L2-Complexity

Monday, August 10, 2020 10:03 AM

Finding Timing Complexity of a piece of code.

- Finding Exact Running Time
  - Hardware & Software Dependency
  - Very Time Consuming
  - Error Prone
- Finding Complexity profile as function of input size

What is ***input size***?

Operations Counting Technique:

- Assign each primitive operation (+,-,*,/, etc.) equal cost (1 unit)
- Assign unit cost to object creation
- Assign unit cost to initialization, copy & assignment of primitive objects
- Assign unit cost to return statement of primitive objects
- Assign unit cost to function call excluding argument/parameter passing

- Find the number of operations of each line of code
- Find the frequency of execution of each line of code.

Developing Timing Equation

		operations(Cost)	*	Freq
1.	a = 2*a + 3*k	4	*	1

2. a = a * k; 2 * 1

T(n) = 4*1 + 2*1 = 6

Developing Timing Equation

operations(Cost) * Freq

//for (i=1 to n)

1. for (int i=1; i<=n; ++i) (1+1) + 2*n (end-start+2)
2. a = a * k; 2 * n
3. }

T(n) = 2*n +2*n + 1+1
= 4*n +2

operations(Cost) * Freq

//for (i=1 to n-1)

4. for (int i=1; i < n; ++i) { (3+1)+2*(n-2) (1+1) + 2 * (n-1) (end-start+1)
5. a = a * k; 2 * (n-1)
6. }

T(n) = 2*(n-1) +2*(n-1) +1+1
= 4*n - 2

operations(Cost) * Freq

//for (i=1 to n step 2)

1. for (int i=1; i<=n; i=i+2){ (4+1)+3*(n/2 - 1) = (1+1)+ 3*(n/2)
2. a = a * k; 2*(n/2)
3. }

Let n=8
i=1,3,5,7,9

T(n) = 5*n/2 + 2

operations(Cost) * Freq

//for (i=1 to n step 3)

1. for (int i=1; i<=n; i=i+3){ (4+1) + 3*(8/3-1) = (1+1) + 3*(n/3)
2. a = a * k; 2*(n/3)
3. }

Let n=9
i=1,4,7,10

Let n=8
i=1,4,7,10

T(n) = 5*(n/3) + 2

Developing Timing Equation

operations(Cost) * Freq

1. for (int i=1; i<=n; i=i*2) { (4+1) + 3* $\log_2(n)$ (1+1) + 3* ($\log_2(n)$+1)

2. a = a * k; 2 * ($\log_2(n)+1$)
3. }

Let n=5
i=1,2,4,8
Let n=8
i=1,2,4,8,16

T(n) = 3*($\log_2(n)+1$) +2*($\log_2(n)+1$) + 1+1
= 5*$\log_2(n)$ + 7

Home Work: Find the operations cost & Frequency of each Line.

Developing Timing Equation

operations(Cost) * Freq

1. for (int i=n; n>=1; i=i/2) {
2. a = a * k;
3. }

T(n) =