COSC 2306
Data Programming

Algorithm Analysis

Exam 1

- Date: **Oct 15** during the class time
- Duration: **60 mins** (but will give you **full class time**)
- Questions: multiple choices, coding
- Open-booked, but **only paper-based materials** such as notes/slides/books/documents are allowed
- Same as the quiz, except the laptop running Respondus, no access to any electronic devices/internet/AI, no discussion/communication in any kind
- You need to sit at least 3 feet from your neighboring students during the exam.
- If you have questions or issues, please only talk to the TA who supervises the exam, but never talk to other students.

Determine Big-O

```
result = 0
for i in range(0, n, 2):
    result += i
print(result)
```

```
result = 0
for i in range(n):
    for j in range(i):
        result += i*j
print(result)
```

Determine Big-O

```
result = 0
for i in range(0, n, 2):
    result += i
print(result)
```

O(n)

$$f(n) = 1 + \frac{n}{2} + 1$$

$$O(n)$$

```
result = 0
for i in range(n):
    for j in range(i):
        result += i*j
print(result)
```

$O(n^2)$

Outer loop: i = = 0, 1, 2, …, n-1 → total of **n iterations**

Inner loop: For each i, the inner loop runs **j = 0 … (i-1)** → exactly i times.

$$nested\ loop:\ 1 + 2 \ldots + n - 1 + n = \frac{n(n-1)}{2}$$

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

$$O(n^2)$$

Best case and worst case

- Due to the characteristics of input, the time cost for a single program may vary
  - So does its big-O notation

```
def Find(A, v):# Find an index i so that A[i] = v
    n = len(A)
    for i in range(n):
        if A[i] == v:
            return i
    return -1
```

If A[0] = v : Loop goes only one round. O(1)

If v is not in A: Need to iterate through A. O(n)

Best case and worst case

```
# Generate a list of the sums of pairs of odd elements.
# For each pair, one element is in A and the other is in B.
# A and B have the same length

def oddSums(A, B):
    n = len(A)
    results = []
    for i in range(n):
        if A[i] % 2 == 1:
            for j in range(n):
                if B[j] % 2 == 1:
                    results.append(A[i] + B[j])
return results
```

Best case and worst case

- If A contains only even elements:
  - Best case
  - O(n)
- If A contains only odd elements:
  - Worst Case
  - $O(n^2)$
- What if A and B contain random elements?

Best case and worst case

- Can expect the best case
- But need to care for the worst case
  - If it happens, it will matter a lot, especially when the problem size is large
  - Always Check the Big-O for **the worst case**
- Beside best case and worst case, the time cost for "average case" is also often concerned.

Big-O with multiple variables

- Big-O can be in form of O(f(m, n))

```
result = 0
for i in range(n):
    for j in range(m):
        result += i*j
print(result)
```

O(nm)

```
result = 0
for i in range(n):
        result += i
for j in range(m):
        result += j
print(result)
```

O(n+m)

Program optimization

- Big-O is not the only factor
- Inaccurate with small problem sizes
  - 100n is larger than n^2 when $n < 100$
- Memory cost matters
  - Faster algorithms may cost more memory space
- Think all-roundly when optimize a program
  - What is the problem size?
  - What is the bottleneck (time or space)?
  - How fast are the "constant" operations?

Program optimization

- Tips:
  - Focus on the most time-costing section
  - Reduce loop layers
  - Reduce redundant operations
  - Redesign algorithm
  - Use more spaces recording executed computations to save the time of further computation (e.g., dynamic programming)