# 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

Afternoon FREE :D

## 2 x 🧠 💨

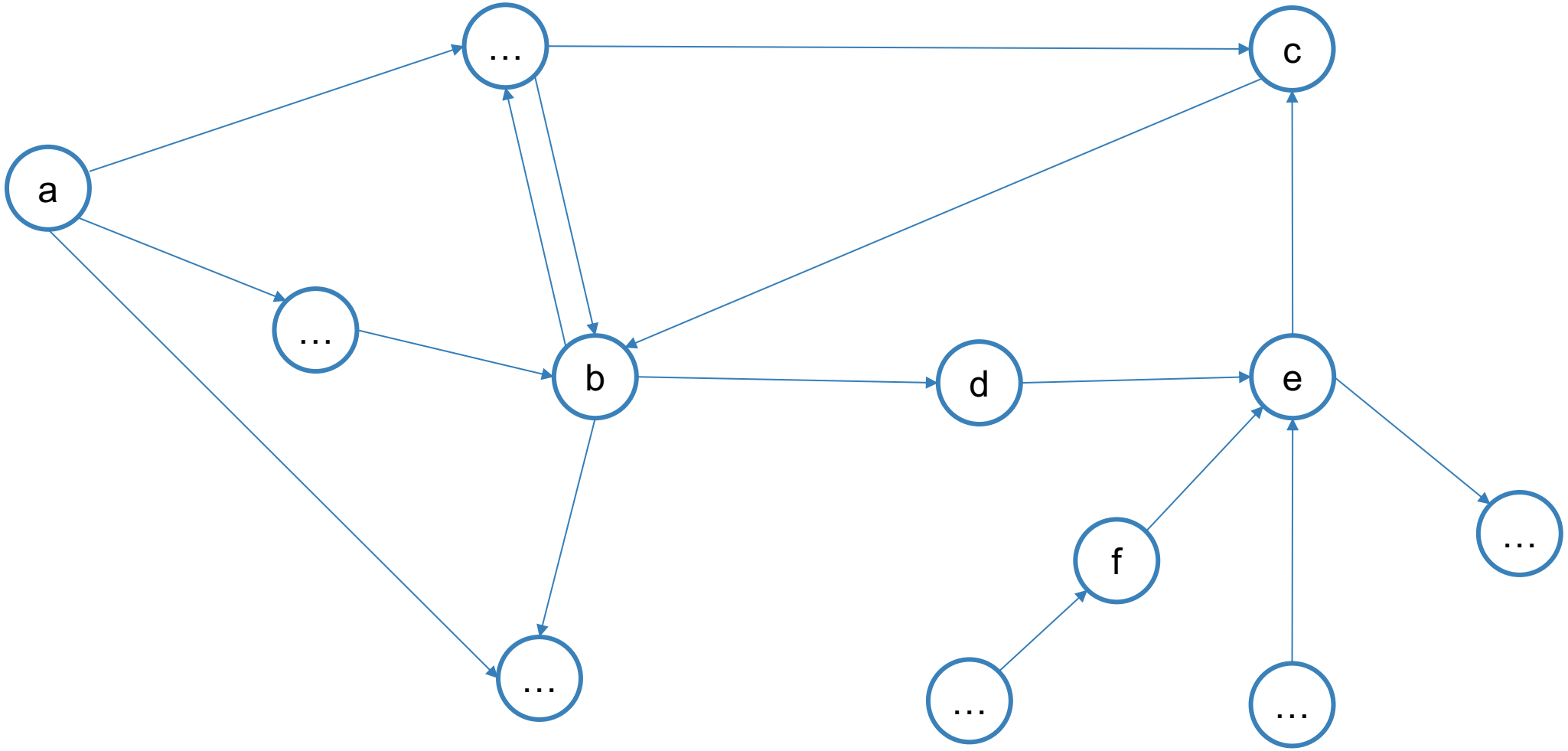
- Quiz question in class
- Bi-directional in direct

