Algorithm Design & Analysis (CSE222)

Lecture-5

Recap

- Counting Inversions
  - Merge-Count()

MERGE-COUNT(L, R)

Initialize $pt_\ell = 1; pt_r = 1; i = 1$ and $Count = 0$
Initialize an empty array A of size $|L| + |R|$.
while $(pt_\ell \leq |L|$ and $pt_r \leq |R|)$
    if $(L[pt_\ell] \leq R[pt_r]$
        $A[i] = L[pt_\ell]$ and $pt_\ell = pt_\ell + 1$
    else
        $A[i] = R[pt_r]$; $pt_r = pt_r + 1$ and $Count = Count + |L| - pt_\ell + 1$
    $i = i + 1$
if $(pt_\ell > |L|)$
    Append remaining elements of R into A
if $(pt_r > |R|)$
    Append remaining elements of L into A
Return $(Count, A)$

COUNT-INVERSION(A)

if $|A| = 1$
else
    $m = \left\lceil \frac{|A|}{2} \right\rceil$
    $A_{Left} = A[1, \ldots, m]$ and $A_{Right} = A[m+1, \ldots, |A|]$
    $(C_{Left}, A_{Left}) =$ COUNT-INVERSION(A_{Left})
    $(C_{Right}, A_{Right}) =$ COUNT-INVERSION(A_{Right})
    $(C_{Split}, A) =$ MERGE-COUNT(A_{Left}, A_{Right})
    $Count = C_{Left} + C_{Right} + C_{Split}$
Return $(Count, A)$

Recap

- Counting Inversions
  - Merge-Count()
- Select k Smallest Elements
  - Randomized Algorithm: Partition()

$\textsc{RandQuickSelect}(A, k)$

if $(|A| == 1)$
    Return A
$p = \textsc{ChoosePivot}(A)$
$Lesser = \{A[i] : A[i] < p\}$; $Greater = \{A[i] : A[i] > p\}$
if $(|Lesser| == k - 1)$
    Return p
if $(|Lesser| > k - 1)$
    Return $\textsc{RandQuickSelect}(Lesser, k)$
else
    Return $\textsc{RandQuickSelect}(Greater, k - |Lesser| - 1)$

Outline

- Select k-Smallest Element
- Fast Fourier transform

Deterministic Algorithm (B. F. P. R. & T. 1973)

Input: Array *A* of size n, integer *k*.
Output: k^{th} smallest element in *A*.

DetQuickSelect(*A*,*k*)

1. Group *A* into n/5 groups, each of size 5. Find median of each group.
2. Recursively find median of the medians. Lets call it p.
3. Split A into subarrays *Lesser* & *Greater*. Set $L = |Lesser|$
4. If $L = k - 1$ then return *p*.
5. If $L > k - 1$ then return DetQuickSelect(*Lesser*, *k*).
6. If $L < k - 1$ then return DetQuickSelect(*Greater*, $k - L - 1$)

Claim: The worst case running time of DetQuickSelect() is O(n).

Deterministic Algorithm (B. F. P. R. & T. 1973)

Input: Array *A* of size n, integer *k*.
Output: k^{th} smallest element in *A*.

$\text{DetQuickSelect}(A, k)$

if $(|A| == 1)$
    Return A
$p = \text{ChoosePivot}(A)$
$Lesser = \{A[i] : A[i] < p\}$; $Greater = \{A[i] : A[i] > p\}$
if $(|Lesser| == k - 1)$
    Return p
if $(|Lesser| > k - 1)$
    Return $\text{DetQuickSelect}(Lesser, k)$
else
    Return $\text{DetQuickSelect}(Greater, k - |Lesser| - 1)$

$\text{ChoosePivot}(A)$

Initialize an empty array C of size $|A|/5$; $i = 1$
Split A into $|A|/5$ groups as $B_1, \ldots B_g$ where $g = |A|/5$.
for $j = 1 \cdots g$
    $q = Median(B_j)$
    $C[i] = q$; $i = i + 1$
$p = \text{DetQuickSelect}(C, |C|/2)$
Return p

Claim: The worst case running time of DetQuickSelect() is O(n).

