CSC 151 Spring 2020
Topic: Merge Sort

May 4, 2020

Day 39

Self Checks
Divide and Conquer

- Why is merge sort called a “divide and conquer” algorithm?

- Give another divide and conquer algorithm we have studied.

Self Checks
Divide and Conquer

- Why is merge sort called a "divide and conquer" algorithm?

Divide the input in half, to only deal with half the input at a time.

- Give another divide and conquer algorithm we have studied.

Binary Search

Self Checks
Merging

- Give an expression to merge the lists (1 2 3) and (1 1.5 2.3)
- Give an expression to merge two identical lists of numbers.
- Give an expression to merge two lists of strings.

Self Checks
Merging

- Give an expression to merge the lists (1 2 3) and (1 1.5 2.3)

(merge (list 1 2 3) (list 1 1.5 2.3) <=)

- Give an expression to merge two identical lists of numbers.

(merge (list 1 2 3) (list 1 2 3) <=)

- Give an expression to merge two lists of strings.

(merge (list “acorn” “tree”) (list “blue sky” “swing”) string-ci<=?)

Self Checks
Splitting

- What will the output be: (split (list 1 2 3 12 2 3))
- What will the output be: (split (list 1 2 3 12 2))

Reviewing Insertion Sort

Suppose we have a list of length n

1. What is the worst case scenario for insertion sort? In other words, which type of list takes the *longest* to sort using insertion sort?
2. How many steps does the worst case take, in terms of n? Explain.
3. Suppose you add one new element to the list, so it's length n+1, how many *more steps* does it take?

Reviewing Insertion Sort

Suppose we have a list of length n

1. What is the worst case scenario for insertion sort? In other words, which type of list takes the *longest* to sort using insertion sort? **If its already sorted**
2. How many steps does the worst case take, in terms of n? Explain. **n(n+1)/2**
3. Suppose you add one new element to the list, so it's length n+1, how many *more steps* does it take? **n**

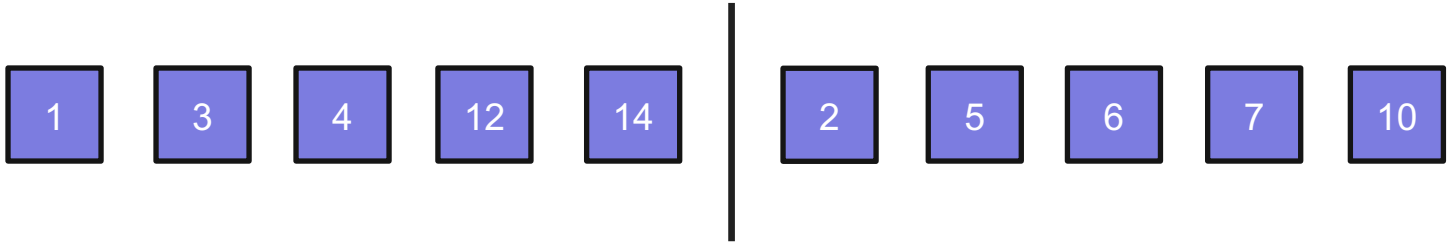
Practice with MergeSort

The following slides go through many examples of practicing merging two lists together.

Write out the steps on paper. It may seem tedious, but it is very beneficial to understanding the algorithm!

Ex1: Divide and Conquer

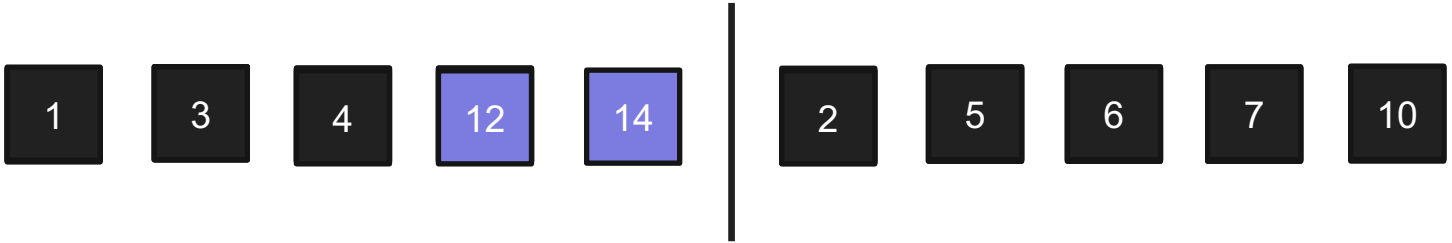
Suppose we have 2 sorted lists, each of length n/2.



