**Introduction**

In this discussion, we give an overview of projects p03, p04, and p05. In these projects, the object is the same. The object of these projects is to find $T(n)$ and, subsequently, the time complexity, $O(f(n))$ of a given code fragment. We find the time complexity by first analyzing the code fragment and, then, testing our analysis by implementing the code fragment and testing for selected values of $n$. In every case our analytical results should match our empirical results.

```
int sum=0;
for (int a=n;a>0;a--) {
    for (int b=a;b>0;b--) {
        sum++;
    }
}
```

**Figure 1.** Code Fragment

| Line | Code | Cost |
|---|---|---|
| 1 | **int** *sum*=**0;** | $1$ |
| 2 | **int** *a*=*n*; | $1$ |
| 3 | **while** (*a*>**0**) **{** | $\sum_{a=1}^{n} 1 = n$ |
| 4 | **int** *b*=*a*; | $\sum_{a=1}^{n} 1 = n$ |
| 5 | **while** (*b*>**0**) **{** | $\sum_{a=1}^{n}\sum_{b=1}^{a} 1$ |
| 6 | *sum*++; | $\sum_{a=1}^{n}\sum_{b=1}^{a} 1$ |
| 7 | *b*--; | $\sum_{a=1}^{n}\sum_{b=1}^{a} 1$ |
| 8 | **}** | $\sum_{a=1}^{n} 1 = n$ |
| 9 | *a*--; | $\sum_{a=1}^{n} 1 = n$ |
| 10 | **}** | $1$ |

| | | |
|---|---|---|
| | **Total** | $T(n) = 3 + 4n + 3\sum_{a=1}^{n}\sum_{b=1}^{a} 1$ |
| | | $T(n) = \frac{3}{2}n^2 + \frac{3}{2}n + 4n + 3$ |
| | | $\sum_{a=1}^{n}\sum_{b=1}^{a} 1 = \sum_{a=1}^{n} a = \frac{1}{2}n^2 + \frac{1}{2}n$ |
| | | $T(n) = \frac{3}{2}n^2 + \frac{11}{2}n + 3$ |

**Figure 2.** Code Fragment Analysis

**Finding $O(f(n))$:**

1. $f(n) = n^2$
2. $C = \frac{3}{2} + 1 = \frac{5}{2}$
3. $|T(n_0)| \leq C|f(n_0)|$
   $\left|\frac{3}{2}n_0^2 + \frac{11}{2}n_0 + 3\right| \leq \frac{5}{2}|n_0^2|$
   $n_0^2 - \frac{11}{2}n_0 - 3 \geq 0$
   $n_0 = \left\lceil \frac{-\frac{11}{2} \pm \sqrt{\left(-\frac{11}{2}\right)^2 + 4 \cdot 3}}{2} \right\rceil = 1$

