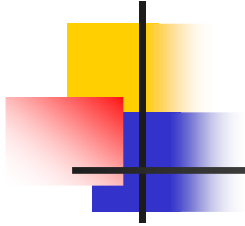




# Algorithms 2005

---

Ramesh Hariharan



# Polynomial Computation



# Multiplying Univariate Polynomials

---

- Of degrees  $n$  and  $m$  gives a new polynomial of degree  $n+m$
- Time taken is  $O(nm)$
- Can one do better?



# Multiplying 2 Polynomials

---

- Consider  $n=m$  first
- Evaluate each polynomial at  $2n+1$  distinct values
- Compute products of  $2n+1$  evaluations
- Interpolate between the resulting  $2n+1$  values to obtain the product polynomial (prove that it is possible to uniquely recover the polynomial from its evaluations at  $2n+1$  distinct values)



# Multi-location Polynomial Evaluation

---

- Consider evaluating a polynomial of degree  $n$  at  $k=2n+1$  distinct locations
- Each evaluation takes  $O(n)$ ; Total  $O(n^2)$ ; Can one do better?
- Also. an evaluation could produce a huge number, how do we handle these numbers?



# Multi-location Polynomial Evaluation

---

- Consider evaluating a polynomial  $P(x)$  of degree  $2n+1$  at distinct locations  $a_1..a_k$ ,  $k=2n+1$
- Same as computing  $P(x) \bmod (x-a_1), \dots, P(x) \bmod (x-a_k)$   
Show this.

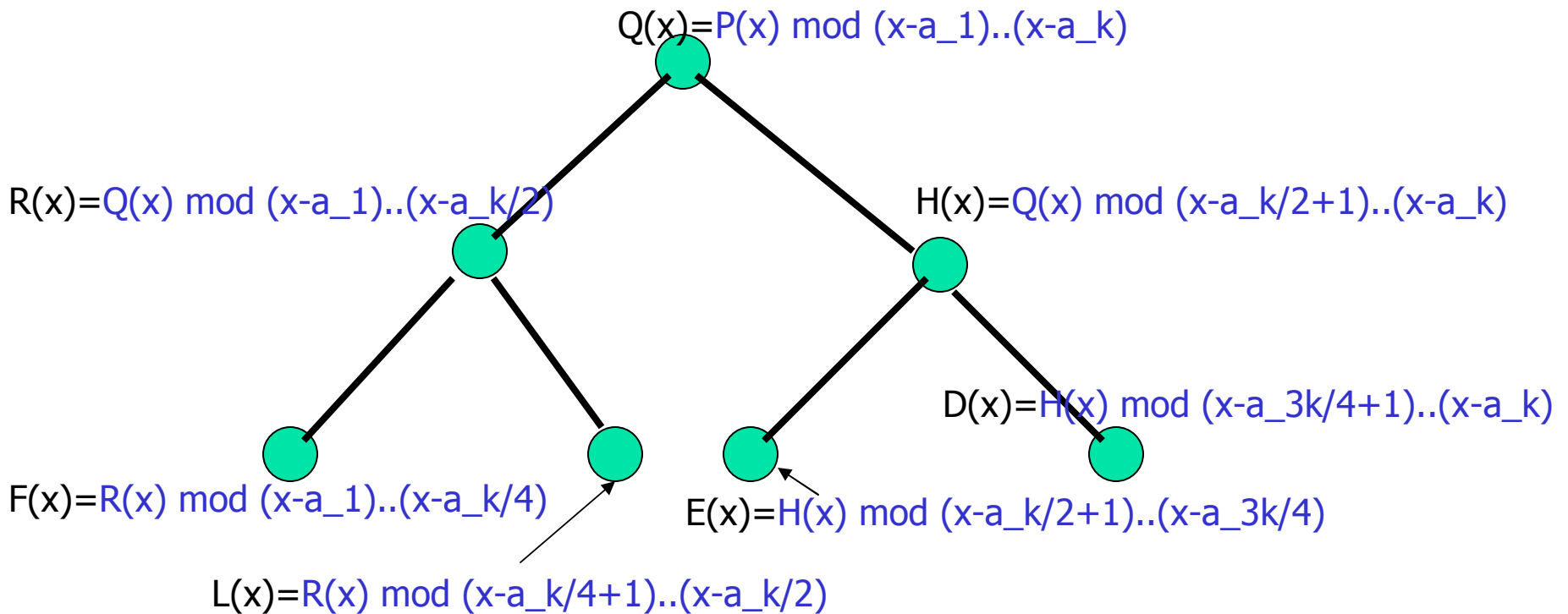
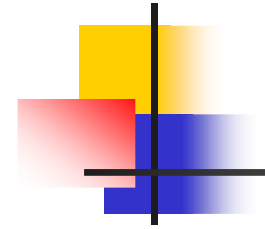


