

# Introduction to Algorithmic Problem Solving

Dr. Chia-Ling Tsai

---

---

---

---

---

---

---

---

## Outline

- Definition of an algorithm
- Procedural decomposition
- Algorithm design issues
- Sorting algorithms
- Searching algorithms

---

---

---

---

---

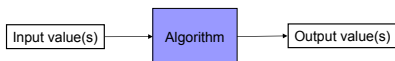
---

---

---

## What is an algorithm?

- An example in our everyday life is cooking
  - Algorithms  $\Leftrightarrow$  Recipes
  - Various levels of details in the description
- An algorithm is *a sequence of steps* that transform the input into the output




---

---

---

---

---

---

---

---

# Oatmeal Cookies

- **Input:**  
A bunch of ingredients including lots of oats, flour, etc..
- **Output:**  
A batch of oatmeal cookies
- **Steps (algorithm):**
  - Prepare the mixture
  - Prepare the oven and baking sheets
  - Bake the mixture
- Enough details?

---

---

---

---

---

---

---

---

# Procedural Decomposition

- Tackle complex problem at different levels of details
  - Difficult to solve complex problem all at once
  - Divide the problem into smaller pieces
  - Tackle each piece individually
  - Divide-and-Conquer
- Advantages:
  - Easier for implementation
  - Easier for maintenance

---

---

---

---

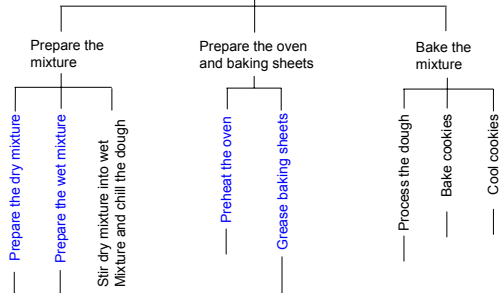
---

---

---

---

## Cookie recipe



---

---

---

---

---

---

---

---

## Prepare the dry mixture

- Place the flour in a large bowl
- Add baking soda
- Add salt
- Add cinnamon
- Add oats

---

---

---

---

---

---

---

---

## Prepare the wet mixture

- Soften butter
- Mix white sugar, brown sugar, and butter
- Repeat
  - Beat in one egg
- Until
  - No more egg .
- Stir in vanilla

---

---

---

---

---

---

---

---

## Preheat the oven

- Set the temperature to 375 degrees F.
- Repeat
  - If the preheat light not off, wait for one minute
- Until
  - Preheat-light off
- Indicate oven-ready

---

---

---

---

---

---

---

---

## Grease the baking sheets

### ■ Repeat

Apply "PAM" butter spray evenly on one ungreased baking sheet

### Until

No more ungreased baking sheets

---

---

---

---

---

---

---

---

## Process the dough

### ■ Repeat

- Shape a piece of dough into walnut-size ball.
- Place the ball two inches away from other cookies on the sheet.
- Flatten the ball with a large fork dipped in sugar

### Until

The dough is gone

---

---

---

---

---

---

---

---

## Bake cookies

### ■ Repeat

leave each sheet in the oven for 8 min

### Repeat

If cookies not firm enough, wait for 30 seconds

Until cookies are firm

### Until

No unbaked cookies

---

---

---

---

---

---

---

---

## Cool cookies

- Repeat

If the sheet is out of oven after 5 minutes, transfer cookies to a wire rack

- Until

No cookies on the sheets

---

---

---

---

---

---

---

---

## Algorithm Design Issues

- It terminates
- It generates correct answers
- It is efficient
  - Computers are not infinitely fast
  - Memory space is limited
  - Can be much more significant than hardware/software differences.

---

---

---

---

---

---

---

---

## Greatest Common Divisor

- Given two non-zero natural numbers  $a$  &  $b$
- Euclid's method:
  - 1<sup>st</sup> try:  $a=15, b=9, 15 \bmod 9 = 6$
  - 2<sup>nd</sup> try:  $a=9, b=6, 9 \bmod 6 = 3$
  - 3<sup>rd</sup> try:  $a=6, b=3, 6 \bmod 3 = 0!$
- Algorithm?
  - Describe the method in plain English.
  - What is the repeated action?