**$T(n)$ is $O(n^2)$ because we found witnesses $C = \frac{5}{2}$ and $n_0 = 1$ that make $|T(n_0)| \leq C|f(n_0)|$ for $n \geq n_0$**

```
//-----------------------------------------------------------------------------
//File p03.cpp is the example of Lecture 18 designed to show how project p03
//should be completed.
//-----------------------------------------------------------------------------
//Author:     Ms. Ada Lovelace
//Student ID: *00000000
//E-Mail:     alovelace@uco.edu
//Course:     CMSC 2123 - Discrete Structures
//CRN:        11094, Autumn, 2020
//Project:    p03
//Due:        October 2, 2020
//Account:    tt000
//-----------------------------------------------------------------------------
//C++ standard include files
//-----------------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <cstring>
#include <cstdlib>
#include <cmath>
using namespace std;
//-----------------------------------------------------------------------------
//Function af3 finds a value for T(n) analytically
//-----------------------------------------------------------------------------
int af3(int n)
{   return (3*n*n+11*n)/2+3;
}
//-----------------------------------------------------------------------------
//Function ef3 finds a value for T(n) empirically
//-----------------------------------------------------------------------------
int ef3(int n)
{   int c=0;
    int sum=0;                          c++;
    int a=n;                            c++;
    while (a>0) {                       c++;
        int b=a;                        c++;
        while (b>0) {                   c++;
            sum++;                      c++;
            b--;                        c++;
        }                               c++;
        a--;                            c++;
    }                                   c++;
    return c;
}
//-----------------------------------------------------------------------------
//Function title print a title for the columns of integers and their squares
//-----------------------------------------------------------------------------
void title(ostream& o,int fn)
{   o << endl;
    o << "Code Fragment " << fn;
    o << endl;
    o << setw( 5) << right << "n";
    o << " ";
    o << setw(10) << right << "analytical";
    o << " ";
    o << setw(10) << right << "empirical";
```

```
}
//-----------------------------------------------------------------------------
//Function row prints a row in the columns of integers and their squares
//-----------------------------------------------------------------------------
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << right << n;
    o << " ";
    o << setw(10) << right << a;
    o << " ";
    o << setw(10) << right << e;
}
//-----------------------------------------------------------------------------
//Function FragmentMgr copies the contents of one file to another
//-----------------------------------------------------------------------------
void FragmentMgr(istream& i,ostream& o)
{   title(o,3);
    for (;;) {
        int n;
        i >> n;
        if (i.eof()) break;
        row(o,n,af3(n),ef3(n));
    }
    o << endl;
}
//-----------------------------------------------------------------------------
//A CommandLineException is thrown when too many file names appear on the
//command line.
//-----------------------------------------------------------------------------
struct CommandLineException {
    CommandLineException(int m,int a)
    {   cout << endl;
        cout << "Too many file names on the command line.";
        cout << endl;
        cout << "A maximum of " << m << " file names can appear on the command
line.";
        cout << endl;
        cout << a << " file names were on the command line.";
        cout << endl;
    }
};
//-----------------------------------------------------------------------------
//A FileException is thrown when a file cannot be opened.
//-----------------------------------------------------------------------------
struct FileException {
    FileException(const char* fn)
    {   cout << endl;
        cout << "File " << fn << " could not be opened.";
        cout << endl;
    }
};
//-----------------------------------------------------------------------------
//Function main directs the process of copying one file to another
//-----------------------------------------------------------------------------
int main(int argc,char* argv[])
{   try {
        char ifn[255];                  //Input File Name
        char ofn[255];                  //Output File Name
```

```
        switch (argc) {                     //Process Command Line Arguments
            case 1:                         //Prompt for both file names
                cout << "Enter the input file name. ";
                cin  >> ifn;
                cout << "Enter the output file name. ";
                cin  >> ofn;
            break;
            case 2:                         //One file name on the command line.
                strcpy(ifn,argv[1]);        //Copy the input file name from argv.
                cout << "Enter the output file name. ";
                cin  >> ofn;
            break;
            case 3:                         //Two file names on the command line.
                strcpy(ifn,argv[1]);        //Copy the input file name from argv.
                strcpy(ofn,argv[2]);        //Copy the output file name from argv.
            break;
            default:                        //Too many file names on the command
line
                throw CommandLineException(2,argc-1);
             break;
         }//end switch
         ifstream i(ifn); if (!i) throw FileException(ifn);
         ofstream o(ofn); if (!o) throw FileException(ofn);
         FragmentMgr(i,o);
         o.close();
         i.close();
    } catch(...) {
         cout << endl;
         cout << "Program terminated!";
         cout << endl;
         cout << "I won't be back!";
         cout << endl;
         exit(EXIT_FAILURE);
    }
    return 0;
}
```

