**Big O - Best, Average, Worst Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Searches** | **Best Case** | **Average Case** | **Worst Case** |
| Linear or Sequential Search | O(1) - 1st element | O(n) - middle element | O(n) - last element |
| Binary Search (sorted data) | O(1) - 1st element | O(log n) | O(log n) |
| Search Binary Search Tree | O(1) - root | O(log n) | O(n) - tree is a linear |
| **Sorts** | **Best Case** | **Average Case** | **Worst Case** |
| Selection Sort | O(n2) | O(n2) | O(n2) |
| Insertion Sort | O(n) - already sorted | O(n2) | O(n2) |
| MergeSort | O(n log n) | O(n log n) | O(n log n) |
| QuickSort | O(n log n) | O(n log n) | O(n2) - pivot consistently not close to median |
| Arrays.sort(*primitive*[])  uses modified QuickSort | O(n log n) | O(n log n) | O(n2) |
| Arrays.sort(*Object*[])  uses modified MergeSort | O(n log n) | O(n log n) | O(n log n) |
| Collections.sort()  uses modified MergeSort | O(n log n) | O(n log n) | O(n log n) |

**Java Regional/State Review #5 – Sorts, Searches and Big O**

1. Which search efficiency has the worst time efficiency?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

2. Which search efficiency has the best time efficiency?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

3. An algorithm Bug operates with order O(n). If Bug processes 1000 items in .05 seconds, how many seconds will it take to process n similar items if n equals:

a. 3000 \_\_\_\_\_\_\_ seconds

b. 5000 \_\_\_\_\_\_\_ seconds

c. 2000 \_\_\_\_\_\_\_ seconds

d. 4000 \_\_\_\_\_\_\_ seconds

4. An algorithm Bug operates with order O(n2). If Bug processes 2000 items in .05 seconds, how many seconds will it take to process n similar items if n equals:

a. 3000 \_\_\_\_\_\_\_ seconds

b. 5000 \_\_\_\_\_\_\_ seconds

c. 1000 \_\_\_\_\_\_\_ seconds

d. 4000 \_\_\_\_\_\_ seconds

e. 10000 \_\_\_\_\_\_ seconds

f. 800 \_\_\_\_\_\_ seconds

5. How many comparison operations will be made to find the value 22 using a binary search if the array contains the following elements.

a. 2 5 8 9 12 13 14 15 16 17 18 20 22 23 24 25 \_\_\_\_\_\_\_\_\_

b. 12 13 14 15 17 18 19 21 22 25 26 27 28 29 33 35 36 37 38 39 \_\_\_\_\_\_\_\_\_

c. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 \_\_\_\_\_\_\_\_\_

d. 25 26 27 28 29 30 35 37 39 41 43 44 45 47 56 64 70 72 73 \_\_\_\_\_\_\_\_\_

6. Using the arrays in #5, what is the value of *index* at the end of a binary search for the element 25 if *midpoint = Array[index]*?

a. \_\_\_\_\_\_\_ b. \_\_\_\_\_\_\_ c. \_\_\_\_\_\_\_ d. \_\_\_\_\_\_\_

7. A search item exists in a sorted array of 100 elements. If \_\_\_\_\_\_\_\_\_

a linear search is used to find the item, what is the maximum

number of comparisons that will be made.

8. A search item exists in an unsorted array of 100 elements. If \_\_\_\_\_\_\_\_\_

a linear search is used to find the item, what is the maximum

number of comparisons that will be made.

9. A search item exists in a sorted array of 100 elements. If \_\_\_\_\_\_\_\_\_

a binary search is used to find the item, what is the maximum

number of comparisons that will be made.

