Daniel Vu

Project 1 – Report

I. Sorting implement explanation

Bubble Sort: I followed this pseudocode from the lecture :

```
public static void bubbleSort(int[] a) {
   for (int i = 0; i < a.length; i++) {
      for (int j = 1; j < a.length - i; j++) {
         // swap adjacent out-of-order elements
         if (a[j-1] > a[j]) {
            swap(a, j-1, j);
         }
      }
   }
}
```

Insertion Sort: I followed this pseudocode from the lecture :

```
public static void insertionSort(int[] a) {
   for (int i = 1; i < a.length; i++) {
      int temp = a[i];
      // slide elements down to make room for a[i]
      int j = i;
      while (j > 0 && a[j - 1] > temp) {
         a[j] = a[j - 1];
         j--;
      }
      a[j] = temp;
   }
```

}

Spin-the-bottle Sort: I followed this pseudocode from the lecture

while A is not sorted **do**
  for $i = 1$ to n **do**
    Choose s uniformly and independently at random from $\{1, 2, \ldots, i-1, i+1, \ldots, n\}$.
    if $(i < s$ **and** $A[i] > A[s])$ **or** $(i > s$ **and** $A[i] < A[s])$ **then**
      Swap $A[i]$ and $A[s]$.

To get uniformly and indepently number, I use use mt19937 for random generation of the numbers in the algorithm .

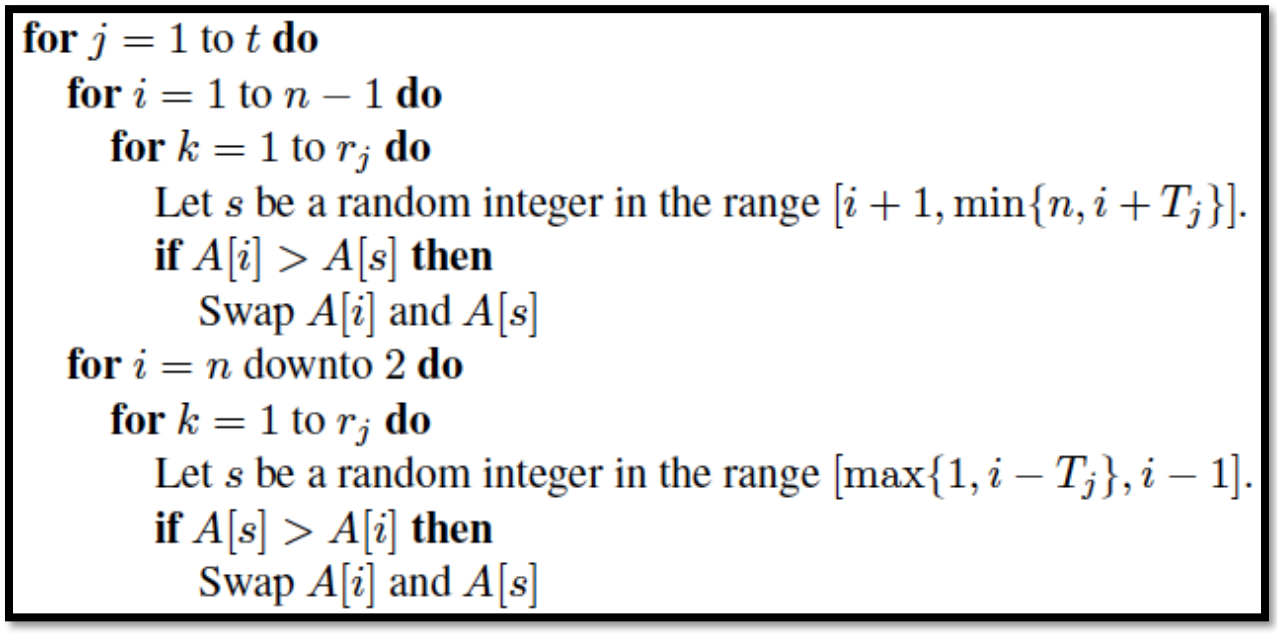
Shell sort: I followed this pseudocode from the lecture

```
public static void shellSort(int[] a) {
   for (int gap = a.length / 2; gap > 0; gap /= 2) {
      for (int i = gap; i < a.length; i++) {
         // slide element i back by gap indexes
         // until it's "in order"
         int temp = a[i];
         int j = i;
         while (j >= gap && temp < a[j - gap]) {
            a[j] = a[j – gap];
            j -= gap;
         }
         a[j] = temp;
      }
   }
}
```

