AVIEL RESNICK

26.5 Problem Set

Page 445 1-5

1. Why is a linear search called “linear”?
   1. A linear search is called linear, because it iterates over every item, leading to a runtime efficiency of O(n), aka linear time.
2. Write a linear search method that searches an array of objects for a target object.

|  |
| --- |
| public static int linearSearch(Object[] array, targetObject) {  int targetIndex = -1;    for(int i = 0; i < array.length; i++){  if(array[i].compareTo(targetObject))  targetIndex = i;  }    return targetIndex; } |

1. Which elements are examined during a binary search of the array 34 56 78 85 99 for the target element 100?
   1. The examined elements are 78, 85, and 99.
2. Jack advises Jill of a modification to linear search that improves its performance when the array is sorted: If the target element is less than the current element, the target cannot be in the array. Modify the linear search method for integers to accomplish this.

|  |
| --- |
| public static int modifiedLinearSearch(int[] array, targetInt) {  int targetIndex = -1;   for(int i = 0; i < array.length; i++){  if(targetInt < array[i]) {  return(-1);  }   if(array[i].equals(targetInt))  targetIndex = i;  }   return targetIndex; } |

1. Describe what the following code segment does:

|  |
| --- |
| boolean inOrder = true;  for (int i = 0; i < a.length - 1; i++)  if (a[i] > a[i + 1]){   inOrder = false;  break; } |

The code segment above checks if the elements of an array are in ascending order, and returns false if not.

Page 452 1-4

1. Draw a diagram that shows the contents of the array 8 7 6 5 4 after each number is moved in a selection sort.
   1. 8-7-6-5-4
   2. 4-8-7-6-5
   3. 4-5-8-7-6
   4. 4-5-6-8-7
   5. 4-5-6-7-8
2. Draw a diagram that shows the contents of the array 8 7 6 5 4 after each number is moved in a bubble sort, until the 8 arrives at the end of the array.
   1. 8-7-6-5-4 |
   2. -------------
   3. 7-8-6-5-4 |
   4. 7-6-8-5-4 | -- First Iteration
   5. 7-6-5-8-4 |
   6. 7-6-5-4-8 |
   7. -------------
   8. 6-7-5-4-8 |
   9. 6-5-7-4-8 | -- Second Iteration
   10. 6-5-4-7-8 |
   11. -------------
   12. 5-6-4-7-8 |
   13. 5-4-6-7-8 | -- Third Iteration
   14. -------------
   15. 4-5-6-7-8 | -- Final Iteration
3. Describe the behavior of the selection sort, bubble sort, and insertion sort with an array that is already sorted. How many exchanges are made in each sort for an array of size n?
   1. Selection sort still traverses the entire array, but does not swap any elements.
   2. Bubble sort does the first iteration, and stops, as no changes were made.
   3. Insertion Sort similarly passes through and makes no swaps.
4. Modify the bubble sort method so that it sorts an array of objects.

|  |
| --- |
| public static void bubbleSort(Object[] array){  int k = 0;  boolean swapOccured = true;   while ((k < array.length() - 1) && swapOccured){  swapOccured = false;  k++;    for (int j = 0; j < array.length() - k; j++)   if (array[j].compareTo(array[j + 1]) > 0) {  int temp = array[j];  array[j] = array[j + 1];  array[j + 1] = temp;    swapOccured = true;  }  } } |