

Introduction to Algorithms

What are algorithms?

algorithm
 /əlˈgoθɪm/ (ə)

noun
 noun: algorithm; plural noun: algorithms

a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
 "a basic algorithm for division"

Origin



late 17th century (denoting the Arabic or decimal notation of numbers); variant (influenced by Greek *arithmos* 'number') of Middle English *algorism*, via Old French from medieval Latin *algorismus*. The Arabic name, al-Kwārizmī 'the man of Kwārizm' (now Khiva), was a name given to the 9th-century mathematician Abu al-Hasan Muhammad ibn Mūsā, author of widely translated works on algebra and arithmetic.

<https://www.dictionary.com/browse/algorithm>

British Dictionary definitions for algorithm

algorithm

noun

1. a logical arithmetical or computational procedure that if correctly applied ensures the solution of a problem; Compare [heuristic](#)
2. **logic maths** a recursive procedure whereby an infinite sequence of terms can be generated

French name: **algorithme**

Derived Forms

algorithmic, adjective
algorithmically, adverb

Word Origin

C17: changed from algorism, through influence of Greek *arithmos* number

Collins English Dictionary - Complete & Unabridged 2012 Digital Edition
 © William Collins Sons & Co. Ltd. 1979, 1986 © HarperCollins
 Publishers 1998, 2000, 2003, 2005, 2006, 2007, 2009, 2012

algorithm in Science

algorithm

[ăl'gə-rīð'ĕm]

1. A finite set of **unambiguous** instructions performed in a prescribed sequence to achieve a goal, especially a mathematical rule or procedure used to compute a desired result. Algorithms are the basis for most computer programming.

The American Heritage® Science Dictionary
 Copyright © 2011. Published by Houghton Mifflin Harcourt Publishing Company. All rights reserved.

Combining definitions...

Algorithm (noun):

A *finite* sequence of *unambiguous* steps that *solves* a specific problem.

Combining definitions...

```
while True:  
    print("foo")
```

Algorithm (noun):

A *finite* sequence of *unambiguous* steps that solves a specific problem.

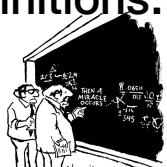
Combining definitions...

~~while True:
 print("Do")~~

Algorithm (noun):

A finite sequence of *unambiguous* steps that solves a specific problem.

Combining definitions...



Algorithm (noun):

A finite sequence of *unambiguous* steps that solves a specific problem.

Cartoon from <http://www.sciencecartoonsplus.com/gallery/math/math07.gif>
Copied under fair use, see <https://rationalwiki.org/wiki/File:Math07.gif>

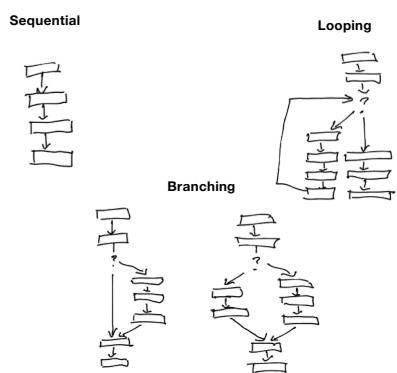
Combining definitions...

Algorithm (noun):

A *finite* sequence of *unambiguous* steps that solves a specific problem.

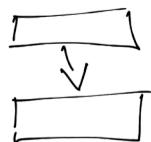


Building-blocks in Programming and in Algorithms



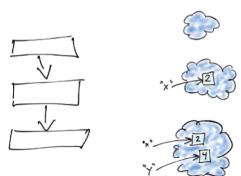
Sequential execution

```
print("Hello")
```



Variables (program state)

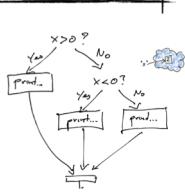
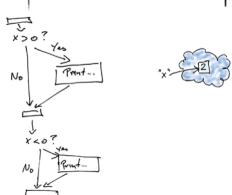
```
x = 2  
y = 2 * x  
print(x, y)
```