# Multi-location Polynomial Evaluation

---

- Computing  $P(x) \bmod (x-a_1)$  and  $P(x) \bmod (x-a_2)$  involves 2 mod computations on a high degree polynomial.
- Can this be reduced to one high degree mod computation (and possibly several low degree mod computations if required?)
- Compute  $Q(x) = P(x) \bmod (x-a_1)(x-a_2)$   
Compute  $Q(x) \bmod (x-a_1)$  and  $Q(x) \bmod (x-a_2)$

# Multi-location Polynomial Evaluation





# Multi-location Polynomial Evaluation

---

## Time taken

- At each node at level  $i$ , find the remainder of a degree  $k/2^{(i-1)}$  with respect to a polynomial of degree  $k/2^i$
- Time is  $O(k^2/2^{2(i-1)})$  at each level  $i$  node. There are  $2^i$  nodes at level  $i$ .
- Total time taken is still quadratic.
- What happens if we could do each level  $i$  node in time  $O(k/2^{(i-1)})$ ? Total time then is  $O(n \log n)$ .
- Can this be achieved if the remaindering polynomials has only a constant number of terms each?



# Multi-location Polynomial Evaluation

---

- How do we ensure that the remaindering polynomials have few terms

$$(x-a_1)(x-a_2)=x^2-(a_1+a_2)x+a_1a_2$$

If  $a_1+a_2=0$  then the result has only 2 terms



# Multi-location Polynomial Evaluation

- How do we ensure that the remaindering polynomials have few terms

Iterating this principle up the tree gives the following constraints

$$\begin{aligned} \blacksquare a_1 + a_2 &= 0 \\ \blacksquare a_3 + a_4 &= 0 \\ \blacksquare a_5 + a_6 &= 0 \\ \blacksquare a_7 + a_8 &= 0 \end{aligned}$$

$$\begin{aligned} \blacksquare a_1 a_2 + a_3 a_4 &= 0 \\ \blacksquare a_5 a_6 + a_7 a_8 &= 0 \end{aligned}$$

$$a_1 a_2 a_3 a_4 + a_5 a_6 a_7 a_8 = 0$$

# Multi-location Polynomial Evaluation

- Solving the above constraints gives

a	-a	b	-b	c	-c	d	-d	e	-e	f	-f	g	-g	h	-h
a	-a	ia	-ia	c	-c	ic	-ic	e	-e	ie	-ie	g	-g	ig	-ig
a	-a	ia	-ia	ja	-ja	ija	-ija	e	-e	ie	-ie	je	-je	ije	-ije
a	-a	ia	-ia	ja	-ja	ija	-ija	ha	-ha	iha	-iha	jha	-jha	ijha	-ijha

$$i^2 = -1 \quad j^4 = -1 \quad h^8 = -1$$

- a is unconstrained, so setting  $a=1$  gives
- 1 -1 i -i j -j ij -ij h -h ih -ih jha -jh ijh -ijh



# Multi-location Polynomial Evaluation

---

- Sequence for general  $k=2^*$ 
  - $S(k)=S(k/2) \cdot \text{Primitive-}k\text{th-root-of-unity} \cdot S(k/2)$
  - $S(1)=1$
- We need the  $k$ th,  $k/2$ th,  $k/4$ th .. roots of unity
- A primitive  $k$ th root of unity  $x$  satisfies  $x^k=1$  but  $x^{\{k/2\}} \neq 1$



# Multi-location Polynomial Evaluation

---

- $r$ th root of unity is  $\text{Cos}(2\pi/r) + i \text{Sin}(2\pi/r)$
- This is irrational, so we can only do approximate computation
- To what precision should we compute?
- How do these approximations add up?



