

Learning Objectives

By the end of this worksheet, you will:

- Analyse the running time of functions containing nested loops.

1. **Nested loop variations.** Each of the following functions takes as input a non-negative integer n and contains at least one nested loop. For each loop, determine the *exact* number of iterations that will occur (in terms of n), and then use this to determine a Theta expression for the running time of each function. For nested loops, do the following:

- First, determine an expression for the exact cost (# of steps) of the inner loop for a fixed iteration of the outer loop. This may or may be the same for each iteration of the outer loop.
- Determine the total cost of the outer loop by adding up the costs of the inner loop from (i). If the cost of the inner loop is the same for each outer loop iteration, you can simply multiply this cost by the total number of iterations of the outer loop. Otherwise, you'll need to set up and evaluate a summation (Σ).
- Repeat steps (i) and (ii) if there is more than one level of nesting, starting from the innermost loop and working your way outwards. Your final result should depend *only* on the function input, not any loop variables.

(a)

```

1 def f1(n: int) -> None:
2     i = 0
3     while i < n:
4         j = 0
5         while j < n:
6             j = j + 1
7         i = i + 5

```

Solution

For a fixed iteration of the outer loop, the inner loop takes exactly n iterations, with each iteration costing a single step. This is the same for each iteration of the outer loop.

There are $\left\lceil \frac{n}{5} \right\rceil$ iterations of the outer loop. We multiply with this by the cost of each outer loop iteration to get a total cost of $\left\lceil \frac{n}{5} \right\rceil \cdot n$, which is $\Theta(n^2)$.

(b)

```

1 def f2(n: int) -> None:
2     i = 4
3     while i < n:
4         j = 1
5         while j < n:
6             j = j * 3
7         k = 0
8         while k < n:
9             k = k + 2
10        i = i + 1

```

Solution

The first inner loop takes $\lceil \log_3 n \rceil$ iterations,* and each iteration counts as a single step. The second inner loop takes $\left\lceil \frac{n}{2} \right\rceil$ iterations, with each iteration also costing a single step. Again, this is the same for each iteration of the outer loop.

There are $\max(n - 4, 0)$ iterations of the outer loop, for a total cost of $\max(n - 4, 0) \cdot \left(\lceil \log_3 n \rceil + \left\lceil \frac{n}{2} \right\rceil \right)$, which is $\Theta(n^2)$.

*See Worksheet 13, Question 2.

(c)

```

1 def f3(n: int) -> None:
2     i = 0
3     while i < n:
4         j = n
5         while j > 0:
6             k = 0
7             while k < j:
8                 k = k + 1
9             j = j - 1
10            i = i + 4

```

Solution

We start with the innermost loop (with k) for a fixed iteration of the other loops. This executes exactly j times, with each iteration costing a single step. This cost depends on the value of j .

Next, we consider the cost of the middle loop for a fixed iteration of the outer loop. We need to add up the cost of the innermost loop for each value of j from n down to 1. (The summation expression will be the same as if j went up from 1 to n .) This total cost is

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

This cost does *not* depend on the iteration number of the outermost loop.

Finally, the outer loop has $\left\lceil \frac{n}{4} \right\rceil$ iterations. This means that the total cost is $\left\lceil \frac{n}{4} \right\rceil \cdot \frac{n(n+1)}{2}$, which is $\Theta(n^3)$.

(d)

```

1 def f4(n: int) -> None:
2     i = 1
3     while i < n:
4         j = 0
5         while j < i:
6             j = j + 1
7
8         i = i * 2

```

Note: you can look up a formula for “sum of powers of 2” or “geometric series” for the analysis in this question. This analysis is trickier than the others.

Solution

The inner loop takes exactly i iterations for a fixed iteration of the outer loop, and the cost per iteration is 1 step.

The outer loop runs $\lceil \log n \rceil$ iterations, where i takes on the values $2^0, 2^1, 2^2, \dots, 2^{\lceil \log n \rceil - 1}$. So the total running time of this function is:

$$\sum_{i=0}^{\lceil \log n \rceil - 1} 2^i = 2^{\lceil \log n \rceil} - 1$$

So the total cost is $2^{\lceil \log n \rceil} - 1$, which is $\Theta(n)$.*

*It is a good exercise to prove that $2^{\lceil \log n \rceil} \in \Theta(n)$. Helpful: for all $x \in \mathbb{R}$, $0 \leq \lceil x \rceil - x < 1$.

