March 2-6

Topics this week:

- Hour 1 - Step counting / 2 typical loop examples
- Hour 2 - Worksheets: loops - different increment patterns
- Hour 3 - Upper and lower bounds: is_prime / print_primes examples
- Hour 4 - Worksheets: nested loops

CSC165 Student Questions / Announcements

- Problem Set 3 is out and due March 6th (we have extra office hours - see schedule on course website)

Go to menti.com and enter the code 64 20 49 (For Q+A during lecture)

CSC 165 Lecture 14

Analyzing Algorithm Running Time Cont’d

Recap on step counting:

Recall our goal:

Given an algorithm, find an approximate # of steps, as a function of the input size, for large inputs.

End goal: $\Theta(-)$
What counts as a step? (a constant time operation)

Exs): 1. Math, +, -, *

2. Comparison, >, <, <=

3. Assignments/lookups, \( y = x + 3 \)

4. Return statements

and passing arguments to a function call.

What operations can take non-constant time?

Exs) 1. Loops, \( \text{csc 165} \)

2. Helper functions.

3. Operations on compound data structures, \( \text{csc 148, csc 263, csc 773} \)

4. Recursive functions, \( \text{csc 256} \)
Analyzing loop running time

Basic Idea: \( \text{"Add up the total runtime of each loop iteration."} \)

\[
\left( \frac{\text{# of steps}}{\text{iteration}} \right) \times \left( \text{# of iterations} \right) = \text{Total # of steps.}
\]

Two examples of typical analysis of loops (more on worksheets)

\( \text{Ex)} \)

\[
def f(n: \text{int}): \# n \geq 0
\]

Loop \#1

\[
\text{for } i \text{ in range}(n): \# i \leq 0, \ldots, n-1.
\]

Loop \#2

\[
\text{for } j \text{ in range}(n \times n): \# j = 0, \ldots, n^2 - 1
\]

\[
\text{print}(i + j). \# 1 \text{ step.}
\]

Two nested loops (independent)

Analysis: For Loop \#2:

1. The loop body (\text{print}(i+j)) takes constant time, so it counts as 1 step per iteration.
2. There are \( n^2 \) iterations in total (\( j=0, \ldots, n^2 \)).
1 + 1 + \ldots + 1 = n^2 \text{ steps for Loop 2.}
\begin{align*}
j = 0 & \quad j = 1 & \quad j = n^2 - 1
\end{align*}

For Loop 1:

1. Each iteration takes $n^2$ steps.

2. $n$ iterations of Loop #1.

\[ n^2 + n^2 + \ldots + n^2 = n^3 \]
\begin{align*}
i = 0 & \quad i = 1 & \quad i = n^2 - 1
\end{align*}

So the total runtime is $\Theta(n^3)$.
Two nested loops (loop 2 depends on loop 1)

```python
def f2(n: int):
    for i in range(n):
        for j in range(i * i):
            print(i * j)
```

Loop #2: We start with Loop 2, assuming we're on iteration i of Loop 1.

[replace n with i] ... $i^2$ steps on iteration i of Loop 1.

Loop #1:

1. Iteration i takes $i^2$ steps.
2. i takes on the values 0, 1, ..., n - 1.

Cost: $0^2 + 1^2 + 2^2 + ... + (n-1)^2$

\[
\sum_{i=0}^{n-1} i^2 = \frac{n(n+1)(2n+1)}{6}
\]

From Last Wed.

\[
\begin{align*}
\text{2 thurs.} & \quad \frac{(n-1)n(2n-1)}{6}
\end{align*}
\]
So the runtime of $f_2$ is $O(n^3)$

CSC165 Student Questions / Announcements

- Problem Set 3 is out and due March 6th

Go to menti.com and enter the code $642049$ (For Q+A during lecture)

Follow up example from last class

```python
def g(n):
    r = 0
    for i in range(n):
        for j in range(n*n):
            r = r + j
    for i in range(n):
        for j in range(i*i):
            r = r + j
    return r
```