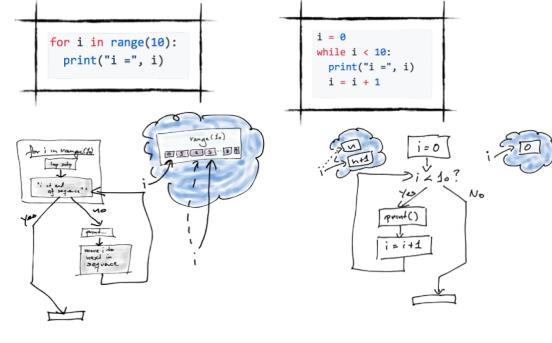
Branching

```
if x > 0:  
    print("x is positive")  
if x < 0:  
    print("x is negative")
```

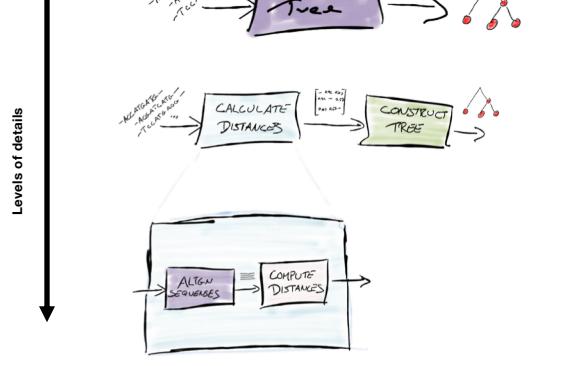
```
if x > 0:  
    print("x is positive")  
elif x < 0:  
    print("x is negative")  
else:
```

