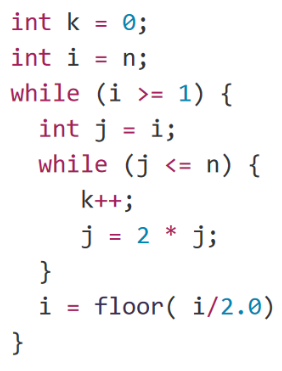
Christopher Singh September 29, 2019

CUS 715 Assignment 3



```
int k = 0;
int i = n;
while (i >= 1) {
  int j = i;
  while (j <= n) {
     k++;
     j = 2 * j;
  }
  i = floor( i/2.0)
}
```

1. Let n = 2^k for k > 0.

Since n = 2^k the outer loop has the following values:

Outer Loop:

i	=	$\frac{n}{1}$	$\frac{n}{2}$	$\frac{n}{4}$	$\frac{n}{\dots}$	$\frac{n}{2^k}$
i	=	1	2	4	…	n
i	=	2^0	2^1	2^2	$2^{\dots}$	2^k

Inner Loop: You can choose any value for n. I chose n = 16

Since n =2^k , and n = 16:

16 = 2^k. → $\frac{\log 16}{\log 2}$ = 4.

Current values at n = 16 and k = 4:

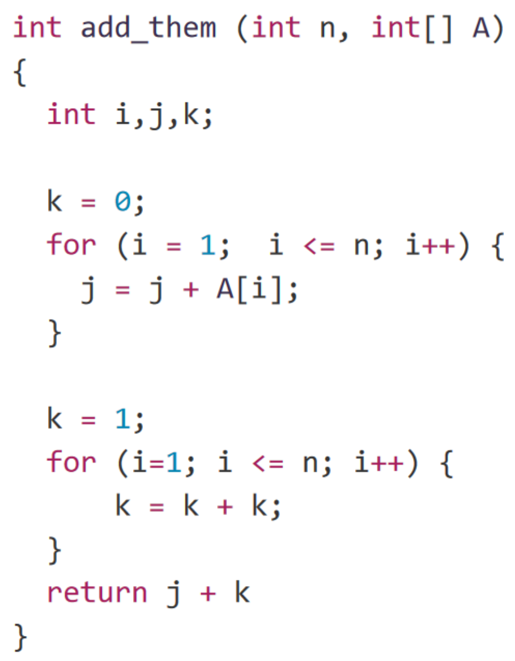
n	i	j	Iteration Count
16	16	16	1 iteration
	8	8	2 iterations
	4	4	
		8	3 iterations
		16	
	2	2	4 iterations

Observations:

The outer loop runs for values from 1 to n by incrementing it by 1. This means that the outer loop will run n times depending on the size of the input set. As a result, the outer loop will have a run time as O(n). The inner loop will run at log(n) time because after each iteration, we are decreasing the value by using the floor and half method.

Both the outer and the inner loops iterates n times. The variable j starts out at a high value based on the input size but decreases by a factor of two.

T(n) = O(n) * O(log n) = **O(n log n)**



```
int add_them (int n, int[] A)
{
  int i,j,k;

  k = 0;
  for (i = 1;  i <= n; i++) {
    j = j + A[i];
  }

  k = 1;
  for (i=1; i <= n; i++) {
      k = k + k;
  }
  return j + k
}
```

2.

Values of j and k after each iteration. Note that we must start at position 1 of the array not position 0 because the variable i is not set to the start of the array.

Iteration Count	j	k
1	5	2
2	8	2
3	15	2
4	23	2
Totals:	51	8

J + K = 51 + 8 = 59

T(n) = O(n) * O(n) = O(n^2**).** Both for loops have a runtime of O(n). To calculate the runtime, you must take into consideration all lines of code and multiple the runtime of each. A better and more efficient approach would be to get rid of the second for loop.

By getting rid of the second for loop and keeping the entire code in one for loop, the runtime will be enhanced. We can do this because each for loop is setup the same way with the same conditions/boundaries.

More efficient code:

```
int add_me (int n, int [] A) {

        int i, j, k;
        k = 1;
        for ( int i = 0; i  <=n; i ++){

                j = j + A[i];
                k = k + k;
        }

        return k
```

3. Show that f(n) belongs to the set of $O(n^3)$.

$T(n) = n^2 + 3n^3 \leq c * n^3$

$n^2 + 3n^3 \leq c * n^3$ Replace c with $3n^3$ to match the n^3 term on the left hand side

$n^2 + 3n^3 \leq n^3 + 3n^3$

$n^2 + 3n^3 \leq 4n^3$

T(n) = $\boldsymbol{n^2 + 3n^3 \leq 4n^3 \leq cn^3}$

c = 4 holds true for n > 1