Analyze

Let T(n) be the running time of DetQuickSelect()

Step1: Takes O(n)
Step2: Takes T(n/5)
Step3: Takes O(n)
Step4: Takes O(1)
Max of Step5 and Step6 is T(7n/10).

DetQuickSelect(*A*,*k*)

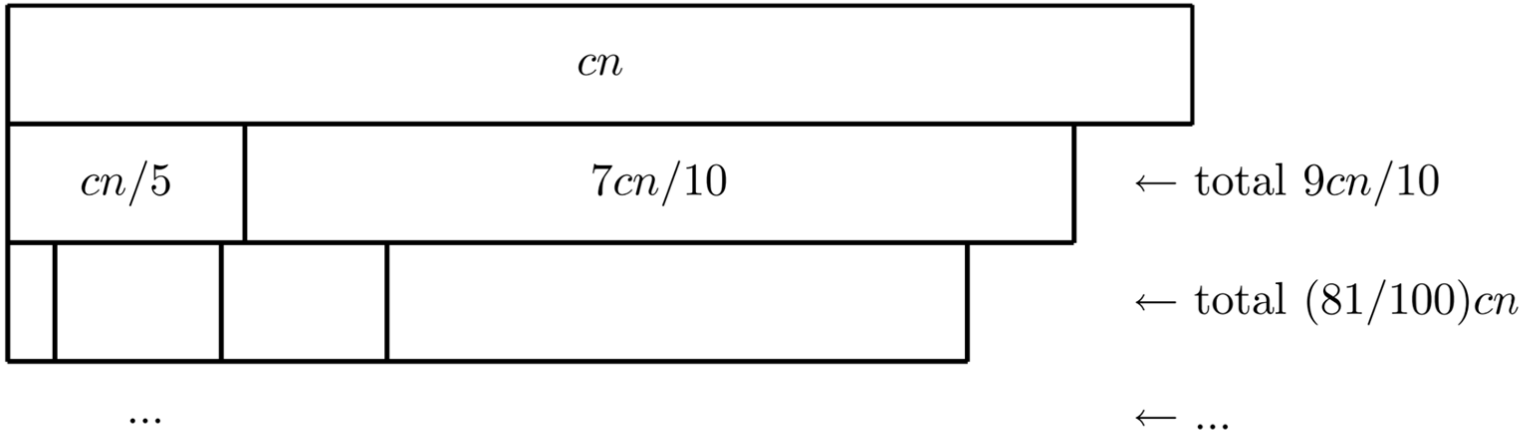
1. Group *A* into n/5 groups, each of size 5. Find median of each group.
2. Recursively find median of the medians. Lets call it p.
3. Split A into subarrays *Lesser* & *Greater*. Set *L* = |*Lesser*|
4. If *L* = *k* - 1 then return *p*.
5. If *L* > *k* - 1 then return DetQuickSelect(*Lesser*, *k*).
6. If *L* < *k* - 1 then return DetQuickSelect(*Greater*, *k* - *L* - 1)

Recurrence: $T(n) \quad \leq \quad cn + T(n/5) + T(7n/10)$

Analyze

$$T(n) \quad \leq \quad cn + T(n/5) + T(7n/10)$$

Recurrence Tree



$$cn(1 + (9/10) + (9/10)^2 + (9/10)^3 + \ldots)$$

Sum of gp when r < 1: a/(1-r).

Outline

- Select k-Smallest Element
- Fast Fourier transform

Fast Fourier Transform

Global Positioning System (GPS)

Television network

Wireless Communications

<u>Signal Processing</u>

- Spectrum analysis: Determine frequency content of a signal.
- Filtering: Remove unwanted frequency component.
- Compression: Lossless data compression.
- Convolution: Mathematical operation used to combine two signals.