10. What is the Order of the Insertion Sort if the data is already sorted?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

11. What is the Order of the Merge Sort if the data is already sorted?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

12. What is the Order of the Selection Sort if the data is already sorted?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

13. Find the time complexity of the algorithms shown.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm a** | | **Algorithm b** | | **Algorithm c** | | **Algorithm d** | | **Algorithm e** | |
| **Data** | **Time** | **Data** | **Time** | **Data** | **Time** | **Data** | **Time** | **Data** | **Time** |
| 1000 | .003 | 1000 | .0042 | 1000 | .002 | 1000 | .003 | 1000 | .0030 |
| 2000 | .012 | 2000 | .0042 | 2000 | .004 | 2000 | .0033 | 2000 | .0066 |
| 4000 | .048 | 4000 | .0041 | 4000 | .008 | 4000 | .0036 | 4000 | .0144 |
| 8000 | .192 | 8000 | .0040 | 8000 | .016 | 8000 | .0039 | 8000 | .3120 |
| Big-O |  |  | |  | |  | |  | |

14-22. How many actual times will the method Z be called if n = 10?

for(i=0;i<n;i++)

for(j=0;j<i;j++)

Z(i\*j);

for(i=0;i<n;i++)

for(j=0;j<n;j++)

Z(i\*j);

14. 15. 16.

for(i=0;i<n;i++)

for(j=0;j<n-j;j++)

Z(i\*j);

for(i=0;i<n;i++)

for(j=0;j<n-1;j++)

Z(i\*j);

17. 18. 19.

for(i=0;i<n;i++)

for(j=0;j<n+i;j++)

Z(i\*j);

for(i=0;i<n;i++)

for(j=i;j<n;j++)

Z(i\*j);

for(i=0;i<n-1;i++)

for(j=i;j<n;j++)

Z(i\*j);

for(i=0;i<n-i;i++)

for(j=i;j<n;j++)

Z(i\*j);

for(i=0;i<n;i++)

for(j=0;j<n-i;j++)

Z(i\*j);

20. 21. 22.

23. Which code should be placed in

// This code is supposed to locate an

// item in array A, if it exists.

int[] A = new int[100];

int Num;

boolean Found = false;

int i = 0;

while (\_\_\_\_\_\_\_\_blank 1 \_\_\_\_\_\_\_\_\_)

{

if (A[i] == Num)

Found = true;

else

i++;

}

if (\_\_\_ blank 2 \_\_\_)

System.out.println(Num+" is in A");

else

System.out.println(Num+" is not in A");

}

blank 1 to accomplish this purpose?

1. i < A.length && Found
2. i < A.length && !Found
3. i < A.length-1 && Found
4. i < A.length-1 && !Found
5. i < A.length || !Found

24. Which code should be placed in

blank 2 to display whether or not

the item was indeed in the list?

1. Found==false
2. Found==Num
3. !Found
4. Found
5. None of these

25. What is the value returned by Z?

public int Z(int[] W)

{

int i, t=0;

for (i=W.length-1; i>0; i--)

if (W[i] > W[0])

t++;

return t;

}

1. the number of elements of Z

less than the first element?

b. the number of elements of Z

greater than the first element?

c. the first element of Z

d. the largest element in Z

e. None of these

26. What is the Order of the Quick Sort if the data is already sorted?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

27. What is the Order of the Quick Sort if the data is random?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

28. What is the Order of the Insertion Sort if the data is random?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

29. How many comparisons would need to be made to sort an array of 100 items using a "Smart" Bubble Sort in the best case scenario?

A. 1000 B. 8100 C. 99 D. 500 E. None of these

30. Each element of a 100 element array is 0. What would be the execution requirement of a Quick Sort if the pivot element is the first element in the array?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

31. Each element of a 100 element array is 0. What would be the execution requirement of an Insertion Sort?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

32. Each element of a 100 element array is 0. What would be the execution requirement of a Merge Sort?

A. O(n) B. O(n2) C. O(log2 n) D. O(n log2 n) E. O(2n)

33. Which is true about a Quick Sort?

* + 1. A recursive version executes faster than a non-recursive version
    2. A recursive version has fewer lines than a non-recursive version
    3. A recursive version takes less run-time space than a non-recursive version

A. I only B. II only C. III only D. I and III only E. None of these

public Object S(List list,

Comparable item)

//list is an ArrayList that is

// sorted in ascending order.

