**Big-Oh Analysis**

Give a tight bound of the runtime complexity class for each of the following code fragments in Big-Oh notation, in terms of the variable *N*.

Source: <https://courses.cs.washington.edu/courses/cse373/13wi/exams/final-practice-2.pdf>

**Problems**:

|  |
| --- |
| a)  Map<Integer, Integer> map = new TreeMap<Integer, Integer>();  for (int i = 1; i <= N; i \*= 2) {  } map.put(i, i + 1);  System.out.println("done!"); |
| b)  Set<Integer> set = new HashSet<Integer>();  for (int i = 0; i < 9999999; i++) {  } set.add(i \* N);  System.out.println("done!"); |
| c)  int sum = 0;  for (int j = 0; j < 10000 \* N; j++) {  for (int i = N; i > 0; i /= 2) {  } } sum++;  System.out.println(sum); |
| d)  List<Integer> list = new ArrayList<Integer>();  for (int i = 0; i < N; i += 3) {  } list.add(i);  Queue<Integer> pq = new PriorityQueue<Integer>();  for (int i = 0; i < list.size(); i++) {  } pq.add(list.remove(0));  System.out.println("done!"); |

**Solutions**:

Source: <https://courses.cs.washington.edu/courses/cse373/13wi/exams/final-practice-2-key.pdf>

|  |
| --- |
| a)  **Original:**  Map<Integer, Integer> map = new TreeMap<Integer, Integer>();  for (int i = 1; i <= N; i \*= 2) {  } map.put(i, i + 1);  System.out.println("done!");  **Runtime:**  O(1)  for O(logN)  O(logN)  O(1)  **Answer:**  = O(1) + O(logN \* logN) + O(1)  **= O((logN)^2)** |
| b)  **Original:**  Set<Integer> set = new HashSet<Integer>();  for (int i = 0; i < 9999999; i++) {  } set.add(i \* N);  System.out.println("done!");  **Runtime:**  O(1)  for O(1)  O(1)  O(1)  **Answer:**  **= O(1)** |
| c)  **Original:**  int sum = 0;  for (int j = 0; j < 10000 \* N; j++) {  for (int i = N; i > 0; i /= 2) {  } } sum++;  System.out.println(sum);  **Runtime:**  O(1)  for O(N)  O(logN)  O(1)  O(1)  **Answer:**  = O(1) O(N(logN(1))) + O(1)  **= O(NlogN)** |
| d)  **Original:**  List<Integer> list = new ArrayList<Integer>();  for (int i = 0; i < N; i += 3) {  } list.add(i);  Queue<Integer> pq = new PriorityQueue<Integer>();  for (int i = 0; i < list.size(); i++) {  } pq.add(list.remove(0));  System.out.println("done!");  **Runtime:**  O(1)  for O(N)  O(1)    O(1)  for O(N)  O(N)  O(logN)  O(1)  **Answer:**  = O(1) + O(N(N + logN)) + O(1)  = O(N^2 + NlogN)  **= O(N^2)** |