

To calculate the time complexity of this algorithm:

i j j-runtime

1 2 to n n-1 times

2 3 to n n-2 times

3 4 to n n-3 times

n-2 n-1 to n 1 time

n-1 n 0 time

Taking only the highest-powered polynomial:



To calculate the time complexity of this algorithm (FIND-MAXIMUM-SUBARRAY, call it ‘T’):

Base Case:

When n = 1, then just one statement will run. Hence, time complexity is O(1)

When n > 1:

T(n) runs three functions: T(n/2) twice, and FIND-MAX-CROSSING-SUBARRAY(n) once, call it ‘P’.

Since the algorithm P(n) takes an array and runs two loops starting from the middle, one backwards and the other forwards – it *de facto* spans the whole array of size ‘n’ once, hence its time complexity is O(n).

Thus, we can write:

Using the substitution method:

If the input to eq 1.1 is half, then: