CMPSC 250
Analysis of algorithms

Dr. Aravind Mohan
Allegheny College

February 16, 2018
Lecture 11 - Randomized Quick Sort and Merge Sort
Characteristics of Randomized Quick Sort

- Assume all elements are distinct.
- Partition around a random element.
- Consequently, all splits (1: n-1, 2: n-2, ... , n-1: n)
- Randomization is a general tool to improve algorithms with bad worst-case but good average case complexity.
Randomized Partition

**function** RANDOMIZED-PARTITION(A, p, r)

\[ i \leftarrow \text{Random}(p, r) \]

swap A[r] and A[i]

**return** Partition(A, p, r)

**end function**
Randomized QuickSort

```plaintext
function RANDOMIZED-QUICKSORT(A, p, r)
    if p < r then
        q ← Randomized-Partition(A, p, r)
        Randomized-QuickSort(A, p, q-1)
        Randomized-QuickSort(A, q+1, r)
    end if
end function
```
Expected Running Time

- Randomizing pivot selection
- Splits are going to be better than $0: n-1$
- Take average for running time using different pivot selection.

$O(n \times \log(n))$
Characteristics of Merge Sort

- sort out of “place”, i.e., does require an additional array
- uses divide and conquer principle
- worst case running time is $O(n \times \log(n))$
Some background about Merge Sort

- Invented by John von Neumann in 1945
Merge Sort - the Principle

To understand merge-sort, let’s look at a high-level description of the algorithm.

A divide-and-conquer algorithm.

- **Divide:** if S has at least two elements (nothing needs to be done if S has zero or one elements), remove all the elements from S and put them into two sequences $S_1$ and $S_2$, each containing about half of the elements of S. (i.e., $S_1$ contains the first floor $(n/2)$ elements and $S_2$ contains the remaining floor $(n/2)$ elements.)
Merge Sort - the Principle

- **Conquer**: Sort sequences $S_1$ and $S_2$ using Merge Sort.
- **Combine**: Put back the elements into $S$ by merging the sorted sequences $S_1$ and $S_2$ into one sorted sequence.
function \texttt{MERGE}(A, p, q, r)

\begin{align*}
n_1 & \leftarrow q - p + 1 \\
n_2 & \leftarrow r - q
\end{align*}

Let \(L[1, \cdots, n_1 + 1]\) and Let \(R[1, \cdots, n_2 + 1]\) be a new array

\begin{align*}
\text{for } i = 1 \text{ to } n_1 & \text{ do} \\
& L[i] \leftarrow A[p+i-1]
\end{align*}

\textbf{end for}

\begin{align*}
\text{for } j = 1 \text{ to } n_2 & \text{ do} \\
& R[j] \leftarrow A[q+j]
\end{align*}

\textbf{end for}

\begin{align*}
L[n_1 + 1] & \leftarrow \text{Integer.MAX} \\
R[n_2 + 1] & \leftarrow \text{Integer.MAX}
\end{align*}

[Continue Next Page ....]

end function
function \textsc{merge}(A, p, q, r) \\
[Continue Next Page ....] 
\\ni \leftarrow 1 ; j \leftarrow 1 
\\for k = p to r do 
\\ \quad \text{if } L[i] \leq R[j] \text{ then} 
\\ \quad \quad A[k] \leftarrow L[i] 
\\ \quad \quad i \leftarrow i+1 
\\ \quad \text{else} 
\\ \quad \quad A[k] \leftarrow R[j] 
\\ \quad \quad j \leftarrow j+1 
\\ \quad \text{end if} 
\\ \text{end for} 
\\end function
Merge Sort Procedure

function MERGESORT(A, p, r)
    if p < r then
        q ← Floor \((p + r)/2\)
        MERGESORT(A, p, q)
        MERGESORT(A, q+1, r)
        MERGE(A, p, q, r)
    end if
end function
Let us look at an example (make notes..)
Complexity Analysis

- Space complexity: $O(\log(n))$
- Time complexity: $O(n \times \log n(n))$ for Worst, Average, and Best case
Questions