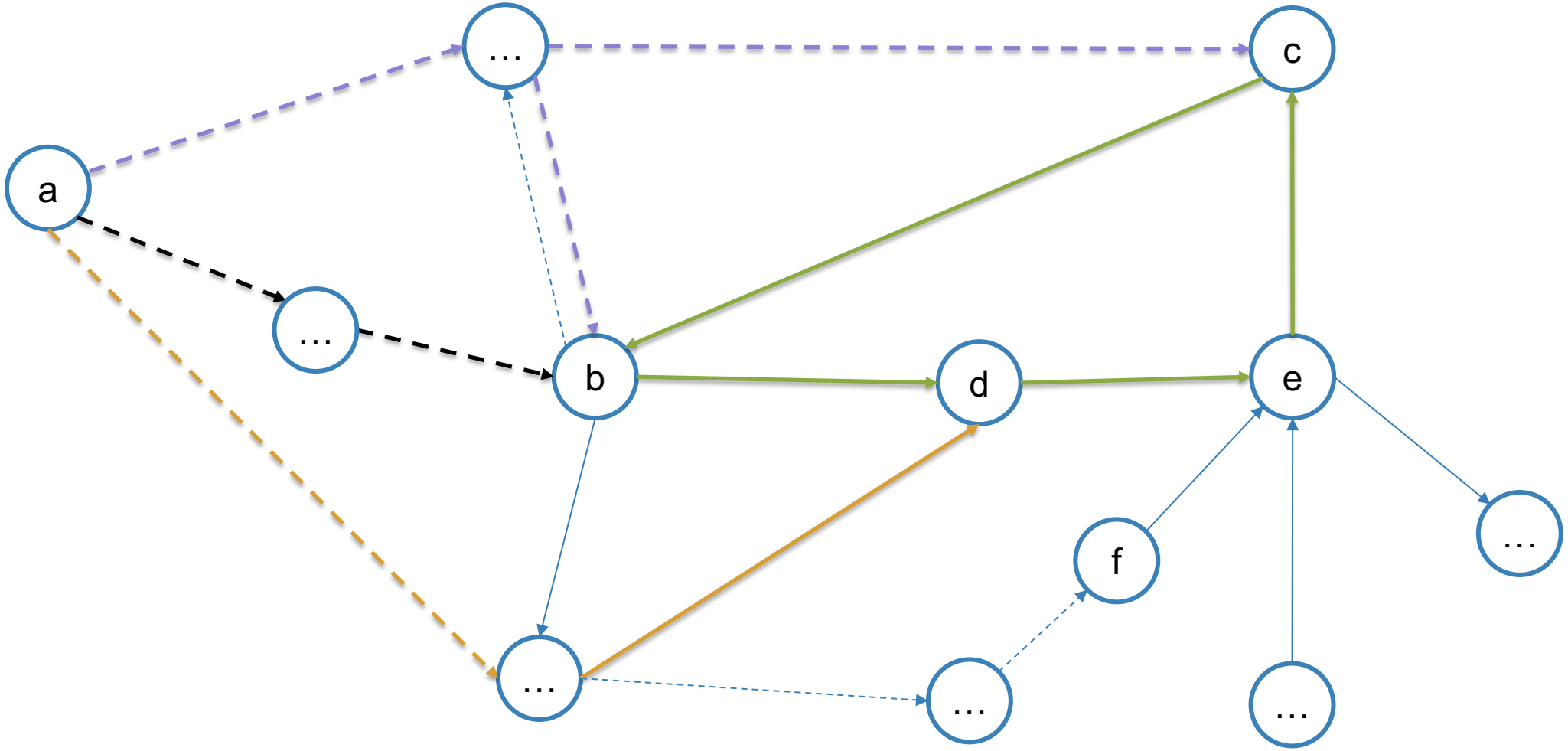
## 2 x 🧠 💨

- Quiz question in class

```
for n in b:
        while n>1 :
                n=n/2
```

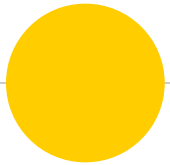
b ≈ n -> O(n*log(n))


a
…
…
b
…
d
c
e
f
…
…
…


a
…
…
b
d
c
e
f
…
…
…
…

# Graphs

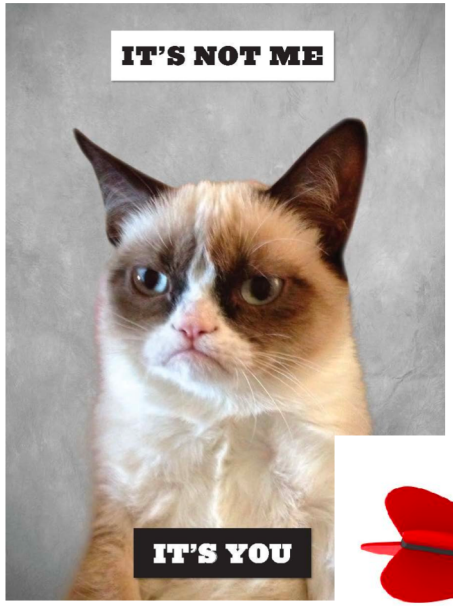
- a -> b -> a
- Cycles, cycles, cycles
- Directed -> undirected
- Null, trivial, non-directed, directed, simple, connected, disconnected, regular, complete,  cycle, wheel, cyclic, acyclic, bipartite, complete bipartite, star, complement, ...

