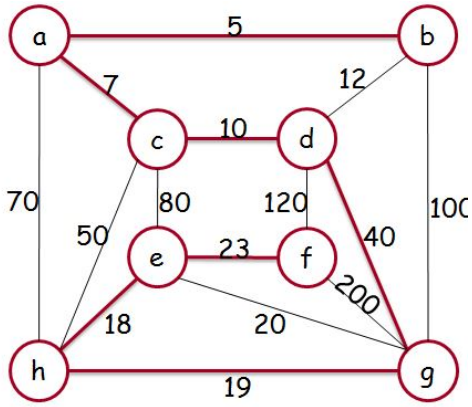
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Χρήσιμα πειραματικά επακόλουθα

- Αρχικές υλοποιήσεις με σωρούς Fibonacci και relaxed έδειχναν μη ανταγωνιστικές
- Υιοθέτηση ιδέας των (Driscoll et al, 1988): ομαδοποίηση κορυφών σε μεγάλες μονάδες (*«σάκους»*) έτσι ώστε αλλαγές τιμής κλειδιών να επιλύονται συνήθως εντός της μονάδας και να μην απαιτείται η αναδόμηση του σωρού
- Υλοποίηση σωρών με σάκους: αποδείχθηκε μια πολύ κρίσιμη απόφαση !

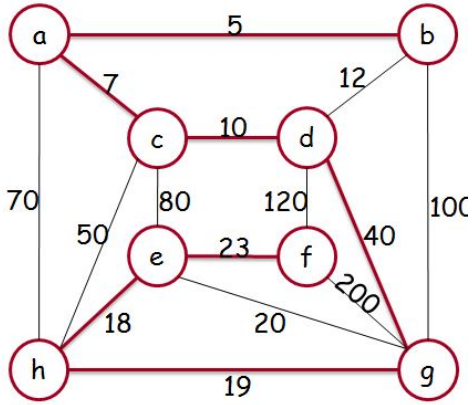
Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Πειραματικά αποτελέσματα

- Ο ταχύτερος αλγόριθμος ήταν μακράν ο απλούστερος: αλγόριθμος Prim υλοποιημένος με σωρό pairing ή απλό δυαδικό σωρό
- Ούτε οι πιο εκλεπτυσμένες υλοποιήσεις για λογικά μεγέθη γραφημάτων, αλλά ούτε και οι πιο εκλεπτυσμένοι αλγόριθμοι απέδωσαν (στην πράξη) την αναμενόμενη αποδοτικότητα
- Η τελική υλοποίηση του αλγορίθμου Prim με σωρούς Fibonacci ήταν σχεδόν 10 φορές ταχύτερες από την αρχική (λόγω των σάκων)
- Οι πολυλογαριθμικοί παράγοντες δεν παίζουν σημαντικό ρόλο

Περίπτωση Μελέτης: Ελάχ. Γεννητικά Δένδρα (Moret & Shapiro, 1995)



Διδάγματα

- Είναι αναγκαία η εκτέλεση πειραμάτων σε διαφορετικές υπολογιστικές πλατφόρμες και μεταγλωττιστές
- Μια πολύ μεγάλη ποικιλία μεγεθών είναι ουσιώδης
- Χρειάζεται εξαιρετικά μεγάλη προσοχή στην παραγωγή στιγμιοτύπων εισόδου
- Η κανονικοποίηση μέσω μιας κατάλληλης ρουτίνας βάσης είναι πολύ επιτυχής στην εξομάλυνση των διαφόρων διακυμάνσεων που οφείλονται στην αρχιτεκτονική και την λανθάνουσα μνήμη, όπως και στην ανάδειξη της ασυμπτωτικής συμπεριφορας και της σχετικής αποδοτικότητας των ανταγωνιστικών αλγορίθμων

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

- Μεταβατική κλειστότητα (transitive closure), ή προσπελασιμότητα (reachability): δεδομένου ενός κατευθυνόμενου γραφήματος G, να προσδιοριστεί, $\forall u, v \in G$, εάν η v είναι προσπελάσιμη από την u ($\exists$ μια διαδρομή u-v)

- Πλήρως δυναμικό πρόβλημα: διατήρηση μεταβατικής κλειστότητας όταν το G υφίσταται μεταβολές (εισαγωγές νέων ακμών και διαγραφές υπαρχόντων ακμών)

