Johnathan Soto

CS 315

Lab # 3

For each of the following code segments, determine the number of operations of the specified type. You are to answer these questions *without using a computer*.

1. **for(intj=1;j<100;j++)// 99**

**{**

**B[j] = 0; // set B 1 - 99 arr to 0’s // executes 99 times**

**for (int k = j+ 1; k < 100; ++k) // 2 - 99**

**B[j] += B[k]; // executes 98 times per for loop**

**}**(a) How many times is the instruction **B[j] += B[k]** executed?

(99\*98) / 2 = 4,851 times B[j] += B[k] is executed

(b) What is the number of times that the instruction B[j] = 0 is executed?

99 times B[j] = 0 is executed

1. **for(inti=0;i<30;++i) // 30 times**

**for (int j = 0; j < 40; ++j) // 40 times**

**{**

**for (int k = 0; k < 20; ++k) // 20 times  
C[i][j] = C[i][j] + A[i][k]\*B[k][j];**

**C[i][j] = C[i][j]\*D[i][j];**

**}**

How many multiplications are performed by the above code segment ?

(3 for loops multiplication) 24,000 + (2 for loop multiplication) 1,200 = 25,200 multiplications are preformed

**3)** A[0]=1;A[1]=1;  
for (int j = 2; j < 1000; ++j)

if (j % 2 == 0)  
A[j] = A[j-1]+3\*A[j-2];

else  
A[j] = A[j-1]+A[j-2];

What is the total number of operations performed by the above program? (Include all the following operations: assignments, increments, comparisons, additions, multiplications and % operations. For example, the assignment A[j] = A[j-1] + A[j-2] would be counted as 4 operations – two subtractions, one addition and one assignment. In your solution, indicate the number of operations associated with each instruction separately and the number of times each instruction is executed, then calculate the total number of operations. (Count the increment as two operations since it involves an addition and an assignment.)

Total:

Assign(1999) + Compare(1997) + Modulus(998) + Add(1497) + Subtraction(1497) + Multiply(499) + Increment(999) = ***9486 operations executed in the program above.***

For the following problems, first determine the exact number of operations of the specified type as a function of n, the input size. Then express your answer using O notation.

**4)** Shown below is a function that takes as input a vector of n integers and determines the maximum and the minimum numbers in the array. Determine (a) the number of key comparisons and (b) the number of key assignments performed by findminmax as a function of n in the best-case and in the worst-case. In the code, swap(x,y) swaps the keys x and y. Each swap involves three assignments. (Assume that n is even.)

void findminmax (vector<int> A, int& min, int& max) { int n = A.size();  
for (int j = 0; j < n-1; j = j+2)

if (A[j] > A[j+1]) swap(A[j], A[j+1]); min = A[0]; max = A[1];  
for (int j = 0; j < n-1; j = j+2) {

if (A[j] < min) min = A[j];

if (A[j+1] > max) max = A[j+1]; }

} }

Make sure to give all the four answers. (The number of key assignments in the best case, the number of key assignments in the worst case, the number of key comparisons in the best case and the number of key comparisons in the worst case.) Key assignment does not include j = 0, j = j + 2 etc. because they are index assignment. Similarly, j < n – 1 is an index comparison, not key comparison so it is not counted.

Best case Key Assignment: 2 or O(1)

Worst Case Key Assignment: 5n/2 or O(n)

Best Case Key Assignment: 3n/2 or O(n)

Worst Case Key Assignment: 3n/2 or O(n)

**5)** Shown below is a code segment that implements insertion sorting, an important sorting algorithm:

void insertionSort (vector<int> A) {

for (int j = 1; j < A.size(); ++j) {

temp = A[j]; for (k = j – 1; k >= 0 && A[k] > temp; k--) A[k+1] = A[k];

A[k+1] = temp; }

}

1. Determine the exact number of key comparisons performed by the above program in the worst-case.

((n - 1) \* n) / 2 = ((n^2) – n)/ 2 or O(N^2)

1. Do the same thing for the best-case.

(n – 1) \* 1 = n – 1 or O(N)

1. Assuming that the keys used are the integers 1 to n, what input is the best-case input?

The best-case would be going from 1 to n increasing in value till integer n, starting at integer 1. (eg: 1,2,3,4)

(d) Assuming that the keys used are the integers 1 to n, what input is the worst-case input?

