COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

Divide and Conquer

Problem

Median Finding: Given an array of unsorted numbers and an integer k, design an algorithm that finds the k^{th} smallest number in the array. Assume that A contains distinct numbers.

Problem

Median Finding: Given an array of unsorted numbers and an integer k, design an algorithm that finds the k^{th} smallest number in the array. Assume that A contains distinct numbers.

Algorithm

`Kth-smallest-incomplete`(A, k)

- Pick a number p as *pivot*
- Partition the numbers in A into A_L (all numbers $< p$) and A_R (all numbers $> p$)
- If ($|A_L| = k - 1$), then return(p)
- If ($|A_L| > k - 1$), then return(`Kth-smallest-incomplete`(A_L, k))
- If ($|A_L| < k - 1$), then return(`Kth-smallest-incomplete`$(A_R, k - |A_L| - 1)$)

Problem

Median Finding: Given an array of unsorted numbers and an integer k, design an algorithm that finds the k^{th} smallest number in the array. Assume that A contains distinct numbers.

Algorithm

`Kth-smallest-incomplete`(A, k)

- Pick a number p as *pivot*
- Partition the numbers in A into A_L (all numbers $< p$) and A_R (all numbers $> p$)
- If $(|A_L| = k - 1)$, then return(p)
- If $(|A_L| > k - 1)$, then return(`Kth-smallest-incomplete`(A_L, k))
- If $(|A_L| < k - 1)$, then return(`Kth-smallest-incomplete`$(A_R, k - |A_L| - 1)$)

- What is the running time of this algorithm if the pivot is picked arbitrarily?

Problem

Median Finding: Given an array of unsorted numbers and an integer k, design an algorithm that finds the k^{th} smallest number in the array. Assume that A contains distinct numbers.

Algorithm

`Kth-smallest-incomplete`(A, k)
- Pick a number p as *pivot*
- Partition the numbers in A into A_L (all numbers $< p$) and A_R (all numbers $> p$)
- If $(|A_L| = k - 1)$, then return(p)
- If $(|A_L| > k - 1)$, then return(`Kth-smallest-incomplete`(A_L, k))
- If $(|A_L| < k - 1)$, then return(`Kth-smallest-incomplete`$(A_R, k - |A_L| - 1)$)

- What is the running time of this algorithm if the pivot is picked arbitrarily? $O(n^2)$

Algorithm

`Kth-smallest-incomplete`(A, k)

- Pick a number p as *pivot*
- Partition the numbers in A into A_L (all numbers $< p$) and A_R (all numbers $> p$)
- If $(|A_L| = k - 1)$, then return(p)
- If $(|A_L| > k - 1)$, then return(`Kth-smallest-incomplete`(A_L, k))
- If $(|A_L| < k - 1)$, then return(`Kth-smallest-incomplete`$(A_R, k - |A_L| - 1)$)

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

40	13	70	62	30
19	64	67	24	94
49	83	77	3	78
50	4	46	49	6
85	25	29	11	60

- Consider groups of 5 elements.

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements.
- Sort Individual groups.

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements
- Sort Individual groups
- Consider the median of the medians:
  - Here it is 46.
- Use this as the pivot element.

Divide and Conquer
Median finding

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements
- Sort Individual groups
- Consider the median of the medians:
  - Here it is 46.
- Use this as the pivot element p.

- How many elements in A are larger than p?

Divide and Conquer
Median finding

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements
- Sort Individual groups
- Consider the median of the medians:
  - Here it is 46.
- Use this as the pivot element p.

- How many elements in A are larger than p?
  - Claim 1: There are at least $(3n/10 - 6)$ numbers in A that are larger than p.

Divide and Conquer
Median finding

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements
- Sort Individual groups
- Consider the median of the medians:
  - Here it is 46.
- Use this as the pivot element p.

- How many elements in A are larger than p? at least $(3n/10 - 6)$
- How many elements in A are smaller than p?
  - Claim 2: There are at least $(3n/10 - 6)$ numbers in A that are smaller than p.

Divide and Conquer
Median finding

- How do we pick a good pivot number?
  - Randomly: We will look at this later.
  - Deterministically:

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements
- Sort Individual groups
- Consider the median of the medians:
  - Here it is 46.
- Use this as the pivot element p.

- How many elements in A are larger than p? at least $(3n/10 - 6)$
- How many elements in A are smaller than p? at least $(3n/10 - 6)$

Algorithm

`Find-Kth-smallest`(A, k)

- ... *//Base cases*
- Consider groups of 5 numbers, sort each group and create another array B containing the median number from each group
- $p \leftarrow$ `Find-Kth-smallest`$(B, \lfloor \frac{|B|}{2} \rfloor)$
- Partition the array A into A_L and A_R using p as the pivot
- If $(|A_L| = k - 1)$, then return(p)
- If $(|A_L| > k - 1)$, then return(`Find-Kth-smallest`(A_L, k))
- If $(|A_L| < k - 1)$, then return(`Find-Kth-smallest`$(A_R, k - |A_L| - 1)$)

- What is the running time of the above algorithm?

