CS 105: Introduction to Computer Science

Prof. Thao Nguyen

Spring 2025

Materials adapted from Dave Wonnacott

Office hours

- TA hours:
  - Mondays: 4-6pm and 8-10pm
  - Tuesdays: 4-6pm
- Email Suzanne to schedule meeting

Algorithm Design #1: Relate to a Solved Problem

Sometimes you may see a similarity to an existing problem or a math fact

Suppose we wanted to know if a point (x, y) is inside a circle

- use the distance formula, to the center of the circle: $\sqrt{(x-x_c)^2+(y-y_c)^2}$
- note that sometimes you'll want a *library function*, e.g. "sqrt", two notations:
  - import math
    ... math.sqrt((x-xcenter)**2, (y-ycenter)**2) ... # assuming xcenter is the x of the center
  - from math import sqrt # refrain from using “import *”
    ... sqrt((x-xcenter)**2, (y-ycenter)**2) ...

Algorithm Design #1: Relate to a Solved Problem

Sometimes you may see a similarity to an existing problem or a math fact

Suppose we wanted to know if a point (x, y) is inside a circle

- use the distance formula, to the center of the circle: $\sqrt{(x-x_c)^2+(y-y_c)^2}$
- note that sometimes you'll want a *library function*, e.g. "sqrt", use "import"
- you can also import from your own files, or other files in the project, e.g.,
    - from ShapeLibraryForCG import Circle, center_x, center_y, radius
      from math import sqrt
      c1 : Circle = Circle(100, 100, 20) # radius 20 at 100, 100
      ... math.sqrt((x-center_x(c1))**2, (y-center_y(c1))**2) ... # call, e.g. "center_x" function
      ... math.sqrt((x-c1.center_x())**2, (y-c1.center_y())**2) ... # send, e.g., "center_x" message
- **Note**: some libraries use function notation, some use message notation

Algorithm Design #1: Relate to a Solved Problem

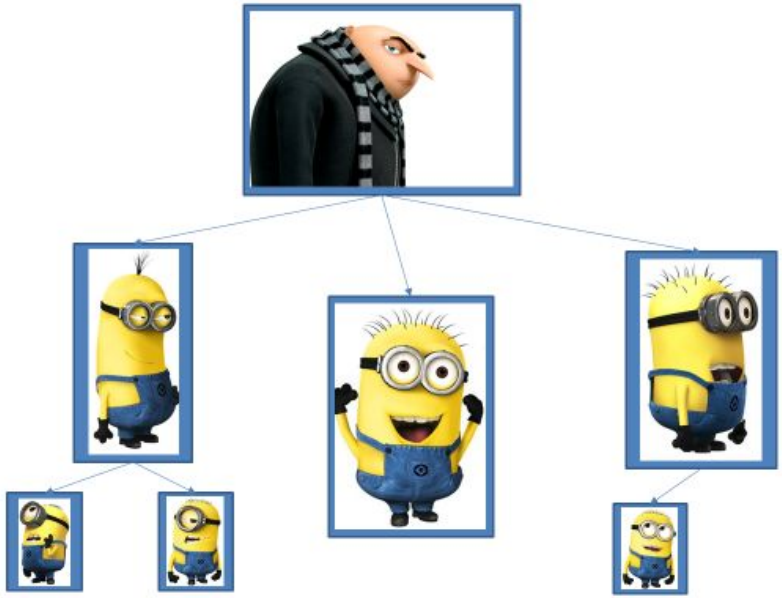
Sometimes you may see a similarity to an existing problem or a math fact

Suppose we wanted to know if a point (x, y) is inside a circle

- use the distance formula, to the center of the circle: $\sqrt{(x-x_c)^2+(y-y_c)^2}$
- note that sometimes you'll want a *library function*, e.g. "sqrt", use "import"
- you can also import from your own files, or other files in the project
- Note: some libraries use function notation, some use message notation
  - our ShapeLibraryForCG allows you to choose either one
- **Caution:** Beware of "circular reasoning"
  - Can define circle_overlap_two_figures in terms of circle_overlap, or the other way, **not both!**

Algorithm Design #2: Top-Down Design

Sometimes a problem contains a simpler problem that's *not yet solved*
(if the simpler problem is solved, this is "relate to a solved problem")



Algorithm Design #2: Top-Down Design

Pre-condition: must be true before entering the function

Post-condition: must be true before leaving the function

Benefits of TDD:

- Creates code that is easier to implement, debug, modify, and extend
- Avoids going off in the wrong direction

Algorithm Design #3: Design By Cases

Sometimes it's easiest to solve specific cases of a problem

Example discussion: identify specific cases in which windows can't overlap?

For more notes, see discussion in "From Vision to Execution"

Note: *Usually*, if doing a True/False design by cases, it's best to either

- list all True cases, have the final "else" return False
- or, list all False cases, have the final "else" return True

Python details: If/elif/else

If/elif/else statements can contain other statements:

- variable definitions
- return statements
- other if/else statements

It's also possible to have an if without an else, we'll see this later

Other uses of tests

We can check things like "radius >= 0" or "x1 < x2"

- In "if", when we want to choose one option or another in our algorithm
- In an if controlling a "throw", to indicate an "exception"
  - brief discussion in lecture, we'll see more of this later
- In a "precondition" statement, to indicate our algorithm can't handle this case
- In an "and" or "or"
  - **Warning**: "x < y and z" doesn't mean what you want it to mean!
  - What should we write instead?
- **Note:** comparing for equality uses **==** rather than =
  - number_of_chickens **=** 5 # "=" *gives a value* to a variable
  - if number_of_chickens == 5: print("still have all the chickens") # "==" *checks* the values

Algorithm Design and Encoding (programming)

Algorithm-Design is an art, many approaches, including

- find a formula/use math
- split into cases
- rewrite in terms of another simpler (or already-solved) problem

Programming tools & discussion of their use for the problems we discussed

- function definitions (as before)
- variables
- if/else