Lesson 5 Merge sort design and analysis

# Learning objectives

1. Write the pseudocode for merge sort and the merge operation
2. Using precise mathematical arguments, explain why merge sort runs in O(n log n) time.
3. Implement merge sort as a computer program.

# Agenda

1. Let students go through the PowerPoint document Merge Sort Examples.ppt at their own pace
2. Have teams of 2 or 3 try to derive the pseudocode for merge sort, assuming the merge method has already been coded.
3. Pick one team to present their solution, and discuss.
4. Analysis of the run-time and why it’s O(n log n).
5. Teams go through The Merge Step.ppt at their own pace and try to derive the pseudocode for it.
6. HW: Code it all in Java!

# Analysis of the run-time for merge sort and why it’s O(n log n)

Let T(n) = the time it takes merge sort to sort n elements

T(n) = T(n/2) + T(n/2) + O(n)

time to merge the two

time to sort the

right half

time to sort the

left half

T(n) = 2 T(n/2) + O(n)

T(1) = 0

(Why? How long does it take to sort 1 element? Zero! I don’t even need to look at the element!)

Illustration

Suppose n = 64

T(64) = 2T(32) + 64

= 2(2T(16) + 32) + 64

= 4T(16) + 64 x 2

= 4(2T(8) + 16) + 64 x 2

= 8T(8) + 64 x 3

= 8(2T(4) + 8) + 64 x 3

= 16T(4) + 64 x 4 Can you finish it from here? Stop when you get to T(1)

= 16(2T(2) + 4) + 64 x 4

= 32T(2) + 64 x 5

= 32(2T(1) + 2) + 64 x 5

= 64T(1) + 64 x 6

= 0 + 64 x 6

= 64 x 6

= n x log2n

What is the significance of the 6? That’s just log264!

|  |  |
| --- | --- |
| Let’s do this for generic n  T(n) = 2T(n/2) + n  = 2(2T(n/4) + n/2) + n  = 4T(n/4) + 2n  = 8T(n/8) + 3n  = …  = kT(1) + kn  This took k steps  What is k? log2 n!  So T(n) = 0 + kn = n log n |  |
|  |  |