CSI 3334 Data Structures
Homework 1

Yufan Xu

February 5, 2018

Do your own work for this assignment; do not work with others. Consult the book and your professor for help if you need it. Please write neatly, or preferably type your answers. Use good grammar, correct spelling, and complete sentences.

1. (30 points) Exercise 2.7 in your textbook. For parts 1-4, label each finite-time statement with its own constant as we have in class (i.e. c1, c2, etc.). For example, code fragment (1) should be labeled as:

```
sum = 0;                      // c1
for (int i = 0;               // c2
              i < n;          // c3
                     i++) {   // c4
    sum++;                    // c5
}
```

   For parts 2-6,

   - show how you assigned constants for each code fragment,
   - give a precise T(n)
   - simplify the precise T(n)
   - giving your big-Oh answer
   - Give your code that tests and times each function
   - Construct a table for 5 different values of N and the runtime for each case

   When timing these code fragments, use values of n that vary enough to show clear trends in the running times. Also, you should run each function many times in a row so that the running time is not zero (which isn't meaningful). You can time your code by calling the C function clock() before the code fragment runs, saving its result, then running the code fragment many times, then calling clock() again and finding the difference between the two calls. The value that clock returns is in units of CLOCKS_PER_SEC. Both clock and CLOCKS_PER_SEC are defined in the <ctime>library.

Solution to (2):

```
sum = 0;                                // c1(1)
for (int i = 0;                         // c2(1)
               i < n;                   // c3(n+1)
                      ++i) {            // c4(n)
      for (int j = 0;                   // c5(1)n
                     j  < n;            // c6(n+1)n
                            ++j) {      // c7(n)n
          ++sum;                        // c8(n)n
      }
}
```

$T(n) = c_1(1) + c_2(1) + c_3(n+1) + c_4(n) + c_5(1)n + c_6(n+1)n + c_7(n^2) + c_8(n^2)$

$T(n) = c_1 + c_2 + c_3n + c_3 + c_4n + c_5n + c_6n^2 + c_6n + c_7n^2 + c_8n^2$

$T(n) = c_1 + c_2 + c_3 + (c_3 + c_4 + c_5 + c_6)n + (c_6 + c_7 + c_8)n^2$

So $T(n) = c_9 + c_{10}n + c_{11}n^2$, where $c_9 = c_1 + c_2 + c_3$, $c_{10} = c_3 + c_4 + c_5 + c_6$, and $c_{11} = c_6 + c_7 + c_8$

So, $T(n) = \Theta(c_9 + c_{10}n + c_{11}n^2)$

$T(n) = O(n^2)$

Solution to (3):

```
sum = 0;                                // c1(1)
for (int i = 0;                         // c2(1)
               i < n;                   // c3(n+1)
                      ++i) {            // c4(n)
      for (int j = 0;                   // c5(1)n
                     j  < n*n;          // c6(n^2+1)n
                            ++j) {      // c7(n^2)n
          ++sum;                        // c8(n^2)n
      }
}
```

$T(n) = c_1(1) + c_2(1) + c_3(n+1) + c_4(n) + c_5(1)n + c_6(n^2+1)n + c_7(n^2)n + c_8(n^2)n$

$T(n) = c_1 + c_2 + c_3n + c_3 + c_4n + c_5n + c_6n^3 + c_6n + c_7n^3 + c_8n^3$

$T(n) = c_1 + c_2 + c_3 + (c_3 + c_4 + c_5 + c_6)n + (c_6 + c_7 + c_8)n^3$

So $T(n) = c_9 + c_{10}n + c_{11}n^3$, where $c_9 = c_1 + c_2 + c_3$, $c_{10} = c_3 + c_4 + c_5 + c_6$, and $c_{11} = c_6 + c_7 + c_8$

So, $T(n) = \Theta(c_9 + c_{10}n + c_{11}n^3)$

$T(n) = O(n^3)$

Solution to (4):

```
sum = 0;                                // c1(1)
for (int i = 0;                         // c2(1)
               i < n;                   // c3(n+1)
                      ++i) {            // c4(n)
      for (int j = 0;                   // c5(1)n
                     j  < i;            // c6(n)n
                            ++j) {      // c7(n-1)n
          ++sum;                        // c8(n-1)n
      }
}
```