//Method S returns a reference to the

// element in list with value item or

// null if the item is not in the

// list

{

int left=0, right=list.size()-1;

int mid;

while (left <= right)

{

middle = (left + right) / 2;

if (\_\_\_\_blank 1\_\_\_\_\_)

return list.get(mid);

else if (\_\_\_\_\_blank 2\_\_\_\_\_)

left = mid + 1;

else

right = mid – 1;

}

return \_\_\_\_\_blank 3\_\_\_\_\_;

}

34. Which of the following

fills blank 1 correctly?

a. mid == item

b. list[mid] == item

c. list[mid].equals(item)

d. list.get(mid) == item

e. list.get(mid).equals(item)

35. Which of the following

fills blank 2 correctly?

a. mid > item

b. list[mid] > item

c. list.get(mid) > item

d. ! list.get(mid).equals(item)

e. list.get(mid).compareTo(item) < 0

36. Which of the following

fills blank 3 correctly?  
  
 a. null b. NULL

c. 0 d. new Object()

**For questions 37-38 Use the following answer choices:**  
  
 **A. O(1) B. O(n) C. O(log2 n) D. O(n log2 n) E. None of these**

1. What is the Order of each of the following **java.util.ArrayList** methods?
2. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(int, java.lang.Object))(int index, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) element) – average case
3. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(int, java.lang.Object))(int index, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) element) – worst case
4. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
5. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worst case
6. **\_\_\_\_\_** [**get**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#get(int))(int index)
7. **\_\_\_\_\_** [**indexOf**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractList.html#indexOf(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o)
8. **\_\_\_\_\_** [**iterator**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractList.html#iterator())()
9. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#remove(int))(int index)
10. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o)
11. **\_\_\_\_\_** [**set**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#set(int, java.lang.Object))(int index, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) element)
12. **\_\_\_\_\_** [**size**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#size())()
13. **\_\_\_\_\_** [**toString**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#toString())() – assume each member's toString() requires O(1) time.
14. What is the Order of each of the following **java.util.LinkedList** methods?
15. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(int, java.lang.Object))(int index, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) element) – average case
16. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(int, java.lang.Object))(int index, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) element) – worst case
17. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case – average case
18. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case – worst case
19. **\_\_\_\_\_** [**addFirst**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/LinkedList.html#addFirst(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o)
20. **\_\_\_\_\_** [**addLast**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/LinkedList.html#addLast(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o)
21. **\_\_\_\_\_** [**get**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#get(int))(int index)
22. **\_\_\_\_\_** [**getFirst**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/LinkedList.html#getFirst())()
23. **\_\_\_\_\_** [**getLast**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/LinkedList.html#getLast())()
24. **\_\_\_\_\_** [**indexOf**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractList.html#indexOf(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o)
25. **\_\_\_\_\_** [**iterator**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractList.html#iterator())()
26. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#remove(int))(int index)
27. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o)
28. **\_\_\_\_\_** [**removeFirst**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/LinkedList.html#removeFirst())()
29. **\_\_\_\_\_** [**removeLast**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/LinkedList.html#removeLast())()
30. **\_\_\_\_\_** [**set**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#set(int, java.lang.Object))(int index, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) element)
31. **\_\_\_\_\_** [**size**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#size())()
32. **\_\_\_\_\_** [**toString**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#toString())() – assume each member's toString() requires O(1) time.

**For questions 39-41 Use the following answer choices:**  
  
 **A. O(1) B. O(n) C. O(log2 n) D. O(n log2 n) E. None of these**

1. What is the Order of each of the following **java.util.HashSet** methods?
2. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
3. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worst case
4. **\_\_\_\_\_** [**contains**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashSet.html#contains(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
5. **\_\_\_\_\_** [**contains**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashSet.html#contains(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worse case
6. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
7. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worse case
8. **\_\_\_\_\_** [**size**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#size())()
9. **\_\_\_\_\_** [**toString**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#toString())() – assume each member's toString() requires O(1) time.
10. What is the Order of each of the following **java.util.TreeSet** methods?
11. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
12. **\_\_\_\_\_** [**add**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/ArrayList.html#add(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worst case
13. **\_\_\_\_\_** [**contains**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashSet.html#contains(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
14. **\_\_\_\_\_** [**contains**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashSet.html#contains(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worst case
15. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – average case
16. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) o) – worse case
17. **\_\_\_\_\_** [**size**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#size())()
18. **\_\_\_\_\_** [**toString**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#toString())() – assume each member's toString() requires O(1) time.
19. What is the Order of each of the following **java.util.HashMap** methods?
20. **\_\_\_\_\_** [**containsKey**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#containsKey(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)– average case
21. **\_\_\_\_\_** [**containsKey**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#containsKey(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)– worst case
22. \_\_\_\_\_ [**containsValue**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#containsValue(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) value)
23. **\_\_\_\_\_** [**entrySet**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#entrySet())()
24. **\_\_\_\_\_** [**get**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#get(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)
25. **\_\_\_\_\_** [**keySet**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#keySet())()
26. **\_\_\_\_\_** [**put**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#put(java.lang.Object, java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) value)
27. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)
28. **\_\_\_\_\_** [**size**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#size())()
29. **\_\_\_\_\_** [**toString**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#toString())() – assume each member's toString() requires O(1) time.
30. **\_\_\_\_\_** [**values**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#values())()