**Code Fragment 3**

| n | analytical | empirical |
|---|---|---|
| 0 | 3 | 3 |
| 10 | 208 | 208 |
| 20 | 713 | 713 |
| 30 | 1518 | 1518 |
| 40 | 2623 | 2623 |
| 50 | 4028 | 4028 |
| 60 | 5733 | 5733 |
| 70 | 7738 | 7738 |
| 80 | 10043 | 10043 |
| 90 | 12648 | 12648 |
| 100 | 15553 | 15553 |

| **while (*n*>1) *n*=*n*/4;** |
|---|

**Figure 3.** Code Fragment 4

| Line | Code | Cost |
|---|---|---|
| **1** | **while (*n*>1) {** | $k$ |
| **2** | ***n*=*n*/4;** | $2k$ |
| **3** | **}** | $1$ |
| | | $T(n) = 3k + 1$ |

**Figure 4.** Code Fragment 4 Analysis

Analysis:

1. $n_i = \frac{1}{4} n_{i-1}$
2. $n_1 = \frac{1}{4} n_0 = \frac{1}{4} n$
3. $n_2 = \frac{1}{4} n_1 = \frac{1}{4}\left(\frac{1}{4} n\right) = \frac{1}{4^2} n$
4. $n_k = \frac{1}{4^k} n$, where $k$ is the number of iterations
5. $n_k = 1$ on the $k^{th}$ iterations and, thus
6. $\frac{1}{4^k} n = 1 \Rightarrow n = 4^k \Rightarrow k = \log_4 n$

```
//-----------------------------------------------------------------------------
//File p04.cpp is the example of Lecture 18 designed to show how project p04
//should be completed.
//-----------------------------------------------------------------------------
//Author:     Ms. Ada Lovelace
//Student ID: *00000000
//E-Mail:     alovelace@uco.edu
//Course:     CMSC 2123 - Discrete Structures
//CRN:        11094, Autumn, 2020
//Project:    p04
//Due:        October 2, 2020
//Account:    tt000
//-----------------------------------------------------------------------------
//C++ standard include files
//-----------------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <cstring>
#include <cstdlib>
#include <cmath>
using namespace std;
//-----------------------------------------------------------------------------
//Function log4 returns the log to the base 4
//-----------------------------------------------------------------------------
double log4(double x){return log(x)/log(4.0);}
//-----------------------------------------------------------------------------
//Function af4 finds a value for T(n) analytically
//-----------------------------------------------------------------------------
int af4(int n)
{   if (n<1) return 1;
    return 3*round(log4((double)n))+1;
```

```
}
//-------------------------------------------------------------------------
//Function ef4 finds a value for T(n) empirically
//-------------------------------------------------------------------------
int ef4(int n)
{   int c=0;
    while (n>1) {                    c++;
        n=n/4;                       c+=2;
    }                                c++;
    return c;
}
//-------------------------------------------------------------------------
//Function title print a title for the columns of integers and their squares
//-------------------------------------------------------------------------
void title(ostream& o,int fn)
{   o << endl;
    o << "Code Fragment " << fn;
    o << endl;
    o << setw( 5) << right << "n";
    o << " ";
    o << setw(10) << right << "analytical";
    o << " ";
    o << setw(10) << right << "empirical";
}
//-------------------------------------------------------------------------
//Function row prints a row in the columns of integers and their squares
//-------------------------------------------------------------------------
void row(ostream& o,int n,int a,int e)
{   o << endl;
    o << setw( 5) << right << n;
    o << " ";
    o << setw(10) << right << a;
    o << " ";
    o << setw(10) << right << e;
}
//-------------------------------------------------------------------------
//Function FragmentMgr copies the contents of one file to another
//-------------------------------------------------------------------------
void FragmentMgr(istream& i,ostream& o)
{   title(o,4);
    for (;;) {
        int n;
        i >> n;
        if (i.eof()) break;
        row(o,n,af4(n),ef4(n));
    }
    o << endl;
}
//-------------------------------------------------------------------------
//A CommandLineException is thrown when too many file names appear on the
//command line.
//-------------------------------------------------------------------------
struct CommandLineException {
    CommandLineException(int m,int a)
    {   cout << endl;
        cout << "Too many file names on the command line.";
        cout << endl;
```

```
        cout << "A maximum of " << m << " file names can appear on the command
line.";
        cout << endl;
        cout << a << " file names were on the command line.";
        cout << endl;
    }
};
//-----------------------------------------------------------------------------
//A FileException is thrown when a file cannot be opened.
//-----------------------------------------------------------------------------
struct FileException {
    FileException(const char* fn)
    {   cout << endl;
        cout << "File " << fn << " could not be opened.";
        cout << endl;
    }
};
//-----------------------------------------------------------------------------
//Function main directs the process of copying one file to another
//-----------------------------------------------------------------------------
int main(int argc,char* argv[])
{   try {
        char ifn[255];                  //Input File Name
        char ofn[255];                  //Output File Name
        switch (argc) {                 //Process Command Line Arguments
            case 1:                     //Prompt for both file names
                cout << "Enter the input file name. ";
                cin  >> ifn;
                cout << "Enter the output file name. ";
                cin  >> ofn;
            break;
            case 2:                     //One file name on the command line.
                strcpy(ifn,argv[1]);    //Copy the input file name from argv.
                cout << "Enter the output file name. ";
                cin  >> ofn;
            break;
            case 3:                     //Two file names on the command line.
                strcpy(ifn,argv[1]);    //Copy the input file name from argv.
                strcpy(ofn,argv[2]);    //Copy the output file name from argv.
            break;
            default:                    //Too many file names on the command
line
                throw CommandLineException(2,argc-1);
             break;
         }//end switch
         ifstream i(ifn); if (!i) throw FileException(ifn);
         ofstream o(ofn); if (!o) throw FileException(ofn);
         FragmentMgr(i,o);
         o.close();
         i.close();
    } catch(...) {
         cout << endl;
         cout << "Program terminated!";
         cout << endl;
         cout << "I won't be back!";
         cout << endl;
         exit(EXIT_FAILURE);
    }
```

```
    return 0;
}
```

