**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-13wi.pdf>

**Problems**:

|  |
| --- |
| a)  Map<Integer, String> map = new TreeMap<Integer, String>();  for (int i = 1; i <= N; i++) {  map.put(i, "foo");  if (map.containsKey(i / 100)) {  map.remove(i - 1);  map.put(i, "bar");  }  } |
| b)  List<String> list = new ArrayList<String>();  for (int i = N; i >= 1; i--) {  for (int j = 1; j <= i; j++) {  if (!list.contains(i \* j)) {  } } } list.add(i \* j);  for (int j = 1; j <= N; j++) {  } list.add(j);  System.out.println(list); |
| c)  Map<Integer, Integer> map = new HashMap<Integer, Integer>();  for (int i = 1; i <= N; i++) {  } map.put(i, i+1);  int sum = 0;  for (int i : map.keySet()) {  Set<Integer> copy = new HashSet<Integer>(map.values());  if (copy.contains(i \* 2)) {  } } sum++;  System.out.println(sum); |

**Solutions**:

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

|  |
| --- |
| a)  **Original:**  Map<Integer, String> map = new TreeMap<Integer, String>();  for (int i = 1; i <= N; i++) {  map.put(i, "foo");  if (map.containsKey(i / 100)) {  map.remove(i - 1);  map.put(i, "bar");  }  }  **Runtime:**  O(1)  for O(N)  O(logN)  O(logN / 100)  O(logN)  O(logN)  **Answer:**  = O(1) + O(N(logN + logN + logN + logN))  = O(NlogN + NlogN + NlogN + NlogN))  **= O(NlogN)** |
| b)  **Original:**  List<String> list = new ArrayList<String>();  for (int i = N; i >= 1; i--) {  for (int j = 1; j <= i; j++) {  if (!list.contains(i \* j)) {  } } } list.add(i \* j);  for (int j = 1; j <= N; j++) {  } list.add(j);  System.out.println(list);  **Runtime:**  O(1)  for O(N)  for O(N)  O(N)  O(1)  for O(N)  O(1)  O(1)  **Answer:**  = O(1) + O(N(N(N + 1))) + O(N(1)) + O(1)  = O(N(N^2 + N) + O(N)  = O(N^3 + N^2) + O(N)  **= O(N^3)** |
| c)  **Original:**  Map<Integer, Integer> map = new HashMap<Integer, Integer>();  for (int i = 1; i <= N; i++) {  } map.put(i, i+1);  int sum = 0;  for (int i : map.keySet()) {  Set<Integer> copy = new HashSet<Integer>(map.values());  if (copy.contains(i \* 2)) {  } } sum++;  System.out.println(sum);  **Runtime:**  O(1)  for O(N)  O(1)  O(1)  for O(N)  O(N)  O(1)  O(1)  O(1)  **Answer:**  = O(1) O(N(N + 1 + 1)) + O(1)  = O(N^2 + N + N)  **= O(N^2)** |