# **Analysis of Algorithms:** More Practice Examples

This worksheet complements the lecture slides on runtime analysis. You might also need the math cheatsheet.

**Question.** For each of the following pieces of code, find the number of times **op()** is called as a function of the input size $n$. Express your answer using Big-$O$ notation.

**Ex.1**

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

**Solution.** $O(n^3)$.

**Ex.2**

```
for (i = 0; i < n; i++)
   for (j = i - 4; j < i; j++)
      op()
```

**Solution.** $O(n)$.

The inner loop always repeats 4 times. The outer loop repeats $n$ times.

**Ex.3**

```
for (i = 1; i < n; i *= 4)
   for (j = 4; j < n; j++)
      op()
```

**Solution.** $O(n \log n)$.

**op()** is called $\log_4 n \times (n-4) = n \log_4 n - 4 \log_4 n$ times. We drop the coefficients and lower order terms.

**Ex.4**

```
for (i = 1; i < n; i *= 2)
   for (j = 1; j < n; j *= 3)
      for (k = 1; k <= n; k++)
         op()
```

**Solution.** $O(n \log^2 n)$.

**op()** is called $\log_2(n) \times \log_3(n) \times n$ times (analyze each loop separately, multiply and drop the log bases).

**Ex.5**

```
for (i = 1; i < 64; i *= 2)
   for (j = 1; j <= 100; j++)
      for (k = 1; k <= 100; k += 5)
         op()
```

**Solution.** $O(1)$.

**op()** is called a constant number of times ( $\log_2(64) \times 100 \times 20$ )

**Ex.6**

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

**Solution.** $O(n^2)$.

**op()** is called $n-1$ times when $i = 1$, $n-2$ times when $i = 2$, $n-3$ times when $i = 3$, etc.

$$\text{Total} = (n-1) + (n-2) + (n-3) + \ldots + 1$$

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

**Ex.7**

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

**Solution.** $O(n)$.

Explanation. The following is a trace of the code:

| i | j | *number of* op() *calls* |
|---|---|---|
| 1 | [1] | 1 = $2^0$ |
| 2 | [1, 2] | 2 = $2^1$ |
| 4 | [1, 2, 3, 4] | 4 = $2^2$ |
| 8 | [1, 2, 3, …, 8] | 8 = $2^3$ |
| ... | ... | ... |
| n | [1, 2, 3, …, n] | n = $2^k$, $(k = \log_2 n)$ |

$$\text{Total} = 2^0 + 2^1 + 2^2 + \ldots + 2^{\lg n}$$

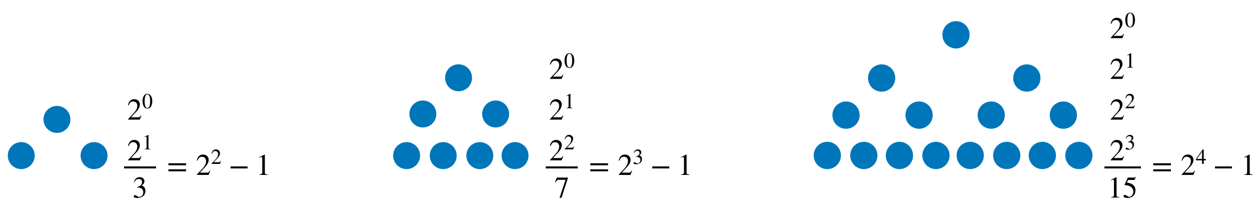
$$= \sum_{i=1}^{\lg n} 2^i = 2^{\lg n \; + 1} - 1 \quad = \quad 2n - 1$$

- Using the identity $a^m \times a^n = a^{m+n}$, we can represent the answer as $2^1 \times 2^{\lg n} - 1$.
- Using the identity $x^{\lg y} = y^{\lg x}$, we can represent the answer as $2^1 \times n^{\lg 2} - 1 = 2^1 \times n^1 - 1$.

This is a geometric sum that can be calculated using the geometric sum formula:

$$\sum_{i=0}^{m} r^i = \frac{r^{m+1} - 1}{r - 1}$$

where $m = \lg n$ and $r = 2$ in this exercise. The following is a visual explanation for this special case:




For extra Muscles. Hint 1: Find a sum and then solve the sum. Hint 2: Refer to the math cheatsheet.

```
for (i = 1; i < n; i *= 2)
   for (j = 4; j < i; j++)
       op()
```

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

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

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