// A is an n-by-n two-dimensional array

// swap() swaps the values of its arguments

for (int i=1 ; i < n ; i++) {

for (int j=0; j < i ; j++) {

swap(A[j][i], A[i][j]) ;

}

}

because the program loops times and within each loop it loops anywhere from 0 to times depending on so the absolute maximum is never more than *2*

because the program only has a static number of loops depending solely on the value of , it will always loop times

// A is an n-long array

// v is a value to search for

bool found = false;

for (int i=0; i < n; i++) {

if (A[i] == v) {

found = true;

break;

}

}

because the maximum times the program will compare with

if is in at a certain point, then regardless of how long the array is, it will find after a constant number of operations

// A is a \*sorted\* n-long array

// v is a value to search for

int p = 0 ;

int r = n-1 ;

while ( p != r ) {

int q = (p + r) / 2 ;

if (v <= A[q]) {

r = q ;

} else {

p = q + 1 ;

}

}

because each iteration of the loop divides the search radius effectively in half

because no matter if is in the array or not or how soon the program locates , it goes until the search radius is reduced to 1, after iterations

t = 1 ;

for (int i=0 ; i< n ; i++) {

t++ ;

}

while ( t > 1 ) {

t = t / 2 ;

}

because the process of incrementing -times is almost always longer than the process of halving until it reaches 1

because is never less than

int total = 0 ;

int i = 1 ;

while ( total <= n ) {

total = total + i ;

i++ ;

}

because the program loops through times, where can be determined through

Thus, the program iterates times, which means

because given any , the program iterates exactly , no more, no less, so the upper and lower bounds are the same