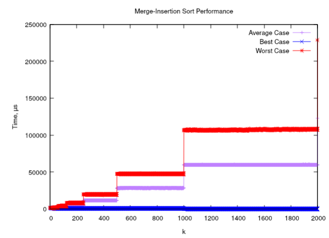
# Homework 4

Rumen Mitov

March 4, 2024

# Problem 4.1




To calculate the time complexity of Merge-K-Sort, we first need to examine the height, h, of the recursion tree based on the inputs, n, and the minimum subarray length. The height of the recursion tree for regular merge-sort comes from:
$2^h = n$
Hence, if we stop merge-sort when our subarrays have at most k elements:
$\frac{2^h}{k} = n$
$\implies h = \log k \cdot n$
Merging the subarrays will be a $\Theta(n)$ operation. So time complexity before taking into account the insertion-sort step will be:
$T(n) = n \cdot \log k \cdot n$
Insertion sort is a O($k^2$) and a $\Omega(k)$:
$O(n) = n \cdot \log k \cdot n \cdot k^2$
$\Theta(n) = n \cdot \log k \cdot n \cdot k$

Based on my answers for c) and d) we can see that when $k = 1$ we will have the same time complexity as regular merge-sort. Hence I would pick $k = 1$.

# Problem 4.2

$T(n) = 36 \cdot T(\frac{n}{6}) + 2n$
Using Master Theorem:
$a = 36$
$b = 6$
$n^{\log_b a} = n^2$
Hence:
$f(n) = 2n$ is polynomially smaller than $n^2$, so:
$T(n) = \Theta n^2$

$T(n) = 5 \cdot T(\frac{n}{3}) + 17n^{1.2}$
Base case: n = 2

$T(2) = 5 \cdot T(\frac{2}{3}) + 17 \cdot (2)^{1.2} = \Theta(1)$
Assuming $T(k) \leq c \cdot k^2 \| k < n$, show that $T(n) \leq c \cdot n^2$.
$T(n) \leq 5c \cdot (\frac{n}{3})^2 + 17n^{1.2}$
$\implies T(n) \leq cn^2 - (\frac{4}{9}n^2 \cdot c - 17n^{1.2})$
Since $n \geq 2$:
$\frac{4}{9}n^2 \cdot c - 17n^{1.2} > 0$
So $T(n) = O(n^2)$.
And since each operation requires at least $n^{1.2}$ time cost, we can say that $T(n) = \Omega(n^{1.2})$.

$T(n) = 12 \cdot T(\frac{n}{2}) + n^2 \cdot \log n$
Base case: n = 2
$T(2) = 12 \cdot T(\frac{2}{2}) + 2^2 \cdot \log 2 = \Theta(1)$
Assuming $T(k) \leq c_1 \cdot k^3 - c_2 \cdot k^2 - c_3 \cdot k \| k < n$, show that $T(n) \leq c_1 \cdot n^3 - c_2 \cdot n^2 - c_3 \cdot n$.
$T(n) \leq 12(c_1(\frac{n}{2})^3 - c_2(\frac{n}{2})^2 - c_1\frac{n}{2}) + n^2 \log n$
$\implies T(n) \leq c_1 n^3 - c_2 n^2 - c_1 n - (-n(\frac{1}{2}c_1 n^2 + 2c_2 n + 5c_3 + n \log n))$
Since $n \geq 2$:
$-n(\frac{1}{2}c_1 n^2 + 2c_2 n + 5c_3 + n \log n) < 0$
So we can establish that:
$T(n) = \Omega(n^2 \log n)$
We now must examine a higher power of n.
Assuming $T(k) \leq k^4$:
$T(n) = 12(c(\frac{n}{2})^4) + n^2 \log n$
$\implies T(n) = cn^4 - (n^2(\frac{1}{4}cn^2 - \log n))$
Since $n \geq 2$:
$n^2(\frac{1}{4}cn^2 - \log n) > 0$
So $T(n) = O(n^4)$.

$T(n) = 3T(\frac{n}{5}) + T(\frac{n}{2}) + 2^n$
Base case: n = 2
$T(2) = 3T(\frac{2}{5}) + T(\frac{2}{2}) + 2^2 = \Theta(1)$
Assuming $T(k) \leq c \cdot 2^k \| k < n$, show that $T(n) \leq c \cdot 2^n$.
$T(n) = 3(c2^{\frac{n}{5}}) + (c2^{\frac{n}{2}}) + 2^n$
$\implies T(n) = c2^n - (2^n(c2^{\frac{-n}{2}} - 3c2^{\frac{-4}{5}} - 1))$
Since $n \geq 2 \wedge c2^{\frac{-n}{2}} > -3c2^{\frac{-4}{5}} - 1$:
$T(n) = c2^n - (2^n(c2^{\frac{-n}{2}} - 3c2^{\frac{-4}{5}} - 1)) > 0$
So $T(n) = O(2^n)$ and since each operation has a time complexity of at least $2^n$, we can say that $T(n) = \Omega(2^n)$. Therefore, we can conclude that:
$T(n) = \Theta(2^n)$

$T(n) = T(\frac{2n}{5}) + T(\frac{3n}{5}) + \Theta(n)$
Using the recursive tree method we can represent $T(n)$ as such:
$T(n) = \Theta(n) + \Theta(c_1 n) + \Theta(c_2 n) + \ldots$
And since we are dealing with $\Theta$ we can drop the constants and merge terms of the same time complexity, resulting in:
$T(n) = \Theta(n)$