 Big O

  Consider an algorithm that takes 5 n \*2 steps for input size n. ie f(n)= 5 \* n+ 2.

There are two steps to find Big O  of f(n)

1.       Remove all but highest power terms seeing the graph for example 2 ^n  is greater than n ^2  (check for values n=100)

2.       Remove the coefficient of highest power.

 The Big O of f(n)=5n+2 will be O(n)  but how to prove it using the definition that it is upper bound.

This is upper bound than n=6 T(n)=6 and f(n)=32 so how come n is upper bound of 5n+2 when for n=6 it is less than 32. To see it we go do definition of Big-O. For any constant c(we can select the value) c g(n) should be greater than f(n) but according to definition for certain values on n >= n0.

The rules for selection of c for Big O not for omega, theta.

If f(n) has minus after largest term and most of the term are subtracting other than largest term that c can be what is the coefficient of largest term.

If f(n) has addition after largest term and most of the term are adding with the largest term that c can be a value greater than what is the coefficient of largest term.

Solution: c= 6 cg(n)= 6n

For n=1 cg(n) 6\*1=6 f(n)=5\*1+2=7 so cg(n) is not greater than f(n) so n0 will be greater than 1

For n=2 cg(n) 6\*2=12 f(n)=5\*2+2=12 so cg(n) is equal to ( greater than and equal to required) f(n) so n0 will be 2

So c is 6 n0 is 2 to prove it according to definition.

Have a good time.