In analysis, clearly indicate you have considered all lines of code (including all constant time operations),
Recap of how we perform a runtime analysis

1. Determine “exact” step count.
2. Convert to $\Theta()$.

Alternative:
1. Find a lower bound on # of steps.
   $L.B \leq \text{exact # of steps} \leq U.B$
2. Go from lower bound to $\Omega$.
   1. Find an upper bound.
   4. Convert to $\Theta$.
   * If equal, then $\Theta$. 
Prime(n): n > 1 ∨ \forall d \in \mathbb{Z}, \, d | n ⇒ d = 1 \lor d = n.

Example: \(5\) is prime:

```python
def is_prime(n: int) -> bool:
    # n > 1 (precondition)
    for d in range(2, n):  # 2, 3, ..., n-1
        if n % d == 0:
            return False
    return True
```

Analysis:

1. Each iteration takes \(\Omega(1)\) step.
2. The number of iterations:
   - at least 1,
   - at most \(n-2\).

Lower bound

- Since the loop takes at least 1 iteration
  then the total number of steps is \(\Omega(1)\).
- So the total runtime of \(\text{is} \text{prime}\) is at least \(\Omega(1)\).
  So the runtime is \(\Omega(1)\).
Upper bound $\rightarrow$ runtime is at most $n - 2$.

So, $O(n)$.

$1 \leq \text{RT}_{ip}(h) \leq n - 2$
Example: \[ \text{5.17} \] .

def print_primes(n: int) -> None:
    \#n > 1 (precondition)
    for k in range(2,n+1): #2,3,...,n
        if is_prime(k):
            \[ \text{1} \leq R_{\text{ip}}(k) \leq k-2 \]
            print(k)

Analysis:
1) k goes from 2 to n.
2) Cost for iteration \( k \), i.e., \( R_{\text{ip}}(k) \).

\[ RT_{\text{ip}}(n) = \sum_{k=2}^{n} R_{\text{ip}}(k) \]

Upper bound:

\[ RT_{\text{ip}}(n) \leq \sum_{k=2}^{n} (k-2) \sim O(n^2) \]

We know that \( R_{\text{ip}}(k) \leq k-2 \).

Given closed forms in exams.
We know \( 1 \leq R_{tip}(k) \)
\[
R_{te}(n) \geq \sum_{k=2}^{n} 1 = n-1 \quad \Omega(n)
\]
\[
R_{pe}(n) \in \Omega(n) \quad R_{te}(n) \in O(n^2).
\]

**Lower bound:**
\[
R_{pc}(n) = \sum_{k \leq n} R_{tip}(k) + \sum_{k \leq n \text{ and } k \text{ prime}} R_{tip}(k).
\]

**Improved lower bound:**
\[
\sum_{k \text{ prime}} R_{tip}(k) \approx \sum_{k \leq n} (k-2) \Omega(n^2) \text{ from properties of prime #s.}
\]
Improved upper bound (from last year's final exam):
Can we get a better upper bound on is_prime?
How many iterations, at most, if n is not prime? $\sqrt{n}$

$O\left(\frac{n^2}{\log n}\right)$

Rough proof by contradiction:
Assume $d > \sqrt{n}$ and $d | n$:
then, \( \exists k, dk = n \)
and \( k < \sqrt{n} \)

On an earlier iteration we would have found that \( k \sqrt{n} \) though.
Given the above improved upper bound, what would be a better algorithm for is_prime? (Should it take $n-2$ iterations to conclude if $n$ is prime?)

$$\text{Can stop iterating when } d = \sqrt{n} + 1 \text{ rather than } d = n.$$ 

i.e. a more efficient algorithm would replace “for $d$ in range(2, n)” with “for $d$ in range(2, int(\sqrt{n}) + 1)”.

A collection of math / programming problems:
https://projecteuler.net/about (many of which relate to prime number/number theory).