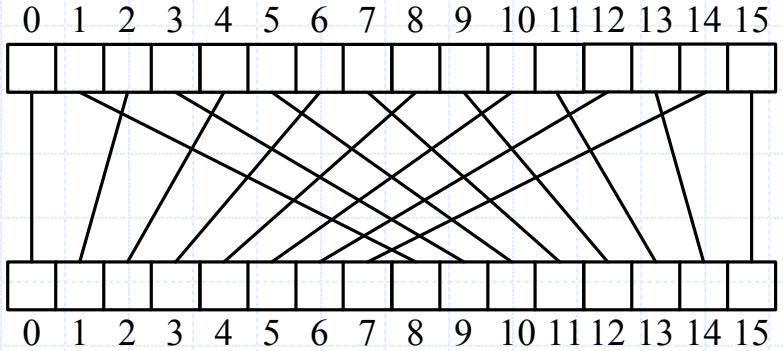
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Αλγόριθμοι και πολυπλοκότητα

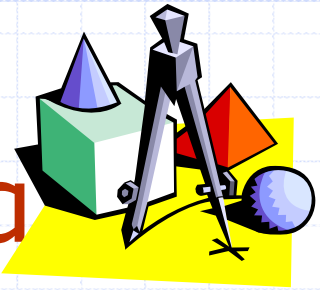
Ο Μετασχηματισμός Fast Fourier

Ιωάννης Τόλλης
Τμήμα Επιστήμης Υπολογιστών

Ο Μετασχηματισμός Fast Fourier



Γενικές Γραμμές και Διάβασμα



- Πρόβλημα Πολ/σμού Πολυωνύμων
- Πρωταρχικές Ρίζες της Μονάδος
- Μετασχηματισμός Fourier Διακριτού Χρόνου
- Αλγόριθμος FFT
- Πολ/σμός Ακεραίων
- FFT Πολ/σμός Ακεραίων σε Java

Πολυώνυμα

- Πολυώνυμα:

$$p(x) = 5 + 2x + 8x^2 + 3x^3 + 4x^4$$

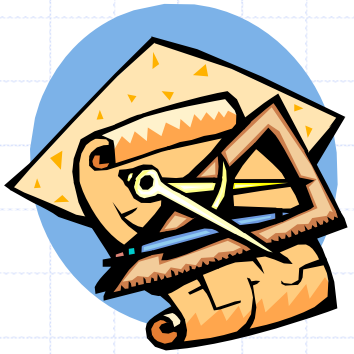
- Γενικά,

$$p(x) = \sum_{i=0}^{n-1} a_i x^i$$

or

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

Polynomial Evaluation



- **Κανόνας Horner**
  - Δοσμένων συντελεστών: $(a_0,a_1,a_2,\ldots,a_{n-1})$, ορίζουμε πολυώνυμο

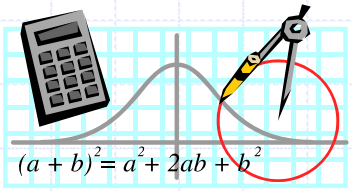
$$p(x)=\sum_{i=0}^{n-1} a_i x^i$$

  - Δοσμένου x, αποτιμούμε το p(x) σε χρόνο O(n) με χρήση της εξίσωσης

$$p(x)=a_0+x(a_1+x(a_2+\cdots+x(a_{n-2}+xa_{n-1})\cdots))$$

- **Eval**(A,x): [όπου $A=(a_0,a_1,a_2,\ldots,a_{n-1})$]
  - If n=1, then return a_0
  - Else,
    - Let $A'=(a_1,a_2,\ldots,a_{n-1})$ [υπόθεση: αυτό γίνεται σε σταθερό χρόνο]
    - return a_0+x***Eval**(A',x)

Polynomial Multiplication Problem



- Δοσμένων συντελεστών: $(a_0,a_1,a_2,\ldots,a_{n-1})$ και$(b_0,b_1,b_2,\ldots,b_{n-1})$ και ορίζοντας δύο πολυώνυμα, p() και q(), και τον αριθμό x, υπολογίζουμε το γινόμενο p(x)q(x).