To create 2 gap sequences for testing, I use different formula for calculate sequences. These methods are shown in Wikipedia with BigO is O^(4/3)

I. gap[i]=4^(i+1)+3*2^i
II. gap[i] = 8*2^i -6 * 2^((i+1)/2) +1

Annealing sort: I followed this pseudocode from the lecture

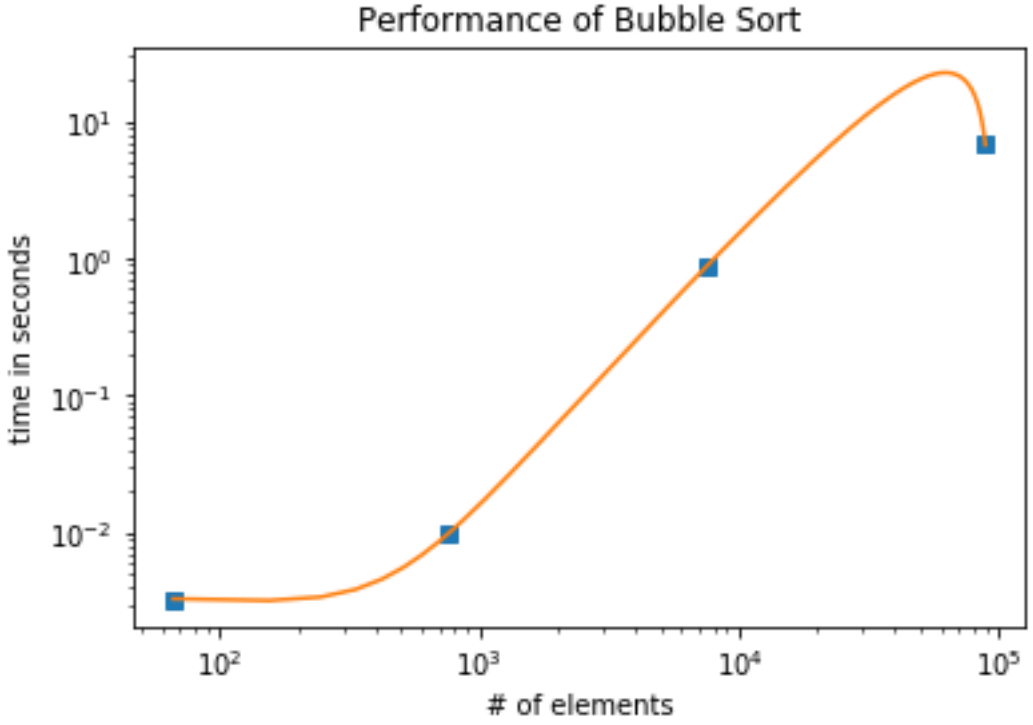


