Sorting Competition

Group 8: Ai and Elsa

7th place in the competition

ALGORITHM

- Quicksort
- Modified (median-of-3 partition)
- The two “if’s” somehow made it faster

```
public static void quickSort(int[][] array, int first, int last, Comparator c){
        if (first < last){
                int pivot = randomizedPartition(array, first, last, c);
                if (first < pivot - 1) quickSort(array, first, pivot - 1, c);
                if (last > pivot + 1) quickSort(array, pivot + 1, last, c);
        }
}
```

DATA

Array of arrays

- Second array length 3
  - First: x coordinate
  - Second: y coordinate
  - Third: time stamp

- Sorted using comparator
  - Sorts based on distance and time stamps

Efficiency

Worst case running time of quicksort

- $\Theta(n^2)$
    - Unbalanced partitioning
    - Completely sorted already

BUT… Picking a pivot using median-of-3 method

⇒ Lowers the odds of unbalanced partitioning at each level of recursion

⇒ Lowers the odds of having the worst case

⇒ More likely to be $\Theta(n \log n)$

Non-constant Memory

- Worst case O(n)
- Common case O(log n)

The quicksort algorithm uses recursion
⇒ At each recursive call a stack frame is allocated
    ⇒ The number of recursive calls
      (i.e., Depth of recursion)
        ⇒ Worst case: n times
        ⇒ Common case: log n times

Additional Things...

Stuff we tried:

- Heap
    - Tried it out of curiousity--quite bad
- Radix/Counting/Bucket
    - Useful for unique keys
    - Started each of these several times, did not get very far

Stuff we wanted to try:

- Mergesort
    - Extremely optimized
- Bucketsort
    - Get any version of this working to see how well it does