Self-Practice Week 1 - Fundamentals (analysis)

Run-time Analysis of Algorithms

The goal of this assignment is to understand how to perform time complexity analysis of algorithms. For each of the following fragments of code, determine the number of times `op()` is called as a function of the input size `n`. Express your answer in terms of the Big-Theta Θ notation.

You might need a quick math refresher on logarithms and summations (e.g., https://www.tug.org/texshowcase/cheat.pdf)

Remember that when studying order of growth, we drop constant coefficients.

1. Single Loops

```
for (i=10; i<n+5; i=i+2)
    op();
```

$T(n) = \frac{n+5-10}{2} = \frac{n-5}{2}$

$\Rightarrow \Theta(n)$

```
for (i=1; i<n; i= i*2)
    op();
```

$\Theta(\log n)$

```
for (i=n; i>1; i=i/2)
    op();
```

$\Theta(\log n)$

```
for (i=0; i*i < n; i++)
    op();
```

$\Theta(\text{sqrt } n) \;/\; \Theta(n^{\frac{1}{2}})$

2. Nested Independent Loops

```
for (i=0; i<n; i++)
    for (j=0; j < 100; j++)
        op();
```

$T(n) = 100 * n$

$\Rightarrow \Theta(n)$

```
for (i=0; i<n; i++){
    for (j=0; j < n; j++)
        op();

    for (j=1; j<=n; j=j*2)
        op();
}
```

n

$T(n) = n(n + \log n)$

$= n^2 + n\log n.$

$\Rightarrow \Theta(n^2)$

3. Nested Dependent Loops

```
for (i=1; i<=n; i++)
    for (j=1; j <=i; j++)
        op();
```

$T(n) = \sum_{i=1}^{n} i = 1+2+3+\cdots+n = \frac{(1+n)n}{2}$

$\Rightarrow \Theta(n^2)$

```
for (i=1; i<=n; i++)
    for (j=1; j <=i; j++)
        for (k=1; k<=i; k++)
            op();
```

$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

$\Rightarrow \Theta(n^3)$

```
for (i=1; i<=n; i++)
    for (j=1; j <=i; j++)
        for (k=1; k<=j; k++)
            op();
```

$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{j} = \sum_{i=1}^{n} \sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{i(i+1)}{2}$

$= \frac{1}{2}\sum_{i=1}^{n}(i^2+i) = \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right) = \Theta(n^3)$

```
for (i=1; i<=n; i=i*2)
    for (j=1; j <=i; j++)
        op();
```

$T(n) = 1+2+4+8+\cdots+n$

$= 2^{\log_2 n+1} - 1 = 2n+1$

$\Rightarrow \Theta(n)$

```
for (i=1; i<=n; i++)
    for (j=1; j <i; j=j*2)
        op();
```

$T(n) = n \cdot (\log n) = n \log n$

$\Theta(n \log n)$

4. Recursion

```
void f (int n) {
    if (n==0) return;
    op();
    f(n-1);
}
```

$\Theta(n)$

```
void f (int n) {
    if (n==0) return;
    op();
    f(n/2)
}
```

$\Theta(\log n)$

```
void f (int n) {
    if (n==0) return;
    for (i=0; i<n; i++)
        op();
    f(n-1);
}
```

$\Theta(n^2)$

$T(n) = 2^0 + 2^1 + 2^2 + \cdots + 2^{\log_2 n}$

```
void f (int n) {
    if (n==0) return;
    op();
    f(n/2);
    f(n/2);
}
```

$= 2^{\log_2 n+1} - 1 = 2n - 1$

$\Rightarrow \Theta(n)$

```
void f (int n) {
    if (n==0) return;
    op();
    f(n-1);
    f(n-1);
}
```

$T(n) = 2^0 + 2^1 + 2^2 + \cdots + 2^n$

$= 2^{n+1} \Rightarrow \Theta(n^2)$