CS 5720 Design and Analysis of Algorithms
Homework #5

Your Name

October 17, 2024

Question 1: MergeSort and QuickSort Analysis

(a) MergeSort on Sorted Arrays

REcurrence Relation:

$$T(n) = 2T\left(\frac{n}{2}\right) + n - 1$$

1. Splitting the array takes $2T(n/2)$ comparisons.
2. Merging two sorted arrays of size $n/2$ takes $n-1$ comparisons (since each comparison merges one element).

Solution Using Master Theorem:

1. Here, $a = 2$, $b = 2$, and $f(n) = n - 1 = \Theta(n)$.
2. Calculate $\log_b a = \log_2 2 = 1$.
3. Since $f(n) = \Theta(n)$, we use Case 2 of the Master Theorem.

$$T(n) = \Theta(n \log n)$$

(b) MergeSort on Reverse-Sorted Arrays

REcurrence Relation:

$$T(n) = 2T\left(\frac{n}{2}\right) + n - 1$$

$\longrightarrow$ The recurrence relation is the same as for sorted arrays because MergeSort's comparison count does not depend on the order of the elements.

Solution Using Master Theorem:

$\longrightarrow$ Same as part (a).

$$T(n) = \Theta(n \log n)$$

(c) QuickSort on Sorted Arrays

Recurrence Relation:

$$T(n) = T(n-1) + n - 1$$

1. The first partition step compares each element to the pivot, making $n-1$ comparisons.
2. The lazy pivot selection (first element in sorted array) leads to one partition of size $n-1$ and one partition of size 0.

Solution:

:

1. Unroll the recurrence:

$$T(n) = T(n-1) + (n-1) = T(n-2) + (n-2) + (n-1) = \cdots = T(1) + \sum_{i=1}^{n-1} i = T(1) + \frac{n(n-1)}{2}$$

2. Since $T(1) = 0$ (base case), we have:

$$T(n) = \Theta(n^2)$$

(d) QuickSort on Arrays of All Equal Elements

Recurrence Relation:

$$T(n) = 2T(n-1) + (n-1)$$

1. Each partition step still compares each element to the pivot, making $n-1$ comparisons.
2. For arrays with all equal elements, each partition splits the array into two equal parts.

Solution:

1. Unroll the recurrence:

$$T(n) = T(n-1) + (n-1) = T(n-2) + (n-2) + (n-1) = \cdots = T(1) + \sum_{i=1}^{n-1} i = T(1) + \frac{n(n-1)}{2}$$

2. Since $T(1) = 0$ (base case), we have:

$$T(n) = \Theta(n^2)$$

Question 2: Optimizing QuickSort for Sorted Arrays

We cannot choose the first or lasst elements of the arrays as one partition might contain most of the elements of the array and further down the recursion the resulting partitions will have one empty subarray and one subarray containing all the remaining elements, the depth of the recursion tree will be at it's maximum yeilding a time complexity of $\Theta(n^2)$. Choosign randomly might help in some scenarios but we can be certian that in some cases this will result in sub-optimal performance.

To optimize QuickSort for sorted arrays, we should select the pivot such that the partitions are balanced.Since we know that the array is sorted.we just have to select the middle element of the array and make it our pivot.

Pivot Selection Rule

- **Rule**: Select the median of the array as the pivot. In sorted arrays, this will divide the array into two equal parts.

Resulting Recurrence Relation

$$T(n) = 2T\left(\frac{n}{2}\right) + n - 1$$

$\longrightarrow$Selecting the median as the pivot results in two partitions of size $n/2$.

Solution using Master Theorem:

1. Here, $a = 2$, $b = 2$, and $f(n) = n - 1 = \Theta(n)$.
2. Calculate $\log_b a = \log_2 2 = 1$.
3. Since $f(n) = \Theta(n)$, we use Case 2 of the Master Theorem.

$$T(n) = \Theta(n \log n)$$