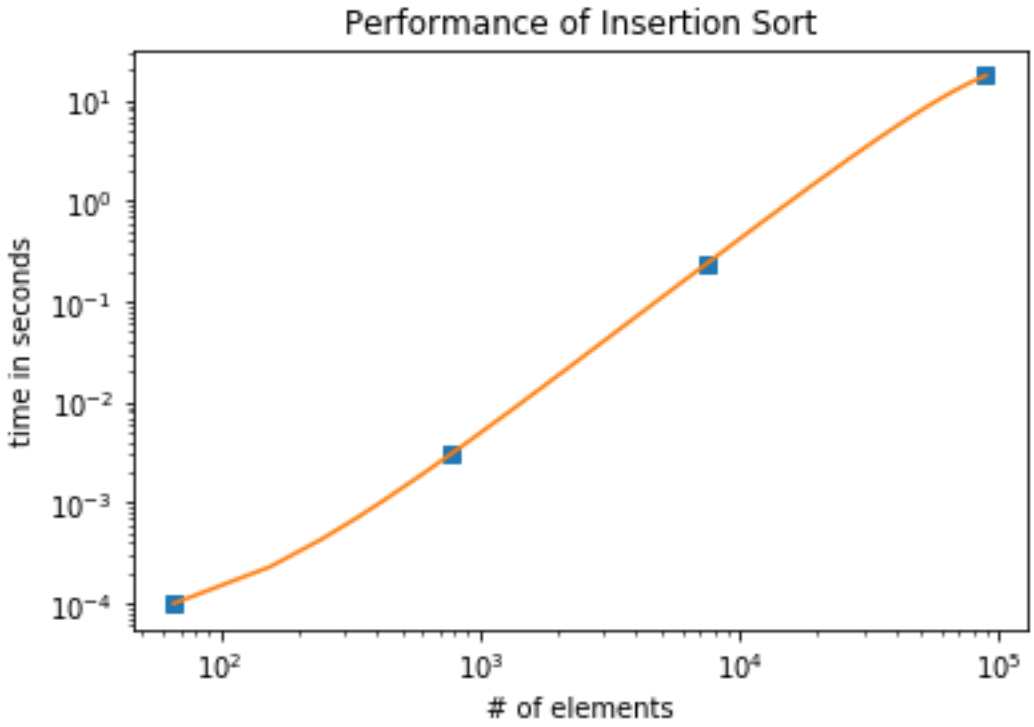
And to create different temp-rep sequence, I also use 3-phase method in the lecture but use mt19937 to get the first and last element in each phase.

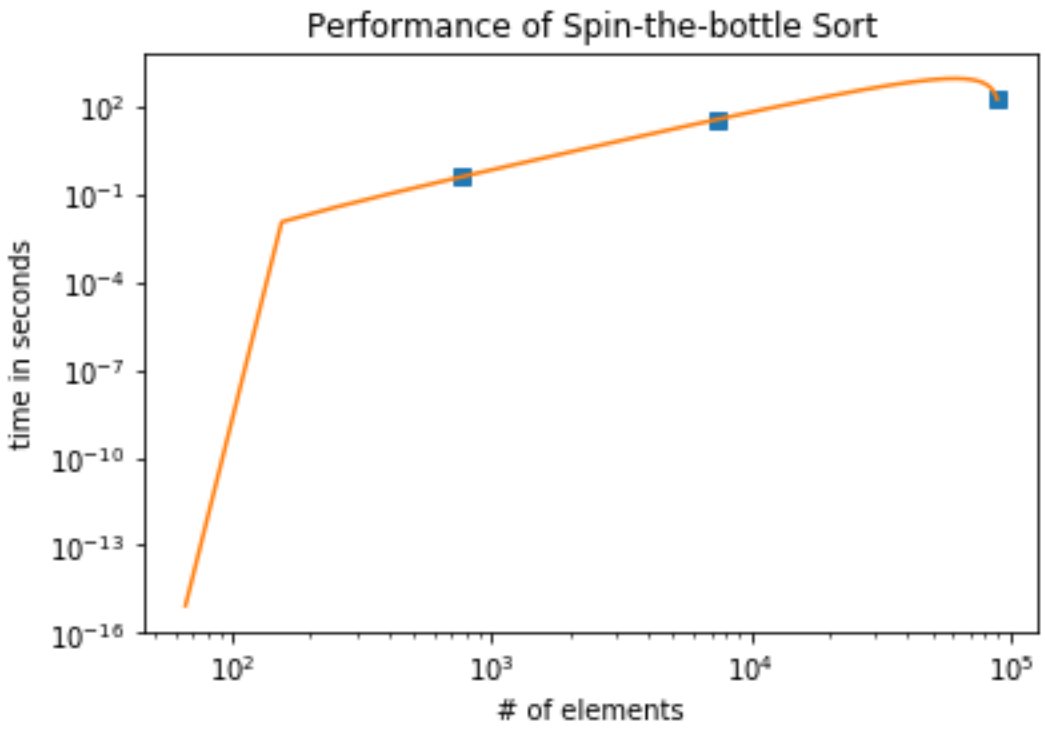
Uniformly distributed and almost-sorted shuffle: I use mt19937 to randomly choose indexes to swap elements in vectors.