**For questions 42-43 Use the following answer choices:**  
 **A. O(1) B. O(n) C. O(log2 n) D. O(n log2 n) E. None of these**

1. What is the Order of each of the following **java.util.TreeMap** methods?
2. **\_\_\_\_\_** [**containsKey**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#containsKey(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)– average case
3. **\_\_\_\_\_** [**containsKey**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#containsKey(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)– worst case
4. \_\_\_\_\_ [**containsValue**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#containsValue(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) value)
5. **\_\_\_\_\_** [**entrySet**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#entrySet())()
6. **\_\_\_\_\_** [**get**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#get(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)
7. **\_\_\_\_\_** [**keySet**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#keySet())()
8. **\_\_\_\_\_** [**put**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#put(java.lang.Object, java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key, [Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) value)
9. **\_\_\_\_\_** [**remove**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/lang/Object.html) key)
10. **\_\_\_\_\_** [**size**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#size())()
11. **\_\_\_\_\_** [**toString**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/AbstractCollection.html#toString())() – assume each member's toString() requires O(1) time.
12. **\_\_\_\_\_** [**values**](mk:@MSITStore:C:\Documents%20and%20Settings\gmart039.000\Desktop\jdk142.chm::/jdk142/api/java/util/HashMap.html#values())()
13. What is the worst case Order of each of the following methods?
14. **\_\_\_\_\_** Arrays.sort(int[] a)
15. **\_\_\_\_\_** Arrays.sort(Object[] o)
16. **\_\_\_\_\_** Collections.sort()
17. What is the best case Order of each of the following methods?
18. **\_\_\_\_\_** Arrays.sort(double[] a)
19. **\_\_\_\_\_** Arrays.sort(Object[] o)
20. **\_\_\_\_\_** Collections.sort()

Answers - Review #5 – Sorts, Searches, and Big-O

1. E

2. C

3 a. .15

b. .25

c. .10

d. .2

4. a. .1125

b. .3125

c. .0125

d. .2

e. 1.25

1. .008

5. a. 4

b. 5

c. 5

d. 5

6. a. 15

b. 8

c. 21

d. 0

7. 100

8. 100

9. 7

10. A

11. D

12. B

13. a. n2

b. 1

c. n

d. log2n

e. n log2n

14. 100

15. 55

16. 45

17. 90

18. 54

19. 50

20. 55

21. 40

22. 145

23. B

24. D

25. B

26. B

27. D

28. B

29. C

30. B

31. A

32. D

33. B

34. E

35. E

36. A

37. a. B

1. B
2. A
3. B
4. A
5. B
6. A
7. B
8. B
9. A
10. A
11. B

38. a. B

1. B
2. A
3. A
4. A
5. A
6. B
7. A
8. A
9. B
10. A
11. B
12. B
13. A
14. A
15. B
16. A
17. B

39. a. A

1. B
2. A
3. B
4. A
5. B
6. A
7. B

40. a. C

1. C
2. C
3. C
4. C
5. C
6. A
7. B

41. a. A

1. B
2. B
3. B
4. A
5. B
6. A
7. A
8. A
9. B
10. B

42. a. C

1. C
2. B
3. B
4. C
5. B
6. C
7. C
8. A
9. B
10. B

43. a. E (n2)

1. D
2. D

44. a. D

b. D

c. D