Introduction

The purpose of this lecture is to demonstrate how the analysis of time complexity values for $T(n)$ can be validated empirically.

Example 1

Find and verify the time complexity of the code fragment in Figure 1.

```
int sum=0;
for (int i=0;i<n;i++) sum++;
```

Figure 1. Program for Example 1.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	`int sum=0;`	1
2	`int i=0;`	1
3	`while (i<n) {`	$k = \sum_{i=0}^{n-1} 1 = n$
4	`sum++;`	k
5	`i++;`	k
6	`}`	1
	Total	$T(n) = 3k + 3$
	Total	$\boldsymbol{T(n) = 3n + 3}$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n){return 3*n+3;}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{   int c=0;
    int sum=0;          c++;
    int i=0;            c++;
    while (i<n){        c++;
        sum++;          c++;
        i++;            c++;
    }                   c++;
    return c;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{   o << endl;
    o << setw( 5) << "n";
    o << " ";
    o << setw(15) << "analytical";
    o << " ";
    o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << n;
    o << " ";
    o << setw(15) << a;
    o << " ";
    o << setw(15) << e;
}
int main()
{   int n[]={0,1,10,20,30,40,50};
    title(cout);
    for (int a=0;a<7;a++) row(cout,n[a],analytical(n[a]),empirical(n[a]));
    cout << endl;
    return 0;
}
```

Output

```
    n analytical  empirical
    0          3          3
    1          6          6
   10         33         33
   20         63         63
   30         93         93
   40        123        123
   50        153        153
```

Example 2

Find and verify the time complexity of the code fragment in Figure 2.

```
int sum=0;
for (int i=0;i<n;i+=2) sum++;
```

Figure 2. Program for Example 2.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	**int** *sum*=**0;**	1
2	**int** *i*=**0;**	1
3	**while (***i*<*n*) **{**	$k = \sum_{2i=0}^{n-1} 1 = \lceil \frac{n}{2} \rceil$
4	*sum*++**;**	k
5	*i*+=**2;**	k
6	**}**	1
	Total	$T(n) = 3k + 3$
	Total	$\boldsymbol{T(n) = 3\lceil \frac{n}{2} \rceil + 3}$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n)
{   int k=n/2;
    if (n%2) k++;
    return 3*k+3;
}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{   int c=0;
    int sum=0;                  c++;
    int i=0;                    c++;
    while (i<n){                c++;
        sum++;                  c++;
        i+=2;                   c++;
    }                           c++;
    return c;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{   o << endl;
    o << setw( 5) << "n";
    o << " ";
    o << setw(15) << "analytical";
    o << " ";
    o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << n;
    o << " ";
    o << setw(15) << a;
    o << " ";
    o << setw(15) << e;
}
int main()
{   int n[]={0,1,10,11,20,21,30,31,40,41,50,51};
    Title(cout);
    for (int a=0;a<12;a++) Row(cout,n[a],analytical(n[a]),empirical(n[a]));
    return 0;
}
```

Output

```
    n analytical  empirical
    0          3          3
    1          6          6
   10         18         18
   11         21         21
   20         33         33
   21         36         36
   30         48         48
   31         51         51
   40         63         63
   41         66         66
   50         78         78
   51         81         81
```

Example 3 Find and verify the time complexity of the code fragment in Figure 1.

```
int sum=0;
for (int i=0;i<n;i++)
    for (int j=0;j<n;j++)
        sum++;
```

Figure 3. Program for Example 3.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	`int sum=0;`	1
2	`int i=0;`	1
3	`while (i<n) {`	$a = \sum_{i=0}^{n-1} 1$
4	`int j=0;`	a
	`while (j<n) {`	$b = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} 1$
5	`sum++;`	b
6	`j++;`	b
7	`}`	a
8	`i++;`	a
9	`}`	1
	Total	$T(n) = 4a + 3b + 3$
a		$a = \sum_{i=0}^{n-1} 1 = n$
b		$b = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} 1 = \sum_{i=0}^{n-1} n = n^2$
	Total	$T(n) = 3n^2 + 4n + 3$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n)
{   return 3*n*n+4*n+3;
}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{   int c=0;
    int sum=0;          c++;
    int i=0;            c++;
    while (i<n){        c++;
        int j=0;        c++;
            sum++;      c++;
            j++;        c++;
        }               c++;
        i++;            c++;
    }                   c++;
    return c;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{   o << endl;
    o << setw( 5) << "n";
    o << " ";
    o << setw(15) << "analytical";
    o << " ";
    o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << n;
    o << " ";
    o << setw(15) << a;
    o << " ";
    o << setw(15) << e;
}
int main()
{   int n[]={0,1,10,20,30,40,50};
    Title(cout);
    for (int a=0;a<7;a++) Row(cout,n[a],analytical(n[a]),empirical(n[a]));
    return 0;
}
```

Output

```
    n analytical  empirical
    0          3          3
    1         10         10
   10        343        343
   20       1283       1283
   30       2823       2823
   40       4963       4963
   50       7703       7703
```

Example 4 Find and verify the time complexity of the code fragment in Figure 1.

