Examples of Analysis of Loops

In the notes below, we will analyze the runtime of functions written in pseudocode.

1.

```
  Func1(n)
1 x ← 0;
2 for i ← 1 to n do
3 |   x ← x + i;
4 |   x ← x − 1;
5 end
6 return (x);
```

In example 1, lines 1, 3, 4, and 6 take constant time. Since what we are really interested in is the runtime for very large values of n, we'll ignore the execution time for lines 1 and 6, since each will only execute once regardless of the value of n. Each pass through the for-loop takes constant time c. Since the for-loop executes n times in total, the total execution time is cn. Therefore, the function is $\Theta(n)$.

We will find it helpful to use summation notation to analyze more complicated functions with nested loops. Here is how we would use a summation to analyze the function above:

$$\sum_{i=1}^{n} c = cn$$

2.

```
  Func2(n)
1 x ← 0;
2 for i ← 1 to n do
3 |   for j ← 1 to n do
4 |   |   x ← x + (i − j);
5 |   end
6 end
7 return (x);
```

Line 4 takes constant time c. Thus we can model this function as follows. Note that the inner summation represents the inner for-loop and the outer summation represents the outer for-loop.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c$$

We will simplify the nested summation below by starting with the inner summation.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c = \sum_{i=1}^{n} cn = cn^2 = \Theta(n^2)$$

3.

```
Func3(n)
x ← 0;
for i ← 6 to n do
    for j ← 1 to 2n do
        x ← x + (i − j);
    end
end
return (x);
```

Analysis:

$$\sum_{i=6}^{n}\sum_{j=1}^{2n} c = \sum_{i=6}^{n} 2nc = (n-5)2nc = \Theta(n^2)$$

4.

```
Func4(n)
x ← 0;
for i ← 1 to n do
    for j ← 1 to i do
        x ← x + (i − j);
    end
end
return (x);
```

Analysis:

$$\sum_{i=1}^{n}\sum_{j=1}^{i} c = \sum_{i=1}^{n} ci = c\sum_{i=1}^{n} i = c\frac{n(n+1)}{2} = \Theta(n^2)$$

5.

```
Func5(n)
x ← 0;
for i ← 1 to n do
    for j ← 1 to ⌊√n⌋ do
        x ← x + (i − j);
    end
end
return (x);
```

Analysis:

$$\sum_{i=1}^{n}\sum_{j=1}^{\lfloor n \rfloor} c = \sum_{i=1}^{n} c\lfloor n \rfloor = cn\lfloor \sqrt{n} \rfloor \approx cn \times n^{1/2} = \Theta(n^{3/2})$$

6.

```
Func6(n)
x ← 0;
for i ← 1 to n do
    for j ← 1 to ⌊√i⌋ do
        x ← x + (i − j);
    end
end
return (x);
```

Analysis:
The running time is:

$$\sum_{i=1}^{n}\sum_{j=1}^{\lfloor\sqrt{i}\rfloor} c = \sum_{i=1}^{n} c\lfloor\sqrt{i}\rfloor \approx \sum_{i=1}^{n} c\sqrt{i} = c\sum_{i=1}^{n}\sqrt{i}$$

We will analyze $\sum_{i=1}^{n}\sqrt{i}$ by using upper and lower bounds:

We will first find an upper bound. Since the root function is increasing,

$$\begin{aligned}
\sum_{i=1}^{n}\sqrt{i} &= \sqrt{1}+\sqrt{2}+\sqrt{3}+\cdots+\sqrt{n} && (1)\\
&< \underbrace{\sqrt{n}+\sqrt{n}+\sqrt{n}+\cdots+\sqrt{n}}_{\text{n terms}} && (2)\\
&= n\sqrt{n} && (3)
\end{aligned}$$

Therefore $c_1 n\sqrt{n}$ is an upper bound on the summation.

