CSC110 Lecture 23: More Running-Time Analysis

🖨 Print this handout

Exercise 1: Analysing running time 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 t 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.

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

i_k

$i_k \geq n$

$5k + 3 \geq n$

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

Iteration	Loop variable i

Iteration	Loop variable i	
0	$i_0 =$	3
1	$i_1 =$	8
2	$i_2 =$	13
3	$i_3 =$	18
...	...	
k	$i_k =$	$5k+3$

Write your running-time analysis below. Let n be the integer input to f1.
- the function takes 1 step before the while loop
- each iteration of the while loop takes 1 step
- Let i_k be the value of variable i after k iterations
- based on the code, $i_k = 5k+3$
- the loop stopping requires $i \geq n$.
- need to find the smallest value of k s.t. $i_k \geq n$.
  - the smallest value of k to make this so is
    $$k = \lceil \tfrac{n-3}{5} \rceil$$
  - $\therefore$ num iterations is $\lceil \tfrac{n-3}{5} \rceil$

- $i_k \geq n$

$5k+3 \geq n$

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

2.

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

$$RT_{f1} = 1 + \lceil \tfrac{n-3}{5} \rceil \cdot 1$$
$$\in \theta(n)$$

Iteration	Loop variable i	
0	$i_0 =$	0
1	$i_1 =$	$n/10$
2	$i_2 =$	$2n/10$
3	$i_3 =$	$3n/10$

Iteration	Loop variable i
...	...
k	$i_k =$ $k\,\dfrac{n}{10}$

loop stops when $i \geq n$

Write your running-time analysis below.

A rough analysis $\longrightarrow$ $i_k \geq n$

write in style of
the Q1 solution
for a complete solution.

$k\,\dfrac{n}{10} \geq n$

$k \geq 10$

$$RT_{f_2}(n) = 1 + 10 \cdot 1$$

$$\in \Theta(1)$$

3.

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

Iteration	Loop variable i
0	$i_0 =$ 20
1	$i_1 =$ 23
2	$i_2 =$ 26
3	$i_3 =$ 29
...	...
k	$i_k =$ $k \cdot 3 + 20$

Write your running-time analysis below.

The loop exists when
for the first time

$i_k \geq n^2$

$$3k + 20 \geqslant n^2$$

The # of iterations is
the smallest k s.t. $\longrightarrow k \geqslant \dfrac{n^2 - 20}{3}$

$$RT_{f3}(n) = 1 + \left\lceil \dfrac{n^2 - 20}{3} \right\rceil \cdot 1$$

$$\in \Theta(n^2)$$

Exercise 2: Analysing nested loops

Remember, to analyse the running time of a nested loop:

1. First, determine an expression for the exact running time of the inner loop(s) 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 running time of the outer loop by adding up the steps of the inner loop(s) from Step 1. Note that if the number of steps of the inner loop(s) is the same for each iteration, you can simply multiply this number 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 variables.

You will also find the following formula helpful:

$$\forall n \in \mathbb{N}, \ \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Each of the following functions takes as input a non-negative integer and performs at least one loop. Analyse the running time of each function.

4.

```python
def f4(n: int) -> None:
    """Precondition: n >= 0"""
    i = 0
    while i < n:                    # Loop 1
        for j in range(0, n):      # Loop 2
            print(j)
        i = i + 5
```

inner loop:

for a fixed iteration of the outer loop,
the inner loop takes n iterations
of 1 step

outer loop:
- # iterations is $\lceil n/5 \rceil$
- of $(n+1)$ steps $j = i + 5$

(inner loop)

$$RT_{f4}(n) = 1 + \lceil n/5 \rceil (n+1) \in \Theta(n^2)$$

$i = 0$

5.

```python
def f6(n: int) -> None:
    """Precondition: n >= 4"""
    i = 4
    while i < n:                    # Loop 1
        j = 1
        while j < n:                # Loop 2
            j = j * 3
        for k in range(0, i):       # Loop 3
            print(k)
        i = i + 1
```

inner while loop
iterations $\lceil \log_3 n \rceil$ of 1 step

inner for loop
iterations i of 1 step

outer loop:
#iterations n-4
#steps per iteration is
$(j=1)$ (1) + $\lceil \log_3 n \rceil$ + i + (1) — $i = i+1$

$$RT_{f5}(n) = \sum_{i=4}^{n-1} \left(1 + \lceil \log_3 n \rceil + i + 1 \right)$$

$$= \sum_{i=4}^{n} \left(2 + \lceil \log_3 n \rceil \right) + \sum_{i=4}^{n-1} i$$

correction
from lecture
upper bound on
sum should be
n-1 not n.   6.

$$= (n-4)\left(2 + \lceil \log_3 n \rceil\right) + \sum_{i=4}^{n-1} i$$

$$+ (0+1+2+3)$$
$$- (0+1+2+3)$$

$$= \sum_{i=0}^{n-1} i - 6$$

$$= \frac{(n-1)(n)}{2} - 6$$

```python
def f6(n: int) -> None:
    """Precondition: n >= 0"""
    i = 0
    while i < n:                    # Loop 1
        j = n
        while j > 0:                # Loop 2
            for k in range(0, j)    # Loop 3
                print(k)
            j = j - 1
        i = i + 4
```

$$\in \Theta(n^2)$$

rough analysis:

$$RT_{f6}(n) = 1 + \left\lceil \frac{n}{4} \right\rceil \left(1 + \sum_{j=1}^{n} \left(\sum_{k=0}^{j-1} 1 + 1 \right) + 1 \right)$$

inner
for loop $(j = j-1)$

$i=0$

#iter
outer
while

$j=n$

inner
while loop

$i = i+4$

$$= 1 + \left\lceil \frac{n}{4} \right\rceil \left(2 + \sum_{j=1}^{n} (j+1) \right)$$

$$= 1 + \left\lceil \frac{n}{4} \right\rceil \left(2 + n + \frac{n(n+1)}{2} \right)$$

$$\in \Theta(n^3)$$

Additional exercises

1. Analyse the running time of the following function.

```python
def extra(n: int) -> None:
    """Precondition: n >= 1"""
    i = 1
    while i < n:         # Loop 1
        j = 0
        while j < i:     # Loop 2
            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 (https://www.teach.cs.toronto.edu/~csc110y /fall/notes/C-math-reference/01-summations-products.html). 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:

```python
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
```

Analyse 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 $n, a, b \in \mathbb{Z}^+$):

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