- Ο κανόνας Horner δεν βοηθάει, καθώς:

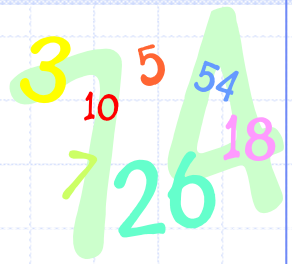
$$p(x)q(x) = \sum_{i=0}^{n-1} c_i x^i$$

όπου

$$c_i = \sum_{j=0}^{i} a_j b_{i-j}$$

- Μία ακριβής αποτίμηση θα έπαιρνε χρόνο $O(n^2)$. Ο "μαγικός" FFT θα το κάνει σε χρόνο O(n log n) .

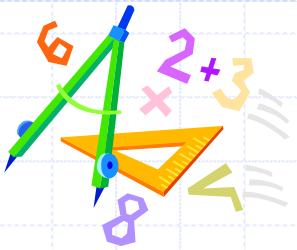
Παρεμβολή Πολυωνύμων & Πολ/σμός Πολυωνύμων



- Δοσμένου συνόλου n σημείων στο επίπεδο με διακριτές συντεταγμένες, υπάρχει **ακριβώς ένα** πολυώνυμο (n-1)-βαθμού που να περνάει από όλα αυτά τα σημεία.
- Εναλλακτική προσέγγιση για τον υπολογισμό του p(x)q(x):
  - Υπολόγισε το p() σε 2n τιμές, $x_0,x_1,\ldots,x_{2n-1}$.
  - Υπολόγισε το q() στις ίδιες 2n x τιμές.
  - Βρες το πολυώνυμο (2n-1)-βαθμού που περνάει απο τα σημεία $\{(x_0,p(x_0)q(x_0)), (x_1,p(x_1)q(x_1)), \ldots, (x_{2n-1},p(x_{2n-1})q(x_{2n-1}))\}$.

- Δυστυχώς, μια ακριβής αποτίμηση θα πάρει πάλι χρόνο $O(n^2)$, όσο θα χρειαζόμασταν εφαρμόζοντας τον κανόνα Horner χρόνου O(n) για την αποτίμιση σε 2n διαφορετικά σημεία.
- Ο "μαγικός" FFT θα το κάνει σε χρόνο O(n log n), διαλέγοντας 2n σημεία που εκτιμώνται εύκολα...

Primitive Roots of Unity



- Το ω είναι η ***n-οστή πρωταρχική ρίζα της μονάδος***, για n>1, αν
  - $\omega^n = 1$
  - Οι αριθμοί 1, ω, ω^2, ..., ω^{n-1} είναι όλοι διακριτοί
- Παράδειγμα 1:
  - Z^*_{11}:

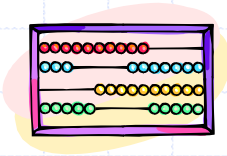
x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^10
1	1	1	1	1	1	1	1	1	1
2	4	8	5	10	9	7	3	6	1
3	9	5	4	1	3	9	5	4	1
4	5	9	3	1	4	5	9	3	1
5	3	4	9	1	5	3	4	9	1
6	3	7	9	10	5	8	4	2	1
7	5	2	3	10	4	6	9	8	1
8	9	6	4	10	3	2	5	7	1
9	4	3	5	1	9	4	3	5	1
10	1	10	1	10	1	10	1	10	1

  - 2, 6, 7, 8 είναι οι 10-th ρίζες της μονάδος στο Z^*_{11}
  - $2^2=4$, $6^2=3$, $7^2=5$, $8^2=9$ είναι 5-th ρίζες της μονάδος στο Z^*_{11}
  - $2^{-1}=6$, $3^{-1}=4$, $4^{-1}=3$, $5^{-1}=9$, $6^{-1}=2$, $7^{-1}=8$, $8^{-1}=7$, $9^{-1}=5$
- Παράδειγμα 2: Ο μιγαδικός $e^{2\pi i/n}$ είναι η πρωταρχική n-οστή ρίζα της μονάδος, όπου $i = \sqrt{-1}$

