**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-1.pdf>

**Problems**:

|  |
| --- |
| a)  int sum = 0;  int X = 100000;  for (int i = 1; i <= 4 \* N; i++) {  for (int j = 1; j <= X + 2; j++) {  } sum++;  for (int j = 1; j <= X \* 100; j += 2) {  for (int k = 1; k <= X \* X; k++) {  sum++;  }  }  sum++;  }  System.out.println(sum); |
| b)  Map<Integer, Integer> map = new TreeMap<Integer, Integer>();  for (int i = 0; i < N; i += 2) {  } map.put(i, N \* N);  System.out.println("done!"); |
| c)  int sum = 0;  for (int j = 0; j < 100 \* N; j++) {  for (int i = N; i > 0; i /= 2) {  } } sum++;  System.out.println(sum); |
| d)  Random rand = new Random();  Queue<Integer> pq = new PriorityQueue<Integer>();  for (int i = 0; i < 99999; i++) {  for (int j = 0; j < 99999; j++) {  } } pq.add(N \* rand.nextInt());  System.out.println("done!"); |

**Solutions**:

Source: <https://courses.cs.washington.edu/courses/cse373/13wi/exams/final-practice-1-key.pdf>

|  |
| --- |
| a)  **Original:**  int sum = 0;  int X = 100000;  for (int i = 1; i <= 4 \* N; i++) {  for (int j = 1; j <= X + 2; j++) {  } sum++;  for (int j = 1; j <= X \* 100; j += 2) {  for (int k = 1; k <= X \* X; k++) {  sum++;  }  }  sum++;  }  System.out.println(sum);  **Runtime:**  O(1)  O(1)  for O(N)  for O(1)  O(1)  for O(1)  for O(1)  O(1)  O(1)  O(1)  **Answer:**  = O(1) + O(1) + O(N((1(1))\*(1(1(1)))\*1) + O(1)  = O(N + N + N)  **= O(N)** |
| b)  **Original:**  Map<Integer, Integer> map = new TreeMap<Integer, Integer>();  for (int i = 0; i < N; i += 2) {  } map.put(i, N \* N);  System.out.println("done!");  **Runtime:**  O(1)  for O(N/2)  O(logN)  O(1)  **Answer:**  = O(1) + O(N(logN)) + O(1)  **= O(NlogN)** |
| c)  **Original:**  int sum = 0;  for (int j = 0; j < 100 \* 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)) + O(1)  **= O(NlogN)** |
| d)  **Original:**  Random rand = new Random();  Queue<Integer> pq = new PriorityQueue<Integer>();  for (int i = 0; i < 99999; i++) {  for (int j = 0; j < 99999; j++) {  } } pq.add(N \* rand.nextInt());  System.out.println("done!");  **Runtime:**  O(1)  O(1)  for O(1)  for O(1)  O(1)  O(1)  **Answer:**  = O(1)  **= O(1)** |