How much effort is it to merge them?

Ex1: Divide and Conquer

Suppose we have 2 sorted lists, each of length n/2



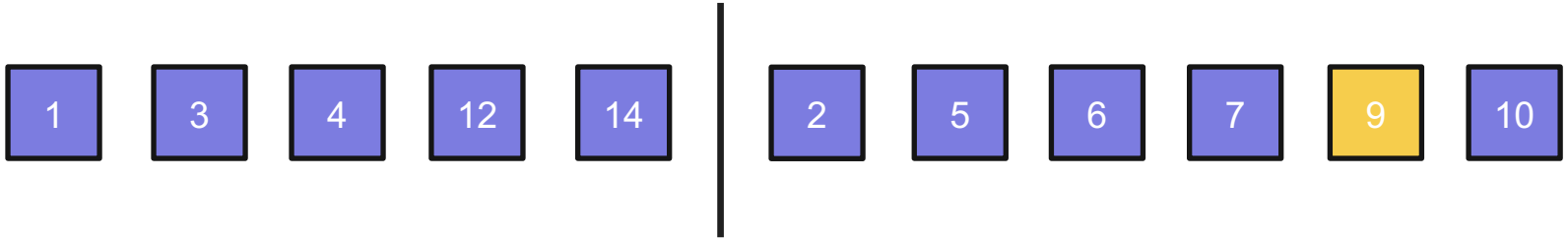
How much effort is it to merge them?



Answer: At most n

Ex2: Merging

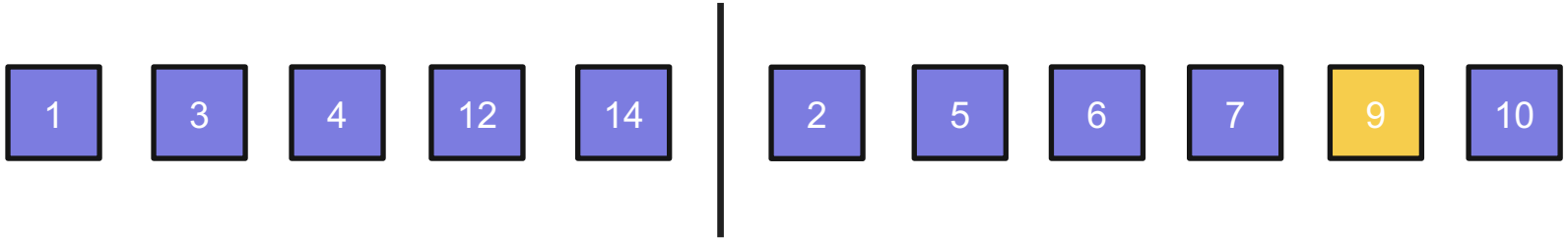
Suppose we have 2 sorted lists, each of length n/2



How many more steps does this take compared with the previous example? Write out the steps on paper.