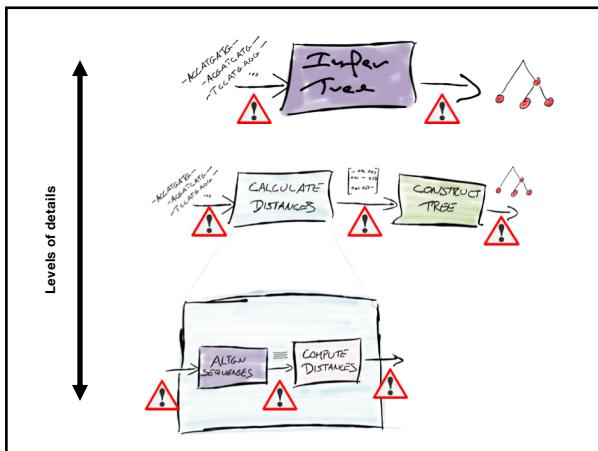


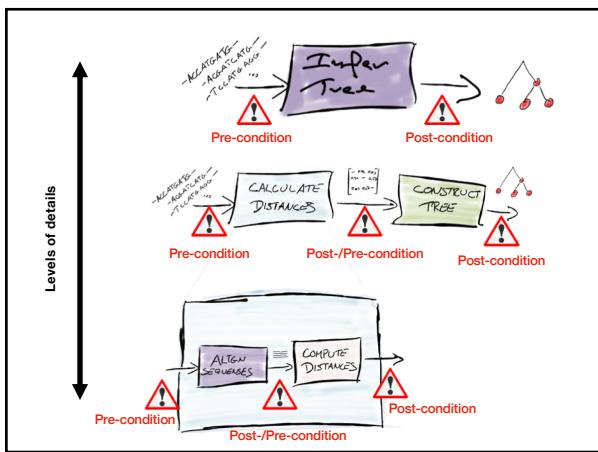
Looping

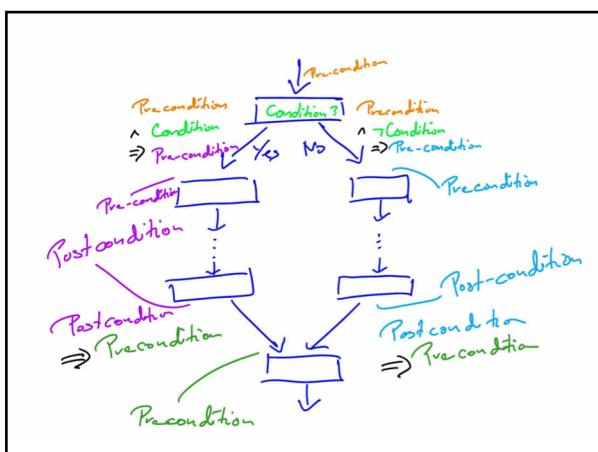


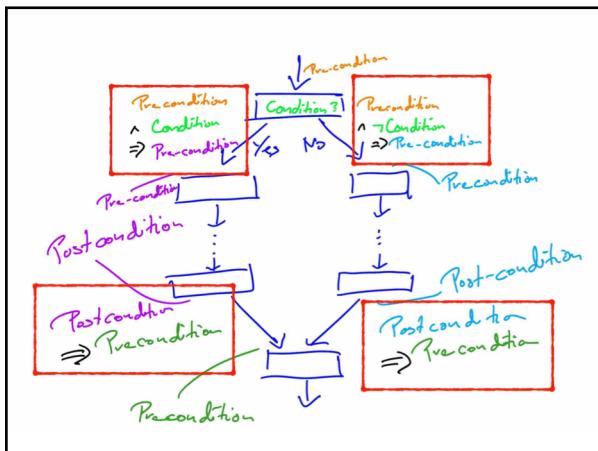
Designing algorithms:
Breaking down problems
to smaller parts

