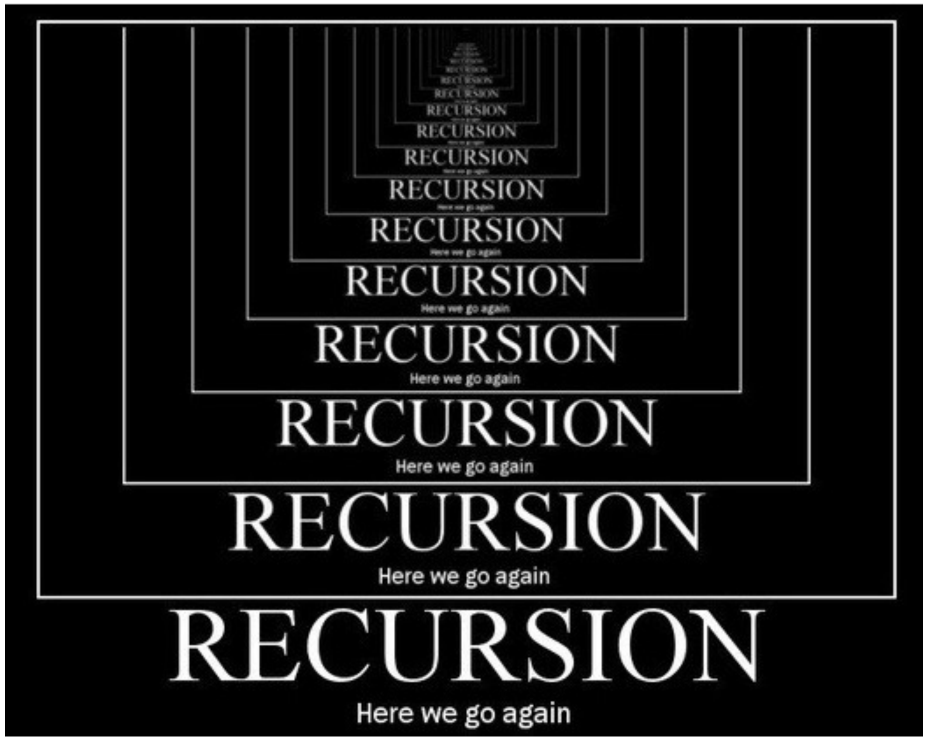
CSC 148: Introduction to Computer Science

Week 6



University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Reminder: Midterm today!

- Tonight is our first midterm!
- Please check Piazza for details.

- If you are unable to make the midterm, please contact the course instructor email ASAP.

“In order to understand recursion ...

you must first understand recursion”

