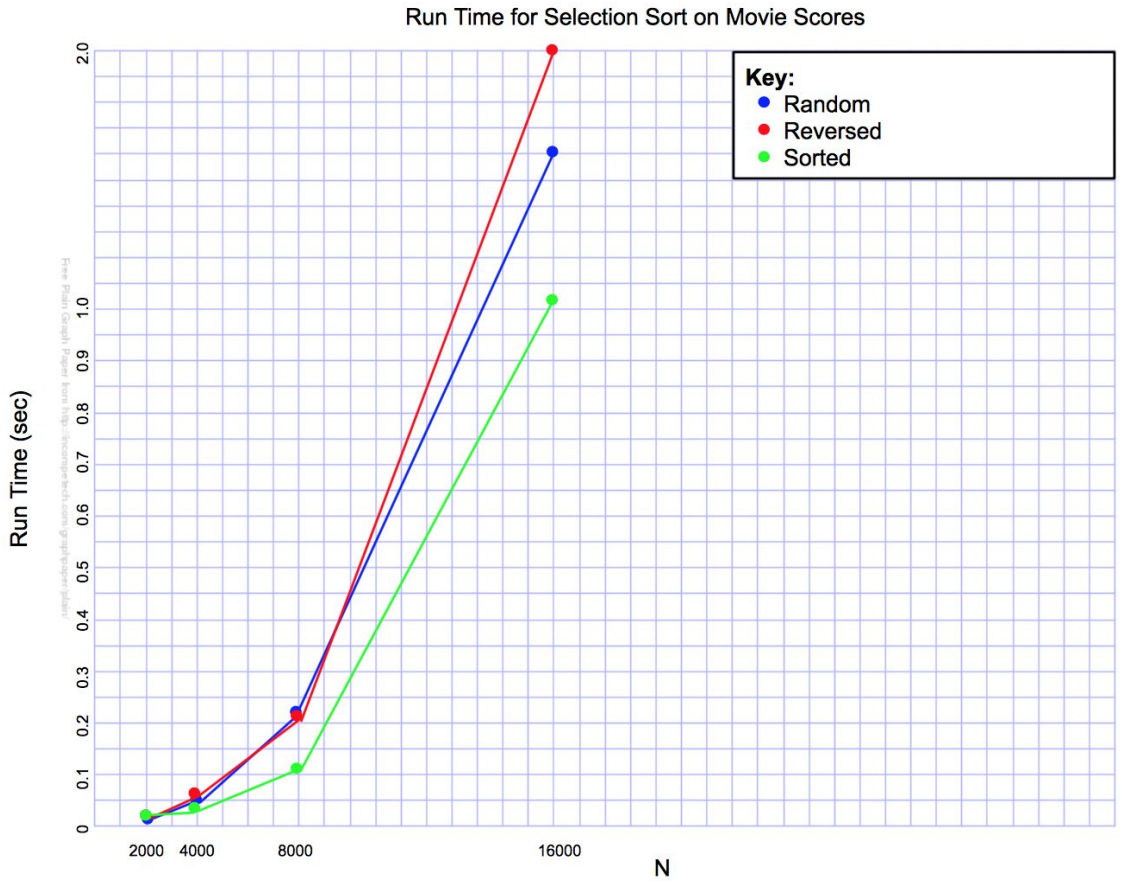
Sunwoo Ha
Introduction to Algorithms
September 21, 2017

Week 4 Workshop

1. See java files in branch.
2. **movie_title_score.random.csv:**
   - **2000**: 0.026, 0.02, 0.022 -> Average runtime is **0.023**
   - **4000**: 0.052, 0.049, 0.051 -> Average runtime is **0.051**
   - **8000**: 0.212, 0.225, 0.237 -> Average runtime is **0.225**
   - **16000**: 1.216, 1.888, 1.861 -> Average runtime is **1.655**

   movie_title_score.reversed.csv:
   - **2000**: 0.022, 0.023, 0.03 -> Average runtime is **0.025**
   - **4000**: 0.057, 0.066, 0.057 -> Average runtime is **0.06**
   - **8000**: 0.21, 0.212, 0.232 -> Average runtime is **0.218**
   - **16000**: 1.697, 2.737, 1.572 -> Average runtime is **2.002**

   movie_title_score.sorted.csv:
   - **2000**: 0.031, 0.023, 0.02 -> Average runtime is **0.0247**
   - **4000**: 0.041, 0.034, 0.037 -> Average runtime is **0.0373**
   - **8000**: 0.121, 0.137, 0.128 -> Average runtime is **0.129**
   - **16000**: 1.57, 0.938, 1.39 -> Average runtime is **1.299**



3. Using the doubling rule, when the input doubles, SelectionSort's runtime is approximately raised to the power of 2 of the last run time. Therefore, the growth of SelectionSort is quadratic.
4. In the SelectionSort code, there are two array accesses. When an array access occurs, the cost is ~ n. So, the cost model for this algorithm would be ~ n^2.
5. Looking at the code for SelectionSort, I do not think that the order of the inputs make much of a difference on efficiency of the algorithm because the **exch** method is called outside of one of the for loops and the **less** method is inside of an if statement. Due to the location of these methods, the algorithm will output a sorted list in about the same time as one another, making SelectionSort insensitive to input.