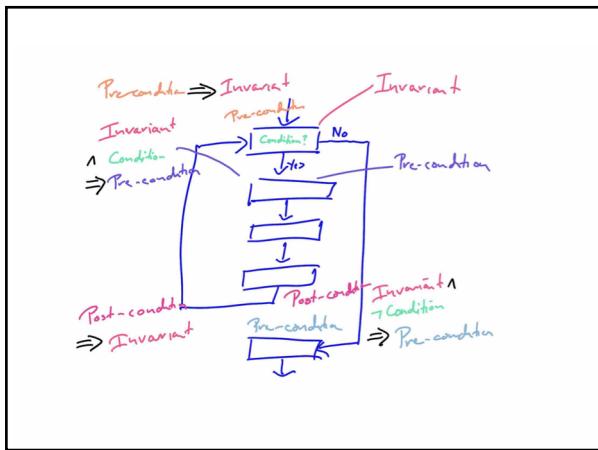


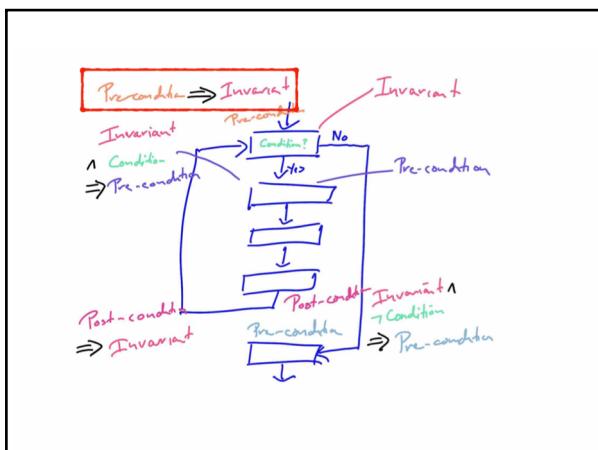


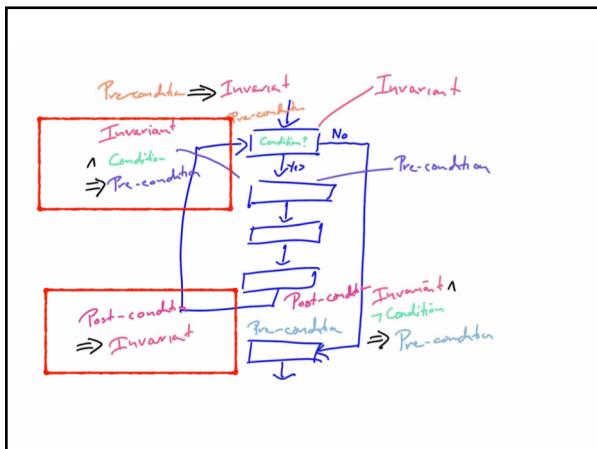


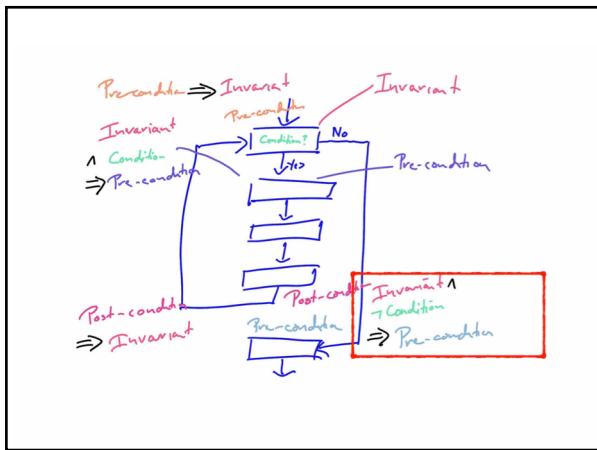


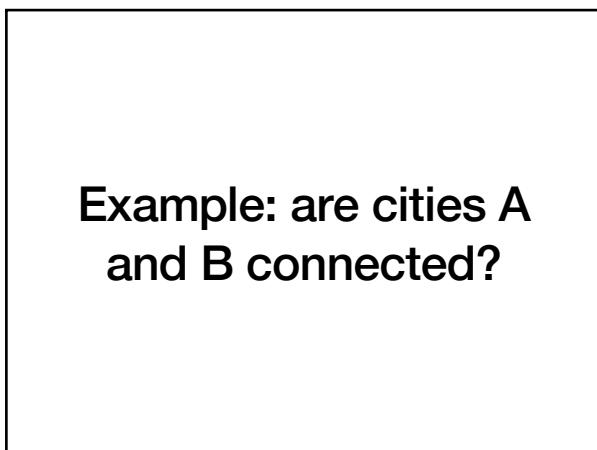