Ex2: Merging

Suppose we have 2 sorted lists, each of length n/2

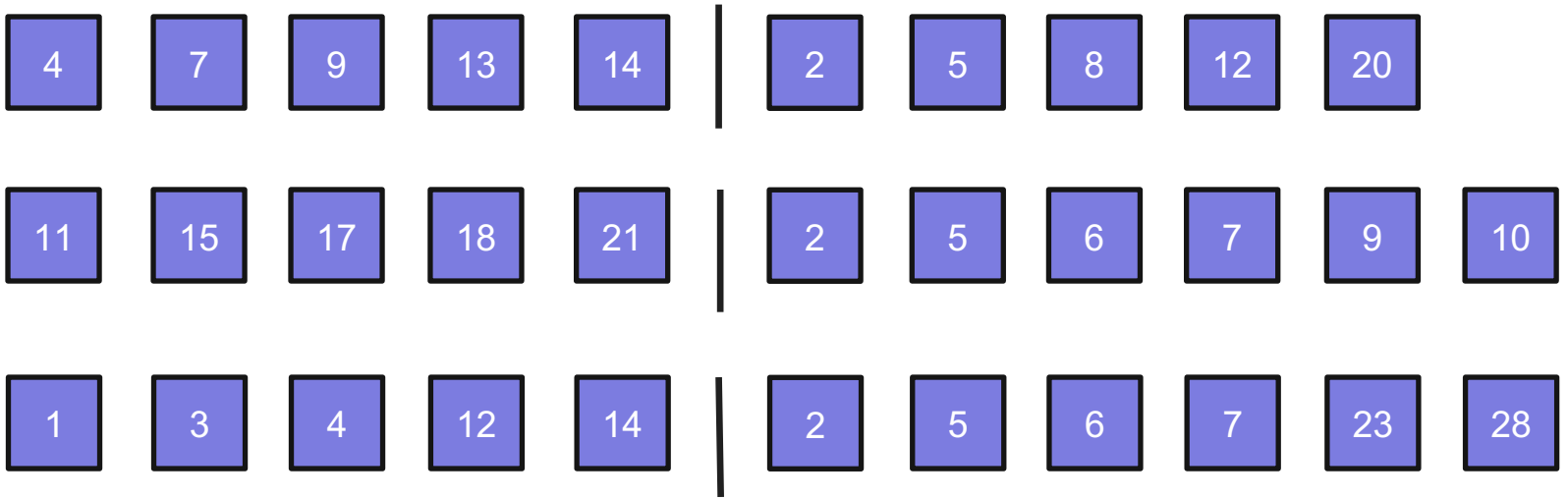


How many more steps does this take compared with the previous example? Write out the steps on paper.

Ans: This only takes 1 (one) more step! Compared with insertion sort – which took n more steps! :o

Ex3: Merging

Suppose we have 2 sorted lists. Practice merging them.



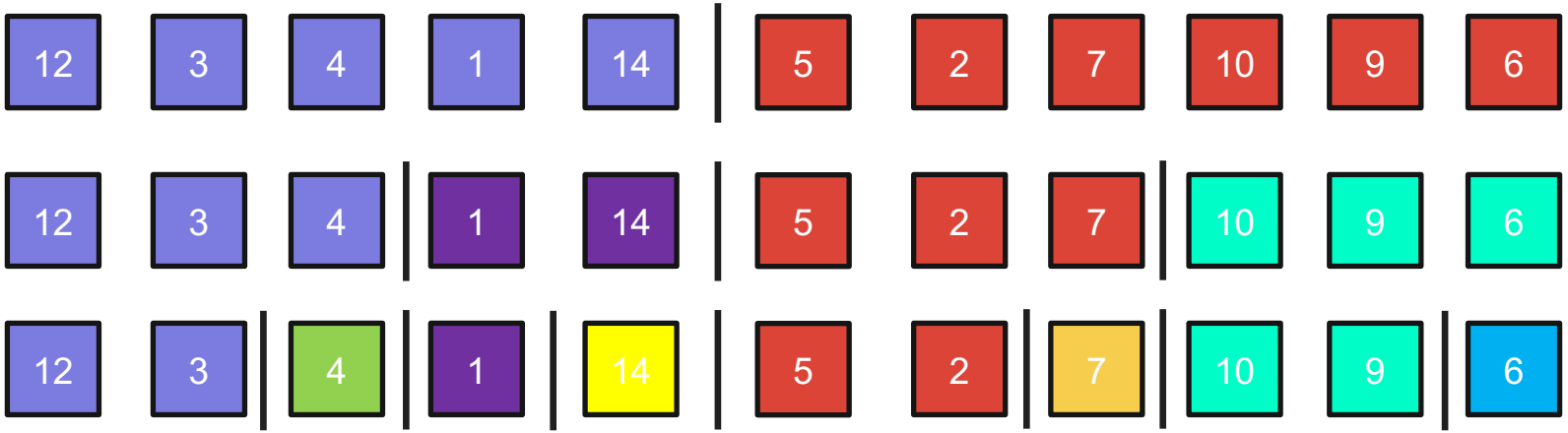
Based on your experimentation with the lists above, what types of lists take the least and most amount of time to merge?

