FindMax & FindMode: Query Versions

Giri Narasimhan

Programming Team

Fall 2024

Problem Solving

- **Correct** Solutions
- **Efficient** Solutions

FindMax: Basic Problem

- **Input**: Array A of size n
- **Output**: Find largest value in A
- Simple iterative solution
- **Time Complexity**?
  - $O(n)$ **$n - 1$ comparisons**
- **Implications**:
  - Time to solve the problem grows linearly with size
  - Doubling the size should double the time
- Is this **optimal**, i.e., the best we can do?

FindMax: Query Version

FindMax-Basic

- **Input**: Array A of size n
- **Output**: Find largest value in A
- Naïve Iterative solution
- **Time Complexity**?
  - Naïve solution: $O(n)$

FindMax-Query

- **Input**: Array A of size n
- **Query**: $1 \leq i \leq j \leq n$
  - # queries, k, may be large
- **Output**: Find largest value in $A[i..j]$
- **Time Complexity**?
  - Naïve solution: $O(kn)$
- Is this **optimal**, i.e., the best we can do?

Idea: Preprocess the input

FindMax-Query

- **Input**: Array A of size n
- **Query**: $1 \leq i \leq j \leq n$
  - # queries, k, may be large
- **Output**: Find largest value in $A[i..j]$
- **Time Complexity**?
  - Naïve solution: $O(kn)$
- Is this **optimal**, i.e., the best we can do?

Preprocessing

- **Idea**: Remember past queries
  - **Bad idea**: k may be too large
- **Idea**: Preprocess the input A before looking at queries
  - Makes sense since A is fixed
  - But how? What to store?

Idea: Preprocess the input … 2

FindMax-Query

- **Input**: Array A of size n
- **Query**: $1 \leq i \leq j \leq n$
  - # queries, k, may be large
- **Output**: Find largest value in $A[i, j]$
- **Time Complexity**?
  - Naïve solution: $O(kn)$
- Is this **optimal**, i.e., the best we can do?

Preprocessing

- **Idea**: Remember past queries
  - **Bad idea**: k may be too large
- **Idea**: Preprocess the input A before looking at queries
  - Makes sense since A is fixed
  - But how? What to store?
- **New Idea**: Store answers for every possible query

Idea: Preprocess the input … 3

FindMax-Query

- **Input**: Array A of size n
- **Query**: $1 \leq i \leq j \leq n$
  - # queries, k, may be large
- **Output**: Find largest value in $A[i..j]$
- **Time Complexity**?
  - Naïve solution: $O(kn)$
- Is this **optimal**, i.e., the best we can do?

Preprocessing

- **New Idea**: Store answers for every possible query
  - How?
  - Need a data structure B
  - Use a 2D array
  - $B[i,j]$ Stores the answer for query $(i..j)$

Example

Input array A

1	2	3	4	5	6	7	8	9
14	41	19	2	23	11	29	6	31

Preprocessed 2D array B

1	2	3	4	5	6	7	8	9
14	41	41	41	41	41	41	41	41
	41	41	41	41	41	41	41	41
		19	19	23	23	29	29	31
			2	23	23	29	29	31
				23	23	29	29	31
					11	29	29	31
						29	29	31
							6	31
								31

Preprocessing Algorithm

How to fill in 2D array B

For $p = 1$ to n do

  For $q = 1$ to n do

    Compute $B[p, q]$

Time Complexity

- How many entries in B?
  - $O(n^2)$
- How to "Compute $B[p, q]$"?
  - Naïve: $O(q - p) = O(n)$
  - Naïve: $O(n^3)$
- Time for k queries:
  - $O(k + n^3)$
  - Fine if $n^3 = O(k)$
  - What if $n^2 = O(k)$, but $k < n^3$
  - Need better preprocessing

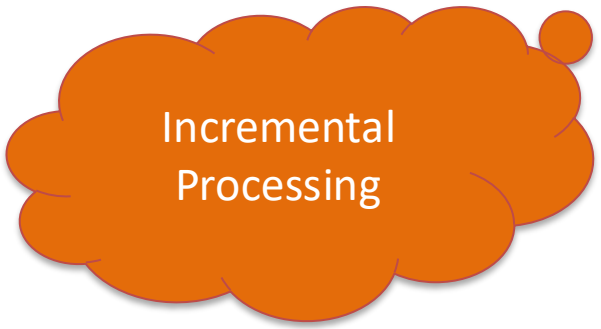
Preprocessing Algorithm

How to fill in 2D array B

For $p = 1$ to n do

  For $q = 1$ to n do

    Compute $B[p, q]$



Improved Preprocessing

- How to "Compute $B[p, q]$"
  - Use: $B[p, q-1]$
  - How?
- $B[p, q] = \max\{B[p, q-1], A[q]\}$
- Time to "Compute $B[p, q]$" $= O(1)$
- Time for k queries:
  - $O(k + n^2)$
  - Fine if $n^2 = O(k)$
  - What if $n = O(k)$, but $k < n^2$
  - Need better preprocessing

General Approach for

- Preprocess input A
- Create data structure B
- For each input query:
  - Ask an appropriate query from B
  - Respond quickly with an answer to input query
- Complexity? Preprocessing & Query time/space
- Tradeoff
  - Preprocessing time and space vs Query time