Fast Fourier Transform

Discrete Fourier Transform

Time domain $\Leftrightarrow$ Frequency domain

and more...

Polynomial Multiplication (Convolution)

Given, two polynomials $P(x)$ and $Q(x)$ compute $R(x) = P(x) \cdot Q(x)$.

Let, $P(x) = 1 + 2x^2$ and $Q(x) = 2x + x^2$ then

$R(x) = 2x + x^2 + 4x^3 + 2x^4$.

PolyMult(P, Q):
Compute R(x) = P(x)·Q(x).

R = [0, 2, 1, 4, 2]

Coefficient Value Representation.

Polynomial Multiplication (Convolution)

PolyMult(P, Q):
Compute $R(x) = P(x) \cdot Q(x)$.

If $P(x)$ and $Q(x)$ are degree d polynomials then PolyMult(P, Q) will take $O(d^2)$ time.

Polynomial Multiplication (Convolution)

How many points do we need to represent a polynomial of degree 1?

Degree 1 polynomial is a line, so two points enough.

$P(x) = p_0 + p_1 \cdot x$.

If (3, 0) & (0, 3) are two points then $P(x) = 3 - x$.

Point Value Representation

Claim: Any d degree polynomial can be uniquely represented by d + 1 points.

Proof

Claim: Any d degree polynomial can be uniquely represented by d + 1 points.

$$\{(x_0, P(x_0)), (x_1, P(x_1)), \ldots, (x_d, P(x_d))\}$$

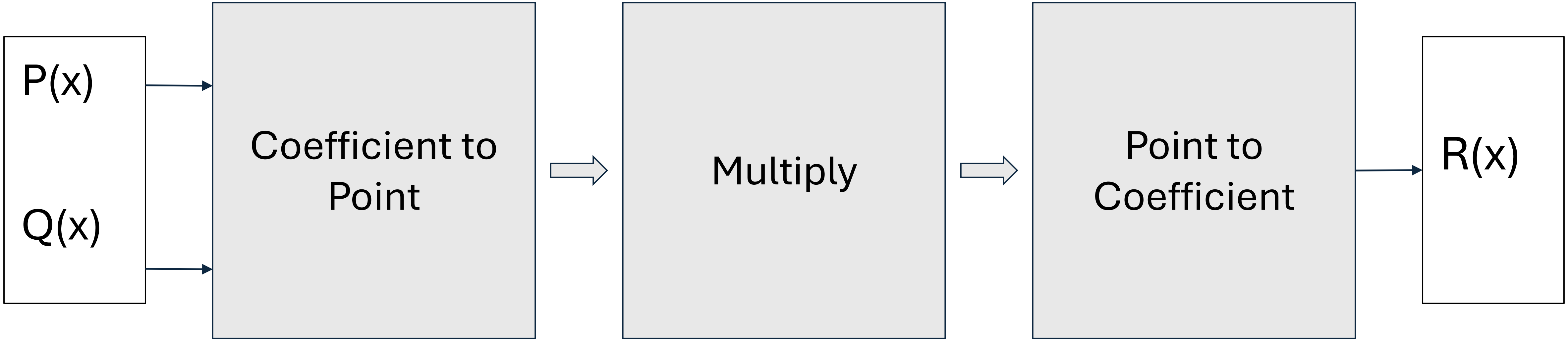
$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d$$

$$\begin{bmatrix} P(x_0) \\ P(x_1) \\ \vdots \\ P(x_d) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^d \\ 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \cdots & x_d^d \end{bmatrix}}_{M} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{bmatrix}$$

- M is invertible for unique $x_0, \ldots, x_d$
- So, unique $p_0, \ldots, p_d$, i.e., $p = M^{-1} \cdot y$
- So, unique polynomial.

Polynomial Multiplication (Convolution)

Given point value representation of P(x) and Q(x) of degree d. Propose an efficient algorithm (assume $M^{-1} \cdot y$ takes (O(1) time).