Ex4: Splitting

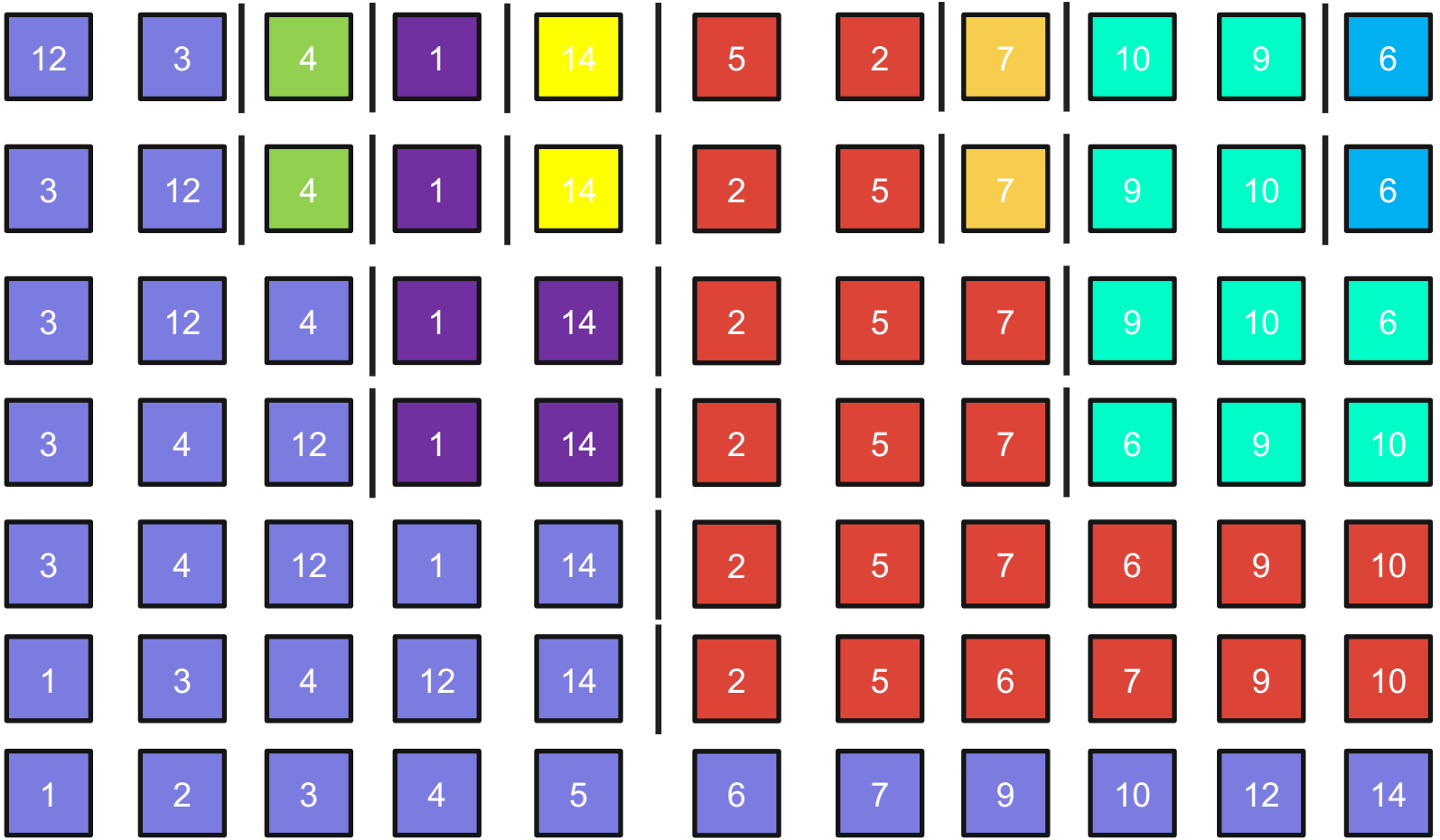
Suppose now that we have 1 unsorted list of length n



Recursively split the list into 2 until we get to the base case



Recursively merge



Cost of Merge Sort

In insertion sort, the average case takes $\frac{n^2}{4}$

In merge sort, the average case takes $n \log_2 n$

$n \log_2 n < \frac{n^2}{4}$ when n is large, meaning that merge-sort is more efficient.

Lab Time



Project Deadlines

~~**Part 1: Project Ideas** *Due on Tuesday April 21*~~
~~Give 3 potential project ideas.~~

~~**Part 2: Project Proposal** *Due on Friday April 24*~~
~~Decide on a data set and describe what you plan to do with it.~~
~~Write about expected work, fall back idea(s), and a reach goal.~~

Part 3: Project *Due on Tuesday May 5*
This includes: 1) Implementation of your project (code), and 2) A project report.
Be sure to read the grading rubric!!

Part 4: Project Presentation *Due Friday May 8*
Describe the project and show the results.

Upcoming Work

Submit all exercises before the extras by Wednesday 5/6 at 10:30pm.
Include 6P's for Problem 5. Include 4P's for any helper procedures you write.

To: csc-151-02-grader@grinnell.edu
Subject: CSC 151-02 Lab 5/4

Project due tomorrow! Read the grading rubric!

Presentations due Friday. See information on the project webpage and on previous days' notes.

Exam 4 (final exam) will go out on Friday.