Algorithm

`Find-Kth-smallest`(A, k)
- ... *//Base cases*
- Consider groups of 5 numbers, sort each group and create another array B containing the median number from each group
- $p \leftarrow$ `Find-Kth-smallest`$(B, \lfloor \frac{|B|}{2} \rfloor)$
- Partition the array A into A_L and A_R using p as the pivot
- If $(|A_L| = k - 1)$, then return(p)
- If $(|A_L| > k - 1)$, then return(`Find-Kth-smallest`(A_L, k))
- If $(|A_L| < k - 1)$, then return(`Find-Kth-smallest`$(A_R, k - |A_L| - 1)$)

- What is the running time of the above algorithm?
  - $T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$; $T(1) = O(1)$
  - What is $T(n)$?

Algorithm

`Find-Kth-smallest`(A, k)

- ... *//Base cases*
- Consider groups of 5 numbers, sort each group and create another array B containing the median number from each group
- $p \leftarrow$ `Find-Kth-smallest`$(B, \lfloor \frac{|B|}{2} \rfloor)$
- Partition the array A into A_L and A_R using p as the pivot
- If $(|A_L| = k - 1)$, then return(p)
- If $(|A_L| > k - 1)$, then return(`Find-Kth-smallest`(A_L, k))
- If $(|A_L| < k - 1)$, then return(`Find-Kth-smallest`$(A_R, k - |A_L| - 1)$)

- What is the running time of the above algorithm?
  - $T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$; $T(1) = O(1)$
  - What is $T(n)$? $O(n)$

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

40	13	70	62	30
19	64	67	24	94
49	83	77	3	78
50	4	46	49	6
85	25	29	11	60

- Consider groups of 5 elements.

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements.
- Sort individual groups

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements.
- Sort individual groups.
- Consider medians of each group.

Divide and Conquer
Median finding

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements.
- Sort individual groups.
- Consider medians of each group.
- Make a recursive call to find median element of medians.

40	64	77	46	29

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements.
- Sort individual groups.
- Consider medians of each group.
- Make a recursive call to find median element of medians.

40	64	77	46	29

29	40	46	64	77

Divide and Conquer
Median finding

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

13	30	40	62	70
19	24	64	67	94
3	49	77	78	83
4	6	46	49	50
11	25	29	60	85

- Consider groups of 5 elements.
- Sort individual groups.
- Consider medians of each group.
- Make a recursive call to find median element of medians.

46

40	64	77	46	29

29	40	46	64	77

Divide and Conquer
Median finding

- Suppose we want to find the 12^{th} smallest element in the following array.

40	13	70	62	30	19	64	67	24	94	47	83	77	3	78	50	4	46	49	6	85	25	29	11	60

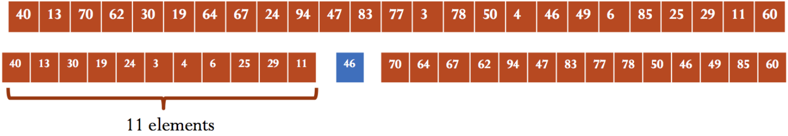
40	13	30	19	24	3	4	6	25	29	11

46

70	64	67	62	94	47	83	77	78	50	46	49	85	60

- Consider groups of 5 elements.
- Sort individual groups.
- Consider medians of each group.
- Make a recursive call to find median element of medians.
- Partition using the pivot as 46.

- Suppose we want to find the 12^{th} smallest element in the following array.



- Consider groups of 5 elements.
- Sort individual groups.
- Consider medians of each group.
- Make a recursive call to find median element of medians.
- Partition using the pivot as 46.

Problem

Given two polynomials:
$A(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a_{n-1} \cdot x^{n-1}$, and
$B(x) = b_0 + b_1 \cdot x + b_2 \cdot x^2 + ... + b_{n-1} \cdot x^{n-1}$, design an algorithm to that outputs $A(x) \cdot B(x)$.

Problem

Given two polynomials:
$A(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a_{n-1} \cdot x^{n-1}$, and
$B(x) = b_0 + b_1 \cdot x + b_2 \cdot x^2 + ... + b_{n-1} \cdot x^{n-1}$, design an algorithm to that outputs $A(x) \cdot B(x)$.

- We have to obtain the polynomial $C(x) = A(x) \cdot B(x)$
- $C(x)$ may be written as:
  $C(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + ... + c_{2n-2} \cdot x^{2n-2}$
- What is c_i in terms of coefficients of A and B?

