**CSE 274 Laboratory Assignment - Complexity Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

For this lab assignment, submit this paper. You may work together, but each person should submit her/his own work.

1. Determine the simplest Big-O notation for the following functions, where K is a constant:

0.001N2 + 1000N 2N + 1000N + 1 N+N∙log N N2+N∙log N  
  
  
 4∙N3+2N 4+37∙N2+0.3∙N3  N2/3+N/K 2K + 1000N + 1  
  
  
 2K + 1000K + 1 2N + 1000N + 1 2N + 1000NK + 1 K2+N

1. Order the functions in order of slowest growth to fastest growth. Identify functions that are equivalent.  
     
    N2 2N Nlog3N N log10N log2N
2. Determine the Big-O of the following code segments  
   1. for (int i=0; i<=N+5; i++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       sum += i;
   2. for (int i=0; i<N; i+=2) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       sum += i;
   3. for (int i=1; i<N; i\*=2) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       sum += i;
   4. for (int i=0; i<N; i++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       for (int j=0; j<3; j++)  
       sum += i \* j;
   5. for (int j=0; j<3; j++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       for (int i=0; i<N; i++)  
       sum += i \* j;  
      for (int i=0; i<N; i++)  
       sum += i;  
      for (int j=0; j<N; j++)  
       sum += j;
   6. for (int r=0; r<N; r++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       for (int c=0; c<N; c++)  
       sum += img[r][c];
   7. for (int r=0; r<N; r++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       for (int c=0; c<r; c++)  
       sum += img[r][c];
   8. for (int r=0; r<N; r++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       for (int c=0; c<N; c++)  
       for (int dr=-1; dr<=+1; dr++)   
       for (int dc=-1; dc<=+1; dc++)  
       sum += r \* c + dr \* dc;
   9. for (int i=1; i<N\*N; i++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       sum += i;
   10. int f(int N) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       { return 2 \* N; }  
       for (int i=0; i<2; i++)  
        sum += f(i);
   11. int f(int N) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       { return 2 \* N; }  
       for (int i=0; i<2; i++) {  
        sum += f(i-1);  
        sum += f(i);  
        sum += f(i+1);  
       }
   12. int sum = 0; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       for (int i=0; i<N\*N; i++)  
        sum++;
   13. int val = N; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
       while (val > 0) {  
        val /= 10;  
       }
   14. int f(int N) { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
        sum = 0;  
        for (int i=0; i<N; i++) {  
        sum += i;  
        return sum;

}  
 int totalSum = 0;  
 for (int i=0; i<N; i++)  
 totalSum += f(N);

* 1. int f(int N) { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
      sum = 0;  
      for (int i=N-1; i<=N+1; i++) {  
      sum += i;  
      return sum;

}  
 int totalSum = 0;  
 for (int i=N-5; i<=N+5; i++)  
 totalSum += f(i);

**Bubble Sort Worst Case Analysis:** All of these questions pertain to the worst case for this bubble sort algorithm. The idea is that in one pass, side by side comparisons are made from beginning to end, swapping whenever two side by side elements are out of order.

**public** **static** **void** bubbleSort(**int** nums[]){

**boolean** swap = **false**;

**for**(**int** pass= 0; pass < nums.length - 1; pass++){

**for**(**int** j = 0; j < nums.length - pass-1; j++){

// introducing a swap flag to monitor swapping

swap = **false**;

**if**( nums[j] > nums[j+1])

{

// swap the elements

**int** temp = nums[j];

nums[j] = nums[j+1];

nums[j+1] = temp;

// if swapping happens update swap to true

swap = **true**;

}

}

// if value of flag is false after all the

// iterations of inner loop then break out

**if**(!swap){

**break**;

}

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| **WORST case analysis** | nums.length = 50 | nums.length = 100 | nums.length = N |
| comparisons on pass 1 |  |  |  |
| comparisons on pass 14 |  |  |  |
| comparisons on pass K |  |  |  |
| comparisons on last pass |  |  |  |
| total passes required |  |  |  |
| total comparisons required |  |  |  |
| total swaps required |  |  |  |

**WORST case, big-O for bubble sort of N elements:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BEST case Bubble Sort analysis:**

In the best case, how many passes are required for an array of length N? \_\_\_\_\_\_\_\_\_\_\_\_\_

In the best case, how many comparisons are required for an array of length N? \_\_\_\_\_\_\_\_\_\_\_\_\_

**BEST case, big-O for bubble sort of N elements:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Selection Sort Worst Case Analysis:** All of these questions pertain to the worst case for this selection sort algorithm. The idea behind this algorithm is to select the smallest element in the array and swap it to position 0. Then select the next smallest element in the array and swap it to position 1, and so on.

**public** **static** **void** selectionSort(**int** nums[]){

**for** (**int** pass = 0; pass < nums.length - 1; pass++) {

count = 0;

// One pass: find smallest and swap

// Find the smallest from pos to end of array

**int** smallestIndex = pass;

**for** (**int** i = pass + 1; i < nums.length; i++) {

**if** (nums[i] < nums[smallestIndex]) {

smallestIndex = i;

}

}

// Found the smallest, so swap it to nums[pos]

**int** temp = nums[pass];

nums[pass] = nums[smallestIndex];

nums[smallestIndex] = temp;

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| **WORST case analysis** | nums.length = 50 | nums.length = 100 | nums.length = N |
| comparisons on pass 0 |  |  |  |
| comparisons on pass 14 |  |  |  |
| comparisons on pass K |  |  |  |
| comparisons on last pass |  |  |  |
| total passes required |  |  |  |
| total comparisons required |  |  |  |
| total swaps required |  |  |  |

**WORST case, big-O for selection sort of N elements:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BEST case analysis:**

In the best case, how many passes are required for an array of length N? \_\_\_\_\_\_\_\_\_\_\_\_\_

In the best case, how many comparisons are required for an array of length N? \_\_\_\_\_\_\_\_\_\_\_\_\_

**BEST case, big-O for selection sort of N elements:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_