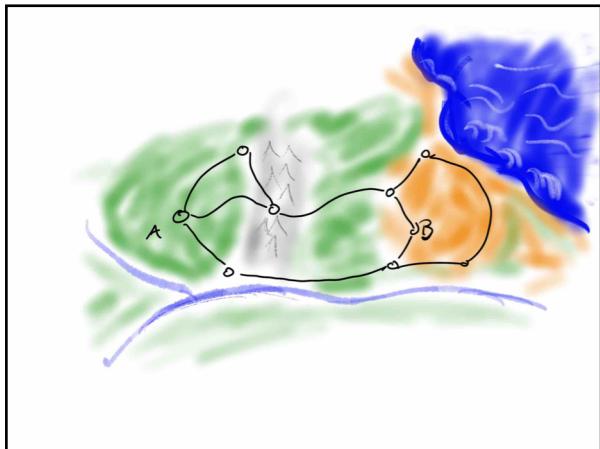


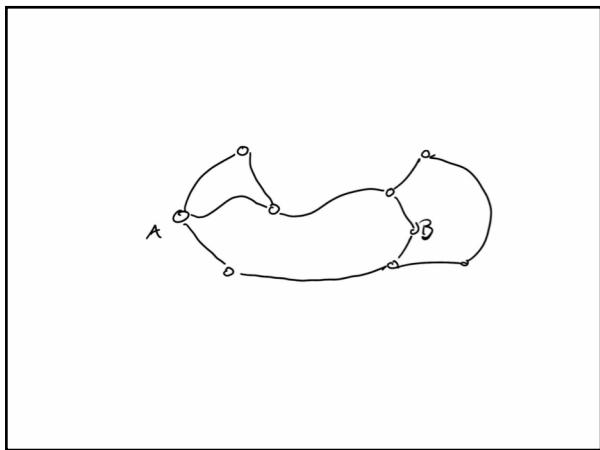


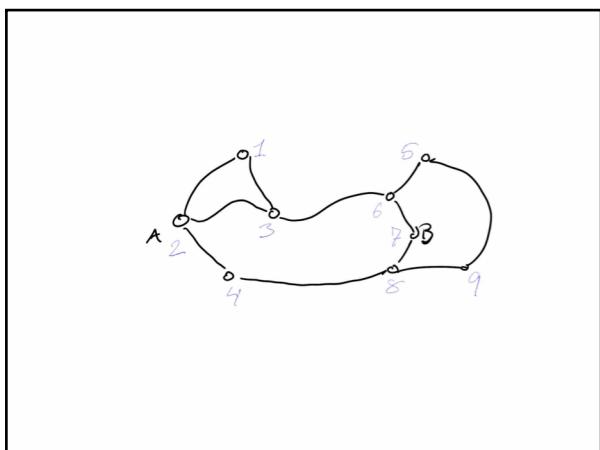


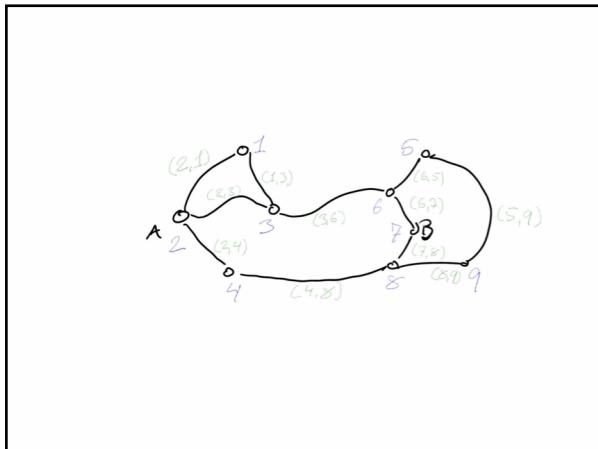


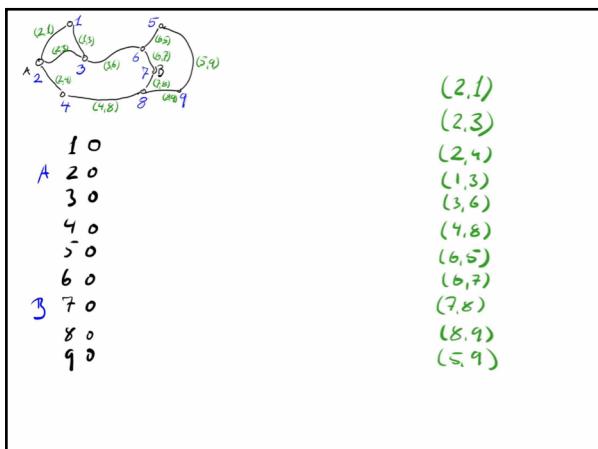