$T(n) = c_1(1) + c_2(1) + c_3(n+1) + c_4(n) + c_5(1)n + c_6(n)n + c_7(n-1)n + c_8(n-1)n$

$T(n) = c_1 + c_2 + c_3n + c_3 + c_4n + c_5n + c_6n^2 + c_7n^2 - c_7 + c_8n^2 - c_8$

$T(n) = (c_1 + c_2 + c_3 - c_7 - c_8) + (c_3 + c_4 + c_5)n + (c_6 + c_7 + c_8)n^2$

So $T(n) = c_9 + c_{10}n + c_{11}n^2$, where $c_9 = c_1 + c_2 + c_3 - c_7 - c_8$, $c_{10} = c_3 + c_4 + c_5$, and $c_{11} = c_6 + c_7 + c_8$

So, $T(n) = \Theta(c_9 + c_{10}n + c_{11}n^2)$

$T(n) = O(n^2)$

Solution to (5):

```
sum = 0;                                   // c1(1)
for (int i = 0;                            // c2(1)
               i < n;                      // c3(n+1)
                      ++i) {               // c4(n)
      for (int j = 0;                      // c5(1)n
                     j < i*i;              // c6((n-1)(2n-1)n+n)
                            ++j) {         // c7(n-1)(2n-1)n
            for(int k = 0;                 // c8(n-1)(2n-1)n
                          k < j;           // c9(n-1)(2n-1)n*n^2
                               ++k) {      // c10((n-1)(2n-1)n*n^2 - n^2)
               ++sum;                      // c11((n-1)(2n-1)n*n^2 - n^2)
            }
      }
}
```

$T(n) = c_1(1) + c_2(1) + c_3(n+1) + c_4(n) + c_5(1)n + c_6((n-1)(2n-1)n + n) + c_7(n-1)(2n-1)n + c_8(n-1)(2n-1)n + c_9(n-1)(2n-1)n * n^2 + c_{10}((n-1)(2n-1)n * n^2 - n^2) + c_{11}((n-1)(2n-1)n * n^2 - n^2)$

$T(n) = c_1 + c_2 + c_3 n + c_3 + c_4 n + c_5 n + c_6((n-1)(2n-1)n + n) + c_7(n-1)(2n-1)n + c_8(n-1)(2n-1)n + c_9(n-1)(2n-1)n * n^2 + c_{10}((n-1)(2n-1)n * n^2 - n^2) + c_{11}((n-1)(2n-1)n * n^2 - n^2)$

So $T(n) = \Theta(c_{12} + c_{13}n + c_{14}n^3 + c_{15}n^5)$

$T(n) = O(n^5)$

Solution to (6):

```
sum = 0;                                   // c1(1)
for (int i = 0;                            // c2(1)
               i < n;                      // c3(n+1)
                      ++i) {               // c4(n)
      for (int j = 0;                      // c5(1)n
                     j < i*i;              // c6((n-1)(2n-1)n+n)
                            ++j) {         // c7(n-1)(2n-1)n
             if (j % i == 0) {             // c8(n+1)*n
                for(int k = 0;             // c9(n+1)*n
                              k < j;       // c10(n-1)(2n-1)n*n
                                   ++k) {  // c11((n-1)(2n-1)n*n -n)
                   ++sum;                  // c12((n-1)(2n-1)n*n -n)
                  }
            }
      }
}
```

$T(n) = c_1(1) + c_2(1) + c_3(n+1) + c_4(n) + c_5(1)n + c_6((n-1)(2n-1)n + n) + c_7(n-1)(2n-1)n + c_8(n+1) * n + c_9(n+1) * n + c_{10}(n-1)(2n-1)n * n + c_{11}((n-1)(2n-1)n * n - n) + c_{12}((n-1)(2n-1)n * n - n)$