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

Υλοποιημένοι Αλγόριθμοι

- Πλήρως δυναμικοί
  - (King 1999)
  - (King & Thorup, 2001)
  - (Roditty & Zwick, 2002)
  - (Roditty 2003)
  - (Roditty & Zwick, 2004)
  - (Demetrescu & Italiano, 2000) (όχι αποκλειστικά συνδυαστικός)
- Ψευδο-πλήρως δυναμικοί
  - `RZ-Opt` βασισμένος στον μειωτικό αλγόριθμο των (Roditty & Zwick, 2002)
  - `Ital-Gen` (Frigioni,Miller,Nanni, & Zaroliagis 1998;2001)
- Απλοϊκοί (Frigioni,Miller,Nanni, & Zaroliagis 1998;2001)

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

Αλγόριθμος	Αναφορά	Επιμερ. Χρόνος Ενημέρωσης		Ερώτημα	Χώρος
Ital-Gen	(FMNZ,01)	$O(n)$ per ins	$O(m)$ per del	$O(1)$	$O(n^2)$
RZ-Opt	(KZ, 05;08)	$O(m)$ per ins	$O(n)$ per del	$O(1)$	$O(n^2)$
RZ-1	(RZ,02)	$O(m\sqrt{n})$		$O(\sqrt{n})$	$O(n^2)$
Rod	(R,03)	$O(n^2)$		$O(1)$	$O(n^2)$
Rod-Opt	(KZ, 05;08)	$O(n^2)$		$O(1)$	$O(n^2)$
RZ-P	(RZ,04)	$O(m + n \log n)$		$O(n)$	$O(nm)$
King-1	(K,99)	$O(n^2 \log n)$		$O(1)$	$O(n^2 \log n)$
King-2	(KZ, 05;08)	$O(n^2 \log n)$		$O(1)$	$O(n^2 \log n)$
King-3	(KZ, 05;08)	$O(n^2 D)$		$O(1)$	$O(n^2)$
DI	(DI,00)	$O(n^2)$		$O(1)$	$O(n^2)$
Simple	(FMNZ,01)	$O(1)$		$O(n + m)$	$O(n + m)$

- $m = m_0 + m'$ (m_0: αρχικός αριθμός ακμών)
- $m' = \Theta(m)$ εισαγωγές/διαγραφές ακμών
- D : διάμετρος γραφήματος

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

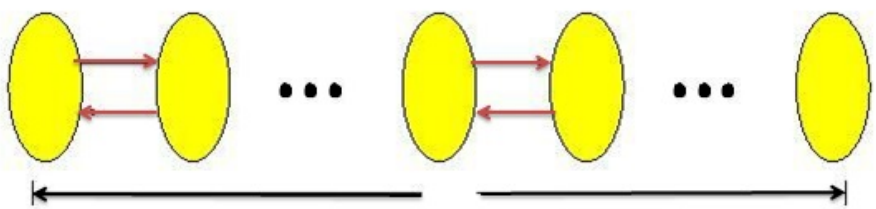
Πειραματικό περιβάλλον και αρχικοποίηση

- C++ με χρήση της LEDA και του LEDA Extension Package on Dynamic Graph Algorithms (LEP-DGA)
- Υπολογιστικά περιβάλλοντα:
  (I) Sun UltraSparc II με 4 επεξεργαστές στα 300 MHz, λειτουργικό σύστημα Solaris 7, 1.2GB κύρια μνήμη, και 2MB λανθάνουσας μνήμης L2 ανά επεξεργαστή
  (II) Intel Pentium 4 (P4) στα 1.6 GHz, με λειτουργικό σύστημα linux SUSE 7.3, 512MB κύρια μνήμη, και 512KB λανθάνουσα μνήμη L2
  (III) AMD Athlon στα 1.9 GHz, με λειτουργικό σύστημα linux Mandrake 10, 512MB κύρια μνήμη, και 256KB λανθάνουσα μνήμη L2

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

Σύνολα δεδομένων εισόδου

- Τυχαία στιγμιότυπα εισόδου
  - $n \in [100, 700]$, $m_0 \in \{0, n/2, n, n \log n, n^{1.5}, n^2/\log n, n^2/4\}$
  - Ακολουθία δυναμικών λειτουργιών σ: $|\sigma| \in [500, 50000]$
    ομοιόμορφα αναμεμειγμένες ενημερώσεις (εισ/διαγ) με ερωτήματα
    ομοιόμορφα αναμεμειγμένες εισαγωγές, διαγραφές, και ερωτήματα
- Συνθετικά (χειρότερης περίπτωσης) στιγμιότυπα είσόδου: n/k κλίκες K_k



- Πραγματικά δεδομένα εισόδου
  - Τμήμα του διαδικτυακού γραφήματος προσπελάσιμο από το RIPE (1259 κορυφές, 5101 ακμές)
  - Οδικό δίκτυο των ΗΠΑ (576 κορυφές, 1762 ακμές)