Next, we will find a lower bound, we will do this by throwing away the lower half of the terms, and then decreasing the argument of each term to $\frac{n}{2}$.

$$\begin{aligned}
\sum_{i=1}^{n}\sqrt{i} &= \sqrt{1}+\sqrt{2}+\sqrt{3}+\cdots+\sqrt{n} && (4)\\
&> \underbrace{\sqrt{\frac{n}{2}+1}+\sqrt{\frac{n}{2}+2}+\cdots+\sqrt{n}}_{\frac{n}{2}\text{terms}} && (5)\\
&> \underbrace{\sqrt{\frac{n}{2}}+\sqrt{\frac{n}{2}}+\sqrt{\frac{n}{2}}+\cdots+\sqrt{\frac{n}{2}}}_{\frac{n}{2}\text{terms}} && (6)\\
&= \frac{n}{2}\sqrt{\frac{n}{2}} && (7)\\
&= \frac{n}{2\sqrt{2}}\sqrt{n} && (8)
\end{aligned}$$

Therefore: Therefore $c_2 n\sqrt{n}$ is lower bound on the summation.

So $\sum_{i=1}^{n}\sqrt{i} = \Theta(n^{\frac{3}{2}})$

7.

```
Func7(n)
x ← 0;
i ← 1;
while (i < n) do
    x ← 2x;
    i ← i + 1;
end
return (x);
```

Analysis: The while-loop executes n times, doing constant work each time, so the runtime is $\Theta(n)$.

8.

```
Func8(n)
x ← 0;
i ← 8;
while (i < n) do
    x ← 2x;
    i ← i + 1;
end
return (x);
```

Analysis: Analysis: The while-loop executes $n - 7$ times, doing constant work each time, so the runtime is $\Theta(n)$.

Note that starting the loop at a constant other than 1 doesn't change the asymptotic runtime.

9.

```
Func9(n)
x ← 0;
i ← 1;
while (i < n) do
    x ← 2x;
    i ← i + 3;
end
return (x);
```

Analysis: Now we are incrementing i by 3 each time through the loop, so the loop should iterate about a third as many times as it did in the previous problem. We can see this by creating a table:

Iteration Number	value of i
0	1
1	1 + 3
2	1 + 3*2
3	1 + 3 * 3
...	...
k	1 + 3 * k

The loop will terminate when the value of i (which is $1 + 3k$) is greater than n. That is:

$$1 + 3k > n$$

Solving for k, this is when $k > \frac{n-1}{3}$, so the loop terminates when k is about $\frac{n}{3}$. In other words there are about $\frac{n}{3}$ iterations of the loop. Each iteration takes constant time c, so the total time is about $\frac{n}{3}c$, which is $\Theta(n)$.

10.

```
Func10(n)
x ← 0;
i ← 1;
while (i < n) do
    x ← 2x;
    i ← 2i;
end
return (x);
```

Analysis: Now we are doubling i each time through the loop. We will use a table again:

Iteration Number	value of i
0	1
1	2
2	2^2
3	2^3
...	...
k	2^k

The loop will terminate when the value of i (which is 2^k) is greater than n. To make the math easier we will solve for when they are equal:

$$2^k = n$$

Solving for k, this is when $k \approx log_2(n)$, so now the loop terminates after about $log_2(n)$ interations. Since constant work is done in each iteration the runtime is now $\Theta(\log_2(n))$.

11.

```
Func11(n)
x ← 0;
i ← 42;
while (i < n) do
    x ← 2x;
    i ← 3i;
end
return (x);
```

Analysis:

Iteration Number	value of i
0	42
1	42×3
2	42×3^2
3	42×3^3
...	...
k	42×3^k

The loop will terminate when the value of i (which is 42×3^k) is greater than n. To make the math easier we will solve for when they are equal:

$$42 * 3^k = n$$

Solving for k, this is when $k = log_3(\frac{n}{42})$, . Since constant work is done in each iteration the runtime is $\Theta(\log_2(n))$.

CSE 2321: Examples of Analysis of Loops 2

Give the asymptotic running time of each the following functions in Θ notation. Justify your answer. (Show your work.)