Problem

Given two polynomials:
$A(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a_{n-1} \cdot x^{n-1}$, and
$B(x) = b_0 + b_1 \cdot x + b_2 \cdot x^2 + ... + b_{n-1} \cdot x^{n-1}$, design an algorithm to that outputs $A(x) \cdot B(x)$.

- We have to obtain the polynomial $C(x) = A(x) \cdot B(x)$
- $C(x)$ may be written as:
  $C(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + ... + c_{2n-2} \cdot x^{2n-2}$
- What is c_i in terms of coefficients of A and B?
  - $c_i = a_i \cdot b_0 + a_{i-1} \cdot b_1 + ... + a_0 \cdot b_i$
- The vector $(c_0, ..., c_{2n-2})$ is called the *convolution* of vectors $(a_0, ..., a_{n-1})$ and $(b_0, ..., b_{n-1})$.

Algorithm

$\texttt{SimpleMultiply}((a_0, ..., a_{n-1}), (b_0, ..., b_{n-1}))$
- For $i = 0$ to $2n - 2$
  - For $j = 0$ to i
    - $c_i \leftarrow c_i + a_j \cdot b_{i-j}$
- return$((c_0, c_1, ..., c_{2n-2}))$

- What is the running time of the above algorithm?

Algorithm

$\texttt{SimpleMultiply}((a_0, ..., a_{n-1}), (b_0, ..., b_{n-1}))$

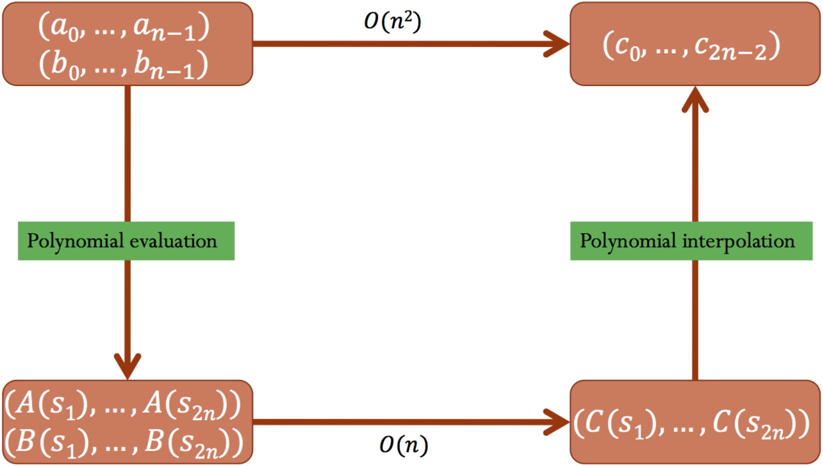
- For $i = 0$ to $2n - 2$
  - For $j = 0$ to i
    - $c_i \leftarrow c_i + a_j \cdot b_{i-j}$
- return$((c_0, c_1, ..., c_{2n-2}))$

- What is the running time of the above algorithm? $O(n^2)$
- Is there another way to compute the polynomial $C(x)$?

- Another way to compute the polynomial $C(x)$:
  - Compute $A(s_1), A(s_2), ..., A(s_{2n})$.
  - Compute $B(s_1), B(s_2), ..., B(s_{2n})$.
  - Compute:
    - $C(s_1) = A(s_1) \cdot B(s_1)$
    - $C(s_2) = A(s_2) \cdot B(s_2)$
    - $\vdots$
    - $C(s_{2n}) = A(s_{2n}) \cdot B(s_{2n})$
  - Interpolate to obtain the polynomial $C(x)$.

- Another way to compute the polynomial $C(x)$:
  - Compute $A(s_1), A(s_2), ..., A(s_{2n})$.
  - Compute $B(s_1), B(s_2), ..., B(s_{2n})$.
  - Compute:
    - $C(s_1) = A(s_1) \cdot B(s_1)$
    - $C(s_2) = A(s_2) \cdot B(s_2)$
    - $\vdots$
    - $C(s_{2n}) = A(s_{2n}) \cdot B(s_{2n})$
  - Interpolate to obtain the polynomial $C(x)$.
- How fast can you compute $A(s)$ given value of s?