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

Στόχοι πειραματικής μελέτης

- Έλλειψη πρακτικής αξιολόγησης μιας πληθώρας νέων θεωρητικών (και πολύπλοκων) αλγορίθμων
- Επίτευξη καλύτερου αποτελέσματος από την τετριμμένη λύση (επανυπολογισμός εξ' αρχής)

Περίπτωση Μελέτης: Πλήρως Δυναμική Μεταβατική Κλειστότητα (Krommudas & Zaroliagis, 2005;2008)

Πειραματικά αποτελέσματα

- Αδόμητα (τυχαία & πραγματικά) στιγμιότυπα εισόδου:
  οι ψευδο-πλήρως δυναμικοί υπερτερούν
  - `RZ-Opt` $<_{2-3}$ `DBFS` $<_{8}$ `Ital-Gen` $<_{3-5}$ `RZ-1` $<_{3-7}$ `RZ-P` $<_{3}$ `Rod-Opt`
  - Μακράν οι χειρότεροι:
    `Rod-Opt` $<_{1.5}$ `Rod` $\ll_{10}$ `DI` $<_{2.5}$ `King-3` $<_{2}$ `King-2` $<_{2}$ `King-1`
    `DI`: μικρότερο ποσοστό αστοχιών λανθάνουσας μνήμης από κάθε άλλο αλγόριθμο

- Δομημένα (συνθετικα) στιγμιότυπα εισόδου:
  πλήρως δυναμικοί αλγόριθμοι έχουν πολύ καλή απόδοση (αλλά είναι σχεδόν πάντοτε χειρότεροι των απλοϊκών)

Αξιολόγηση Απόδοσης – Μέτρηση πραγματικού χρόνου

- Ασχολείται με την μέτρηση και την αξιολόγηση των πραγματικών χρονικών απαιτήσεων ενός προγράμματος
- Ο πραγματικός χρόνος εξαρτάται σε μεγάλο βαθμό από τον συγκεκριμένο μεταγλωττιστή και τις επιλογές του, καθώς και από το υπολογιστικό σύστημα πάνω στο οποίο εκτελείται το πρόγραμμα

Αξιολόγηση Απόδοσης – Μέτρηση πραγματικού χρόνου

- Η C++ παρέχει έναν μηχανισμό ρολογιού που χρησιμοποιεί
  - Την συνάρτηση `clock()` (επιστρέφει τον αριθμό των «χτύπων» από τη στιγμή που το πρόγραμμα άρχισε να εκτελείται)
  - Την μεταβλητή `CLOCKS_PER_SEC` (προσδιορίζει τη σχέση μεταξύ «χτύπων» ρολογιού και δευτερολέπτων)

- $$\text{Πραγματικός χρόνος (sec)} = \frac{\text{Αριθμός «Χτύπων»}}{\texttt{CLOCKS\_PER\_SEC}}$$

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση

- Συνάρτηση ενθετικής ταξινόμησης

```
template<class T>
void InsertionSort(T a[], int n)
{  // Sort a[0:n-1]
   for (int i = 1; i < n; i++)
   {
      // insert a[i] into a[0:i-1]
      T t = a[i];
      int j;
      for (j = i-1; j >= 0 && t < a[j]; j--)
         a[j+1] = a[j];
      a[j+1] = t;
   }
}
```

- Χρονική ΠΧΠ: $O(n^2)$

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση

- Στόχος: μέτρηση των πραγματικών χρονικών απαιτήσεων ΧΠ της συνάρτησης `InsertionSort`

- Προσδιορισμός
  - Μεγέθους εισόδου n
  - Σύνολα δεδομένων που αναγκάζουν την `InsertionSort` να αναδείξει την ΠΧΠ της

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση

- Ποιο στιγμιότυπο εισόσου είναι στιγμιότυπο ΧΠ για την `InsertionSort`;
- *Μια φθίνουσα ακολουθία*

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση – 1η προσπάθεια

```
#include <iostream>
#include <time.h>
#include "insort.h"

int main()
{
   int a[100000], step = 1000;
   clock_t start, finish;
   float seconds;
   cout << "CLOCKS_PER_SEC = " << CLOCKS_PER_SEC << endl;

   for (int n = 0; n <= 100000; n += step)
   {
      // get time for size n
      for (int i = 0; i < n; i++)
         a[i] = n - i; // initialization
      start = clock( );
      InsertionSort(a, n);
      finish = clock( );
      seconds = float(finish - start) / CLOCKS_PER_SEC;
      cout << n << ' ' << finish - start << ' ' << seconds << endl;
      if (n == 10000) step = 10000;
    }
   return 0;
}
```

