Review

Do P5

Efficiency doesn't really matter for small inputs.

- ✎ Algorithms run longer on larger inputs
- ✎ Interest in time efficiency for large inputs
- ✎ Need to isolate algorithm performance from that of machine and code
- ✎ Time efficiency is measured by <u>growth</u> of basic operation count, *C(n)*, as input size *n* increases

Realization

➪ Asymptotic efficiency

Turns Out

Most algorithms fall into a small set of efficiency classes.

Long-term (limiting) runtime behavior, as inputs become large, to classify efficiency .

A system for classifying efficiency based on <u>asymptotic</u> behavior of run time, as input grows very large, avoids dealing with efficiency of 1000s of algorithms individually

Non-recursive Algorithms
General Plan Review

❶ Select suitable input size parameter, *n*

❷ Identify suitable basic operation

❸ Check dependancy of basic op

❹ Determine count $C(n)$: write sum

❺ Determine *order of growth* of $C(n)$

Non-recursive Algorithms
Example 1

Exercise
Discuss choices of basic operation, and dependency of count in each case.

Algorithm $MaxElement$
Input Array $A[0..n-1]$
Output Largest value in A

```
maxval ← A[0]
for i ← 1 to n − 1 do
    if A[i] > maxval then
        maxval ← A[i]
return maxval
```

cs223fig4.cdr
Thursday, January 27, 2022 12:14:38 PM
Color profile: Disabled

Composite Default screen

A Useful Tool

➪ Basic properties

$$\sum_{i=l}^{u} ca_i = c\sum_{i=l}^{u} a_i \qquad \sum_{i=l}^{u}(a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i$$

➪ Basic sums (references)

Quiz
Which sum would you choose to solve Example 1 from previous slide?

$$\sum_{i=l}^{u} 1 = u - l + 1 \text{ where } l \le u$$

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

Non-recursive Algorithms Example 2

Exercise
Compare to solution discussed in previous lecture. What if a *quicksort* was used?

Quiz
Does the basic operation count depend on input size only?

How to read?

Algorithm $UniqueElements$
Input Array $A[0..n-1]$
Output Return true if elements in A distinct, otherwise false

for $i \leftarrow 0$ **to** $n-2$ **do**
    for $j \leftarrow i+1$ **to** $n-1$ **do**
        if $A[i] = A[j]$ **then**
            return false
return true

Non-recursive Algorithms Example 3

➪ Multiplicative constant

Analysis steps are marked.

Quiz
Why it's not important to fully understand how the calculation works for the purposes of this lecture? **Hint**: recall *How to read?* discussion.

Algorithm *MatrixMaultiplication*

Input Square matrices A, B, C indexed $0 \mathrel{..} n-1$

Output Matrix $C = A \times B$

```
for i ← 0 to n − 1 do
    for j ← 0 to n − 1 do
        C[i, j] ← 0.0
        for k ← 0 to n − 1 do
            C[i, j] ← C[i, j] + A[i, k] × B[k, j]
return C
```

Quiz
Should it be preferable to pick × since it is more expensive than + in most cases? (Offers a better approximation of runtime.)

Do Example 4 (P6)