**Blue Waters Petascale Semester Curriculum v1.0**

**Unit 2: Parallel Computing Concepts**

**Lesson 5: Parallel Algorithms 2**

**Instructor Guide**

*Developed by Beau Christ for the Shodor Education Foundation, Inc.*



*Except where otherwise noted, this work by The Shodor Education Foundation, Inc. is licensed under CC BY-SA 4.0. To view a copy of this license, visit*[*https://creativecommons.org/licenses/by-sa/4.0*](https://creativecommons.org/licenses/by-sa/4.0)

*Browse and search the full curriculum at*[*http://shodor.org/petascale/materials/semester-curriculum*](http://shodor.org/petascale/materials/semester-curriculum)

*We welcome your improvements! You can submit your proposed changes to this material and the rest of the curriculum in our GitHub repository at*[*https://github.com/shodor-education/petascale-semester-curriculum*](https://github.com/shodor-education/petascale-semester-curriculum)

*We want to hear from you! Please let us know your experiences using this material by sending email to* [*petascale@shodor.org*](mailto:petascale@shodor.org)

This lesson continues the basic ideas in the lesson **Parallel Algorithms I** toward the purpose of getting students to continue thinking about parallel design.

There is still no formal code in this lesson, and it is not recommended to dive into a programming language at this stage. The goal is to introduce simple problems and their sequential solutions, and then give students time to think about a parallel solution before revealing the answer.

These examples are examined in detail in the textbook **An Introduction to Parallel Programming** by Peter Pacheco (1st Edition), along with sequential implementations and several parallel implementations.

**Common Pitfalls for Students and Instructors**

A common pitfall is moving too quickly through this lesson. After presenting a sequential solution to a problem, it is recommended to pause to give students time to really think about how to parallelize before simply revealing the solution.

Otherwise, no other pitfalls should be expected.