$T(n) = c_1 + c_2 + c_3 n + c_3 + c_4 n + c_5 n + c_6((n-1)(2n-1)n + n) + c_7(n-1)(2n-1)n + c_8(n+1) * n + c_9(n+1) * n + c_{10}(n-1)(2n-1)n * n + c_{11}((n-1)(2n-1)n * n - n) + c_{12}((n-1)(2n-1)n * n - n)$

So $T(n) = \Theta(c_{13} + c_{14}n + c_{15}n^2 + c_{16}n^3 + c_{17}n^4$

$T(n) = O(n^4)$

Code for (2):

```
int main() {

    int n = 10;
    unsigned int start, stop;

    int count = 0;
    start = clock();
    while (clock() - start <= 2000) {
        loop1(n);
        count++;
    }
    stop = clock();
    cout << static_cast<double>(stop - start)/ count<< endl;
}

void loop1(int n){
    int sum = 0;
    for (int i = 0;i < n; ++i) {
        for (int j = 0; j <n; ++j) {
            ++sum;
        }
    }
}
```

when n = 10, time is 0.52879

when n = 20, time is 1.22354

when n = 40, time is 4.04286

when n = 80, time is 13.9742

when n = 160, time is 48.878

As a result, we can know that the algorithm grows at $O(n^2)$ time.

Code for (3):

```
int main() {

    int n = 10;
    unsigned int start, stop;

    int count = 0;
    start = clock();
    while (clock() - start <= 2000) {
        loop1(n);
        count++;
    }
    stop = clock();
    cout << static_cast<double>(stop - start)/ count<< endl;
}

void loop1(int n){
    int sum = 0;
    for (int i = 0;i < n; ++i) {
        for (int j = 0; j <n*n; ++j) {
            ++sum;
        }
    }
}
```

when n = 10, time is 2.59533

when n = 20, time is 19.0762

when n = 40, time is 139.467

when n = 80, time is 1171.5

when n = 160, time is 9187

As a result, we can know that this algorithm grows at $O(n^3)$ time

Code for (4):

```
int main() {

    int n = 10;
    unsigned int start, stop;

    int count = 0;
    start = clock();
    while (clock() - start <= 2000) {
        loop1(n);
        count++;
    }
    stop = clock();
    cout << static_cast<double>(stop - start)/ count<< endl;
}

void loop1(int n){
    int sum = 0;
    for (int i = 0;i < n; ++i) {
        for (int j = 0; j < i; ++j) {
            ++sum;
        }
    }
}
```

when n = 100, time is 11.2179

when n = 200, time is 41.875

when n = 400, time is 158.923

when n = 800, time is 625

when n = 1600, time is 2365

As a result, this algorithm grows at $O(n^2)$ time

Code for (5):

```
int main() {

    int n = 10;
    unsigned int start, stop;

    int count = 0;
    start = clock();
    while (clock() - start <= 2000) {
        loop1(n);
        count++;
    }
    stop = clock();
    cout << static_cast<double>(stop - start)/ count<< endl;
}

void loop1(int n){
    int sum = 0;
    for (int i = 0;i < n; ++i) {
        for (int j = 0; j < i*i; ++j) {
            for (int k = 0; k < j; ++k) {
                ++sum;
            }
        }
    }
}
```

when n = 5, time is 0.808485

when n = 10, time is 16.9076

when n = 20, time is 580

when n = 40, time is 18440

when n = 80, time is 601898

As a result, this algorithm grows at $O(n^5)$ time

Code for (6):

```
int main() {

    int n = 10;
    unsigned int start, stop;
    int count = 0;
    start = clock();
    while (clock() - start <= 2000) {
        loop1(n);
        count++;
    }
    stop = clock();
    cout << static_cast<double>(stop - start)/ count<< endl;
}

void loop1(int n){
    int sum = 0;
    for (int i = 0;i < n; ++i) {
        for (int j = 0; j < i*i; ++j) {
            if (j % i == 0) {
                for (int k = 0; k < j; ++k) {
                    ++sum;
                }
            }
        }
    }
}
```

when n = 10, time is 3.57041

when n = 20, time is 48.0714

when n = 40, time is 704

when n = 80, time is 9826

when n = 160, time is 162473

As a result, this algorithm grows at $O(n^4)$ time