Tutorial #3

CS2800: Design and Analysis of Algorithms

Yadu Vasudev
yadu@cse.iitm.ac.in
IIT Madras

Problem 1: Searching in 2D-array

Problem: Search for an element x in an $n \times n$ array A whose rows and columns are sorted in the increasing order

Problem 1: Searching in 2D-array

Problem: Search for an element x in an $n \times n$ array A whose rows and columns are sorted in the increasing order

Method 1: Linear search on the array

- Time complexity: $O(n^2)$

Problem 1: Searching in 2D-array

Problem: Search for an element x in an $n \times n$ array A whose rows and columns are sorted in the increasing order

Method 1: Linear search on the array

- Time complexity: $O(n^2)$

Method 2: Binary search one row at a time

- Time complexity: $O(n \log n)$

Problem 1: Searching in 2D-array

Problem: Search for an element x in an $n \times n$ array A whose rows and columns are sorted in the increasing order

Method 1: Linear search on the array

- Time complexity: $O(n^2)$

Method 2: Binary search one row at a time

- Time complexity: $O(n \log n)$

Method 3: Compare $A[n, 1]$ and x

- If $x = A[n, 1]$, we have found the element
- If $x < A[n, 1]$, then $x < A[n, i]$ for all i and we can discard the last row
- If $x > A[n, 1]$, then $x > A[i, 1]$ for all i and we can discard the first column

Observation: After each query, we have reduced our search space by one row or one column. So total running time is $O(n)$

Problem 2: Inversions

Problem: An inversion in an array A is a pair (i, j) such that $i < j$ and $A[i] > A[j]$

Problem 2: Inversions

Problem: An inversion in an array A is a pair (i, j) such that $i < j$ and $A[i] > A[j]$

- **Exercise**: Show that the number of swaps performed by Bubble-sort is equal to the number of inversions

Problem 2: Inversions

Problem: An inversion in an array A is a pair (i, j) such that $i < j$ and $A[i] > A[j]$

- **Exercise**: Show that the number of swaps performed by Bubble-sort is equal to the number of inversions
- **A simple algorithm**: For each element, count all the inversions it is part of - running time of $O(n^2)$

Problem 2: Inversions

Problem: An inversion in an array A is a pair (i, j) such that $i < j$ and $A[i] > A[j]$

- $A[1] \leq A[2] \leq A[3] \leq \ldots \leq A[n/2]$
- $A[n/2 + 1] \leq A[n/2 + 2] \leq \ldots \leq A[n]$

Problem 2: Inversions

Problem: An inversion in an array A is a pair (i, j) such that $i < j$ and $A[i] > A[j]$

- $A[1] \leq A[2] \leq A[3] \leq \ldots \leq A[n/2]$
- $A[n/2 + 1] \leq A[n/2 + 2] \leq \ldots \leq A[n]$

Claim

- If $A[i] > A[n/2 + j]$, then all the pairs $(k, n/2 + j)$ such that $i \leq k \leq n/2$ are inversions
- If $A[i] < A[n/2 + j]$, then there are no inversions of the form (i, k) for $n/2 + j \leq k \leq n$

Problem 2: Inversions

Problem: An inversion in an array A is a pair (i, j) such that $i < j$ and $A[i] > A[j]$

- $A[1] \leq A[2] \leq A[3] \leq \ldots \leq A[n/2]$
- $A[n/2 + 1] \leq A[n/2 + 2] \leq \ldots \leq A[n]$

Claim

- If $A[i] > A[n/2 + j]$, then all the pairs $(k, n/2 + j)$ such that $i \leq k \leq n/2$ are inversions
- If $A[i] < A[n/2 + j]$, then there are no inversions of the form (i, k) for $n/2 + j \leq k \leq n$

Algorithm idea: Merge the sorted arrays $A[1, 2, \ldots, n/2]$ and $A[n/2 + 1, \ldots, n]$ while counting the number of inversions using the claim above

- Running time: $T(n) = 2T(n/2) + O(n)$

Problem 3: Closest pair in other metrics

Problem: How does the algorithm for closest pair change when the distance metric changes? Specifically, consider the following metrics:

- L_1 metric: $d_1((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$
- L_∞ metric: $d_\infty((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$

Problem 3: Closest pair in other metrics

Problem: How does the algorithm for closest pair change when the distance metric changes? Specifically, consider the following metrics:

- L_1 metric: $d_1((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$
- L_∞ metric: $d_\infty((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$

The basic algorithmic idea works for all the metrics. The only thing that changes is the number of points within the $2\delta \times 2\delta$ square that has to be considered during the conquer-phase of the algorithm.