---

---

---

---

---

---

---

---

## Greatest Common Divisor

- GCD (a,b)
- Repeat
  - c is (a mod b)
  - if c not equal to 0 then a becomes b and b becomes c
- Until c is 0
- b is the GCD

---

---

---

---

---

---

---

---

## Sorting a Hand of Cards

- There are many ways to sort the cards in ascending order
  - What is your way?
- **Input:** A hand of unsorted playing cards
- **Output:** A hand of sorted playing cards
- **Algorithm:**
  - Again, in plain English first
  - What is the repeated action

---

---

---

---

---

---

---

---

## My sorting method

- Start from the 2<sup>nd</sup> card and make it the “moving” one
- Repeat
  - Search for the first card in front of the moving card that is greater than the moving card
  - If found, insert the moving card before the one that is just greater, else do nothing
  - Move to the next unprocessed card
- Until the last card is processed

---

---

---

---

---

---

---

---

## Revisit your algorithm

- The description is good for human-understanding
- But too ambiguous for a machine
  - How to search?
  - How to insert?
- Redo the algorithm again with the idea that every card occupies a cell



---

---

---

---

---

---

---

---

## Try again ...

- Start from the 2<sup>nd</sup> card and make it the  $m$  card
- Repeat
  - Start from the card before  $m$  and make it the  $c$  card
  - Repeat
    - If  $c$  is greater than  $m$ , swap their positions and make the card before the  $m$  card the  $c$  card
- Until ( $c$  not greater than  $m$ ) or (no  $c$  card)
- Until the last card is reached

---

---

---

---

---

---

---

---

## Sorting Algorithms

- Important for other tasks, such as search and merge algorithms.
- Can be executed hundreds of times in a program
- Efficiency is very important
- **Input:** a sequence of  $n$  numbers  $A = \langle a_1, a_2, \dots, a_n \rangle$
- Output:** A reordering  $\langle a'_1, a'_2, \dots, a'_n \rangle$  of the input sequence such that  $a'_1 \leq a'_2 \leq \dots \leq a'_n$

---

---

---

---

---

---

---

---

## Insertion Sort

### ■ INSERTION-SORT(A)

```
placeholders i,j,key
loop i from 2 to A.length, begin
  key ← A[i]
  j ← i-1
  while j>0 and A[j]>key, do
    A[j+1] ← A[j]
    decrease j by 1
  end while
  A[j+1] ← key
end loop
```

- Let's sort A={5,2,4,6,1,3}

---

---

---

---

---

---

---

---

## Bubble Sort

### ■ BUBBLE-SORT(A)

```
placeholders i,j
loop i from A.length to 2, begin
  loop j from 2 to i, begin
    if A[j-1]>A[j], do swap A[j-1] and A[j]
  end loop j
end loop i
```

---

---

---

---

---

---

---

---

## Lab1: printing R-A triangle

- What is being repeated?
- How do the line number, number of spaces and number of stars relate?
- With input = 5

Line #	# of spaces	# of stars
1	0	5
2	1	4
3	2	3
4	3	2
5	4	1

---

---

---

---

---

---

---

---

## (cont)

```
■ Algorithm: Triangle( base )
placeholders: line, j, k
loop line from 1 to base //counting the line#
  //count and print spaces
  loop j from 1 to line-1
    print(" ")
  end loop j
  //count and print stars
  loop k from 1 to (base-line+1)
    print("*")
  end loop k
end loop line
```

---

---

---

---

---

---

---

---

## Search Algorithms

- Searching a list of data for a particular value.
- Different algorithms have been proposed for data of different complexity.
- **Input:** a sequence of  $n$  sorted or unsorted numbers  $A = \langle a_1, a_2, \dots, a_n \rangle$ , and a target value
- Output:** The position of the target value in the list

---

---

---

---

---

---

---

---

## Sequential Search

```
■ Sequence A is unsorted
■ Sequential-Search(A, target)
placeholders i, pos
pos ← -1
loop i from 1 to A.length, begin
  if target = A[i], do
    pos ← i
  end if
end loop i
return pos
```

---

---

---

---

---

---

---

---

## Binary Search

- Sequence  $A$  is sorted
- `Sequential-Search(A, target)`  
**placeholders** low, high, mid  
low  $\leftarrow 1$   
high  $\leftarrow A.length$   
**while** low **less or equal** high, **do**  
  mid  $\leftarrow (low+high)/2$   
  **if**  $A[mid] > target$ , **do** high  $\leftarrow mid-1$   
  **else if**  $A[mid] < target$ , **do** low  $\leftarrow mid+1$   
  **else return** mid       //found it!  
**end while**  
**return** -1               //Not found!

---

---

---

---

---

---

---

---

## Summary

- Definition of an algorithm
- Procedural decomposition
- Important issues for an algorithm design
- An algorithm can be expressed in pseudo-code or plain English
- Tracing an algorithm to make sure its correctness
- We'll re-visit sorting and searching later in the course to discuss other more advanced algorithm designs.

---

---

---

---

---

---

---

---