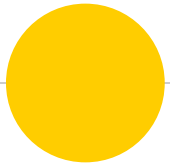
Problem solving with Python

aka MSAN 689

by Laurynas Riliskis

Solving the scheduling problem

- 22 nd Special case:
- 10:00-10:30 quiz
- 11-12 keynote



Last Christmas

recap

Data thingies

- Types
  - Basic building block
  - Composite types
- Build in in most of programming languages
- In Python, mutable vs immutable

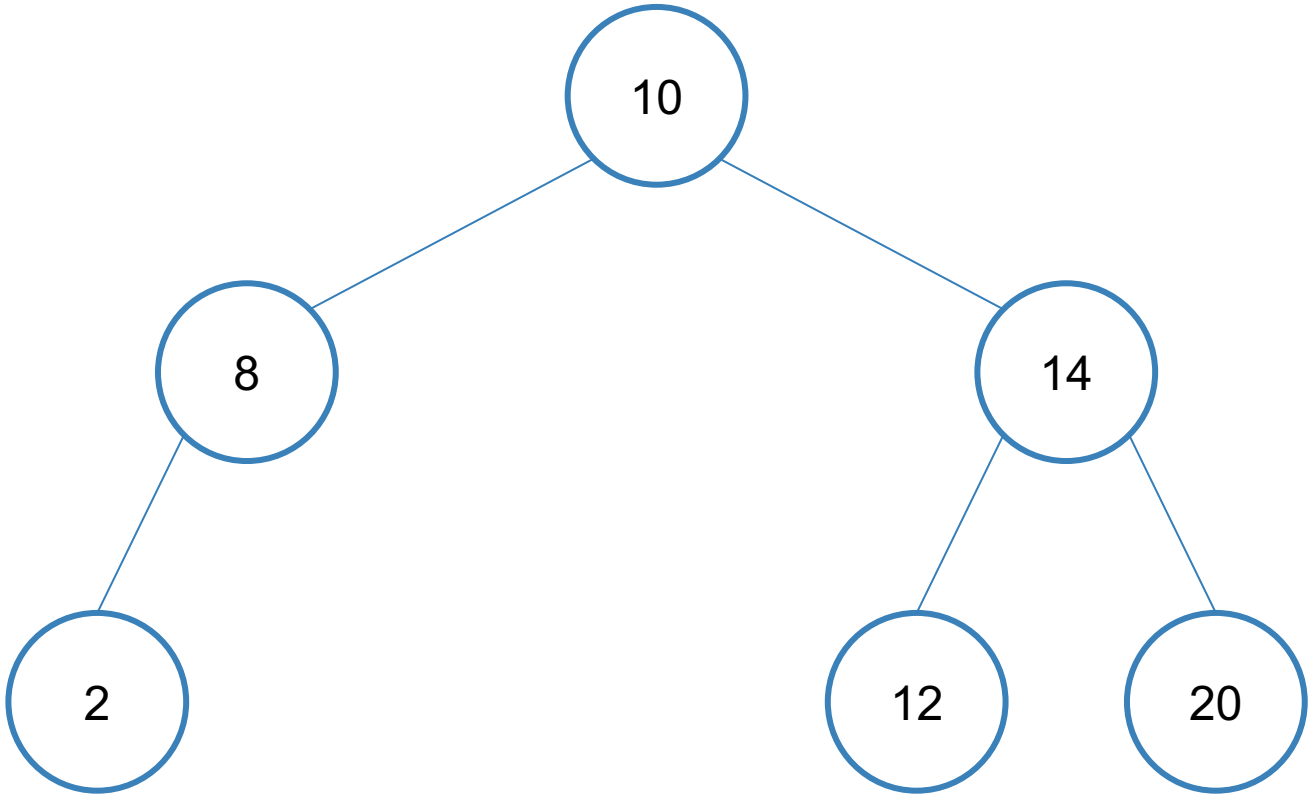
Data structures

- Is a way to organize your data
  - Insert
  - Search
  - Delete
  - Query
  - Update
- Does data structure solve problems?