III. Sorting Algorithm Running Time Comparation

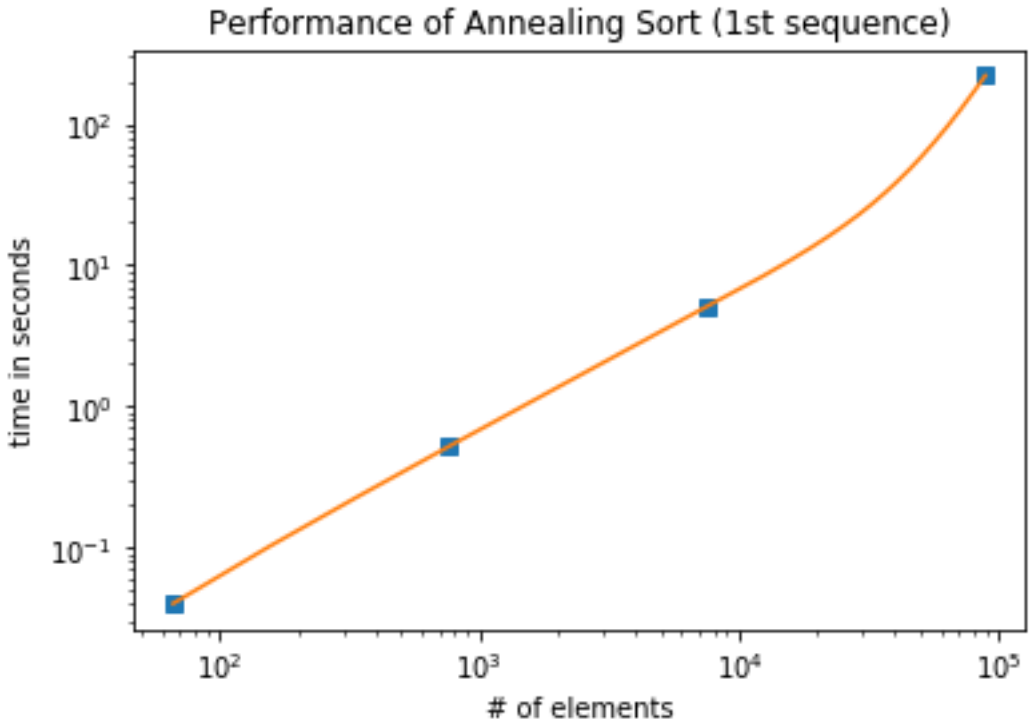
a. Uniformly distributed

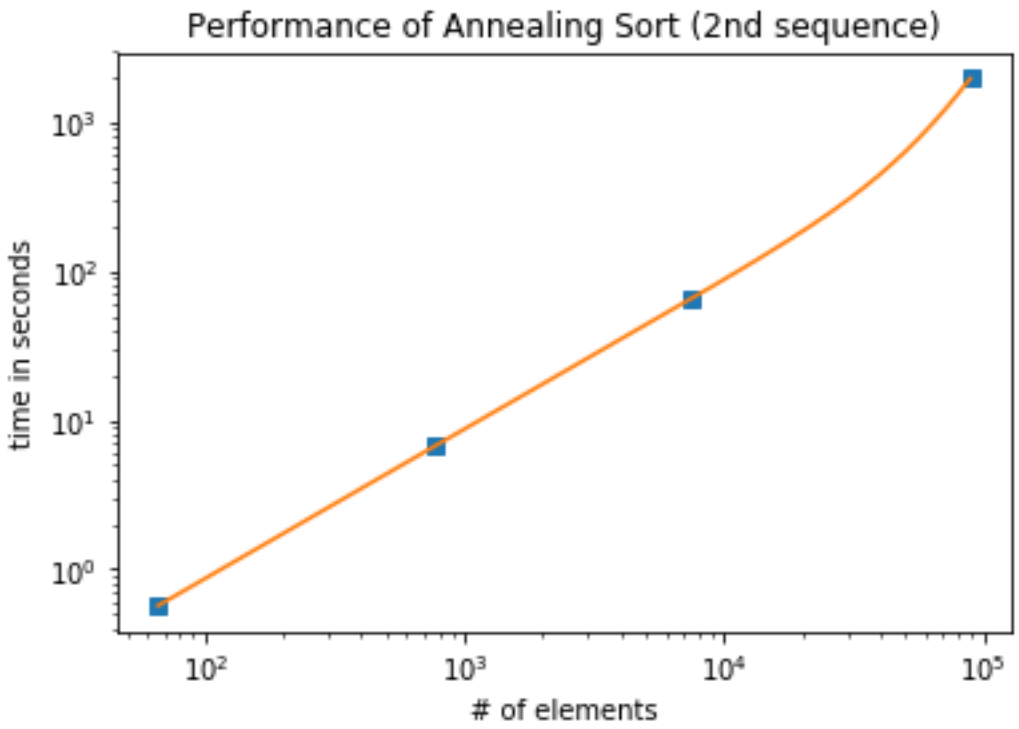