2. Consider the following algorithm:

```

1 def max_subsequence_sum(lst: List[int]) -> int:
2     n = len(lst)
3     max_so_far = 0
4     for i in range(n):           # Loop 1: i goes from 0 to n-1
5         for j in range(i, n):    # Loop 2: j goes from i to n-1
6             s = 0
7             for k in range(i, j + 1): # Loop 3: k goes from i to j
8                 s = s + lst[k]
9             if max_so_far < s:
10                 max_so_far = s
11 return max_so_far

```

Determine the Theta bound on the running time of this function in terms of n , the length of the input list. For practice, do not make any approximations on the *number of iterations* of any of the three loops; that is, your analysis should actually calculate the total number of iterations of the innermost k -loop across all iterations of the outer loop. Go slow! Treat this as a valuable exercise in performing calculations with summation notation.

You may find the following formulas helpful (valid for all $n, a, b \in \mathbb{Z}^+$):

$$\sum_{i=0}^n i = \frac{n(n+1)}{2}, \quad \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=a}^b f(i) = \sum_{i'=0}^{b-a} f(i' + a)$$

Solution

As usual, we start with the innermost loop (Loop 3). For a fixed iteration of Loops 1 and 2, the Loop 3 takes exactly $j - i + 1$ iterations, with a cost of 1 step per iteration. So the cost of the innermost loop for a fixed iteration of Loops 1 and 2 is $j - i + 1$. This number of iterations depends on both i and j .

Now consider the cost of all iterations of Loop 2 for a fixed iteration of Loop 1. We need to sum the cost of Loop 3 for all values of j from i to $n - 1$. The total cost for a fixed iteration of the outer Loop 1 is:

$$\begin{aligned}
 \sum_{j=i}^{n-1} j - i + 1 &= \sum_{j'=1}^{n-i} j' && \text{(change of variable } j' = j - i + 1) \\
 &= \sum_{j'=1}^{n-i} j' \\
 &= \frac{(n-i)(n-i+1)}{2} \\
 &= \frac{n^2 - 2ni + n - i + i^2}{2} \\
 &= \frac{1}{2}i^2 - \left(n + \frac{1}{2}\right)i + \left(\frac{n^2}{2} + \frac{n}{2}\right)
 \end{aligned}$$

[Note: we're grouping by i here because the next step is to sum over possible values of i .]

To obtain the total cost, we have to add up the cost of each iteration of Loop 1, as i goes from 0 to $n - 1$:

$$\begin{aligned}
 & \sum_{i=0}^{n-1} \left[\frac{1}{2}i^2 - \left(n + \frac{1}{2}\right)i + \left(\frac{n^2}{2} + \frac{n}{2}\right) \right] \\
 &= \frac{1}{2} \sum_{i=0}^{n-1} i^2 - \left(n + \frac{1}{2}\right) \sum_{i=0}^{n-1} i + \left(\frac{n^2}{2} + \frac{n}{2}\right) \sum_{i=0}^{n-1} 1 \\
 &= \frac{1}{2} \left(\frac{(n-1)n(2n-1)}{6} \right) - \left(n + \frac{1}{2}\right) \left(\frac{(n-1)n}{2} \right) + \left(\frac{n^2}{2} + \frac{n}{2}\right) n \\
 &= \frac{(n-1)n(2n-1)}{12} - \frac{(2n+1)(n-1)n}{4} + \frac{n^3}{2} + \frac{n^2}{2} \\
 &= -\frac{(n-1)n(n+1)}{3} + \frac{n^3}{2} + \frac{n^2}{2} \\
 &= -\frac{n^3 - n}{3} + \frac{n^3}{2} + \frac{n^2}{2} \\
 &= \frac{n^3}{6} + \frac{n^2}{2} + \frac{n}{3}
 \end{aligned}$$

So then the total running time of `max_subsequence_sum` is $\frac{n^3}{6} + \frac{n^2}{2} + \frac{n}{3}$, which is $\Theta(n^3)$.