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση – 1η προσπάθεια

- Αποτελέσματα (`CLOCKS_PER_SEC = 1000000`)
- Ακρίβεια: μόνο έως 0.01 secs
- *Μπορούμε να πετύχουμε καλύτερη ακρίβεια ;*

n	Clocks	Χρόνος
0	0	0
1000	0	0
2000	0	0
3000	10000	0.01
4000	20000	0.02
5000	20000	0.02
6000	40000	0.04
7000	50000	0.05
8000	60000	0.07
9000	80000	0.08
10000	100000	0.1
20000	390000	0.39
30000	890000	0.89
40000	1570000	1.57
50000	2450000	2.45
60000	3530000	3.53
70000	4790000	4.79
80000	6290000	6.29
90000	7960000	7.96
100000	9840000	9.84

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση – 2η προσπάθεια

```
#include <iostream>
#include <time.h>
#include "insort.h"

int main()
{
   int a[100000], n, i, step = 1000;
   long counter;
   float seconds;
   clock_t start, finish;
   for (n = 0; n <= 100000; n += step)
   {
      // get time for size n
      start = clock( ); counter = 0;
      while (clock( ) - start < 10*CLOCKS_PER_SEC)
      {
         counter++;
         for (i = 0; i < n; i++)
            a[i] = n - i; // initialize
         InsertionSort(a, n);
      }
      finish = clock( );
      seconds = float(finish - start) / CLOCKS_PER_SEC;
      cout << n << ' ' << counter << ' ' << seconds
           << ' ' << seconds / counter << endl;
      if (n == 10000) step = 10000;
   }
   return 0;
}
```

Αξιολόγηση Απόδοσης – Περίπτωση μελέτης: ενθετική ταξινόμηση – 2η προσπάθεια

- Αποτελέσματα (`CLOCKS_PER_SEC = 1000000`)

n	**Επαναλήψεις**	**Συνολικός Χρόνος**	**Χρόνος μιας Ταξινόμησης**
0	11653463	10	0.000000858114
1000	10060	10	0.000994036
2000	2541	10	0.00393546
3000	1129	10	0.0088574
4000	637	10	0.0156986
5000	406	10.01	0.0246552
6000	282	10.02	0.0355319
7000	208	10.02	0.0481731
8000	160	10.04	0.06275
9000	126	10.06	0.0798413
10000	102	10	0.0980392
20000	26	10.22	0.393077
30000	12	10.62	0.885
40000	7	11	1.57143
50000	5	12.27	2.454
60000	3	10.66	3.55333
70000	3	14.46	4.82
80000	2	12.64	6.32
90000	2	16.05	8.025
100000	2	19.72	9.86

Αξιολόγηση Απόδοσης – Μέτρηση πραγματικού χρόνου στην LEDA

Η συνάρτηση `used_time` της LEDA

- `float used_time()`: επιστρέφει τον τρέχοντα χρόνο CPU σε δευτερόλεπτα
- `float used_time(float& T)`: επιστρέφει τον χρόνο CPU που χρειάστηκε το πρόγραμμα από τη χρονική στιγμή T μέχρι αυτή τη στιγμή και καταχωρεί τον τρέχοντα χρόνο CPU στην μεταβλητή T

Αξιολόγηση Απόδοσης – Μέτρηση πραγματικού χρόνου στην LEDA

Το ακόλουθο τμήμα κώδικα

```
...
#include<LEDA/basic.h>
...
float T = used_time(); // sets T to the current CPU time
// experiment 1
...
cout << used_time(T);  // sets T to the current CPU time
                       // and returns the difference to
                       // the previous value of T
// experiment 2
...
cout << used_time(T);
```

εκτυπώνει τον CPU χρόνο που χρειάστηκε για κάθε ένα από τα δύο πειράματα

Αξιολόγηση Απόδοσης – Ενθετική ταξινόμηση `used_time` (1)

```
#include <iostream>
#include <LEDA/system/basic.h>
#include "insort.h"

int main()
{
   int a[100000], step = 1000;
   float T1, T2 ;

   for (int n = 0; n <= 100000; n += step)
   {
      // get time for size n
      for (int i = 0; i < n; i++)
         a[i] = n - i; // initialize
      T1=leda::used_time();  cout << T1 << ' ';
      InsertionSort(a, n);
      T2=leda::used_time(T1); cout << T1 << ' ';
      cout << n << ' ' << T2 << endl;
      if (n == 10000) step = 10000;
      }
   return 0;
}
```

