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

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 |

Possible $\binom{n}{2}$ $\implies \mathcal{O}(n^2)$
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:

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

// $n = \text{size of } A$
```

Counting Inversions Faster: a subproblem

merge-and-count($A$)
- 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 $\Theta(n^2)$?

```
24 45 63 85 17 31 50 96
Count = 0, declare $T[1 \ldots n]$, $K = 1$
while ($j \leq n$ and $i \leq \frac{n}{2}$ )
{ if $A[j] < A[i]$
    Count += # elements 1st half after & including $A[i]$
    $T[K] = A[j]$
    $j$, $K$, $i$, $j$

else
}
// after while loop

while (j ≤ n)
    T[k] = A[j]; k++; j++

while (i ≤ n/2)
    T[k] = A[i]; k++; i++

copy T over to A

return count

Finishing the Merge Portion

- We want sorted list when done
- Let’s keep the rest of the array
Counting Inversions Faster

\texttt{Count}\left(A\right)

\begin{itemize}
  \item Use the algorithm from the previous question
  \item count number of inversions in unsorted array
  \item How fast is your algorithm?
\end{itemize}

\texttt{if} \ n \leq 10 \ \texttt{something small}
\texttt{brute force a count and sort.}
\texttt{return}

\texttt{C}_L = \texttt{Count}\left(A[1 \ldots \left\lfloor n/2\right\rfloor]\right)
\texttt{C}_R = \texttt{Count}\left(A[^{n/2}+1 \ldots n]\right)
\texttt{C}_M = \texttt{merge-and-count}\left(A\right)

\texttt{return} \ C_L + C_R + C_M

Running Time for Counting Inversions

\begin{itemize}
  \item if list has one or zero elements then
  \texttt{return} \ no inversions
  \item Divide into \(L = A[1 \ldots \left\lfloor n/2\right\rfloor]\) and \(R = A[^{n/2}+1 \ldots n]\)
  \item Recursively solve on \(L\); count is \(c_L\)
  \item Recursively solve on \(R\); count is \(c_R\)
  \item Run earlier subproblem on \(L, R\); count is \(c_M\)
\end{itemize}

\texttt{return} \(c_L + c_R + c_M\)

How long does this take?

\[ T\left(n\right) = 2T\left(\frac{n}{2}\right) + \Theta\left(n\right) \]
Running Time for Counting Inversions

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