**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/midterm-practice-3.pdf>

**Problems**:

|  |
| --- |
| a)  int sum = 0;  for (int i = 0; i < N; i++) {  if (i == N - 1) {  for (int j = 0; j < N; j++) {  } } } sum++;  System.out.println(sum); |
| b)  Map<Integer, Integer> map = new TreeMap<Integer, Integer>();  for (int i = 1; i < N; i++) {  } map.put(i, N \* N);  map.clear();  System.out.println("done!"); |
| c)  int sum = 0;  for (int i = 0; i < N; i++) {  } sum++;  for (int i = 100\*N; i >= 0; i--) {  } sum++;  System.out.println(sum); |
| d)  List<Integer> list = new LinkedList<Integer>();  for (int i = 0; i < N; i++) {  } list.add(i);  int sum = 0;  for (int i = 0; i < N; i++) {  } sum += list.get(i);  System.out.println("done!"); |

**Solutions**:

Source: <https://courses.cs.washington.edu/courses/cse373/13wi/exams/midterm-practice-3-key.pdf>

|  |
| --- |
| a)  **Original:**  int sum = 0;  for (int i = 0; i < N; i++) {  if (i == N - 1) {  for (int j = 0; j < N; j++) {  } } } sum++;  System.out.println(sum);  **Runtime:**  O(1)  for O(N)  O(1)  O(1)  O(1)  O(1)  **Answer:**  = O(1) + O(N(1 + 1(1))) + O(1)  = O(N + 1)  **= O(N)** |
| b)  **Original:**  Map<Integer, Integer> map = new TreeMap<Integer, Integer>();  for (int i = 1; i < N; i++) {  } map.put(i, N \* N);  map.clear();  System.out.println("done!");  **Runtime:**  O(1)  for O(N)  O(logN)  O(N)  O(1)  **Answer:**  = O(1) + O(NlogN) + O(N) + O(1)  = O(NlogN) + O(N)  **= O(NlogN)** |
| c)  **Original:**  int sum = 0;  for (int i = 0; i < N; i++) {  } sum++;  for (int i = 100\*N; i >= 0; i--) {  } sum++;  System.out.println(sum);  **Runtime:**  O(1)  for O(N)  O(1)  for O(100N)  O(1)  O(1)  **Answer:**  = O(1) O(N) + O(N) + O(1)  **= O(N)** |
| d)  **Original:**  List<Integer> list = new LinkedList<Integer>();  for (int i = 0; i < N; i++) {  } list.add(i);  int sum = 0;  for (int i = 0; i < N; i++) {  } sum += list.get(i);  System.out.println("done!");  **Runtime:**  O(1)  for O(N)  O(1)  O(1)  for O(N)  O(N)  O(1)  **Answer:**  = O(N) + O(N^2)  **= O(N^2)** |