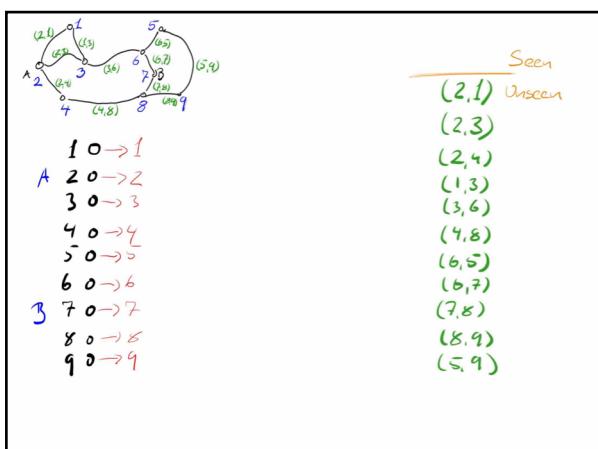


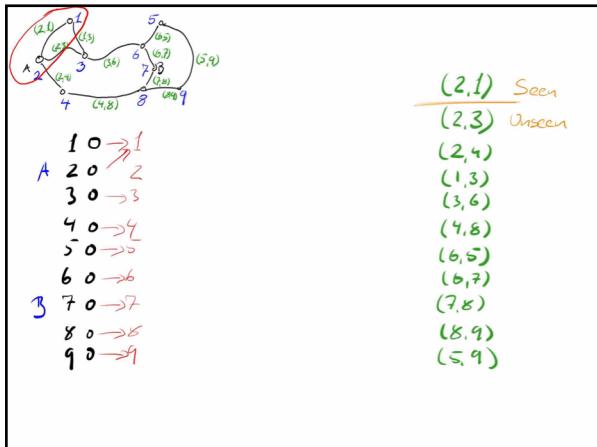


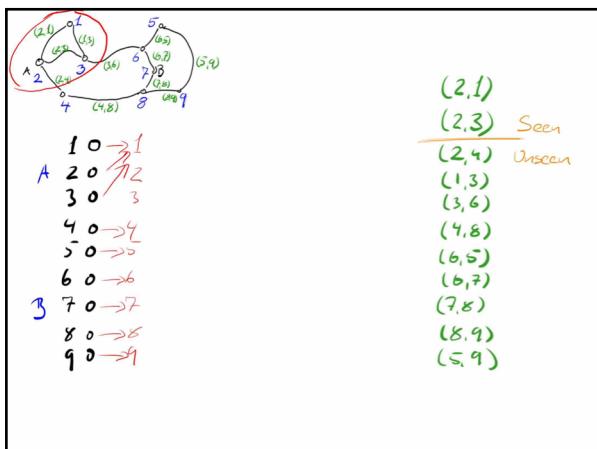


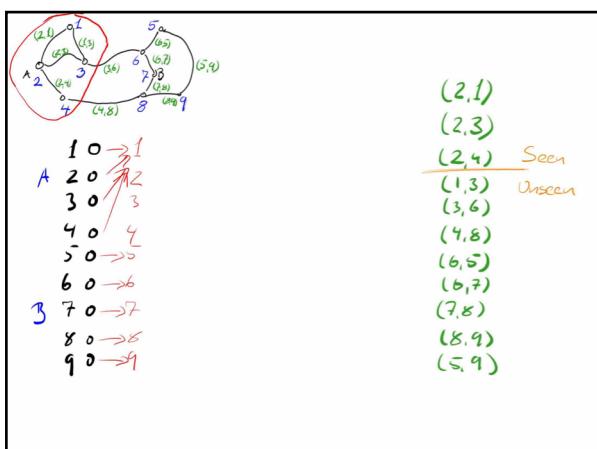


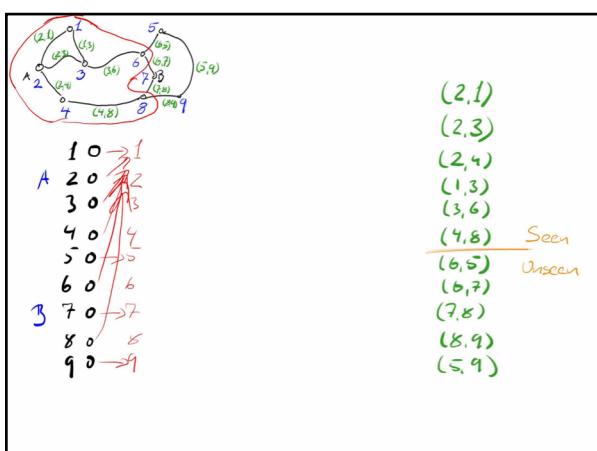