Linear data structures

- Array – like list but constrained
- Contains:
  - Stacks – Last-In-First-Out (LIFO)
  - Queues – First-In-First-Out (FIFO)

Binary search



Tree walking

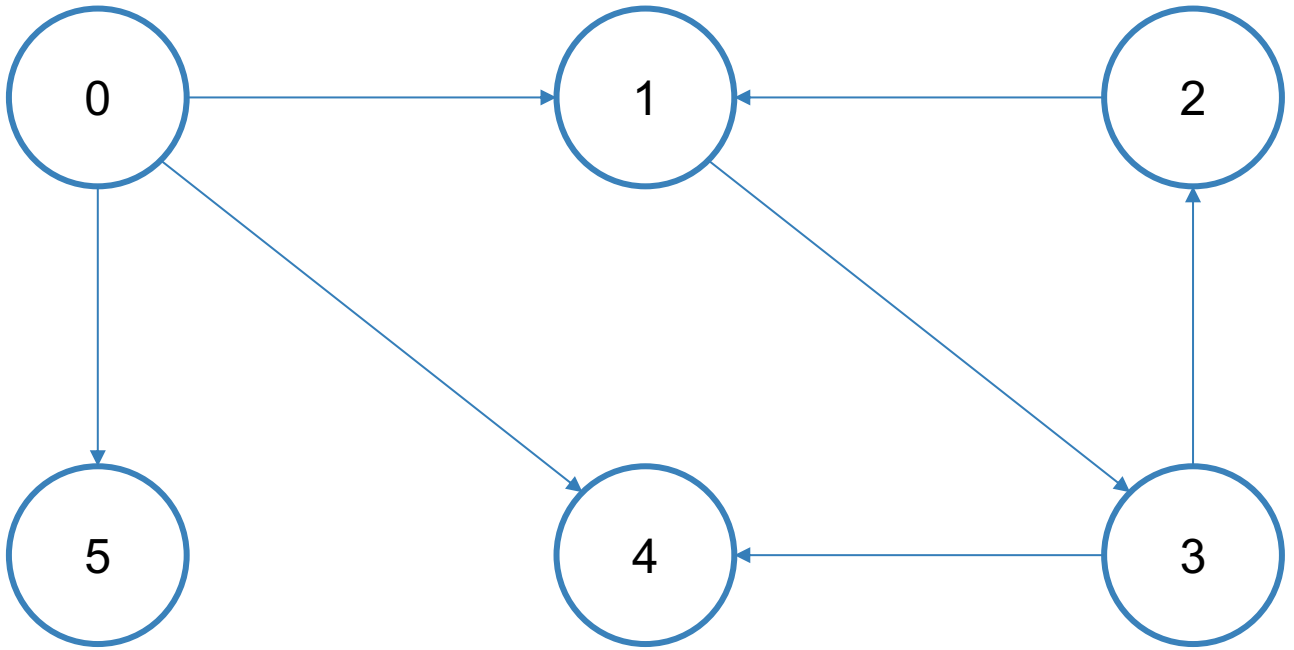
- In order traversal: visit left first (ascending)
- Post-order: visit current node after children
- Pre-order: visit current node before children

Graphs

- 🤯 A tree is a type of graph
- 🤯 Not all graphs are trees

Searching

- Depth-First
- Breadth-First
- Bidirectional



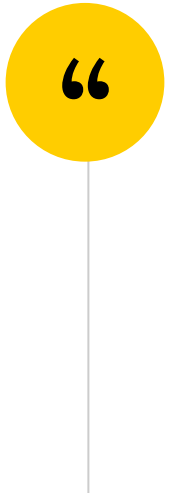
Most common Os

- O(1) – one line of code
- O(log n) – divide in half
- O(n log n) – line of code and divide
- O(n) linear – loop
- $O(n^2)$, $O(2^n)$ – nested loop