# Multi-location Polynomial Evaluation

---

## Precision Questions

- To get a certain accuracy at the output, what input precision must we start with?
- How much does each operation amplify the error?

# Multi-location Polynomial Evaluation

## A Careful Look at Operations

- Each remaindering polynomial is of the form  $x^i - a^2$ ; it is obtained by multiplying  $(x^{\{i/2\}} - a)(x^{\{i/2\}} + a)$
- What is the error in computing  $a^2$ ?
- If the error in  $a$  is  $\epsilon$ , then the error in  $a^2$  is
$$|a^2 - (a + \epsilon)^2| = \epsilon |2a + \epsilon| < 3\epsilon$$
assuming  $|a| \leq 1$  and  $\epsilon \leq 1$  (which will be true, why?)
- The total error in constant term of the remaindering polynomial trebles at each level reaching a max of  $3^{\log k} \epsilon = k^{O(1)} \epsilon$  at the top, where  $\epsilon$  is the starting error at the bottom. By choosing the starting error at the bottom to be  $1/\text{poly}(k)$ , we can keep the error at the top  $1/\text{poly}(k)$  as well.



# Multi-location Polynomial Evaluation

---

## A Careful Look at Operations

- It remains to consider to the mod operations  $Q(x) \bmod x^i - a$
- Note  $Q(x)$  has degree less than  $2i$  (this is important)
- Each step in the mod computation performs  $c - b a$ , where  $c$  and  $b$  are coefficients of  $Q(x)$  (to show this for  $b$ , we need that  $\deg(Q(x)) < 2i$ )
- Assume  $b = O(\text{poly}(k))$  (justify this assumption and show that it holds for intermediate polynomials  $Q(x)$  provided it holds for the original polynomial)

# Multi-location Polynomial Evaluation

## A Careful Look at Operations

- If the coefficients of  $Q(x)$  have error  $\delta$  and  $a$  has error  $\gamma$  then what is the error in the result.

$$|c + \delta - (b + \delta)(a + \gamma) - (c - ab)| \leq \delta + a\delta + b\gamma + \delta\gamma \\ \leq 3\delta + b\gamma$$

since  $|a| \leq 1$  and  $\delta < 1$  and  $\gamma \leq 1$  (which will be true, why?)

- We can choose  $\epsilon$  on the previous slide so that  $\gamma$  is a small enough inverse polynomial so  $b\gamma < \delta$ , in which case the error goes from  $\delta$  to  $4\delta$  in one mod operation. This translates to a  $4^{\log k} \delta$  error at the bottom; choosing  $\delta$  to be at least  $b\gamma$  which is  $1/\text{poly}(k)$  we can bound this error to  $1/\text{poly}(k)$  as well.



# Multi-location Polynomial Evaluation

---

## Completing Precision Details

- How do we compute  $\text{Cos}(2\pi/r)$  and  $\text{Sin}(2\pi/r)$  to  $1/\text{poly}(k)$  precision? How much time does this take?  
Exercise.
- How many bits does it take?  $O(\log k)$  bits; so arithmetic operations on these bits can be performed in  $O(1)$  time.



# Multi-location Polynomial Evaluation

---

## Total Time Taken

- $O(k \log k)$ , assuming  $O(1)$  time for arithmetic operations on  $O(\log k)$  bit words; also assuming roots of unity can be computed to the desired accuracy; also assuming the original coefficients are sub poly( $k$ ).
- Output accuracy is  $1/\text{poly}(k)$



# Multi-location Polynomial Evaluation

---

## Interpolating to Recover the Product

- An almost identical procedure
- Find 2 polynomials  $A(x)$  and  $B(x)$  recursively which interpolate on  $a_1..a_{k/2}$  and  $a_{k/2+1}..a_k$  respectively
- Obtain the full polynomial by computing  $(x-a_1)..(x-a_{k/2}) B(x) + (x-a_{k/2+1})..(x-a_k) A(x)$



# String Matching with WildCards

---

How can Polynomial Multiplication be used?

- Given a text with 0s and 1s and a pattern with 0s, 1s, \*, find all occurrences of the pattern in the text.
- The pattern matches at a location if all 0s in the pattern are aligned to 0s, all 1s are aligned to 1s and \*'s could be aligned to 0s or 1s.
- For text length  $n$  and pattern length  $m$ , time taken is  $O(n \log m)$