Exercise: For each of the metrics, find the number of points that can lie in a $2\delta \times 2\delta$ square.

Takeaway: Constants change, asymptotics will remain the same

Problem 4: Alternate algorithm for closest pair

Problem: Consider the following algorithm

- Choose the leftmost point p according to x-coordinate.
- Recursively find the closest pair and the distance of the remaining $n - 1$ points
- For the point p, look at the position of p in the sorted order using binary search, and search within its neighborhood

Running time: $T(n) = T(n-1) + O(\log n)$

Problem 5: Minkowski sum

Problem: Two sets A and B of n integers each. $A + B$ is defined as $\{x + y \mid x \in A, y \in B\}$. Find $|A + B|$.

Problem 5: Minkowski sum

Problem: Two sets A and B of n integers each. $A + B$ is defined as $\{x + y \mid x \in A, y \in B\}$. Find $|A + B|$.

Method 1:

- Compute $x + y$ for each $x \in A$ and $y \in B$ (at most n^2 of them)
- Sort this set. Traverse the sorted list, counting elements and avoiding repetition

  Running time: $O(n^2 \log n)$

Problem 5: Minkowski sum

Problem: Two sets A and B of n integers each. $A + B$ is defined as $\{x + y \mid x \in A, y \in B\}$. Find $|A + B|$.

Method 2:

- Construct a characteristic vector $S_A[-M, \dots, M]$ where $S_A[i] = 1$ iff $i \in A$. Similarly $S_B[-M, \dots, M]$
- Verify that $S_A * S_B[i]$ is non-zero iff $\exists x \in A$ and $y \in B$ such that $x + y = i$
- Now read the array $S_A * S_B$ and count the number of non-zero values

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \ldots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \ldots, n-1]$, compute $A[\pi(0), \pi(1), \ldots, \pi(n-1)]$.

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \dots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \dots, n-1]$, compute $A[\pi(0), \pi(1), \dots, \pi(n-1)]$.

Method 1:

- For each number i, find the individual bits and invert them - requires $\log n$ operations/number

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \dots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \dots, n-1]$, compute $A[\pi(0), \pi(1), \dots, \pi(n-1)]$.

Method 1:

- For each number i, find the individual bits and invert them - requires $\log n$ operations/number
- Operations on the numbers like addition and multiplication are $O(1)$ - can you use them instead of bit-operations

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \ldots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \ldots, n-1]$, compute $A[\pi(0), \pi(1), \ldots, \pi(n-1)]$.

Method 1:

- For each number i, find the individual bits and invert them - requires $\log n$ operations/number
- Operations on the numbers like addition and multiplication are $O(1)$ - can you use them instead of bit-operations

  Running time: $O(n \log n)$

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \dots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \dots, n-1]$, compute $A[\pi(0), \pi(1), \dots, \pi(n-1)]$.

Method 2: Understand the recursive structure of the permutation

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \ldots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \ldots, n-1]$, compute $A[\pi(0), \pi(1), \ldots, \pi(n-1)]$.

Method 2: Understand the recursive structure of the permutation

$n = 4$: $(0, 1, 2, 3) \to (0, 2, 1, 3)$

$n = 8$: $(0, 1, 2, 3, 4, 5, 6, 7) \to (0, 4, 2, 6, 1, 5, 3, 7)$

- Can you generalize the statement and prove it?

Problem 6: Bit-reversal permutation

Problem: The bit reversal permutation π of $1, 2, \ldots, n$ is the permutation where $\pi(i)$ is the integer obtained by reversing the bit-representation of i. Given $A[0, 1, \ldots, n-1]$, compute $A[\pi(0), \pi(1), \ldots, \pi(n-1)]$.

Method 2: Understand the recursive structure of the permutation

$n = 4$: $(0, 1, 2, 3) \to (0, 2, 1, 3)$

$n = 8$: $(0, 1, 2, 3, 4, 5, 6, 7) \to (0, 4, 2, 6, 1, 5, 3, 7)$

- Can you generalize the statement and prove it?

Claim: Let π be the bit reversal permutation on the set $S = \{0, 1, \ldots, n-1\}$. Consider the set $S' = \{0, 1, \ldots, 2n-1\}$. The bit reversal permutation π' on S' is given by

$$\pi'(i) = \begin{cases} 2\pi(i) & \text{if } i \le n-1, \\ 2\pi(i-n)+1 & \text{if } i \ge n \end{cases}$$