Lecture 3

Solving problems

If all you have is a hammer, everything looks like a nail. – Abraham Maslow

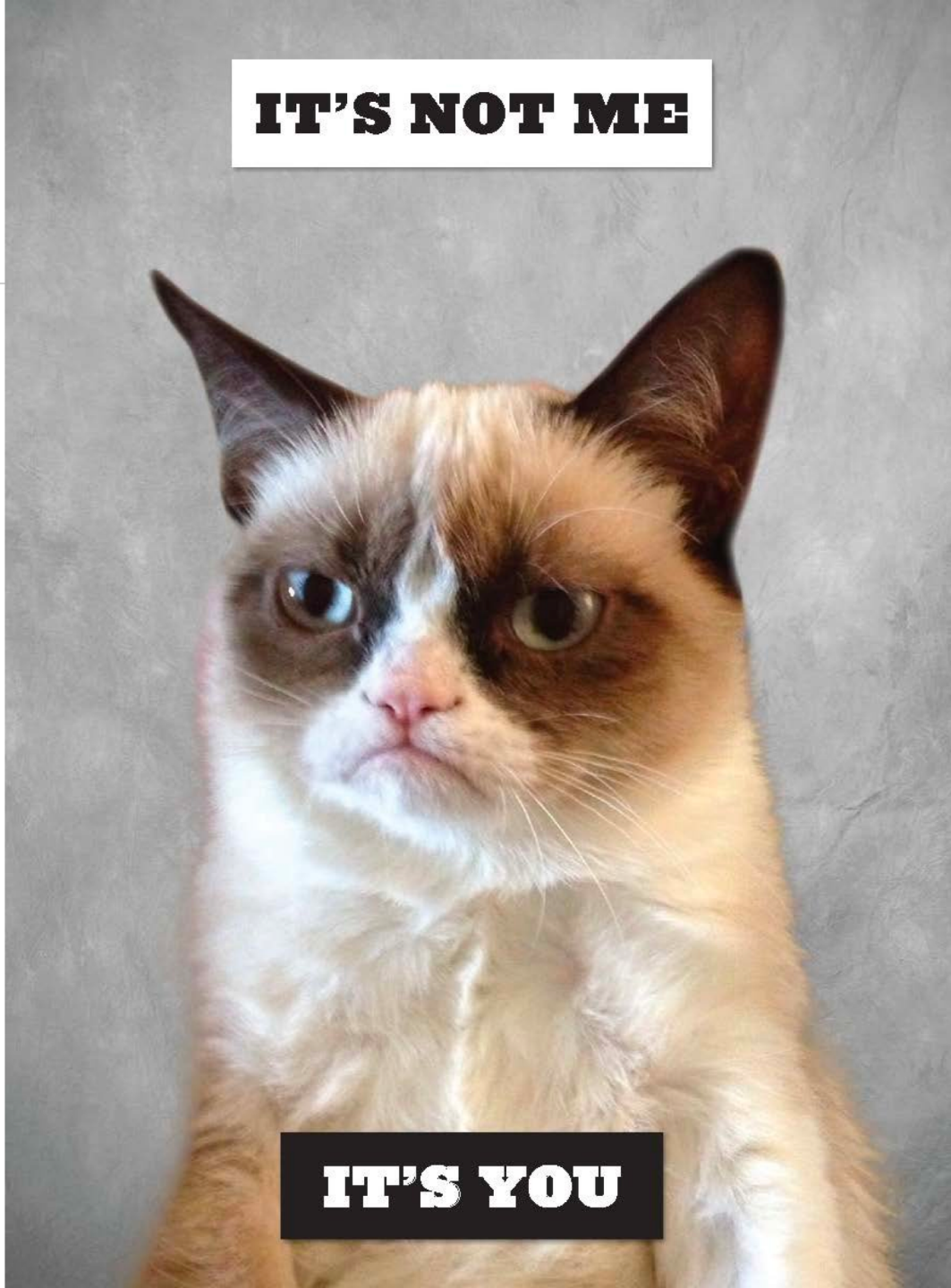


Good problem solver

- Positive attitude

Good problem solver

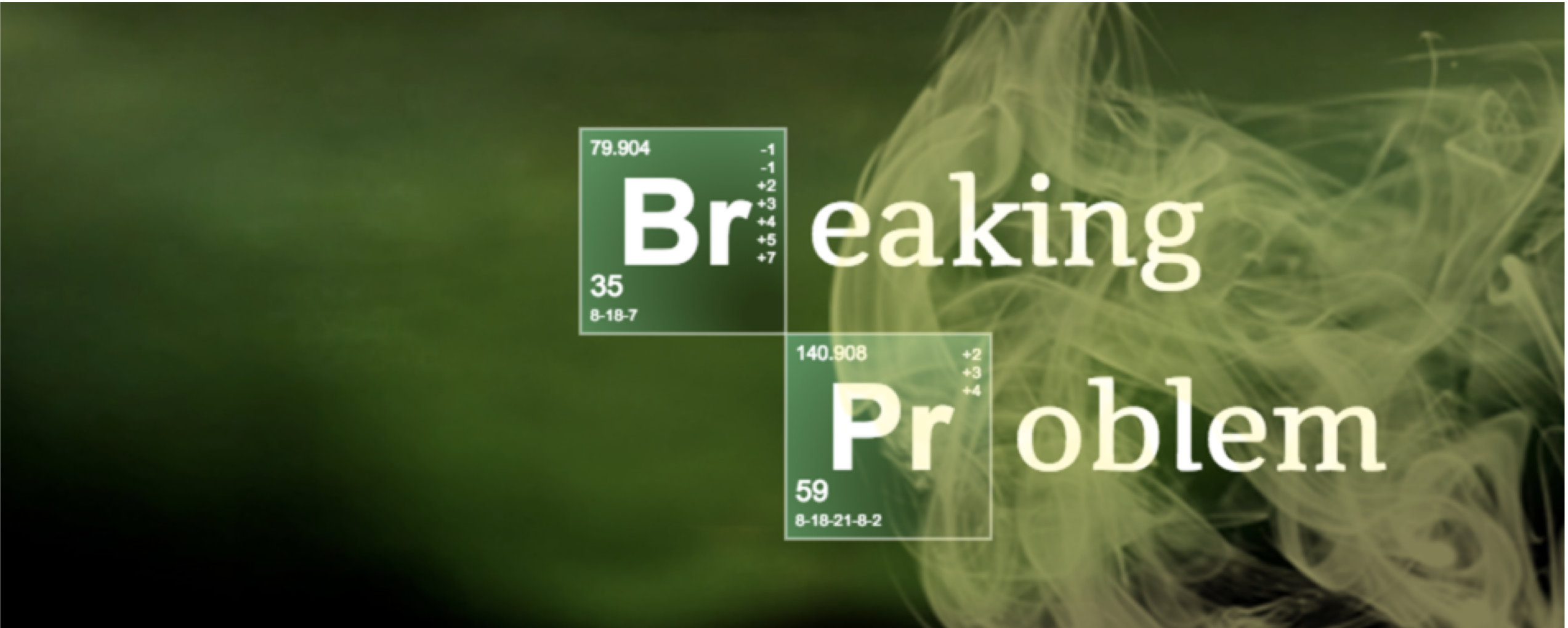
- Positive attitude
  - Solving though persistence



Good problem solver

- Concern for accuracy

Good problem solver



Good problem solver

- Avoid guessing

Good problem solver

- Active problem solving

Good problem solver

- Positive attitude
- Concern for accuracy
- Breaking problem into parts