```
int sum=0;
for (int i=0;i<n;i++)
    sum++;
for (int j=0;j<n;j++)
        sum++;
```

Figure 4. Program for Example 4.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	**int** *sum*=**0;**	1
2	**int** *i*=**0;**	1
3	**while (***i*<*n*) **{**	$a = \sum_{i=0}^{n-1} 1$
	sum++**;**	a
8	*i*++**;**	a
9	**}**	1
2	**int** *i*=**0;**	1
3	**while (***j*<*n*) **{**	$b = \sum_{j=0}^{n-1} 1$
4	*sum*++**;**	b
5	*j*++**;**	b
6	**}**	1
	Total	$T(n) = 3a + 3b + 5$
a		$a = \sum_{i=0}^{n-1} 1 = n$
b		$b = \sum_{i=0}^{n-1} 1 = n$
	Total	$T(n) = 6n + 5$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n)
{    return 6*n+5;
}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{    int c=0;
     int sum=0;                c++;
     int i=0;                  c++;
     while (i<n){              c++;
          sum++;               c++;
          i++;                 c++;
     }                         c++;
     int j=0;                  c++;
     while (j<n){              c++;
          sum++;               c++;
          j++;                 c++;
     }                         c++;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{    o << endl;
     o << setw( 5) << "n";
     o << " ";
     o << setw(15) << "analytical";
     o << " ";
     o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{    o << endl;
     o << setw( 5) << n;
     o << " ";
     o << setw(15) << a;
     o << " ";
     o << setw(15) << e;
}
int main()
{    int n[]={0,1,10,20,30,40,50};
     Title(cout);
     for (int a=0;a<7;a++) Row(cout,n[a],analytical(n[a]),empirical(n[a]));
     return 0;
}
```

Output

```
    n analytical  empirical
    0          5          5
    1         11         11
   10         65         65
   20        125        125
   30        185        185
   40        245        245
   50        305        305
```

Example 5 Find and verify the time complexity of the code fragment in Figure 1.

```
int sum=0;
for (int i=0;i<n;i++)
    for (j=0;j<n*n;j++)
        sum++;
```

Figure 5. Program for Example 5.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	**int** *sum*=**0;**	1
2	**int** *i*=**0;**	1
3	**while (***i*<*n*) **{**	$a = \sum_{i=0}^{n-1} 1$
4	**int** *j*=**0;**	a
		$b = \sum_{i=0}^{n-1} \sum_{j=0}^{n^2-1} 1$
5	**while (***j*<*n***n*) **{**	$2b$
6	*sum*++**;**	b
7	*j*++**;**	b
8	**}**	a
9	*i*++**;**	a
10	**}**	1
	Total	$T(n) = 4b + 4a + 3$

$\boldsymbol{a}$

$$a = \sum_{i=0}^{n-1} 1 = n$$

$\boldsymbol{b}$

$$b = \sum_{i=0}^{n-1} \sum_{j=0}^{n^2-1} 1 = \sum_{i=0}^{n-1} n^2 = n^3$$

Total $T(n) = 4n^3 + 4n + 3$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n)
{   return 4*n*n*n+4*n+3;
}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{   int c=0;
    int sum=0;                  c++;
    int i=0;                    c++;
    while (i<n) {               c++;
        int j=0;                c++;
        while (j<n*n) {         c+=2;
            sum++;              c++;
            j++;                c++;
        }                       c++;
        i++;                    c++;
    }                           c++;
    return c;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{   o << endl;
    o << setw( 5) << "n";
    o << " ";
    o << setw(15) << "analytical";
    o << " ";
    o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << n;
    o << " ";
    o << setw(15) << a;
    o << " ";
    o << setw(15) << e;
}
int main()
{   int n[]={0,1,10,20,30,40,50};
    Title(cout);
    for (int a=0;a<7;a++) Row(cout,n[a],analytical(n[a]),empirical(n[a]));
    return 0;
}
```

Output

```
    n analytical  empirical
    0          3          3
    1         11         11
   10       4043       4043
   20      32083      32083
   30     108123     108123
   40     256163     256163
   50     500203     500203
```

Example 6 Find and verify the time complexity of the code fragment in Figure 1.