Fast Fourier Transform

Fast Fourier Transform

Consider the polynomi$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d$

Compute n ≥ d+1 pairs of (point, value) for P(x).

Coefficient to Point

$$\begin{matrix} (1, P(1)) \\ (2, P(2)) \\ \vdots \\ (n, P(n)) \end{matrix} \longrightarrow \begin{matrix} (1, p_0 + p_1 \cdot 1 + p_2 \cdot 1^2 + \cdots + p_d \cdot 1^d) \\ (2, p_0 + p_1 \cdot 2 + p_2 \cdot 2^2 + \cdots + p_d \cdot 2^d)) \\ \vdots \\ (n, p_0 + p_1 \cdot n + p_2 \cdot n^2 + \cdots + p_d \cdot n^d)) \end{matrix}$$

P(x) at unique n points.

Running time: O(nd) ~ O(d^2) !!!

Lets improvise.

Fast Fourier Transform

Let $P(x) = x^2$ and $n = 4$.

$(1,1)$ & $(-1,1)$

$(2,4)$ & $(-2,4)$

$P(-x) = P(x)$

Let $P(x) = x^3$ and $n = 4$.

$(1,1)$ & $(-1,-1)$

$(2,8)$ & $(-2,-8)$

$P(-x) = -P(x)$

Fast Fourier Transform

General function $P(x) = 3x^5 + 2x^4 + x^3 + 7x^2 + 5x + 1$

Calculate P(x) at $\pm x_1, \pm x_2, \ldots, \pm x_{n/2}$

$$P(x) = (2x^4 + 7x^2 + 1) + (3x^5 + x^3 + 5x)$$

$$P(x) = P_e(x^2) + xP_o(x^2)$$

$$P(x_i) = P_e(x_i^2) + x_i P_o(x_i^2)$$

$$P(-x_i) = P_e(x_i^2) - x_i P_o(x_i^2)$$

$$P(x) = \underbrace{(2x^4 + 7x^2 + 1)}_{P_e(x^2)} + x\underbrace{(3x^4 + x^2 + 5)}_{P_o(x^2)}$$

<u>Observations</u>

- $P_e(x^2)$ and $P_o(x^2)$ are of degree 2, even though P(x) was a 5 degree polynomial.
- Calculate $P_e(x^2)$ and $P_o(x^2)$ $x_1^2, x_2^2, \ldots, x_{n/2}^2$ points.

Fast Fourier Transform

$P(x) : [p_0, p_1, \ldots, p_{n-1}]$
$[\pm x_1, \pm x_2, \ldots, \pm x_{n/2}]$

$$P(x) = P_e(x^2) + xP_o(x^2)$$

$P_e(x^2) : [p_0, p_2, \ldots, p_{n-2}]$
$[x_1^2, x_2^2, \ldots, x_{n/2}^2]$

$P_o(x^2) : [p_1, p_3, \ldots, p_{n-1}]$
$[x_1^2, x_2^2, \ldots, x_{n/2}^2]$

$$P(x_i) = P_e(x_i^2) + x_i P_o(x_i^2)$$
$$P(-x_i) = P_e(x_i^2) - x_i P_o(x_i^2)$$
$$i = \{1, 2, \ldots, n/2\}$$

<u>Observations</u>

- Running time O(n·log(n))
- $[\pm x_1, \pm x_2, \ldots, \pm x_{n/2}]$ ls to be paired.
- So, $[x_1^2, x_2^2, \ldots, x_{n/2}^2]$. For next iteration these needs to paired.
- But these are not +ve/-ve paired.
- Expand the domain of initial points, i.e.,

  $[\pm x_1, \pm x_2, \ldots, \pm x_{n/2}]$ complex numbers.