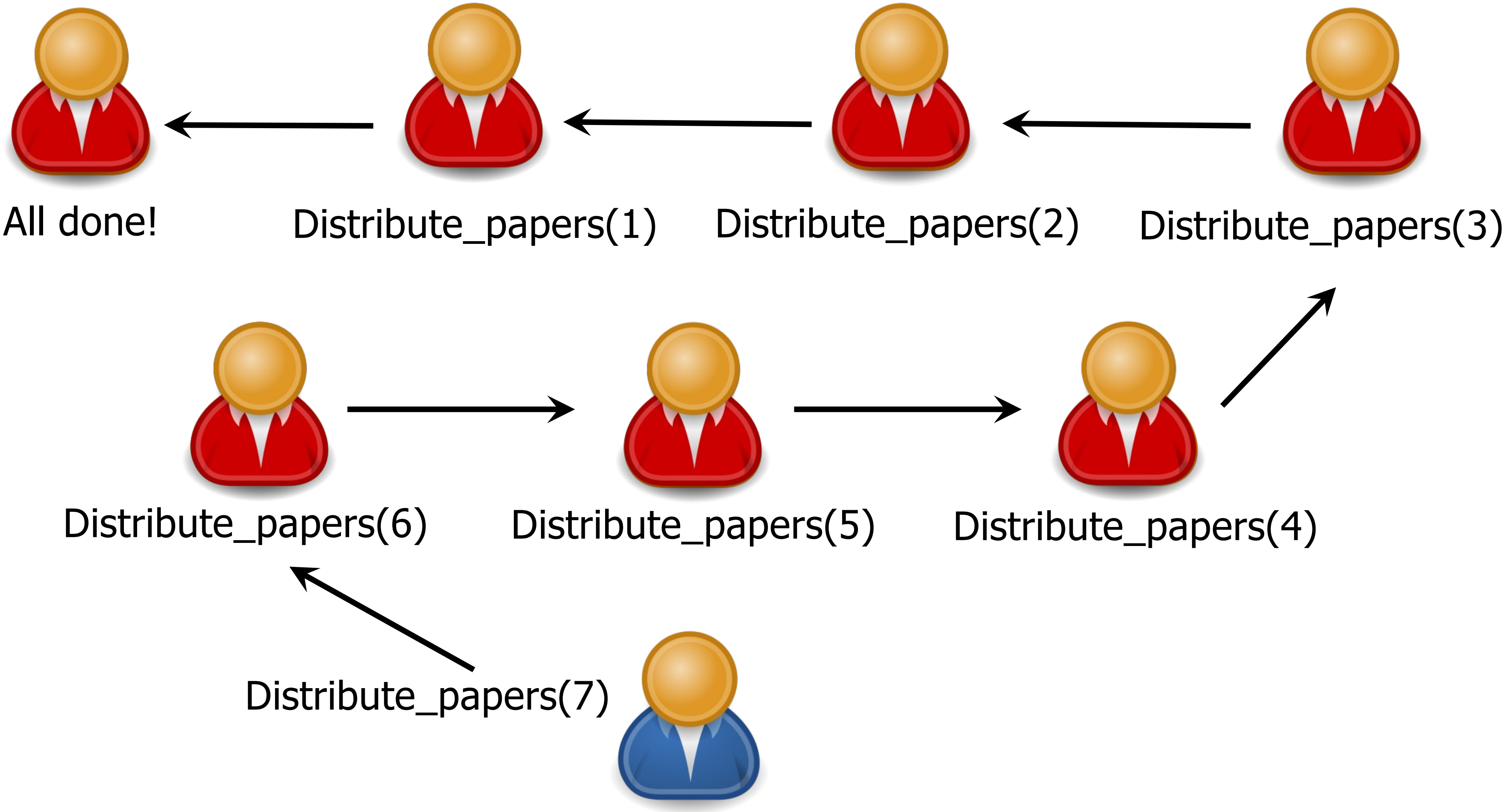
What is Recursion?

- Solve a problem by using an algorithm that calls