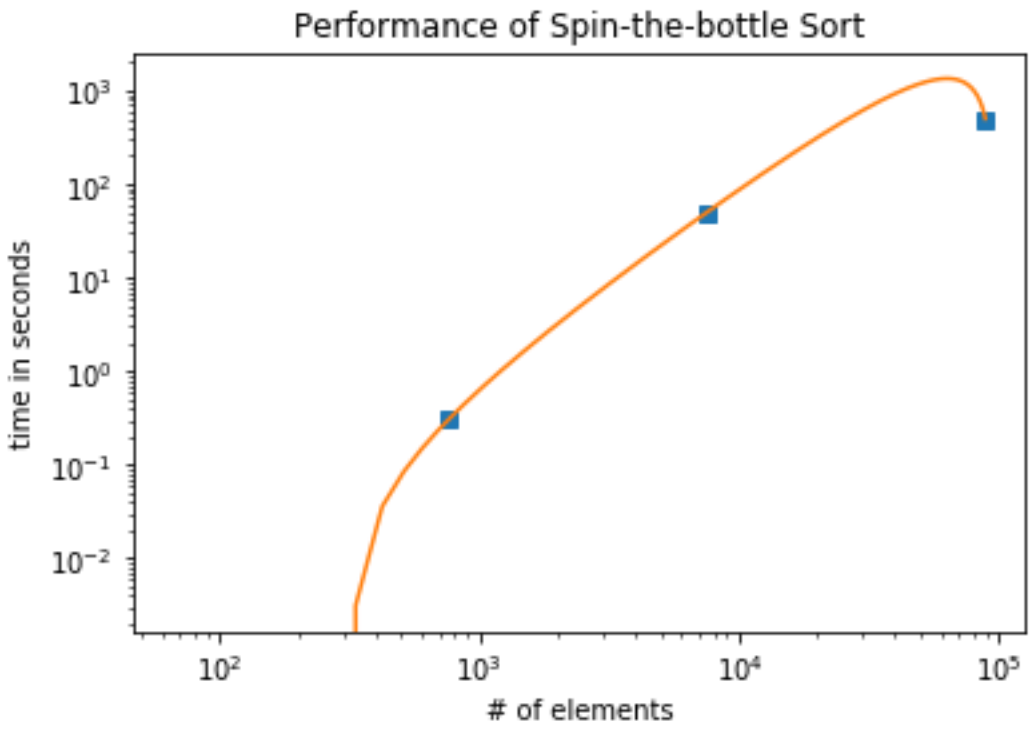
For Shell Sort, since all the time is almost 0 so I cannot create a graph using log-log scale.

