MultiMerge

Isaac Gutt

February 28, 2022

1 Iterative Algorithm

1. Base Case: $T(1) = 1$
   Recursive Case: $T(z) = T(z-1)n$

2. This algorithm first combines the first two n sized arrays, a 2n operation. Then, it combines the third with the first two, the fourth with the first 3, and on. Each iteration, the size of the operation is increased by n. The summation above shows clearly, the operation goes from 2n to 3n and on until (z-1)n, the final operation that returns the sorted array.

3. $T(1) = 1$ (Just return the first array)
   $T(z) = T(z-1)n$
   2n + 3n ... (z-1)n
   $= n(2 + 3...(z-1))$
   $= n((z-1) * z)/2$
   $= n(z^2 - z)/2$
   $=$ O(nz^2)

2 Divide-and-Conquer Algorithm

1. mergeAll(arrays, start, end) is called, start = 0 and end = z-1. MergeAll calls itself twice, the first call starting from 0 to n/2, and the second from n/2+1 to n. The method keeps halving the problem until a base case is reached, where start == end, then it returns the single array corresponding to start. Once that happens, all n arrays are combined

in pairs of two, then the new arrays are combined in pairs of two, recursively until the arrays remain. The two are combined in an O(n) operation and returned as one sorted array. If n is odd, there will be one array that's combined with another not of it's size until there are two left.

2. $T(z) = 2T(z/2) + n$
   z = number of arrays, n = array length

3. Inputs to the master theorem:
   $a = 2$, $b = 2$, $d = 1$
   $a = b^d$ so we have T(z) = n^d log z.
   Plug into the master theorem and we get O(n log(z)) which is better than the iterative algorithm.