itself on a smaller problem

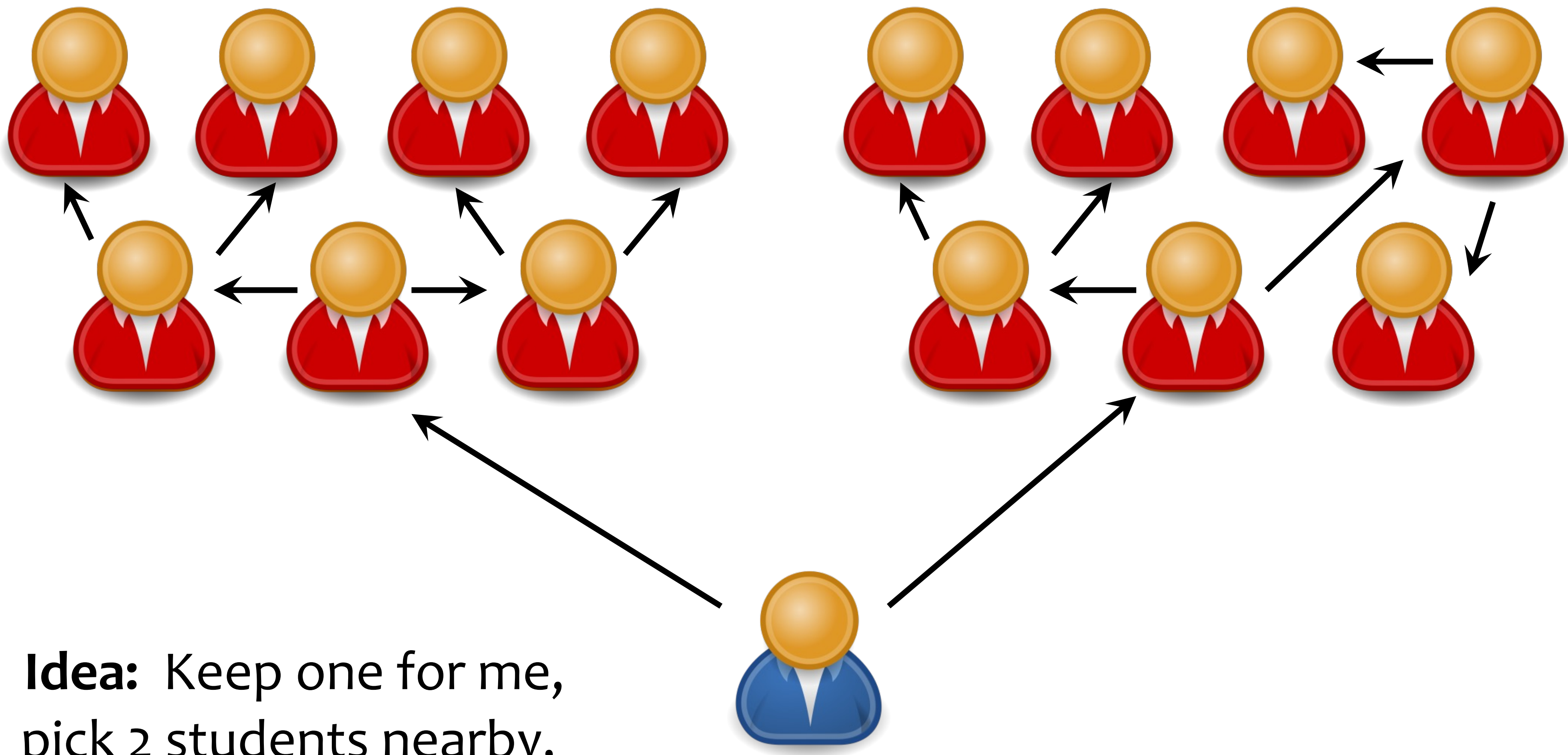
- With each call, the problem becomes simpler