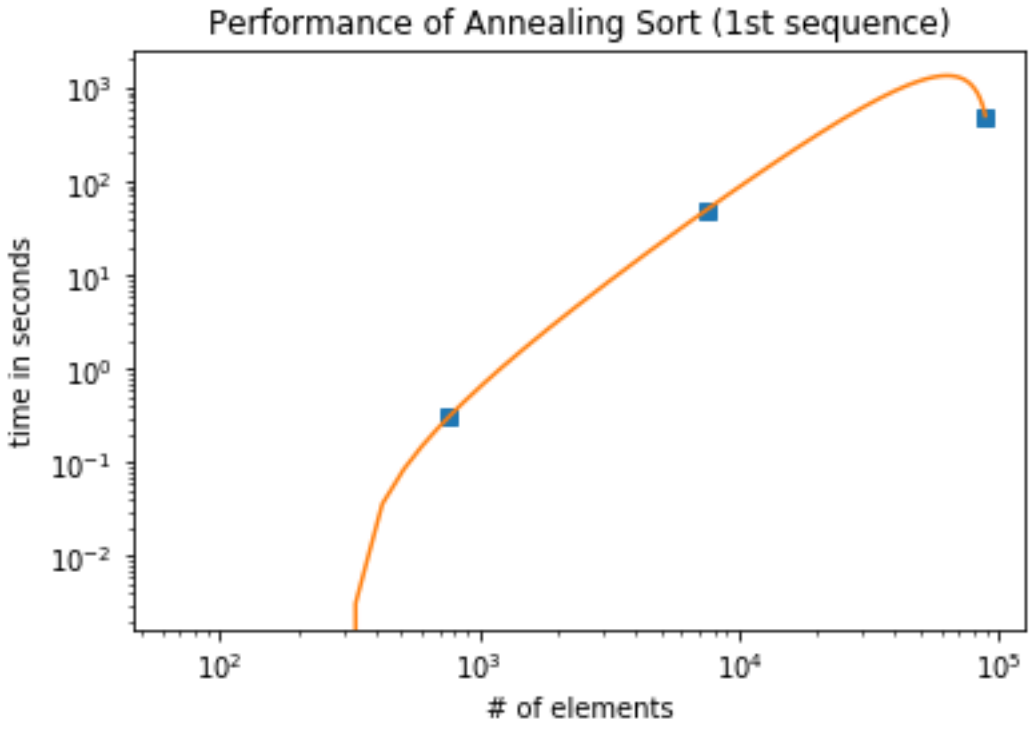


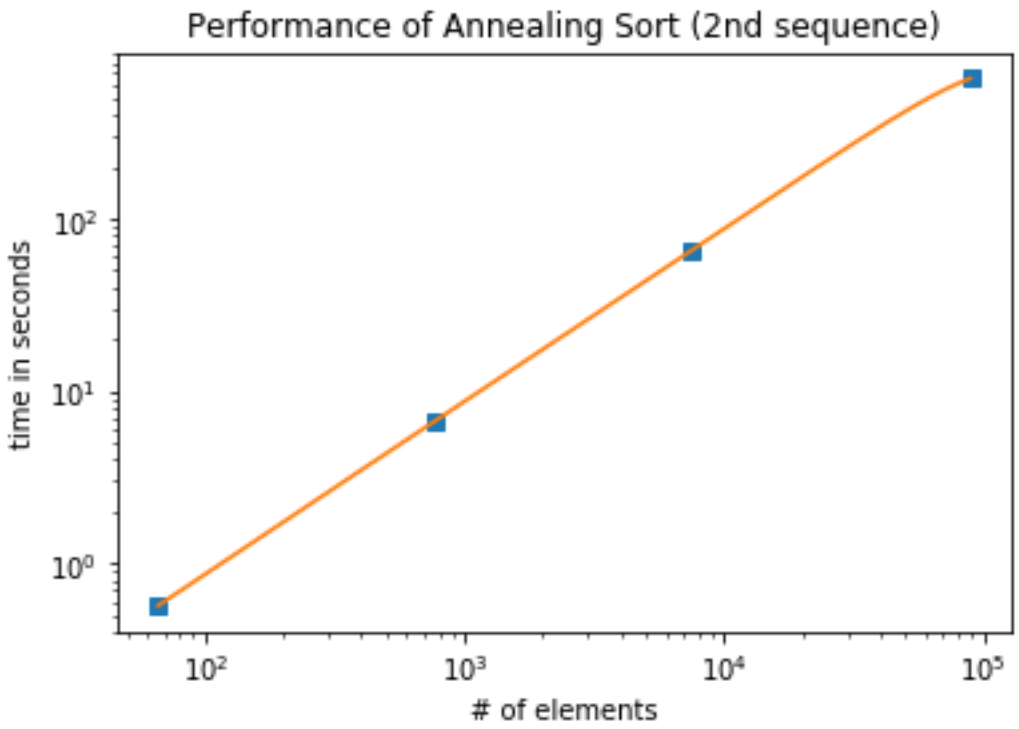


b. Almost-sorted distributed

For almost-sorted distributed, bubble sort, insertion and shell sort both got almost 0 for running time


Performance of Spin-the-bottle Sort
time in seconds
# of elements


Performance of Annealing Sort (1st sequence)
time in seconds
# of elements



IV. Conclusion

I think shell sort is the best sorting algorithm in both distributions. Annealing sort is worst since the probability of unsorted is around 30%.