Assignment 5: Big-O Sorting
Part 1: Runtime
CS3305/W01 Data Structures

Casey Hampson

September 18, 2024

Solution

1. The code:

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

   will run at $\mathcal{O}(n)$ time, since each loop is just incrementing a variable, which runs at constant time.

2. We are considering the code:

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

   For each iteration of the outer loop, the inner loop runs n times, and each iteration of the inner loop runs in constant time (like we just found), so we have that

$$T(n) = c * n * n \to \mathcal{O}(n^2).$$

3. For the code

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

we have an almost identical case to the previous one, except that this inner loop runs $n * n$ times, so

$$T(n) = c * n * n * n \to \mathcal{O}(n^3).$$

4. We are considering the code:

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

This time, we can write out the series for some arbitrarily large n. Each iteration of the outer loop increases the iterations of the inner loop by 1, which itself runs at constant time (i'll just let this constant $c = 1$), so we have

$$T(n) = 1 + 2 + 3 + \ldots + (n-1) = \frac{n(n-1)}{2} \to \mathcal{O}(n^2).$$

5. We are considering the code:

```
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++;
        }
    }
}
```

We need to look at this one a little more closely. For some value of i during one of the n iterations of the outer loop, the middle loop will then run i^2 times, so the innermost loop does its full loop that many times, where each time it does its full loop, its number of iterations increases up from 1 to $i^2 - 1$. This particular series looks like:

$$T(i) = 1 + 2 + 3 + \ldots + (i^2 - 1) = \frac{i^2(i^2-1)}{2} \sim i^4.$$

So, for each value of i or each iteration of the outermost loop, we increment our variable (on the order of) i^4 times. The full series looks like

$$\sum_{n=0}^{n-1} i^4.$$

I'm honestly not entirely sure how we are meant to approach this, since the form of this sum isn't similar to any of the cases presented in the book, but as a physicist my intuition is to take the limiting case for large n, where the sum turns into an integral. Since we basically are just tossing away constants, it makes things really easy:

$$\lim_{n\to\infty} \sum_{n=0}^{n-1} i^4 \sim \int_0^n \mathrm{d}i \; i^4 \sim n^5,$$

So, this particular code has $\mathcal{O}(n^5)$.