CompSci 161
Winter 2023 Lecture 4:
Divide and Conquer I:
Inversion Counting

2

Counting Inversions

- i, j are an *inverted pair* if $i < j$ and $A[i] > A[j]$.
  (the larger element appears earlier in the array)

For example:

85	24	63	45	17	31	96	50

and more!
:)

$\binom{n}{2}$ possible $\longrightarrow O(n^2)$

2

Counting Inversions

- i, j are an *inverted pair* if $i < j$ and $A[i] > A[j]$. (the larger element appears earlier in the array)

The following is an $\Theta(n^2)$ time way to count the inversions in an array:

count = 0
for $i = 1 \ldots n$ **do**
  for $j = i + 1 \ldots n$ **do**
    if $A[i] > A[j]$ **then**
      count++
return count

//n = A, size ( )

– brute force

– polynomial time*

of size of input

3 Counting Inversions Faster: a subproblem

merge-and-count ( )

{

- want to count number of inverted pairs in A,
- we know $A[1 \ldots \frac{n}{2}]$ is sorted, as is $A[\frac{n}{2} + 1 \ldots n]$.
- Can we do better than $\Theta(n^2)$?

~~i~~ i ~~j~~ j

24	45	63	85	17	31	50	96

Count = 0, declare T[1...n], k = 1

while ($j \leq n$ and $i \leq n/2$)

{ if A[j] < A[i]

Count += # elements 1st half after & including A[i]

T[k] = A[j]

j++, k++,

else

T[k] = A[i]; k++; i++

}

// after while loop

while ($j \leq n$)

  $T[k] = A[j]$; k++; j++

while ($i \leq n/2$)

  $T[k] = A[i]$; k++; i++

copy T over to A

return count

4

Finishing the Merge Portion

- We want sorted list when done
- Let's keep the rest of the array

5 **Counting Inversions Faster**

Count (A)
{

- Use the algorithm from the previous question
- count number of inversions in *unsorted* array
- How fast is your algorithm?

if $n \le$?? 10, something small
brute force a count and sort.
return

$C_L =$ Count $(A[1 \dots n/2])$
$C_R =$ count $(A[\frac{n}{2}+1 \dots n])$
$C_M =$ merge - and - count (A)
return $C_L + C_R + C_M$

6 **Running Time for Counting Inversions**

if list has one or zero elements **then**
  return no inversions
Divide into $L = A[1 \dots \frac{n}{2}]$ and $R = A[\frac{n}{2} + 1 \dots n]$
// Should this be slices or parameters?
Recursively solve on L; count is c_L
Recursively solve on R; count is c_R
Run earlier subproblem on L, R; count is c_M
return $c_L + c_R + c_M$

How long does this take?

$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

7

Running Time for Counting Inversions

- Two recursive of size $n/2$, plus local linear work
- $T(n) = 2T(n/2) + n$

work done here, not incl recursion