SIENA COMPUTER SCIENCE

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2025

Mergesort Practice

Basic Idea

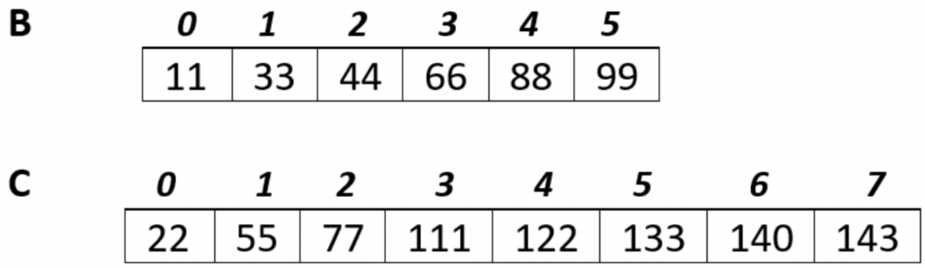
5	3	7	1	2	4	6	8

5	3	7	1

2	4	6	8

Consider the MERGE algorithm on the last page of this packet.

Suppose B and C contain the following values, so $p = 8$ and $q = 6$.



B

0	1	2	3	4	5
11	33	44	66	88	99

C

0	1	2	3	4	5	6	7
22	55	77	111	122	133	140	143

Show the contents of array A when the `while` loop has finished executing.

Show the contents of array A after the `if-else` statement following the `while` loop has finished executing.

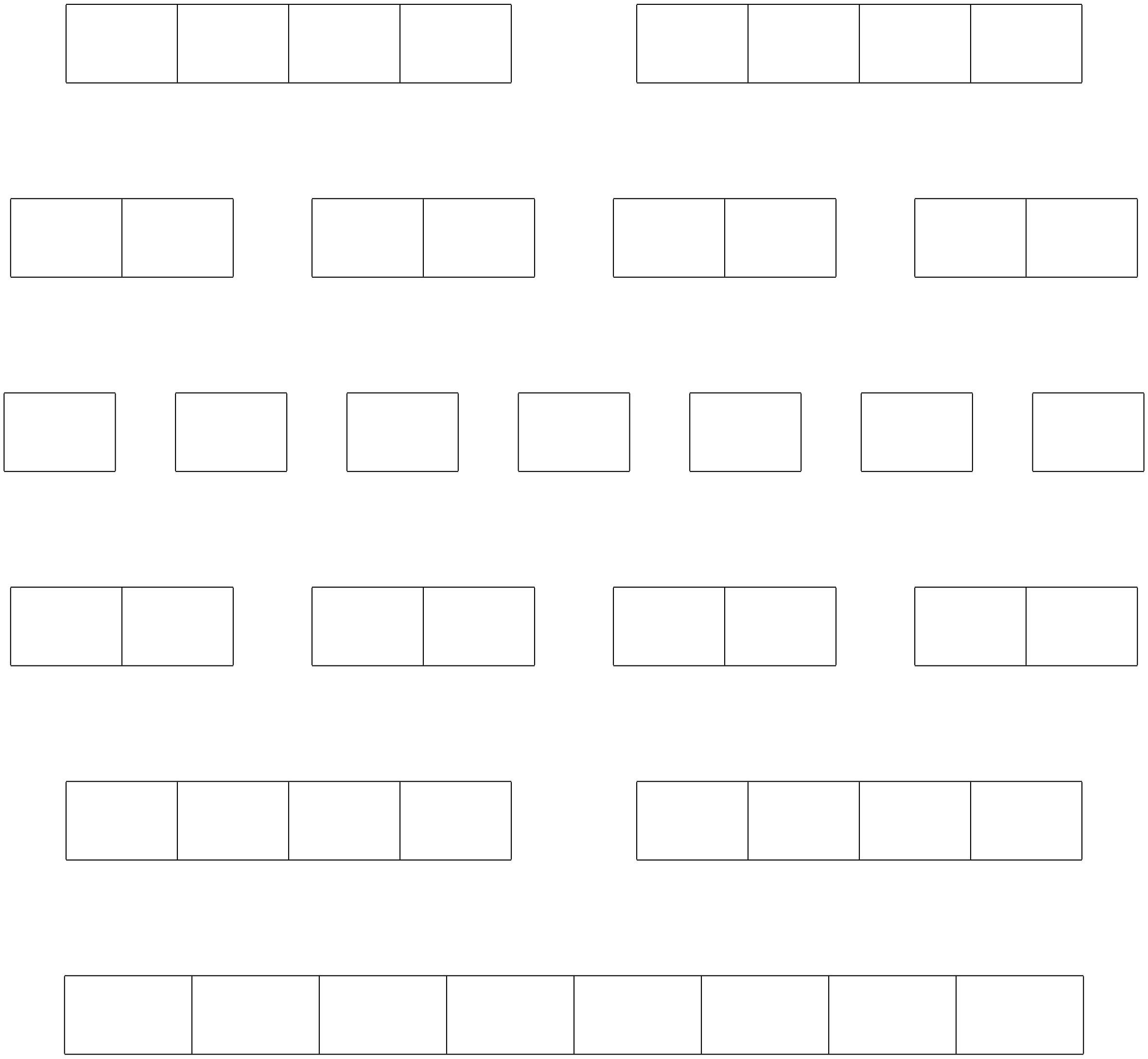
Worst case, how many times is k incremented in the `while` loop?

Worst case, how many items are copied from C to A and from B to A after the `while` loop?

Based on your answers to the previous three questions, what is the Big O worst case running time of algorithm MERGE? Express your answer in terms of n, where $n = p + q$ is the size of array A.

Tracing through MERGESORT

5	3	7	1	2	4	6	8



How many split steps will it take?

Then we will have ______ merge steps

Each merge step involves sub-arrays totaling in size to _____

At the level with k independent merges, each will merge into arrays of size ____ for a total of ____ operations

This suggests an overall complexity of ______

Analysis (assume $n = 2^k$):

Basic operation:

Recurrence:

$$C(n) =$$

Worst case?

$$C_{worst}(n) =$$

By the master theorem,

$$C_{worst}(n) \in \Theta(\qquad\qquad)$$

Mergesort is *not* in place. Space overhead: $\Theta(\qquad)$

```
ALGORITHM Merge($B, C, A$)
    //Input: a sorted array $B[0..p-1]$
    //Input: a sorted array $C[0..q-1]$
    //Output: a sorted array $A[0..(p+q-1)]$
    $i \leftarrow 0$
    $j \leftarrow 0$
    $k \leftarrow 0$
    // take smallest item from $B$ and $C$ and put into $A$
    while $i < p$ and $j < q$ do
        if $B[i] < C[j]$ then
            $A[k] \leftarrow B[i]$
            $i \leftarrow i + 1$
        else
            $A[k] \leftarrow C[j]$
            $j \leftarrow j + 1$
        $k \leftarrow k + 1$
    // if items remain in $B$, put them in $A$
    while $i < p$ do
        $A[k] \leftarrow B[i]$
        $i \leftarrow i + 1$
        $k \leftarrow k + 1$
    // if items remain in $C$, put them in $A$
    while $j < q$ do
        $A[k] \leftarrow C[j]$
        $j \leftarrow j + 1$
        $k \leftarrow k + 1$
```

```
ALGORITHM MergeSort($A$)
    //Input: an array $A[0..n-1]$
    if $n > 1$ then
        copy first half of array $A$ into a temp array $B$
        copy second half of array $A$ into a temp array $C$
        $MergeSort(B)$
        $MergeSort(C)$
        $Merge(B, C, A)$
```