Asymptotic Notations: Big O, Big Omega and Big Theta Explained

We 'll talk a little bit about asymptotic notation. we talked about order. We talked about ordering. We have primarily 3 types of asymptic notation big O, big Theta (Θ) and big Omega (Ω) big O is represented by capital (O), which is in our English. Big O is set to be O (g (n)) if and only if there exist a constant (c) and a constant n -node such that 0 ≤ f (n) ≤ cg (n) is O (g (N) If you watch this video completely then I guarantee that you will understand these three notations. Mathematically, mathematically this function can be anything. When we do analysis of algorithms comparing any 2 algorithms then f (n) will be time and what is n , it 's input ok , size of input. G (n) is your function which will come inside the big O. O (n^2) is Anything Can Be Algorithm it is g (n) that will be here and which is your algorithm. If you guys can find any such constant (C) and (n) -node , then f (n) is O (g (n)" This is the mathematical definition of big O. If you ca n't find it then it 's is not f (n) is O. This question is its own truth , it has validity , it will remain valid. This passage discusses the complexity of an algorithm, which is measured in terms of the size of its big O graph. The author states that the complexity of an algorithm is automatically O(n^5.), O(n^30), and O(n^100). If G (n) is intersecting with f (n). So you will get some complex function Alright so this is the solution to the problem So. What we have done is WE have taken a big function and we have made it so that it is always below the original function and that's what [UNK] means THe definition of [UNK] for a function. F(n) is the largest value of G(n) that is bigger than f(n)..