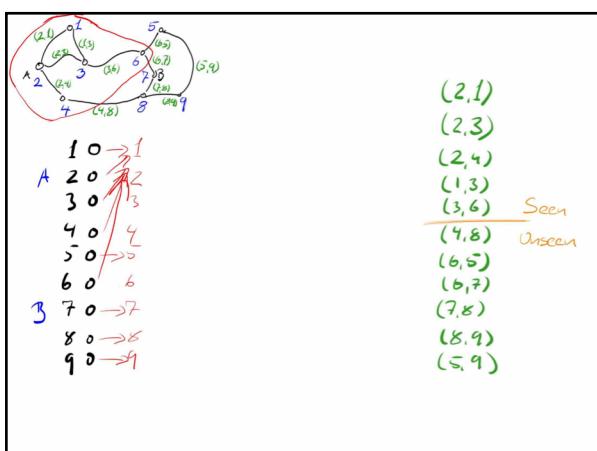
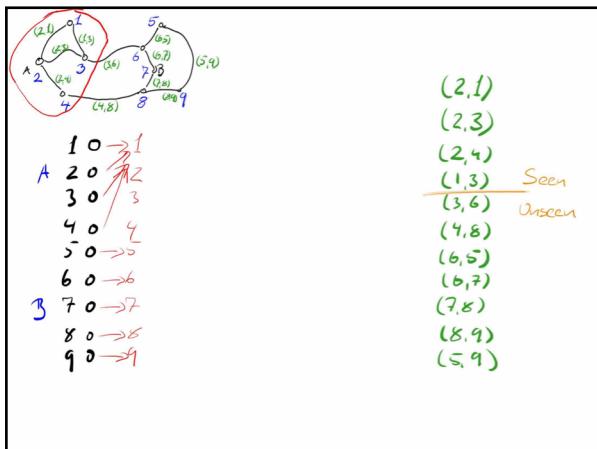


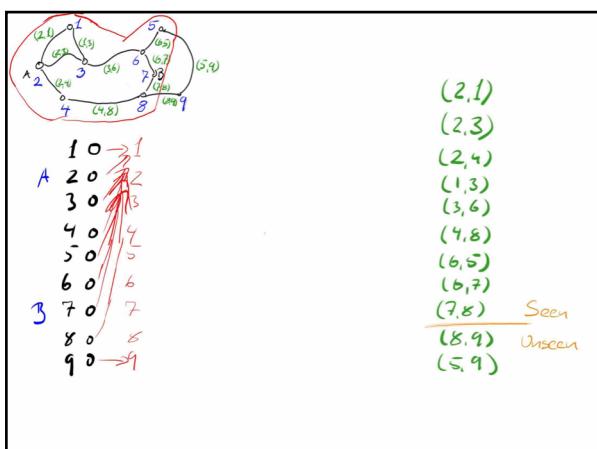
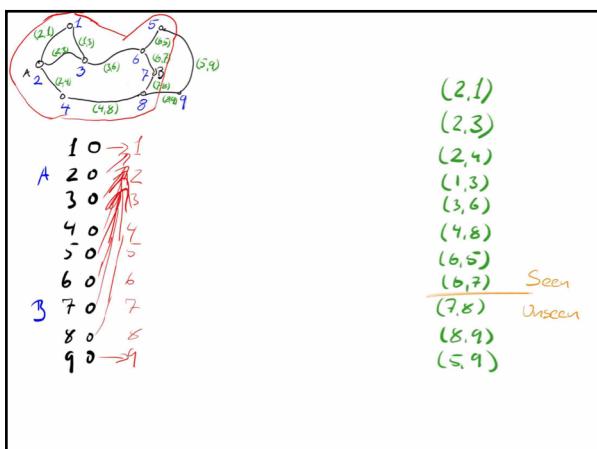
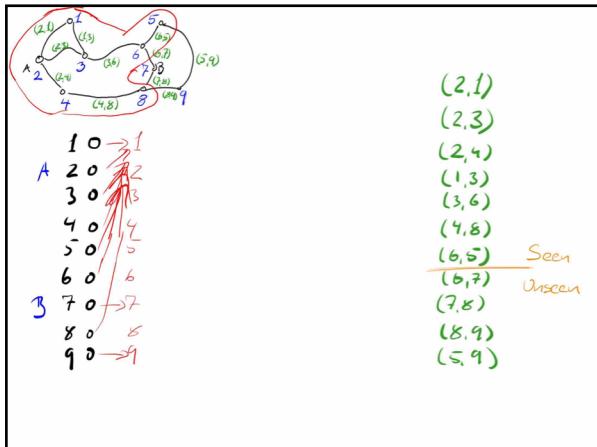


