Mergesort

CSSE 221
Fundamentals of Software Development Honors
Rose-Hulman Institute of Technology
Presentation by Cameron Spry and AJ Piergiovanni

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Fastest Sorts (All O(n log n))

- Quicksort
- **-> Mergesort <-**
- Timsort
- Heapsort
- Introsort

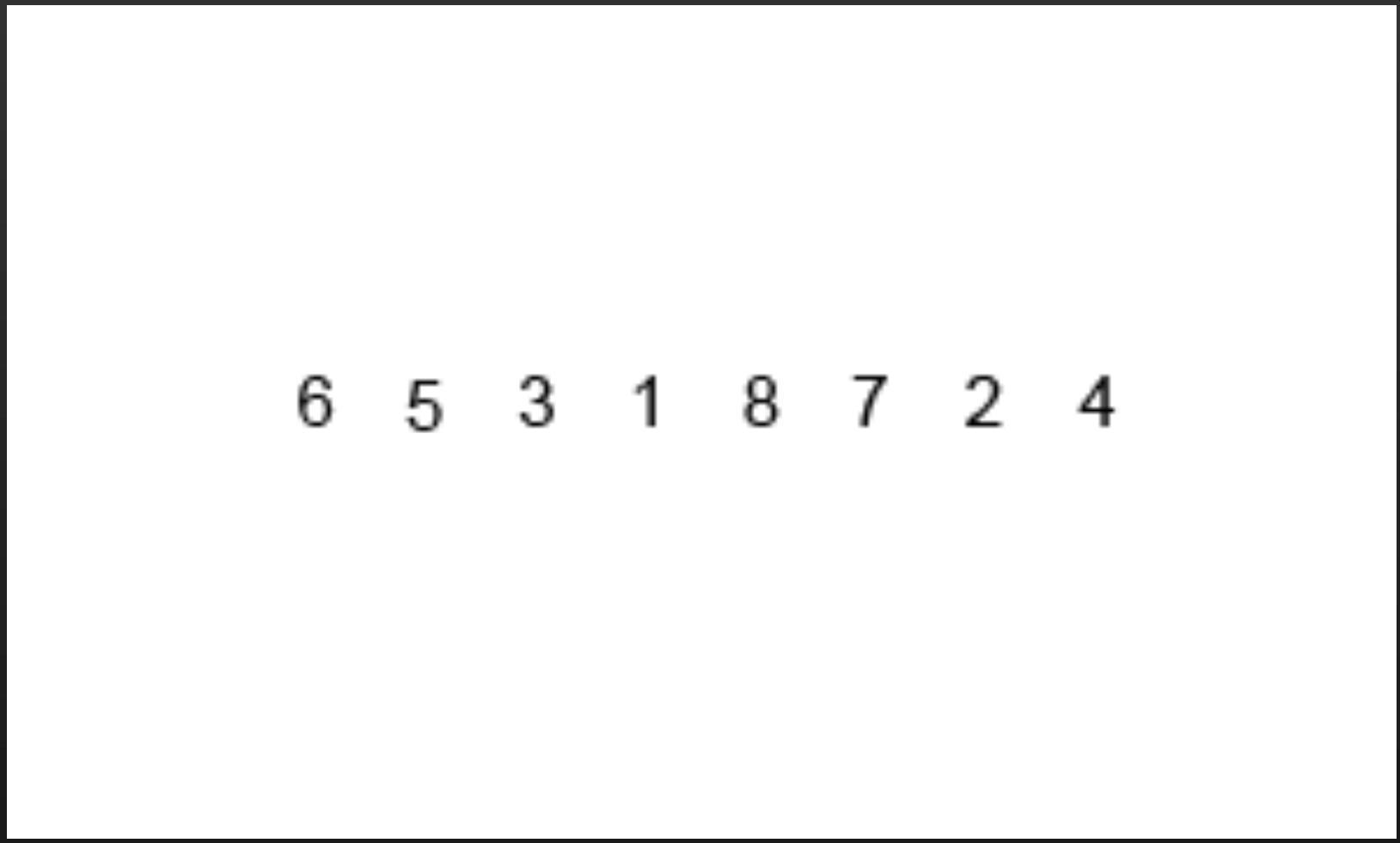
Basics of Mergesort

- A “divide and conquer” recursive algorithm for sorting
- Splits the list to be sorted into smaller, easier to sort lists
- Performance of O(n log n), so it’s fast
- Not typically an in-place sort
- The default sort in Perl and some implementations of Java

Mergesort Algorithm

- If the list is length 0 or 1, the list is sorted
- Divide the list into 2 lists, half the size
- Recursively apply Mergesort by splitting the lists then merging them together
- Merge the lists together to form 1 list

Mergesort (Wikipedia Animation)



Pros and Cons

- Fast
- Always O(n log n)
- Can be concurrent
- It is a stable sort
- Uses lots of memory
- Recursive (uses stack space)

Concurrency

- Mergesort can be optimized by using threads
- When recursively applying the Mergesort, a new Thread can be created to run each Mergesort
- Use fork and join to merge the lists into one sorted list
- This helps limit stack overflows because each thread gets its own stack

Fork and Join

- Fork runs code on a separate thread
  - It uses multiple cores to make you program fast
  - Uses a thread pool
  - Creates tasks that are executed by worker threads
- Forking works great with recursive functions
- More efficient that threads for divide-and-conquer algorithms
- Only in Java 7, but it can be added to Java 6 by including a .jar

Using Fork

- To use fork, you must extend the ForkJoinTask
  - Usually as a RecursiveTask or RecusiveAction
    - RecursiveTask can return a result
    - Recursive action cannot
- You also need a ForkJoinPool
  - The constructor takes a int for cores
  - .invoke(task); starts the process

Fork Join

- This works great with recursive functions because for each recursive call, you can create a new ForkJoinTask to do the work.
  - This speeds up the function because it runs on a separate core
  - It also helps limit Stack Overflow errors
  - ForkJoinTasks are like lightweight threads

Using Fork Join

- In a recursive function, after creating the tasks, there are 3 functions called
  - task1.fork(); starts task1 on a separate core
  - task2.fork(); starts the second task
  - task2.compute(); can also be used to run task2
  - task1.join(); waits until task1 is complete to continue
  - task2.join(); waits until task2 is complete to continue

More Information about Fork/ Join

- http://www.oracle.com/technetwork/articles/java/fork-join-422606.html
- http://www.ibm.com/developerworks/java/library/j-jtp11137/index.html
- http://www.cs.washington.edu/homes/djg/teachingMaterials/grossmanSPAC_forkJoinFramework.html
  - Dan Grossman's work has facilitated our introduction of forkJoin parallelism