CSC110 Lecture 23: More Running-Time Analysis

Hisbaan Noorani

November 04, 2020

Contents

1 Exercise 1: Analysing program runtime of while loops

Your task here is to **analyse the running time of each of the following functions**. Recall the technique from lecture for calculating the number of iterations a while loop takes:

1. Find i_0, the value of the loop variable at the start of the first iteration. (You may need to change the notation depending on the loop variable name.)
2. Find a pattern for i_0, i_1, i_2, etc. based on the loop body, and a formula for a general i_k, the value of the loop variable after k loop iterations, assuming that many iterations occurs.
3. Find the *smallest* value of k that makes the loop condition False. This gives you the number of loop iterations.

   You'll need to use the floor/ceiling functions to get the correct exact number of iterations.

Note: each loop body runs in constant time in this question. While this won't always be the case, such examples allow you to focus on just counting loop iterations here.

```
1 1.    def f1(n: int) -> None:
2           """Precondition: n >= 3."""
3           i = 3
4           while i < n:
5               print(i)
6               i = i + 5
```

$i_0 = 3$

$i_k = 3 + 5k$

$n \leq 3 + 5k$

$k \geq \frac{n-3}{5}$

$RT_{f_1} = \lceil \frac{n-3}{5} \rceil$

$RT_{f_1} \in \Theta(n)$

12.
```python
def f2(n: int) -> None:
    """Preconditions: n > 0 and n % 10 == 0."""
    i = 0
    while i < n:
        print(i)
        i = i + (n // 10)
```

$i_0 = 0$

$i_k = k \cdot \lfloor \frac{n}{10} \rfloor$

$n \leq k \cdot \lfloor \frac{n}{10} \rfloor$

$k \geq 10$

$RT_{f_2} = 10$

$RT_{f_2} \in \Theta(1)$

13.
```python
def f3(n: int) -> None:
    """Precondition: n >= 5."""
    i = 20
    while i < n * n:
        print(i)
        i = i + 3
```

$i_0 = 20$

$i_k = 20 + 3k$

$i_k \geq n^2$

$20 + 3k \geq n^2$

$k \geq \frac{n^2 - 20}{3}$

$RT_{f_3} = 3 + \left\lceil \frac{n^2 - 20}{3} \right\rceil$

$RT_{f_3} \in \Theta(n^2)$

14.
```python
def f4(n: int) -> None:
    """Precondition: n >= 2."""
    i = 2
    while i < n:
        print(i)
        i = i * i
```

$i_0 = 2$

$i_k = 2^{2^k}$

$i_k \geq n$

$$
\begin{aligned}
2^{2^k} &\geq n \\
\log_2(2^{2^k}) &\geq \log_2(n) \\
2k \cdot \log_2(2) &\geq \log_2(n) \\
2^k &\geq \log_2(n) \\
\log_2(2^k) &\geq \log_2(\log_2(n)) \\
k \cdot \log_2(2) &\geq \log_2(\log_2(n)) \\
k &\geq \log_2(\log_2(n))
\end{aligned}
$$

$RT_{f_4} = \lceil \log_2(\log_2(n)) \rceil$

$RT_{f_4} \in \Theta(\log(\log(n))$

2 Exercise 2: Analysing nested loops

Each of the following functions takes as input a non-negative integer and performs at least one loop. Remember, to analyse the running time of a nested loop:

1. First, determine an expression for the exact cost 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.
2. Then determine the total cost of the outer loop by adding up the costs of the inner loop from Step 1. Note that if the cost of the inner loop is the same for each 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 simplify an expression involving summation notation (Σ).
3. Repeat Steps (1) and (2) 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 counters.

You'll also find the following formula helpful:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

```
11.     def f5(n: int) -> None:
2           """Precondition: n >= 0"""
3           i = 0
4           while i < n: # iterates \ciel{n/5} times
5               for j in range(0, n): # iterates n times
6                   print(j)
7               i = i + 5
```

Inner loop: iterates n times

Outer loop: iterates $\lceil \frac{n}{5} \rceil$ times (Same analysis as f_1)

$RT_{f_5} = n \cdot \lceil \frac{n}{5} \rceil$

$RT_{f_5} \in \Theta(n^2)$

12.
```python
def f6(n: int) -> None:
    """Precondition: n >= 4"""
    i = 4
    while i < n:
        j = 1
        while j < n:
            j = j * 3
        for k in range(0, i):
            print(k)
        i = i + 1
```

Inner loop 1: iterates $\lceil \log_3(n) + 1 \rceil$ times.

$j_0 = 1$

$j_k = 3^{j-1}$

$j_k \geq n$

$3^{j-1} \geq n$

$\log_3(3^{j-1}) \geq \log_3(n)$

$(j-1) \cdot \log_3(3) \geq \log_3(n)$

$j \geq \log_3(n) + 1$

Inner loop 2: iterates i times

Outer loop: iterates $n - 4$ times

$i_0 = 4$

$i_k = 4 + k$

$i_k \geq n$

$4 + k \geq n$

$k \geq n - 4$

I'm not really sure how to get this one... sorry. Maybe I'll get it later.

13.
```python
def f7(n: int) -> None:
    """Precondition: n >= 0"""
    i = 0
    while i < n: # iterates \ciel{n/4} times
        j = n
        while j > 0: # iterates
            for k in range(0, j): # iterates j times
                print(k)
            j = j - 1
        i = i + 4
```

Innermost loop: iterates j times

Middle loop: iterates n times

Innermost and Middle loop: $\sum_{i=0}^{n} j = \frac{n(n+1)}{2}$

Outermost loop: iterates $\lceil \frac{n}{4} \rceil$ times

$RT_{f_7} = \lceil \frac{n}{4} \rceil \cdot \frac{n(n+1)}{2}$

$RT_{f_7} \in \Theta(n^3)$

3 Additional exercises

1. Analyse the running time of the following function.

```
def f8(n: int) -> None:
    """Precondition: n >= 1"""
    i = 1
    while i < n:
        j = 0
        while j < i:
            j = j + 1

        i = i * 2
```

Hint: the actual calculation for this function is a little trickier than the others. You may need a formula from Appendix C.1 Summations and Products. 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.

2. Consider the following algorithm:

```
def subsequence_sum(lst: List[int]) -> int:
    n = len(lst)
    max_so_far = 0
    for i in range(0, n):              # Loop 1: i goes from 0 to n-1
        for j in range(i, n):          # Loop 2: j goes from i to n-1
            s = 0
            for k in range(i, j + 1):  # Loop 3: k goes from i to j
                s = s + lst[k]
            if max_so_far < s:
                max_so_far = s
    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 is as a valuable exercise in performing calculations with summation notation.

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

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