- Another way to compute the polynomial $C(x)$:
  - Compute $A(s_1), A(s_2), ..., A(s_{2n})$.
  - Compute $B(s_1), B(s_2), ..., B(s_{2n})$.
  - Compute:
    - $C(s_1) = A(s_1) \cdot B(s_1)$
    - $C(s_2) = A(s_2) \cdot B(s_2)$
    - $\vdots$
    - $C(s_{2n}) = A(s_{2n}) \cdot B(s_{2n})$
  - Interpolate to obtain the polynomial $C(x)$.
- How fast can you compute $A(s)$ given value of s?
  - $O(n)$ arithmetic operations using *Horner's rule*.
  - $A(s) = a_0 + s \cdot (a_1 + s \cdot (a_2 + ... + s \cdot (a_{n-1}) ...))$


$(a_0, \ldots, a_{n-1})$
$(b_0, \ldots, b_{n-1})$
$O(n^2)$
$(c_0, \ldots, c_{2n-2})$
Polynomial evaluation
Polynomial interpolation
$(A(s_1), \ldots, A(s_{2n}))$
$(B(s_1), \ldots, B(s_{2n}))$
$O(n)$
$(C(s_1), \ldots, C(s_{2n}))$

- Polynomial interpolation: We have $C(s_1), ..., C(s_{2n})$ and we need to compute $(c_0, ..., c_{2n-2})$.

$$\begin{pmatrix} 1 & s_1 & (s_1)^2 & \dots & (s_1)^{2n-1} \\ 1 & s_2 & (s_2)^2 & \dots & (s_2)^{2n-1} \\ 1 & s_3 & (s_3)^2 & \dots & (s_3)^{2n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{2n} & (s_{2n})^2 & \dots & (s_{2n})^{2n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-1} \end{pmatrix} = \begin{pmatrix} C(s_1) \\ C(s_2) \\ C(s_3) \\ \vdots \\ C(s_{2n}) \end{pmatrix}$$

- Is the above square matrix *invertible*?

- Polynomial interpolation: We have $C(s_1), ..., C(s_{2n})$ and we need to compute $(c_0, ..., c_{2n-2})$.

$$\begin{pmatrix} 1 & s_1 & (s_1)^2 & \dots & (s_1)^{2n-1} \\ 1 & s_2 & (s_2)^2 & \dots & (s_2)^{2n-1} \\ 1 & s_3 & (s_3)^2 & \dots & (s_3)^{2n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{2n} & (s_{2n})^2 & \dots & (s_{2n})^{2n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-1} \end{pmatrix} = \begin{pmatrix} C(s_1) \\ C(s_2) \\ C(s_3) \\ \vdots \\ C(s_{2n}) \end{pmatrix}$$

- Is the above square matrix *invertible*?
- Fact: A square matrix is invertible iff its determinant is non-zero.

- Polynomial interpolation: We have $C(s_1), ..., C(s_{2n})$ and we need to compute $(c_0, ..., c_{2n-2})$.

$$\begin{pmatrix} 1 & s_1 & (s_1)^2 & \dots & (s_1)^{2n-1} \\ 1 & s_2 & (s_2)^2 & \dots & (s_2)^{2n-1} \\ 1 & s_3 & (s_3)^2 & \dots & (s_3)^{2n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{2n} & (s_{2n})^2 & \dots & (s_{2n})^{2n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-1} \end{pmatrix} = \begin{pmatrix} C(s_1) \\ C(s_2) \\ C(s_3) \\ \vdots \\ C(s_{2n}) \end{pmatrix}$$

- Is the above square matrix *invertible*?
- Fact: A square matrix is invertible iff its determinant is non-zero.
- The square matrix above has a special name: *Vandermonde* matrix.

- Fact: A square matrix is invertible iff its determinant is non-zero.
- The square matrix above has a special name: *Vandermonde* matrix.
- Claim 1: For any Vandermonde matrix V shown below,

$$V = \begin{pmatrix} 1 & s_1 & (s_1)^2 & \dots & (s_1)^{2n-1} \\ 1 & s_2 & (s_2)^2 & \dots & (s_2)^{2n-1} \\ 1 & s_3 & (s_3)^2 & \dots & (s_3)^{2n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{2n} & (s_{2n})^2 & \dots & (s_{2n})^{2n-1} \end{pmatrix}$$

$Det(V) = \prod_{1 \leq j < i \leq 2n}(s_i - s_j)$.

- Fact: A square matrix is invertible iff its determinant is non-zero.
- The square matrix above has a special name: *Vandermonde* matrix.
- Claim 1: For any Vandermonde matrix V shown below,

$$V = \begin{pmatrix} 1 & s_1 & (s_1)^2 & \dots & (s_1)^{2n-1} \\ 1 & s_2 & (s_2)^2 & \dots & (s_2)^{2n-1} \\ 1 & s_3 & (s_3)^2 & \dots & (s_3)^{2n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s_{2n} & (s_{2n})^2 & \dots & (s_{2n})^{2n-1} \end{pmatrix}$$

  $Det(V) = \prod_{1 \leq j < i \leq 2n} (s_i - s_j)$.
- So, as long as we use distict values of $s_1, ..., s_{2n}$, we will be able to do polynomial interpolation.

End