```
int sum=0;
for (int i=0;i<n;i++)
    for (j=0;j<i;j++)
        sum++;
```

Figure 6. Program for Example 6.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	**int** *sum*=**0;**	1
2	**int** *i*=**0;**	1
3	**while** (*i*<*n*) **{**	$a = \sum_{i=0}^{n-1} 1$
4	**int** *j*=**0;**	a
5	**while** (*j*<*i*) **{**	$b = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1$
6	*sum*++**;**	b
7	*j*++**;**	b
8	**}**	a
9	*i*++**;**	a
10	**}**	1
	Total	$T(n) = 3b + 4a + 3$

$$a = \sum_{i=0}^{n-1} 1 = n$$

$$b = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} i = \frac{(n-1)n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

Total

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n + 3$$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n)
{   return (3*n*n+5*n)/2+3;
}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{   int c=0;
    int sum=0;              c++;
    int i=0;                c++;
    while (i<n) {           c++;
        int j=0;            c++;
        while (j<i) {       c++;
            sum++;          c++;
            j++;            c++;
        }                   c++;
        i++;                c++;
    }                       c++;
    return c;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{   o << endl;
    o << setw( 5) << "n";
    o << " ";
    o << setw(15) << "analytical";
    o << " ";
    o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << n;
    o << " ";
    o << setw(15) << a;
    o << " ";
    o << setw(15) << e;
}
int main()
{   int n[]={0,1,10,20,30,40,50};
    Title(cout);
    for (int a=0;a<7;a++) Row(cout,n[a],analytical(n[a]),empirical(n[a]));
    return 0;
}
```

Output

```
    n analytical  empirical
    0          3          3
    1          7          7
   10        178        178
   20        653        653
   30       1428       1428
   40       2503       2503
   50       3878       3878
```

Example 7 Find and verify the time complexity of the code fragment in Figure 1.

```
int sum=0;
while (n>1) {
    sum++;
    n=n/2;
}
```

Figure 7. Program for Example 7.

***Solution*: First, find an expression for $T(n)$**

Line	Statement	Cost
1	**int** *sum*=**0;**	1
2	**while (***n*>**1) {**	k
3	*sum*++**;**	k
4	*n*=*n*/**2;**	$2k$
5	**}**	1
	Total	$T(n) = 4k + 2$

$$n_{i+1} = \frac{n_i}{2}, 0 < n_k \leq 1$$

$n_0 = n_p$ where p signifies parameter. The initial value of *n*, n_0 is the value of *n* passed as the parameter of the function.

$$n_2 = \frac{n_1}{2} = \frac{n_0/2}{2} = \frac{n_0}{4} = \frac{n_0}{2^2}$$

$$n_k = \frac{n_0}{2^k}$$

Also, we know that in the k^{th} iteration, $n \leq 1$, and the loop terminates.

Assume $n_k = \frac{n_0}{2^k} = 1$, then $n_0 = 2^k$ and $k = log_2 n_0$

Now k is an integer value and $log_2 n_0$ does not always produce an integer value. From the previous lecture we know $k = \lfloor log_2 n_0 \rfloor$.

Total $T(n) = 4\lfloor log_2 n \rfloor + 2$

Next, create a C++ function that computes $T(n)$ analytically.

```
int analytical(int n)
{   if (n==0)
        return 2;
    else
        return 4*(int)floor(log2(n))+2;
}
```

Next, create a C++ function that computes $T(n)$ empirically.

```
int empirical(int n)
{   int c=0;
    int sum=0;              c++;
    while (n>1) {           c++;
        sum++;              c++;
        n=n/2;              c+=2;
    }                       c++;
    return c;
}
```

Now, write a program that exercises both functions and prints results

```
void title(ostream& o)
{   o << endl;
    o << setw( 5) << "n";
    o << " ";
    o << setw(15) << "analytical";
    o << " ";
    o << setw(15) << "empirical";
}
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << n;
    o << " ";
    o << setw(15) << a;
    o << " ";
    o << setw(15) << e;
}
int main()
{   int n[]={0,1,2,4,5,10,20,40,80,160,320,640,1280};
    Title(cout);
    for (int a=0;a<13;a++) Row(cout,n[a],analytical(n[a]),empirical(n[a]));
    return 0;
}
```

Output

```
    n analytical  empirical
    0           2          2
    1           2          2
    2           6          6
    4          10         10
    5          10         10
   10          14         14
   20          18         18
   40          22         22
   80          26         26
  160          30         30
  320          34         34
  640          38         38
 1280          42         42
```