**Code Fragment 4**

| n | analytical | empirical |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 2 | 4 | 4 |
| 3 | 4 | 4 |
| 4 | 4 | 4 |
| 5 | 4 | 4 |
| 6 | 4 | 4 |
| 7 | 4 | 4 |
| 8 | 7 | 7 |
| 15 | 7 | 7 |
| 16 | 7 | 7 |
| 17 | 7 | 7 |
| 63 | 10 | 10 |
| 64 | 10 | 10 |
| 65 | 10 | 10 |

```
for (int a=0;a<n;a++) {
    int m=n;
    while (m>1) m=/4;
}
```

**Figure 5.** Code Fragment A

| Line | Code | Cost |
|---|---|---|
| | **int** *a*=**0;** | $1$ |
| | **while (***a*<*n*) **{** | $\sum_{a=0}^{n-1} 1 = n$ |
| | **int** *m*=*n*; | $n$ |
| | **while (***m*>**1) {** | $nk$ |
| | *m*=*m*/**4;** | $2nk$ |
| | **}** | $n$ |
| | **a++;** | $n$ |
| | **}** | $1$ |
| | | $T(n) = 3nk + 4n + 2$ |
| | | Finding $k$ |
| | | $m_0 = n$ |
| | | $m_i = \frac{1}{4} m_{i-1}$ |
| | | $m_1 = \frac{1}{4} m_0 = \frac{1}{4} n$ |
| | | $m_2 = \frac{1}{4} m_1 = \frac{1}{4}\left(\frac{1}{4} n\right) = \frac{1}{4^2} n$ |
| | | $m_k = \frac{1}{4^k} n$ |
| | | $m_k = 1$ |
| | | $1 = \frac{1}{4^k} n$ |
| | | $4^k = n$ |
| | | $\log_4 4^k = \log_4 n$ |
| | | $k \log_4 4 = \log_4 n$ |
| | | $k = \log_4 n$ |
| | | $k = round(\log_4 n)$ |
| | | $T(n) = 3n \times round(\log_4 n) + 4n + 2$ |

**Figure 6.** Code Fragment A Analysis