

How to Calculate Time Complexity of an Algorithm + Solved Questions (With Notes)

CodeWithHarry

Before Solving Some Questions of Time Complexity I will tell you some tricks to get rid of time complexity. After that we will do some set of questions. which will make you a very good grasp in such questions. due to the time complexity of any algorithm when you have to find it so what is the first step that you do and at the same time how to approach this problem. In this way, whatever instructions are going on here , it is taking almost (k) time. We believe that these operations are all (k) time consuming This for loop , that is , how much time is being taken for this fragment It seems (kn) , okay So before this (int i) would have been written here, (int k=0) would be written here. The third technique that I want to tell you is this : That break the code into fragments. The first fragment turned out to be this one , with a little bit of initialization. It took constant time because it is not such that if the value of (n) increases, then its time will increases.

I will go for (n = 100) to determine whether i will be going for n = 1000. I will accept it in (k4) and (n * k4) I will do it in k4 and (n* k4), and it will happen O (n²) If you do it (k=0) , (k < n) and if you look at it , it will come out O (n²) Okay, it will not (N²) ok remember you this thing. There will be some code on it which will take (k1) Now I have become so smart, by doing questions , and you will be done too That (k1) it is will going to be non-dominant , if constant is being added then we will remove it. So once the value of (i) will be zero (0) and then the value of (j) will run for. Then (j=1) will become Then (0,2) Then (i=0 , and j=0) will then run for 0. The value of (i) will be zero (0) for (n) times running the value of (i) will become (1) , it will run again (n) times then it will go on till n. When (n) is running out, watch carefully , watch very closely. Then later I will ask the question, then I am telling if it is not done. value of i will be (1) , (n) , it will be n-1 because I am taking the index (i=) then (i=n-1) will be and here is (n- 1)

I told you guys If it 's not clear to you why it will work (n²) times So I 'd say let 's go look at it for 3 and 3 and print here (i , j) and make a count variable and count it , how many times it is running You write (c) program , write in Python, write in Java, write in Python and write in the Java. But when there are 2 loops inside one , then that will run for n² times. And if another loop is given inside it , then it will run (n³) times. If there is a double for loop, then it becomes straight (n²) I have handpicked some questions which I am going to give to you guys here. And I have also given their programs to you. So you see here I have opened this folder in visual studio code. So it 's saying that Find the time complexity (Func1) function in the program shown in program1. c as follows. Even if you come from another programming language nothing is going to be change. The time of (F1+F2+F3) will be that I will take as the overall time of the whole function. The time is not depending on Array 's length so i 'll accept it (k1) and I can not accept (F2) as (k) , i will accept it as k2 * n. So now if I find a total time complexity that is , if I T (n) come out , then what will it be ? T (n) will be done as T (N) =F1 +F2 +F3.