Ιδιότητες των πρωταρχικών ριζών της μονάδος

- **Ιδιότητα Αντιστροφής:** Αν ω είναι μια πρωταρχική ρίζα της μονάδος, τότε $\omega^{-1}=\omega^{n-1}$
  - Απόδειξη: $\omega\omega^{n-1}=\omega^{n}=1$
- **Ιδιότητα εξουδετέρωσης :** Για μη μηδενικά -n<k<n, $\sum_{j=0}^{n-1}\omega^{kj}=0$
  - Απόδειξη : $$\sum_{j=0}^{n-1}\omega^{kj}=\frac{(\omega^{k})^{n}-1}{\omega^{k}-1}=\frac{(\omega^{n})^{k}-1}{\omega^{k}-1}=\frac{(1)^{k}-1}{\omega^{k}-1}=\frac{1-1}{\omega^{k}-1}=0$$
- **Ιδιότητα αναγωγής:** Αν w είναι μια (2n)-βαθμού πρωταρχική ρίζα της μονάδος, τότε ω^2 είναι n-βαθμού πρωταρχική ρίζα της μονάδος. Απόδειξη:Αν $1,\omega,\omega^2,\ldots,\omega^{2n-1}$ διακριτές, το ίδιο είναι και $1,\omega^2,(\omega^2)^2,\ldots,(\omega^2)^{n-1}$
- **Ιδιότητα Ανάκλασης:** Αν περιττό, τότε $\omega^{n/2} = -1$.
  - Απόδειξη : Από την ιδιότητα εξουδετέρωσης, για k=n/2:
  $$0=\sum_{j=0}^{n-1}\omega^{(n/2)j}=\omega^{0}+\omega^{n/2}+\omega^{0}+\omega^{n/2}+\cdots+\omega^{0}+\omega^{n/2}=(n/2)(1+\omega^{n/2})$$
  - Επομένως : $\omega^{k+n/2}= -\omega^{k}$.

Μετασχηματισμός Fourier Διακριτού Χρόνου



- Δοσμένων συντελεστών $(a_0,a_1,a_2,\ldots,a_{n-1})$ για ένα πολυώνυμο p(x) (n-1)-βαθμού
- Ο **Διακριτός Μετασχ. Fourier** είναι η αποτίμηση του p στις τιμές
  - $1,\omega,\omega^2,\ldots,\omega^{n-1}$
  - Παράγουμε το $(y_0,y_1,y_2,\ldots,y_{n-1})$, όπου $y_j=p(\omega^j)$
  - Δηλαδή, $$y_j = \sum_{i=0}^{n-1} a_i \omega^{ij}$$
  - Σε μορφή πινάκων: **y=Fa,** όπου **F**$[i,j]=\omega^{ij}$.

- Ο **Αντίστροφος Διακριτός Μετασχ. Fourier** επανακτά τους συντελεστές ενός πολυωνύμου (n-1)-βαθμού, δοσμένων των τιμών του, στα σημεία $1,\omega,\omega^2,\ldots,\omega^{n-1}$
  - Σε μορφή πινάκων : **a=F** $^{-1}$**y**, όπου **F** $^{-1}[i,j]=\omega^{-ij}/n$.

Ορθότητα του αντίστροφου DFT

- Ο DFT και ο αντίστροφος DFT είναι πραγματικά αντίστροφες πράξεις
- Απόδειξη: Έστω $\mathbf{A=F^{-1}F}$. Θέλουμε να δείξουμε ότι $\mathbf{A=I}$, όπου

$$A[i,j]=\frac{1}{n}\sum_{k=0}^{n-1}\omega^{-ki}\omega^{kj}$$

- Αν i=j, τότε

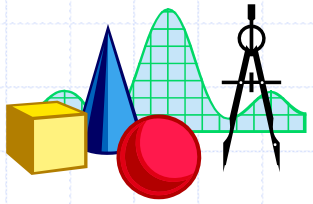
$$A[i,i]=\frac{1}{n}\sum_{k=0}^{n-1}\omega^{-ki}\omega^{ki}=\frac{1}{n}\sum_{k=0}^{n-1}\omega^{0}=\frac{1}{n}n=1$$

