Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Give the complexity of the following functions. Choose the most appropriate notation from among , , and .

1. **void** **function1**(**int** n) {

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

**for** (**int** j = i; j <= n; j += 2) {

cout << "\*";

}

}

}

Answer:

1. **void** **function2**(**int** n) {

**int** count = 0;

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

count++;

}

cout << count;

}

Answer: or

1. **void** **function3**(**int** n) {

**int** count = 0;

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

**for** (**int** j = 1; j + n/2 <= n; j++) {

**for** (**int** k = 1; k <= n; k \*= 2) {

count++;

}

}

}

cout << count;

}

Answer:

1. **void** **function4**(**int** n) {

**int** count = 0;

**for** (**int** i = n/2; i <= n; i++) {

**for** (**int** j = 1; j <= n; j \*= 2) {

**for** (**int** k = 1; k <= n; k \*= 2) {

count++;

}

}

}

cout << count;

}

Answer:

1. **void** **function5**(**int** n) {

**if** (n % 2 == 0) {

**return**;

}

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

**for** (**int** j = 1; j <= n; j++) {

cout << "\*";

**break**;

}

}

}

Answer: , watch out for the break statement, the inner loop is really . For even numbers, the algorithm runs in constant time, so we cannot have .

1. **void** **function6**(**int** n) {

**int** count = 0;

**for** (**int** i = 1; i <= n/2; i++) {

**for** (**int** j = 1; j <= n/3; j++) {

**for** (**int** k = 1; k <= n/4; k++) {

count++;

}

}

}

cout << count;

}

Answer:

1. **void** **function7**(**int** n) {

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

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

cout << "\*";

}

}

}

Answer:

1. **void** **function8**(**int** n) {

**int** i = 1, s = 1;

**while** (s <= n) {

i++;

s += i;

cout << "\*";

}

}

Answer: or

The loop terminates when , or , becomes > . The dominant term is . In terms of asymptotics, this implies the loop terminates when , or when . Therefore, function8 is , and corresponding constants c1, c2, and n0 can be found.

1. Processing Arrays
   1. Suppose you have an unsorted array of integers of length and want to sum all the elements inside it. What is the running time of your algorithm?
   2. Suppose you have an unsorted array of integers of length and want to determine if all the values inside are positive. What is the running time of your algorithm?
   3. Suppose you have a sorted array of integers of length and want to determine the median value. What is the running time of your algorithm?
2. \_\_T\_\_ T / F

If true, prove it by giving *integral* values for the required constants , , and . Choose the tightest values possible for the and constants. If false, show the contradiction.

Choose ; lower bound holds

Choose ; upper bound holds