- Avoid guessing
- Active problem solving

Problem solving is a skill

- Practice practice practice
- Thinking is a skill but invisible
  - Make it shine and visible
- Think aloud
  - Need to explain every step

Problem Solving Components

- Understanding

Problem Solving Components

- Understanding
- Design

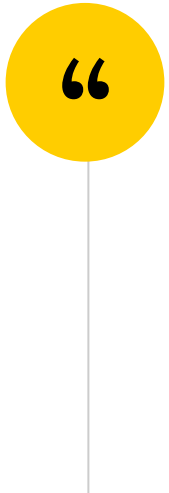
Problem Solving Components

- Understanding
- Design
- Writing

Problem Solving Components

- Understanding
- Design
- Writing
- Review

Example

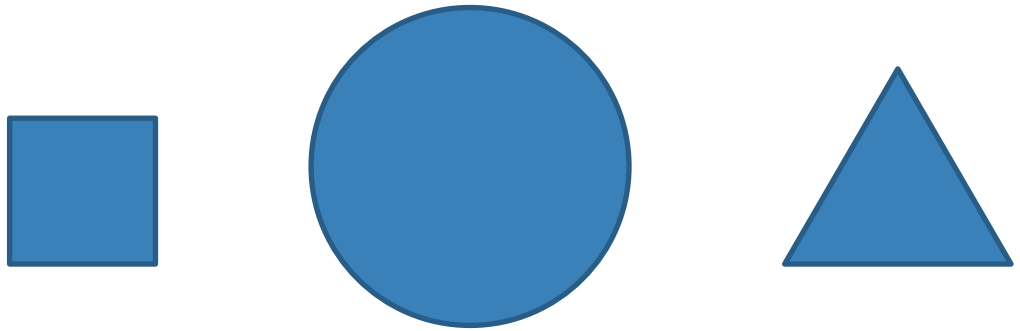


Game on!