- At some point, the problem becomes trivial!

Using recursion – example 1

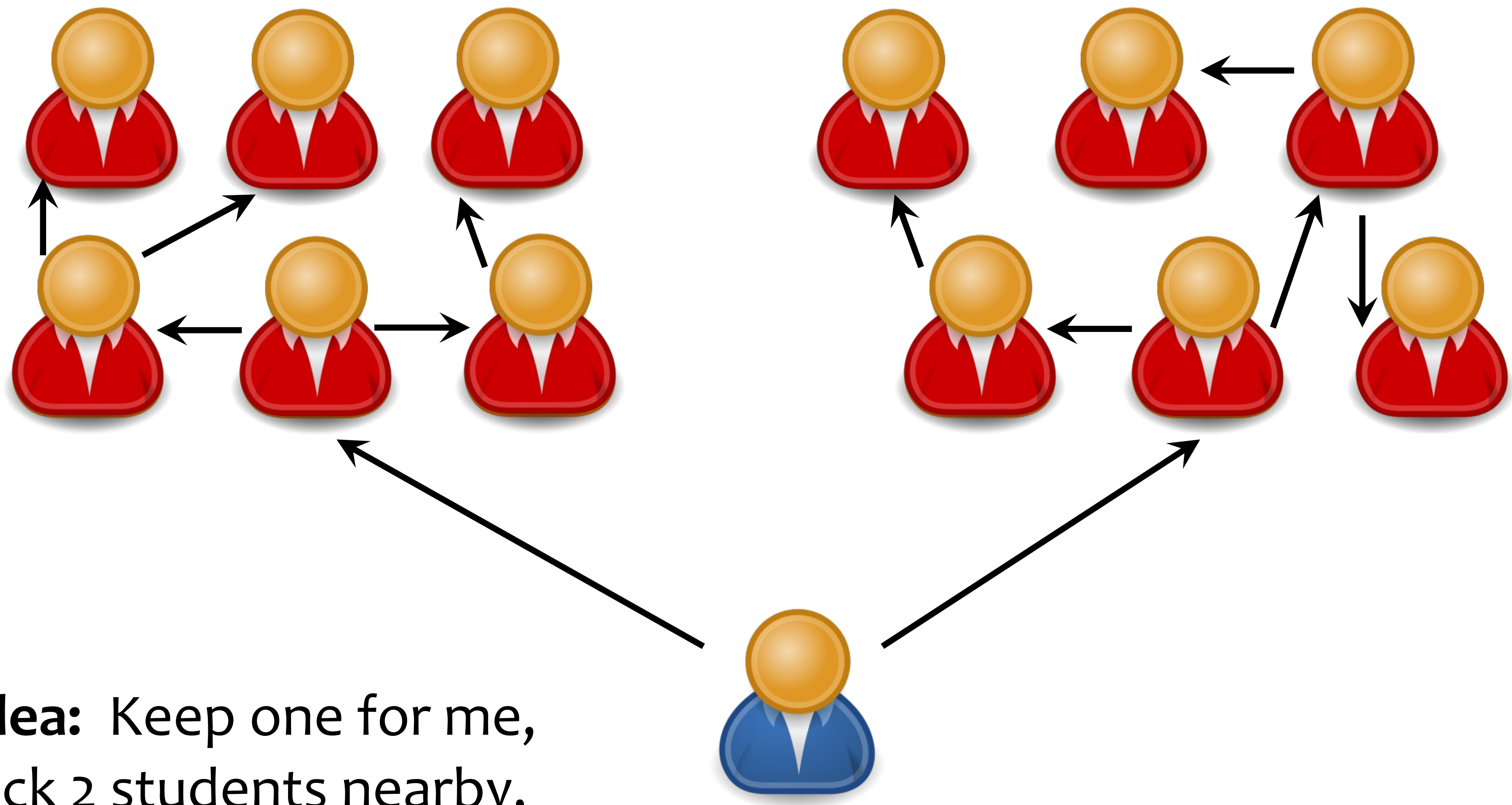


Using recursion – example 2



Idea: Keep one for me, pick 2 students nearby, and give each half of your remaining pile

Using recursion – example 2



Idea: Keep one for me, pick 2 students nearby, and give each half of your remaining pile

What is recursion?

"In order to understand recursion ...

you must first understand recursion"

Actually, to understand recursion, one must understand its components ...

Components of Recursion

- **Base Case**: *A trivial, immediately solvable case.*

Examples: Distributing *zero* papers, sorting 1 number.

- **Recursive Case**: *Any other situation.* Marked by subdividing the problem and solving part of it.

Examples: Distributing 10 papers, sorting a list of size 5000.

Dividing the Problem

- "N-1" approach: handle one entity, then call the recursion for N-1 entities
- Divide in 2 or more subproblems: apply recursion for each half, quarter, etc. of the problem
- Other ways (more later…)

Programmer Perspective

- Recursion is when a function calls itself directly
  - (mostly ... we won't talk about indirect recursion)
- Goal:
  - Calls itself to solve a smaller part of the problem, using the same function/algorithm
- In some cases, we need to combine the solution!

Sum of List Elements

List of integer elements: List = [3, 4, 5]

What's the sum of elements? Solve recursively.

```
def sum_list(L):
    if len(L) == 0:
        return 0
    else:
        return L[0] + sum_list (L[1:])

# main program
...
print(sum_list(List))
```

Sure, we could just use predefined sum(List), or use a simple for loop.

Assume for now that we want an alternative solution using recursion.

Tracing

List = [3, 4, 5]

What's the sum of elements? Solve recursively.

```
def sum_list(L):
    if len(L) == 0:
        return 0
    else:
        return L[0] + sum_list (L[1:])

# main program
...
print(sum_list(List))
```

Main program sum_list([3,4,5]) ?

sum_list([3,4,5]) → 3 + sum_list([4,5])

sum_list([4,5]) → 4 + sum_list([5])

sum_list([5]) → 5 + sum_list([])

sum_list([]) → 0

More Complex Problems

- Why do all this? This could simply be solved with predefined ‘sum’ function
- What if L’s elements can be lists themselves?
  - L = [1, [5,3], 8, [4,[9,7]]]

Will this work?

```
s = 0
for elem in L:
    s += elem
```

What about this?

```
s = 0
for elem in L:
    if isinstance(elem, list):
        for subelem in elem:
            s += subelem
    else:
        s += elem
```

- Nested lists can occur at any depth ⇒ complicated!

Sum of List Elements – Nested Lists

- L = [1, [5,3], 8, [4,[9,7]]]

recursive step

base case

```
def sum_list(L):
    if isinstance(L, list):
        s = 0
        for elem in L:
            # calculate the sum of the sublist "elem" recursively
            s += sum_list(elem)
        return s
    else:
        return L
```

Sum of List Elements – Nested Lists

- L = [1, [5,3], 8, [4,[9,7]]]

```
def sum_list(L):
    if isinstance(L, list) :
        s = 0
        for elem in L:
            # calculate the sum of the sublist "elem" recursively
            s += sum_list(elem)
        return s
    else:
        return L
```

recursive step

base case

Partial Tracing Practice

- Attempting to fully trace recursive code is time-consuming and error prone.

- When tracing recursive code, **don't** trace into recursive calls!
- Instead, assume each call is correct, and make sure the rest of the code uses those calls correctly.

- Worksheet …