1.

```
Func1(n)
s ← 0;
for i ← 3 to n^2 do
    for j ← 7 to 2i⌊log_5(i)⌋ do
        s ← s + i − j;
    end
end
return (s);
```

Solution:

We can model the running-time $T(n)$ of the two loops with the following summation:

$$T(n) = \sum_{i=3}^{n^2} \sum_{j=7}^{2i \log_2 i} c$$

The inner summation evaluates to $c(2i\log_5(i) - 6)$. The dominant term is $2i\log_5(i)$, so we will evaluate the following summation:

$$\sum_{i=3}^{n^2} 2i\log_5(i)$$

Upper Bound:

$$\sum_{i=3}^{n^2} 2i\log_5(i) \leq \sum_{i=1}^{n^2} 2i\log_5(i) \leq \sum_{i=1}^{n^2} 2n^2\log_5(n^2) = n^2 \times 2n^2\log_5(n^2) = 2n^4\log_5(n^2)$$

Therefore,

$$\sum_{i=3}^{n^2} 2i\log_5(i) = O(n^4\log(n))$$

Lower Bound:

$$\sum_{i=3}^{n^2} 2i\log_5(i) \geq \sum_{i=n^2/2+1}^{n^2} 2i\log_5(i) \geq \sum_{i=n^2/2+1}^{n^2} 2(n^2/2)\log_5(n^2/2) = (n^2/2)2(n^2/2)\log_5(n^2/2)$$

Therefore,

$$\sum_{i=3}^{n^2} 2i\log_5(i) = \Omega(n^4\log(n))$$

Since $T(n) = O(n^4\log_2(n))$ and $T(n) = \Omega(n^4\log_2(n))$, we conclude that $T(n) = \Theta(n^4\log_2(n))$.

2.

```
  Func2(n)
1 s ← 0;
2 for i ← 3 to ⌊√n⌋ do
3     j ← i^3;
4     while (j ≥ i) do
5         s ← s + i − j;
6         j ← j − 4 ;                    /* Note: Subtraction */
7     end
8 end
9 return (s);
```

Inner while loop (steps 3-7) iterates $(i^3 - i)/4$ times and takes ci^3 time.

Running time is:

$$T(n) = \sum_{i=3}^{\sqrt{n}} ci^3$$

Upper Bound:

$$\sum_{i=3}^{\sqrt{n}} ci^3 \leq \sum_{i=1}^{\sqrt{n}} c(i)^3 \leq \sum_{i=1}^{\sqrt{n}} c(\sqrt{n})^3 = \sqrt{n}cn^{1.5} = cn^2$$

Therefore,

$$\sum_{i=3}^{\sqrt{n}} ci^3 = O(n^2)$$

Lower Bound:

$$\sum_{i=3}^{\sqrt{n}} ci^3 \geq \sum_{i=\sqrt{n}/2+1}^{\sqrt{n}} c(i)^3 \geq \sum_{i=\sqrt{n}/2+1}^{\sqrt{n}} c(\sqrt{n}/2)^3 = \sqrt{n}/2cn^{1.5}/8$$

Therefore,

$$\sum_{i=3}^{\sqrt{n}} ci^3 = \Omega(n^2)$$

Since $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$, we conclude that $T(n) = \Theta(n^2)$.

3.

```
    Func3(n)
1   s ← 0;
2   i ← n;
3   while (i < 5n^3) do
4   |   j ← 3n^3;
5   |   while (j > 18) do
6   |   |   s ← s + i − j;
7   |   |   j ← ⌊j/4⌋ ;                    /* Note: Division */
8   |   end
9   |   i ← 4 * i ;                         /* Note: Multiplication */
10  end
11  return (s);
```

Solution:

At the end of the k'th iteration of the inner while loop (steps 4-8), variable j equals $3n^3/4^k$. Inner while loop terminates when:

$$\begin{aligned}
3n^3/4^k &= 18, \text{ or} \\
3n^3/18 &= 4^k, \text{ or} \\
k &= \log_4(3n^3/18) = 3\log_4(n) + \log_4(3/18).
\end{aligned}$$

Thus the inner while loop takes $c\log_2(n)$ times for some constant c.

At the end of the k'th iteration of the outer while loop, variable i equals $n4^k$. Outer while loop terminates when:

$$\begin{aligned}
n4^k &= 5n^3, \text{ or} \\
4^k &= 5n^3/n = 5n^2, \text{ or} \\
k &= \log_4(5n^2) = 2\log_4(n) + \log_4(5).
\end{aligned}$$

Thus the outer while loop takes $c_2\log_2(n)$ times for some constant c_2.

Since the running time of the inner while loop does not depend upon the running time of the outer while loop, the total running time is $(c\log_2(n) * c_2\log_2(n)) = \Theta((\log_2(n))^2)$.