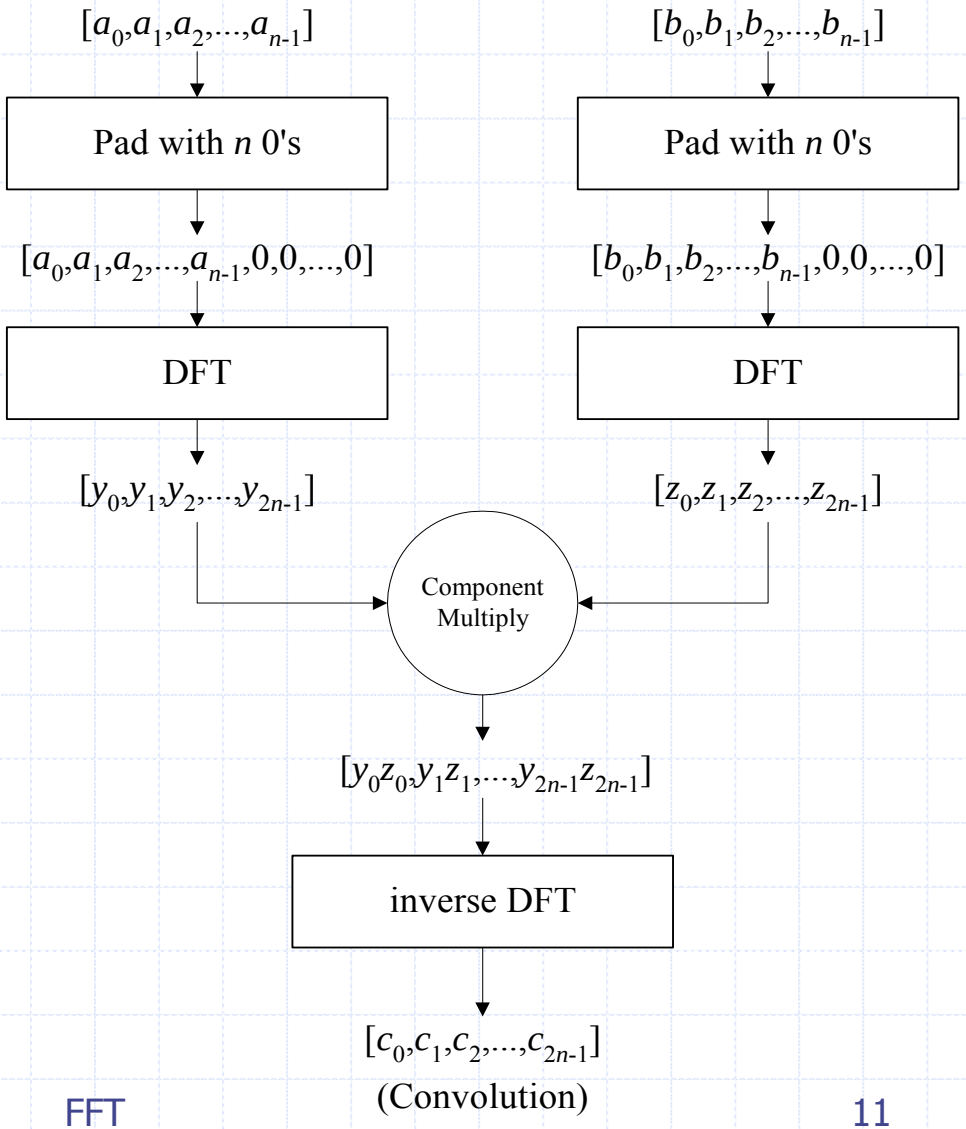
- Αν i διάφορο του j, τότε

$$A[i,j]=\frac{1}{n}\sum_{k=0}^{n-1}\omega^{(j-i)k}=0 \qquad (\text{ιδιότητα εξουδετέρωσης})$$

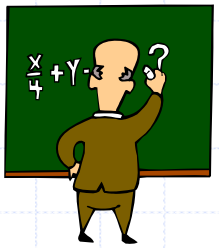
Συνέλιξη

- Ο DFT και ο αντίστροφος DFT μπορούν να χρησιμοποιηθούν για τον πολ/σμό πολυωνύμων
- Έτσι μπορούμε να πάρουμε τους συντελεστές του πολυωνυμικού γινομένου γρήγορα, αν μπορούμε να υπολογίσουμε τον DFT (και τον αντίστροφό του) γρήγορα ...





