Qiao jia Lu
A13638993
11/22/17

hw7 Short Answers

Q1

```
Public void sort(Array) {

    int seperator = 0;
    boolean done = false;

    Int i = 0;
    Int j = Array.length - 1;

    While (!done) {

       //iterate till is not negative
       While(Array[i] <= separator) {
            ++i;
       }

      //iterator till not positive
      While(Array[j] >= separator) {
           --j;
      }

      //stop if index of right is the same as left
      if ( i >= j) {
        done = true;
      }

     //swap elements
     else {
        int temp = Array[i];
        Array[i] = Array[j];
        Array[j] = temp;

       //increase counter
       ++i;
       --j;
     }
    }
}
```

Q2

Selection Sort
Best Case: O(n^2) Worst Case: O(n^2)

Insertion Sort
Best Case: O(n) Worst Case: O(n^2)

According to the Zybook, Insertion algorithm consist of a for loop and a while loop. Such that the for loop iterate over elements(which always run O(n)) and the while loop does the comparison. In the best case, the insertion sort will have a run time of O(n). an example of an array that will cause this run time would be a sorted one or semi sorted one in descending order. EX: 1 2 3 4 5…. The reason that cause this run time is that on each ith iteration, it only have to compare the n element to the n – 1. Since n – 1 is smaller than n it doesn't have to compare any beyond that. Therefore the inner while loop runs O(1) and thus best case is O(n).

On the other hand, the worst case will have a run time of O(n^2). An example of an array will cause this is a unsorted array. EX: 8 2 4 2 4 5… The reason that cause this run time is that on each ith iteration, each n element will have to compare to n – 1 elements. Therefore the inner while loop will run O(n) and thus worse case is O(n^2).

Merge sort
Best Case: O(nlog(n)) Worse Case: O(nlog(n))

Quick sort
Best Case: O(nlog(n)) Worse Case: O(n^2)

Quick sort is an algorithm that sort a list by splitting into partitions, such that in each partition the left side is smaller than pivot point and the right side is larger than the pivot point. A good pivot point is the key to this algorithm's success. A good pivot point will cause a best case run time of O(nlog(n)). An example of a good quick sort would be an array with elements [14, 4, 9, 12, 15, 8, 19, 2] with 12 as the pivot point at start. From this example, quick sort will split its partition size of 4 at start and then choose a new element as pivot point and splitting it again. But each split from this array, the partitions will have equal size. Splitting the partition will have a run time of nlog(n) times and sorting the elements will have a run time of O(n). Thus conclude a best case of O(nlog(n)).

On the other hand, the worst case for quick sort will be picking a minimum element as your pivot point. An example array would be [4 3 2 1 8]. Using the minimum element in this array as pivot point will cause the left side with one element and the right side with the rest of the elements in the array. This will cause quick sort to split partition n times and then sort the elements n times. Which results in the worse case run times of O(n^2).