- Groups of two:
  - 1→ Read and think aloud
  - 2 → Listen, check accuracy, demand vocalization

Problem 1

- Write a program that given a circle, square, and triangle. If circle is taller than the square and the triangle is shorter than the square, put a K in the circle. However, if this is not the case, put a T in the second tallest figure.
- Example:



Problem 2

- Write a program that takes a word as an input and if the word has less than 9 letters and more than 3 vowels, print the first vowel. Otherwise, print the consonant that is farthest to the right in the word.
- Example: *sentence*

Types of problems

- Complete search
- Divide and Conquer
- Greedy
- Dynamic programming

Complete Search

- Brute force
  - Try all possible solutions
  - Rarely optimal
- Recursive backtracking
  - Define contains to eliminate obvious non-candidates

Complete Search

- Filtering:
  - examining lots of candidates to select correct or remove incorrect
  - Iterative
  - Easy to code, run slow
- Generators:
  - gradually build solution by removing incorrect
  - Recursive
- Do the math to determine which

Complete Search

Write a program which takes as input two sorted arrays, and returns a new array containing elements that are present in both of the input arrays. The input arrays may have duplicate entries, but the returned array should be free of duplicates. For example, the input is (2, 3, 3, 5, 5, 6, 7, 7, 8, 12) and (5, 5, 6, 8, 8, 9, 10, 10), your output should be (5, 6, 8).

Jupyter

Complete Search

- Last but not least:
- Using better data structures and algorithms always outperform any optimizations.
- Know you DS, algs, libs

Divide and Conquer

- Make it simpler by dividing it to smaller problems

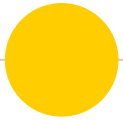
1. Divide the original into sub-problems
2. Find sub-solutions
3. If needed combine them for complete solution

Divide and Conquer

- Binary search

Greedy

- Making locally optimal choice with the hope
- Need two ingredients:
  - Optimal sub-structures
- It has a greedy property



Greedy

- Suppose we have large number of coins with different values: 25, 10, 5, 1.
- We want to make change with least number of coins.
- Give a try!

Greedy

- Keep using the largest possible – greedy
- 47 {25, 10, 5, 1}
- 47 – 25 = 22
- 22 – 10 = 12
- 12 – 10 = 2
- 2 – 1 = 1
- 1 – 1 = 0
-

Greedy

- But!
- Does not work for optimal for *all*
- 6 and {4, 3, 1}
- Greedy -> ?
- Optimal -> ?
-

Dynamic programming

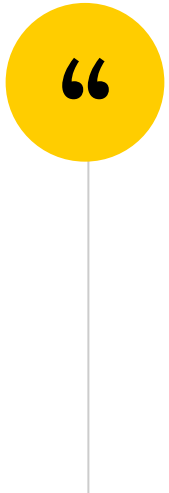
- The most difficult paradigm
- Backtracking on steroids
- Used to solve optimization and counting problems
- Runs naïve exponential in n^2 n^3
- Unlike D&C sub-problems overlap

Dynamic programming

- Longest common subsequence
- Given two sequences, find the length of longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous.
- “abcdefg” –> “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”,

Jupyter

Almost there



Prepare for quiz

- Next quiz: June 15, 11:30 AM!

Problematic Homework

- Solve problem

Good problem solver

- Positive attitude
- Concern for accuracy
- Breaking problem into parts
- Avoid guessing
- Active problem solving

Problem Solving Components

- Understanding
- Design
- Writing
- Review

Types of problems

- Complete search
- Divide and Conquer
- Greedy
- Dynamic programming

The end

- Problems are problematic

That's it for today