Ο Μετασχηματισμός Fast Fourier



- Ο FFT είναι ένας αποτελεσματικός αλγόριθμος για τον υπολογισμό του DFT
- Ο FFT βασίζεται στο παράδειγμα διαίρει-και-βασίλευε:
  - Αν n περιττός, μπορούμε να διαιρέσουμε ένα πολυώνυμο :

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

σε δύο πολυώνυμα

$$p^{\text{even}}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$$

$$p^{\text{odd}}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$$

και μπορούμε να γράψουμε

$$p(x) = p^{\text{even}}(x^2) + x p^{\text{odd}}(x^2).$$

Ο αλγόριθμος FFT

Algorithm FFT$(\mathbf{a}, \omega)$:

Input: An n-length coefficient vector $\mathbf{a} = [a_0, a_1, \ldots, a_{n-1}]$ and a primitive nth root of unity ω, where n is a power of 2

Output: A vector $\mathbf{y}$ of values of the polynomial for $\mathbf{a}$ at the nth roots of unity

if $n = 1$ **then**
  return $\mathbf{y} = \mathbf{a}$.
$x \leftarrow \omega^0$ {x will store powers of ω, so initially $x = 1$.}
{Divide Step, which separates even and odd indices}
$\mathbf{a}^{\text{even}} \leftarrow [a_0, a_2, a_4, \ldots, a_{n-2}]$
$\mathbf{a}^{\text{odd}} \leftarrow [a_1, a_3, a_5, \ldots, a_{n-1}]$
{Recursive Calls, with ω^2 as $(n/2)$th root of unity, by the reduction property}
$\mathbf{y}^{\text{even}} \leftarrow \text{FFT}(\mathbf{a}^{\text{even}}, \omega^2)$
$\mathbf{y}^{\text{odd}} \leftarrow \text{FFT}(\mathbf{a}^{\text{odd}}, \omega^2)$
{Combine Step, using $x = \omega^i$}
for $i \leftarrow 0$ to $n/2 - 1$ **do**
  $y_i \leftarrow y_i^{\text{even}} + x \cdot y_i^{\text{odd}}$
  $y_{i+n/2} \leftarrow y_i^{\text{even}} - x \cdot y_i^{\text{odd}}$ {Uses reflective property}
  $x \leftarrow x \cdot \omega$
return $\mathbf{y}$

Χρόνος εκτέλεσης O(n log n). [όμοια για αντίστροφο FFT]

Πολ/σιάζοντας Μεγάλους Ακεραίους

- Δοσμένων ακεραίων I και J των N-bits integers, υπολογίζουμε το IJ.
- Υπόθεση: πολ/σμός λέξεων των O(log N) bits σε σταθερό χρόνο.
- Προετοιμασία: Βρες πρώτο αριθμό p=cn+1 που να μπορεί να αναπαρασταθεί σε μία λέξη, και θέσε m=(log p)/3, ώστε να αναπαραστήσουμε τα I και J ως διανύσμαρτα μήκους n από λέξεις των m-bits.
- Βρίσκοντας μια πρωταρχική ρίζα της μονάδος.
  - Βρές έναν generator x στο Z^*_p.
  - Τότε $\omega=x^c$ είναι μια πρωταρχική ρίζα της μονάδος n βαθμού στο Z^*_p (arithmetic is mod p)
- Εφάρμοσε τη συνέλιξη και τον αλγόριθμο FFT για τον υπολογισμό της συνέλιξης C της διανυσματικής αναπαράστασης του I με αυτή του J.
- Μετά υπολόγισε: $K = \sum_{i=0}^{n-1} c_i 2^{mi}$
- Το K διάνυσμα αναπαρηστά το IJ, και έχει χρόνο υπολογισμού O(n logn)

Παράδειγμα σε Java: Πολ/σμός Μεγάλων Ακεραίων

- Προετοιμασία: Όρισε την κλάση BigInt class, και περιέλαβε βασικές παραμέτρους, περιέλαβε τον πρώτο, P, και την πρωταρχικλη ρίζα της μονάδος, OMEGA.