The worst-case is going from n to 1 decreasing from n to 1, starting at integer n. (e.g: 4,3,2,1)

Recursive program analysis

To analyze a recursive function, we usually create a recurrence equation in which a function is defined in terms of itself. Here is an example of computing factorial recursively:

int fact (int n) {  
if (n == 1) return 1;

else return n\*fact(n-1); }

Let s(n) be the number of multiplications performed by fact. s(n) is given by the following formula:

s(n) = 0 if n = 1 1 + s(n – 1) else.

We can compute the value of s(n) for small n:  
s(1) = 0, s(2) = 1, s(3) = 2, ... This leads to the guess that s(n) = n – 1. It is easy to prove this guess

by induction on n.

**6)** Recall the recursive algorithm for computing x that you wrote for lab # 1:

long long int exp (long long int x, int n) { if (n == 0) return 1;  
else if (n == 1) return x;  
else if (n % 2 == 0)

return exp(x \* x, n/2); else

return exp(x \* x, (n – 1)/2) \* x; }

1. a)  Compute the number of multiplications performed by the above function for n = 1000, 2000, 3000, ..., 10000 and display the result in a table.

|  |  |
| --- | --- |
| n | multiplications |
| 1000 | 14 |
| 2000 | 15 |
| 3000 | 17 |
| 4000 | 16 |
| 5000 | 16 |
| 6000 | 18 |
| 7000 | 18 |
| 8000 | 17 |
| 9000 | 17 |
| 10000 | 17 |

b)  Write a recursive formula for M(n), the number of multiplications performed by exp. (Note that the number of multiplications performed depends only on n, not x.) You don’t need to solve the recursive formula.

M(n) = { 0 if n = 0 or n = 1

M(n/2) +1 if n > 0 and n is even

M( ((n-1) / 2) +1 if n > 1 and n is odd

**7)** This problem involves experimentally determining the average number of key comparisons performed by insertionSort (Problem 5) as a function of the input size. For various values of n = 100, 200, ..., 1000 experimentally estimate this number by randomly generating m = 10 n inputs (thus for n = 100, you will use 1000 inputs etc.) and counting the number of times the key comparison instructions are executed. Create a table that shows the average-case as well as the worst-case number of comparisons for various values of n. Use this to guess an expression for the average number of comparisons A(n) performed by insertionSort. Use A(n) = an2 + bn + c and use regression to find the values of a, b and c. There are many tools that use online for quadratic regression, for example wolframalpha. We will discuss in the lecture on Wednesday.

a = 0.25 b = 7.59433x10^(-13) c = -9.0555x10^(-9)

Best case: n – 1 or O(N)

Worst case: ( (n^2) – n) / 2 or O(N^2)

Average: (Best case + worst case) / 2

Average A(n) Formula: 0.25(n^2) + (7.59433x10^(-13))(n) – 9.0555x10^(-9)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | Number of key comparisons (best) | Number of key comparisons (worst) | Number of key comparisons ( (Best case + Worst case) / 2) | Number of key comparisons (A(n) formula) |
| 1000 | 999 | 499,500 | 250,249.5 | 250,000.00 |
| 2000 | 1999 | 1,999,000 | 1,000,499.5 | 1,000,000.00 |
| 3000 | 2999 | 4,498,500 | 2,250,749.5 | 2,250,000.00 |
| 4000 | 3999 | 7,998,000 | 4,000,999.5 | 4,000,000.00 |
| 5000 | 4999 | 12,497,500 | 6,251,249.5 | 6,250,000.00 |
| 6000 | 5999 | 17,997,000 | 10,001,499.5 | 9,000,000.00 |
| 7000 | 6999 | 24,496,500 | 12,251,749.5 | 12,250,000.00 |
| 8000 | 7999 | 31,996,000 | 16,001,999.5 | 16,000,000.00 |
| 9000 | 8999 | 40,495,500 | 20,252,249.5 | 20,250,000.00 |
| 10000 | 9999 | 49,995,000 | 25,002,499.5 | 25,000,000.00 |