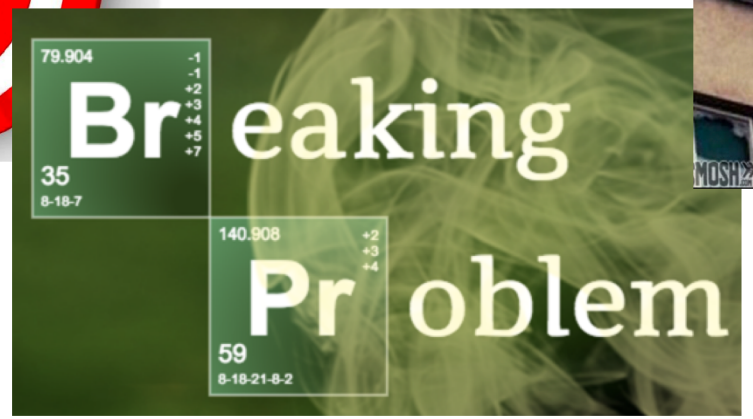


# Last Christmas

recap

# Good problem solver





# Problem Solving Components

- Understanding
- Design
- Writing
- Review

## Types of problems

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

# Lecture 4

String and Math

## String matching

- Find starting index/indices of a(sub)string in long strings
- Pure + short = library

# String accessing

str[0]

str[-1]

str[3:5]

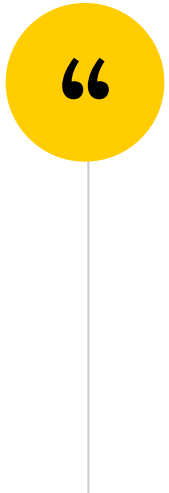
str[7:]

str[:6]

str[7:-4]

# *Junyper: dir('know your libraries')*

*Every substring must be a prefix of one of all possible suffixes*



## String processing

- Suffix Trie (information retrieval)
- A tree of all possible suffixes.
- Each edge -> character
- Each vertex is connected to some other ~26 vertices
- Boolean flags indicating termination/existence of suffix/word
- Demo

## String processing

- Suffix Trie (information retrieval)
- Good for dictionaries
- O(m)

# String processing

- Suffix Tree
- Compressed tree for all suffixes
- Find all, sort, clean, build
- Demo

## String processing

- Suffix Tree, Good for

- Pattern Searching
- Longest repeated substring
- Longest common substring
- Longest palindrome in a string

# String processing

- Generalized suffix tree
  - combines multiple suffix trees
  - Solves longest common substring in O(n)
- Suffix array
  - O(nlogn)
  - MUCH simpler to implement
  - Integer array storing permutations of n indices of sorted suffixes.

*Jupyter*

*Time for math!*

# Fibonacci

- Jupyter

# Prime numbers

- Brain work!
- Let’s generate list of primes
- Create teams of 2, understand, design, write, review

# Prime numbers

- Write a program highly efficient generating a list of prime numbers up to MAX

*Jupyter*

In [27]:
```
def is_prime(n):
    if n==2 or n==3:
        return True
    if n%2==0 or n<2:
        return False
    for i in range(3,n):
        if n%i==0:
            return False

    return True
```

*is prime*

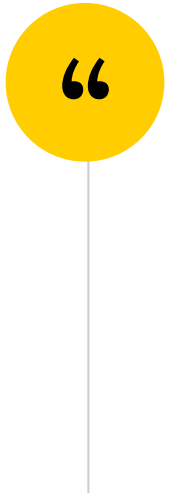
```
def is_prime(n):
    if n==2 or n==3:
        return True
    if n%2==0 or n<2:
        return False
    for i in range(int(n**0.5)+1,
        if n%i==0:
            return False

    return True
```

*is better prime*

# A generic approach

- You have two ropes, and each takes exactly one hour to burn. How would you use them to time exactly 15 minutes? Note that the ropes are of uneven densities, so half the rope length-wise does not necessarily take half an hour to burn.

# *Almost there*



# Prepare for quiz

- Next quiz: June 22, 10:00 AM!
- Keynote 11–12: done!

# Problematic Homework

- Palindrome
- Call me maybe?

*That's it for today*