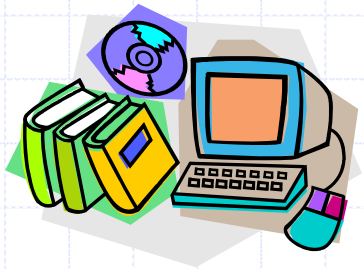
```
import java.lang.*;
import java.math.*;
import java.util.*;

public class BigInt {
  protected int signum=0;                  // neg = -1, 0 = 0, pos = 1
  protected int[] mag;                     // magnitude in little-endian format
  public final static int MAXN=134217728;  // Maximum value for n
  public final static int ENTRYSIZE= 10;   // Bits per entry in mag
  protected final static long P=2013265921; // The prime 15*2^{27}+1
  protected final static int OMEGA=440564289; // Root of unity 31^{15} mod P
  protected final static int TWOINV=1006632961; // 2^{-1} mod P
```

Μέθοδος Πολ/σμού Ακεραίων σε Java

- Χρήση συνέλιξης για τον πολ/σμό δυο μεγάλων ακεραίων, this και val:

```
public BigInt multiply(BigInt val) {
  int n = makePowerOfTwo(Math.max(mag.length,val.mag.length))*2;
  int signResult = signum * val.signum;
  int[] A = padWithZeros(mag,n);   // copies mag into A padded w/ 0's
  int[] B = padWithZeros(val.mag,n); // copies val.mag into B padded w/ 0's
  int[] root = rootsOfUnity(n);      // creates all n roots of unity
  int[] C = new int[n];              // result array for A*B
  int[] AF = new int[n];             // result array for FFT of A
  int[] BF = new int[n];             // result array for FFT of B
  FFT(A,root,n,0,AF);
  FFT(B,root,n,0,BF);
  for (int i=0; i<n; i++)
    AF[i] = (int)(((long)AF[i]*(long)BF[i]) % P);   // Component multiply
  reverseRoots(root);                // Reverse roots to create inverse roots
  inverseFFT(AF,root,n,0,C);         // Leaves inverse FFT result in C
  propagateCarries(C);               // Convert C to right no. bits per entry
  return new BigInt(signResult,C);
  }
```

FFT στο Z^*_p σε Java



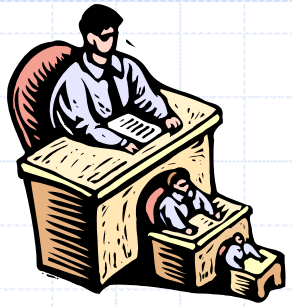
```
public static void FFT(int[] A, int[] root, int n, int base, int[] Y) {
  int prod;
  if (n==1) {
    Y[base] = A[base];
    return;
    }
  inverseShuffle(A,n,base);     // inverse shuffle to separate evens and odds
  FFT(A,root,n/2,base,Y);       // results in Y[base] to Y[base+n/2-1]
  FFT(A,root,n/2,base+n/2,Y); // results in Y[base+n/2] to Y[base+n-1]
  int j = A.length/n;
  for (int i=0; i<n/2; i++) {
    prod = (int)(((long)root[i*j]*Y[base+n/2+i]) % P);
    Y[base+n/2+i] = (int)(((long)Y[base+i] + P - prod) % P);
    Y[base+i] = (int)(((long)Y[base+i] + prod) % P);
    }
}
public static void inverseFFT(int[] A, int[] root, int n, int base, int[] Y) {
  int inverseN = modInverse(n);      // n^{-1}
  FFT(A,root,n,base,Y);
  for (int i=0; i<n; i++)
    Y[i] = (int)(((long)Y[i]*inverseN) % P);
}
```

Μέθοδοι που υποστηρίζουν FFT στο Z^*_p σε Java

```
protected static int modInverse(int n) { // assumes n is power of two
  int result = 1;
  for (long twoPower = 1; twoPower < n; twoPower *= 2)
    result = (int)(((long)result*TWOINV) % P);
  return result;
}
protected static void inverseShuffle(int[] A, int n, int base) {
  int shift;
  int[] sp = new int[n];
  for (int i=0; i<n/2; i++) { // Unshuffle A into the scratch space
    shift = base + 2*i;
    sp[i] = A[shift];          // an even index
    sp[i+n/2] = A[shift+1];  // an odd index
    }
  for (int i=0; i<n; i++)
    A[base+i] = sp[i];         // copy back to A
}
```

```
protected static int[] rootsOfUnity(int n) { //assumes n is power of 2
  int t = MAXN;
  int nthroot = OMEGA;
  for (int t = MAXN; t>n; t /= 2)  // Find prim. nth root of unity
    nthroot = (int)(((long)nthroot*nthroot) % P);
  int[] roots = new int[n];
  int r = 1;              // r will run through all nth roots of unity
  for (int i=0; i<n; i++) {
    roots[i] = r;
    r = (int)(((long)r*nthroot) % P);
    }
  return roots;
}
protected static void propagateCarries(int[] A) {
  int i, carry;
  carry = 0;
  for (i=0; i<A.length; i++) {
    A[i] = A[i] + carry;
    carry = A[i] >>> ENTRYSIZE;
    A[i] = A[i] - (carry << ENTRYSIZE);
    }
  }
```