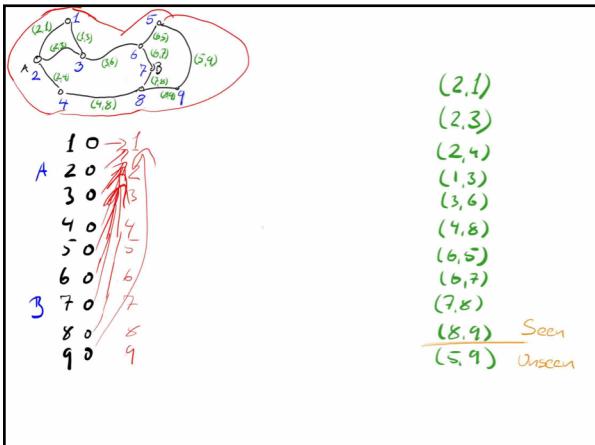


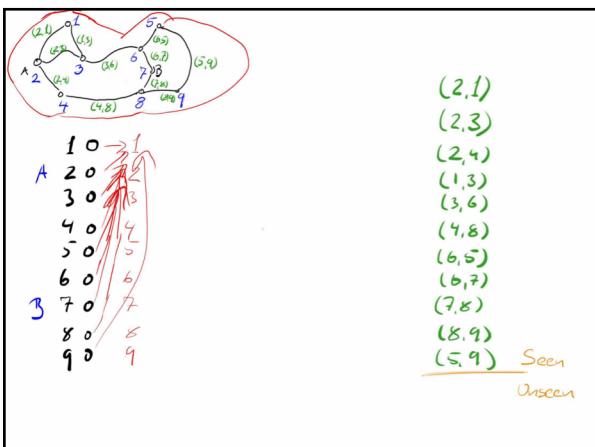










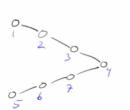
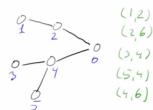


Intermezzo

- How would you represent the components in Python?
- How would you formalise the loop invariant to take into account the representation of the components?

Intermezzo

- Run the algorithm on these two graphs:

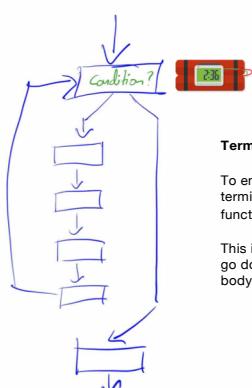


Correctness

How do we prove correctness?

- We prove that all pre- and post-conditions are satisfied through the steps in the algorithm.
- We prove that the post-condition of the last step implies that the overall problem is solved.
- Correctness is just a special case of post-conditions*

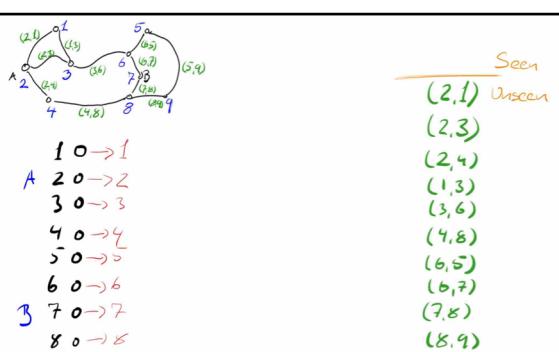
Termination



Termination functions:

To ensure that our algorithm terminates, we add a "termination" function to each loop.

This is a count-down that should go down each time we execute the body of the loop.



Get the binary representation of a number n

```
reverse_bits = []
while n > 0:
    reverse_bits.append(n % 2)
    n //= 2
print(reverse_bits[::-1])
```

Termination function:
 $t(n) = n$

Thats it!

Now it is time to do the exercises to test that you now know how to construct algorithms

