Week 2 Tursday Lecture Notes

Before Class:

- Recitation on Wednesday: 5:45-6:45 reviewing Week 1 Homework problems
- Prepare for Quiz in class.
- Review Common Efficiency Classes (how to tell asymptotic equivalence)
- Review Slides for Inversion Counting on Canvas.
- Lecture Summaries at end of PowerPoints.

My Quiz Review:

- Asymptotic Equivalence.
- Runtime (**Ω, Θ, and O**)
- We care about Worst Case for run time.

Proof by Induction:

1. Prove the base case
2. Induction Hypothesis: assume a statement is true for some k, or all <=k
3. Inductive step: Prove the statement is true for k+1.

Recurrence Relation Master Theorem:

Master Theorem

$$T(n) = aT(\frac{n}{b}) + cn^d$$

Case 1: $a < \mathrm{b}^{\mathrm{d}}$, or equiv. $d > \log_b a$, $T(n) = O\left(n^d\right)$

Case 2: $a = b^d$, or equiv. $d = \log_b a$, $T(n) = O\left(n^d \log n\right)$

Case 3: $a > b^d$, or equiv. $d < \log_b a$, $T(n) = O\left(n^{\log_b a}\right)$

Divide and Conquer:

Inversion Counting: Number of inversions in an array.

Example: [1,2,3,4,5] (no inversions)

[1,3,5,2,7]

- Inversion for 2 and 5, then 2 and 3.

Start on Page 5 of PowerPoint 3: High Level Idea

Counting how many inversions there are in an array.

Divide and conquer the array, break into two parts.

- Count Cross inversions between two parts.

Target is to be able to count Cross Inversions in linear time O(n)

What if left and right are already sorted?

A1: = (2,4,5) A2: (1,3,6)

Cross counting can be done very quickly if both are already sorted.

Building on Merge_sort:

Ls: Sort and Count

Rs: Sort and Count

As: Merge and CountCross (Ls,Rs)

Based on the target runtime you can figure out what needs to change to fit with the Master Theorem.

Pseudo Code for Merge Sort:

We can count how many inversions there are whenever the code enters the else statement in the code.

- The number of elements Left in L when the else is run is how many inversions there will be for that value.
- Use Pseudo code to figure out the sorted array as well as the count for number of inversions.
- Are able to maintain an n log n time.

- Claim: when an element y of R is copied into the output array A, the number of inversion y incurs = the number of elements left in L
- Proof:

Let x be an element of L.

1. If x is copied into A before y, we know $x < y$, which does not cause inversion
2. If x is copied into A after y, we know $y < x$, which causes inversion

Claim: When a **Right** element is copied, the number of cross inversions is the number of elements still in the **Left** side. Assumes that the array is sorted.

- Because of this we are able to count Cross Inversions in Linear time.
- Going to use Proof by Induction to solve that the Merge and CountCross is done in Linear time.

END OF LECTURE 3

Lecture 4: Divide and Conquer: Closest Pair of Points

- Discuss First assignment: Closest Pair of Points.
- (x1,y1), (x2,y2) SquareRoot $(x1-x2)^2+(y1-y2)^2$
- Lots of applications

You can brute force: Check all pairs of points and q with n^2 Comparisons.

Initial Thoughts: Brute force in $O(n^2)$

1-D version: O (nlog n) if points are on one axis.

Assuming that no 2 points have the same x coordinate.

- Find closest pair in each side as well as the closest pair with one point on each side. Return best 3 solutions.
- Once you figure out the smallest between the 2 sets you can restrict what can be between the cross.

How can we further Improve this Algorithm? Food for Thought for next week.

End on Page 5 of Lecture 4 Powerpoint.

Had Week 2 Quiz at end of class Period. Focus on Big-O/Omega/Theta and creating and solving Recurrence Relations.

<u>End of Week 2 Tuesday Lecture Notes</u>

~Information composed by Notetaker Scott Russell for CS 325 **DAS** student