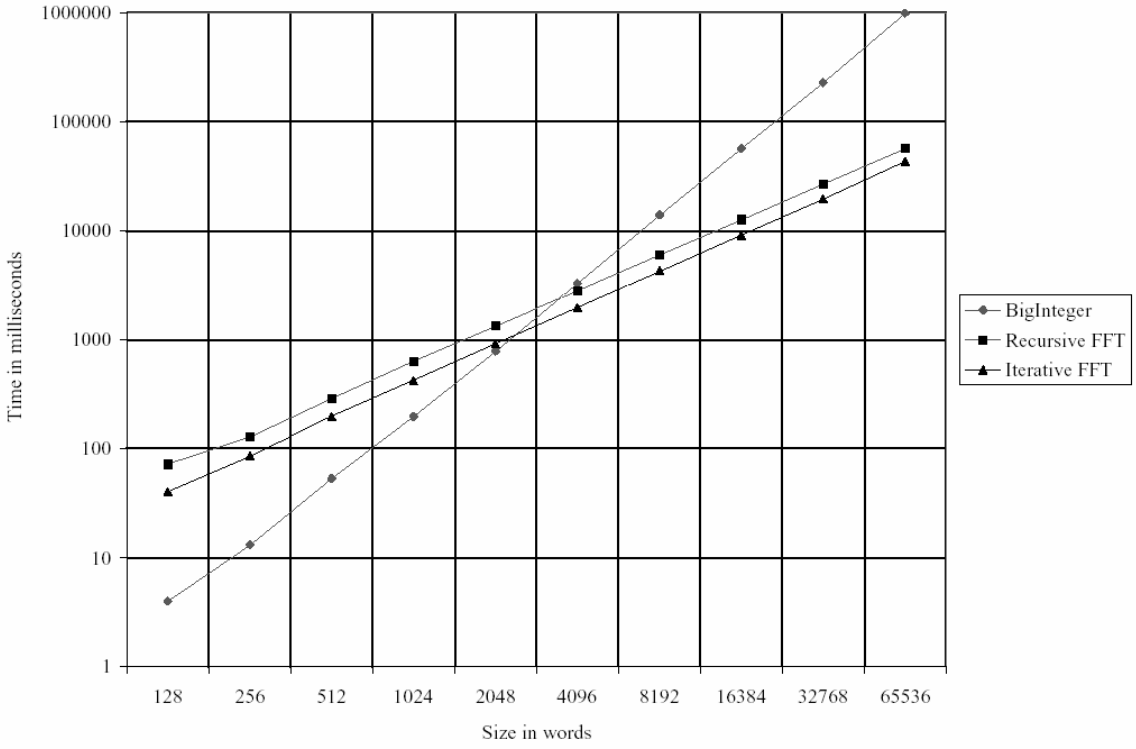
Μη αναδρομικός FFT



- Υπάρχει επίσης μη αναδρομική έκδοση του FFT
  - Εκτελεί τον FFT στον χώρο
  - Προ- υπολογίζει όλες τις ρίζες της μονάδος.
  - Εκτελει μια αθροιστική συλλογή από shuffles στο A και στο B προηγούμενα τοτ FFT, που ισοδυναμεί με την ανάθεση της τιμής με δείκτη i στο δείκτη bit-reverse(i).
- Ο κώδικας είναι πιο πολύπλοκος, αλλά ο χρόνος εκτέλεσης είναι γρηγορότερος κατά μια σταθερά, λόγω βελτιωμένου overhead

Πειραματικά Αποτελέσματα

- Η κλίμακα Log-log δείχνει παραδοσιακές εκτελέσεις πολ/σμού σε χρόνο $O(n^2)$, ενώ οι FFT εκδόσεις είναι σχεδόν γραμμικές.



Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης» 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.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

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

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως Μη Εμπορική ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Σημείωμα Αναφοράς

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.