Complexity of Nested Loops

The purpose of this document is to give methods which can help in computing asymptotic complexity of code. We concentrate on the complexity of nested loops, although we begin with a quick overview of some standard simple loops.

Throughout, we assume n is given outside the code, and we will usually express complexity as a function of n.

We first review some standard single loop examples.

Linear Complexity

```
1. for(int i = 0; i < n; i++)
   for(int i = n; i > 0; i--)
```

The complexity of both of these loops is $\Theta(n)$.

Let $m < n$ be given.

```
2. for(int i = m; i < n; i++)
   for(int i = n; i > m; i--)
```

The complexity of both of these loops is $\Theta(n - m)$.

Logarithmic Complexity

```
3. for(int i = 1; i < n; i=2*i)
   for(int i = n; i > 1; i=i/2)
```

The complexity of both of these loops is $\Theta(\log n)$.

Let $m < n$ be given.

```
4. for(int i = m; i < n; i=2*i)
   for(int i = n; i > m; i=i/2)
```

The complexity of both of these loops is $\Theta(\log(n/m)) = \Theta(\log n - \log m)$.

Nested Loops

We now introduce a counter, which we call `kounter`. In each example, the final value of `kounter` is the number of times the inner loop iterates. Since the inner loop iterates more than the outer loop, `kounter` is asymptotically equivalent to the time complexity. We use the time complexities of examples 1. through 4. above.

```
5. int kounter = 0;
   for(int i = 1; i < n; i = 2*i)
    for(int j = 0; j < i; j++)
     kounter++;
```

```
6. int kounter = 0;
   for(int i = 1; i < n; i = 2*i)
    for(int j = i; j < n; j++)
     kounter++;
```

In both of these examples, the outer loop is logarithmic and the inner loop is linear. We substitute the values given in examples 1. and 2. for the inner loop.

```
5. int kounter = 0;
   for(int i = 1; i < n; i = 2*i)
    kounter = kounter + i;

6. int kounter = 0;
   for(int i = 1; i < n; i = 2*i)
    kounter = kounter + (n-i);
```

In both examples, $i = 2^t$ during the t^{th} iteration of the outer loop. In example 5., the final value of the counter is the sum of a geometric series, namely $1 + 2 + 4 + \cdots + 2^T$, where $T = \Theta(\log n)$ and the last term is $\Theta(n)$. Thus, by the standard formula for the sum of a geometric series, the time complexity is $2 \cdot 2^T - 1 = \Theta(n)$.

In example 6., the final value of the counter is $n - 1 + n - 2 + n - 4 + \cdots + n - 2^T$, which we rewrite as $n(T - 1) - (1 + 2 + 4 + \cdots + 2^T) = nT - n - (2 \cdot 2^T - 1) = \Theta(n \log n)$.

We now consider examples where the outer loop is linear and the inner loop is logarithmic.

```
7. int kounter = 0;
   for(int i = 1; i < n; i++)
    for(int j = 1; j < i; j = 2*j)
     kounter++;

8. int kounter = 0;
   for(int i = 1; i < n; i++)
    for(int j = i; j < n; j = 2*j)
     kounter++;
```

Substituting the formulas given for Examples 5. and 6., we obtain:

```
7. int kounter = 0;
   for(int i = 1; i < n; i++)
    kounter = kounter + log(i)

8. int kounter = 0;
   for(int i = 1; i < n; i++)
    kounter = kounter + log(n) - log(i)
```

The correct asymptotic solutions can be obtained without it, but it's simpler to use calculus. We approximate summations by integrals for both examples. The time complexity for Example 7:

$$\int_{x=1}^{n} \ln x \, dx = (x \ln x - x)\Big|_1^n = \Theta(n \log n)$$

The time complexity for Example 8:

$$\int_{x=1}^{n} (\ln n - \ln x) \, dx = (x \ln n - x \ln x + x)\Big|_1^n = \Theta(n)$$