Fast Fourier Transform

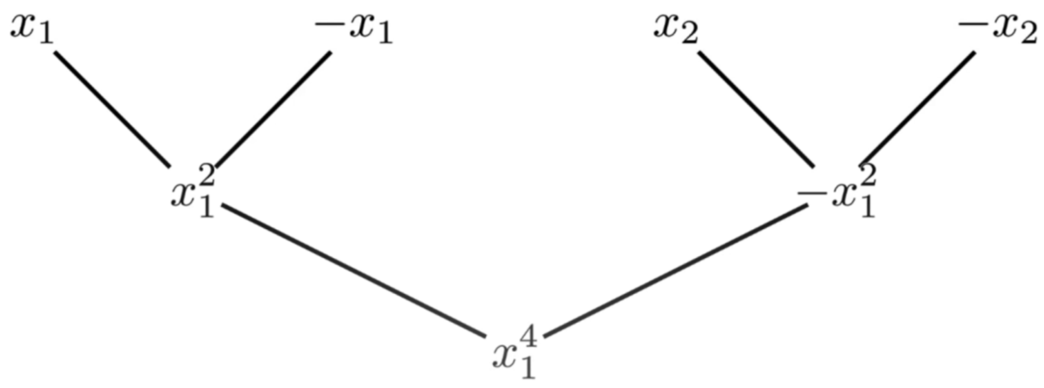
Let $P(x) = x^3 - x^2 + x - 1$ & $n = 4$.

Fast Fourier Transform

Let P(x) = $x^3 - x^2 + x - 1$ & n = 4.

Fast Fourier Transform

Let P(x) = x^3 - x^2 + x - 1 & n = 4.

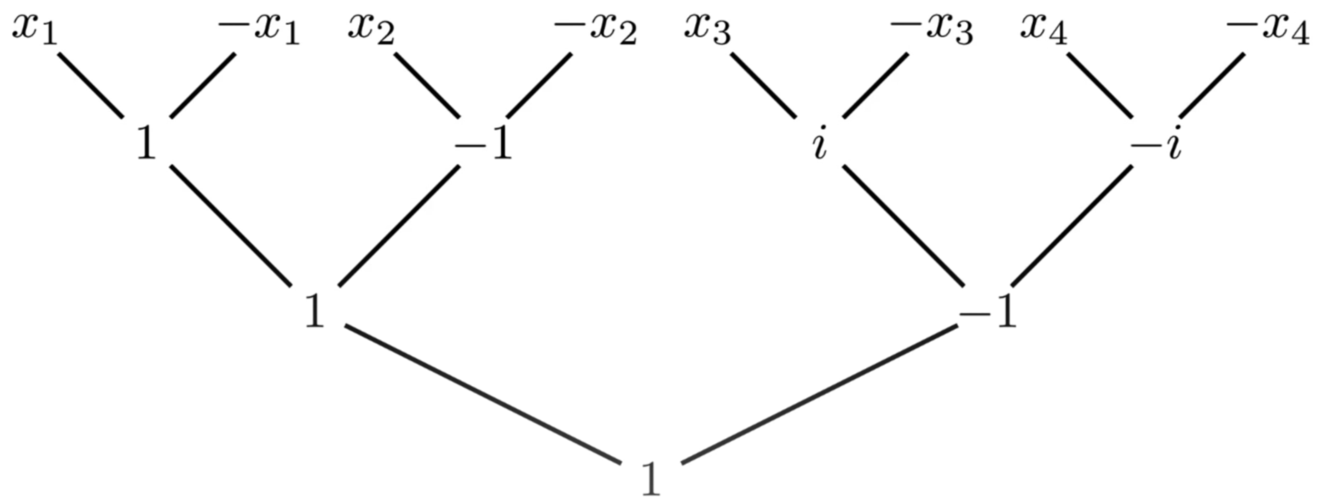


Let x_1 = 1.

So, our initial points are solution to x^4 = 1.

Fast Fourier Transform

Let $P(x) = x^6 - x^5 + x^4 - x^3 + x^2 - x + 1$ & $n = 8$ (>7).



So, our initial points are solution to $x^8 = 1$.

Reference

Slides

Algorithms Design by Kleinberg & Tardos - Chp 5.3 & 5.6

Jeff Erickson's Lecture notes - 1.8