Αξιολόγηση Απόδοσης – Ενθετική ταξινόμηση `used_time` (1)

Αποτελέσματα

T_1 (πριν)	T_1 (μετά)	n	T_2
0	0	0	0
0	0	1000	0
0	0	2000	0
0	0.01	3000	0.01
0.01	0.03	4000	0.02
0.03	0.05	5000	0.02
0.05	0.09	6000	0.04
0.09	0.14	7000	0.05
0.14	0.20	8000	0.06
0.20	0.28	9000	0.08
0.28	0.38	10000	0.10
0.38	0.77	20000	0.39
0.77	1.65	30000	0.88
1.65	3.22	40000	1.57
3.22	5.66	50000	2.44
5.66	9.24	60000	3.58
9.24	14.04	70000	4.80
14.04	20.34	80000	6.30
20.34	28.28	90000	7.94
28.28	38.12	100000	9.84

Αξιολόγηση Απόδοσης – Ενθετική ταξινόμηση `used_time` (2)

```
#include <iostream>
#include <LEDA/system/basic.h>
#include "insort.h"

int main()
{  int a[100000], step = 1000, iters = 10;
   float T1, T2 ;

   for (int n = 0; n <= 100000; n += step)
   {  T2=0;
      for (int j = 0; j < iters; j++)
      {
         // get time for size n
         for (int i = 0; i < n; i++)
            a[i] = n - i; // initialize
         T1 = leda::used_time();  cout << "T1= " << T1 << ' ';
         InsertionSort(a, n);
         T2 = T2 + leda::used_time(T1); cout << "T2= " << T2 << ' ';
      }
      cout << endl;
      cout << n << ' ' << "Avg(T2) = " << T2/iters << endl;
      if (n == 10000) step = 10000;
   }
   return 0;
}
```

Αξιολόγηση Απόδοσης – Ενθετική ταξινόμηση `used_time` (2)

Αποτελέσματα

n	$\text{Avg}(T_2) = \sum_{i=1}^{10} T_2(i)/10$
0	0
1000	0.001
2000	0.004
3000	0.008
4000	0.016
5000	0.024
6000	0.035
7000	0.048
8000	0.063
9000	0.079
10000	0.098
20000	0.394
30000	0.884
40000	1.567
50000	2.451
60000	3.529
70000	4.813
80000	6.294
90000	7.960
100000	9.833

Αξιολόγηση Απόδοσης – Μέτρηση πραγματικού χρόνου στην LEDA

Άλλα βοηθητικά προγράμματα για μέτρηση χρόνου

- Κλάση `timer` (δείτε το εγχειρίδιο της LEDA)

Συμπεράσματα – Χαρακτηριστικά γνωρίσματα καλών πειραμάτων

- Σαφώς προσδιορισμένοι στόχοι
- Δοκιμή ευρείας κλίμακας, τόσο ως προς το μέγεθος των στιγμιοτύπων εισόδου, όσο και ως προς τον αριθμό των στιγμιοτύπων που χρησιμοποιούνται για κάθε μέγεθος
- Μίξη πραγματικών, τυχαίων και συνθετικών στιγμιοτύπων εισόδου, συμπεριλαμβανομένων τυχόν τυποποιημένων δεδομένων δοκιμής (benchmarks) όπου υπάρχουν
- Σαφώς καθορισμένοι παράμετροι, συμπεριλαμβανομένων εκείνων που
  - προσδιορίζουν τεχνητά στιγμιότυπα εισόδου
  - καθορίζουν την συλλογή δεδομένων
  - καθορίζουν το περιβάλλον δοκιμής (υπολογιστές, μεταγλωττιστές, κλπ)

Συμπεράσματα – Χαρακτηριστικά γνωρίσματα καλών πειραμάτων

- Στατιστική ανάλυση των αποτελεσμάτων και προσπάθεια συσχέτισής τους με την φύση των αλγορίθμων και των δεδομένων δοκιμής
- Αναπαραγωγή: δημόσια διάθεση των
  - στιγμιοτύπων εισόδου και των γεννητόρων τους (προκειμένου άλλοι επιστήμονες να μπορούν να εκτελέσουν τους δικούς τους αλγόριθμους στα ίδια στιγμιότυπα)
  - του κώδικα των αλγορίθμων (επιθυμητό)

Τέλος Ενότητας

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση **1.0**.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Σημείωμα Ιστορικού Εκδόσεων Έργου

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει :

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους