Informatics

Algorithmic Thinking
Part 2

Claudio Sartori
Department of Computer Science and Engineering
claudio.sartori@unibo.it
https://www.unibo.it/sitoweb/claudio.sartori/

Elaboration on © Pearsons – Fluency with Information Technology – Snyder – Ch.10

Intersecting *two* Lists (I2L)

Write the friends of Mary and John in 2 pieces of paper and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Julie	Susie	
Anne	Charlie	
Frank	Bob	
Susie	Anne	

Intersecting 2 Lists

- put a marker in the first position of each list
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie *	Susie *	
Anne	Charlie	
Frank	Bob	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie *	Susie	
Anne	Charlie*	
Frank	Bob	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie *	Susie	
Anne	Charlie	
Frank	Bob *	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie *	Susie	
Anne	Charlie	
Frank	Bob	
Susie	Anne *	

Intersecting 2 Lists

- different, but the right marker is at the end: reset the right marker and advance the left marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie *	
Anne *	Charlie	
Frank	Bob	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	
Anne *	Charlie *	
Frank	Bob	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	
Anne *	Charlie	
Frank	Bob *	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	
Anne *	Charlie	
Frank	Bob	
Susie	Anne *	

Intersecting 2 Lists

- equal: copy the element at the end of the intersection
- advance left marker, and reset right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	**Anne**
Anne *	Charlie	
Frank	Bob	
Susie	Anne *	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie *	**Anne**
Anne	Charlie	
Frank *	Bob	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	**Anne**
Anne	Charlie *	
Frank *	Bob	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	**Anne**
Anne	Charlie	
Frank *	Bob *	
Susie	Anne	

Intersecting 2 Lists

- different: advance the right marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie	**Anne**
Anne	Charlie	
Frank *	Bob	
Susie	Anne *	

Intersecting 2 Lists

- different: since the right marker is at the end, reset the right marker and advance the left marker
- compare the elements pointed by the markers: equal or different?

Friends of Mary	Friends of John	Intersection
Julie	Susie *	**Anne**
Anne	Charlie	
Frank	Bob	
Susie *	Anne	

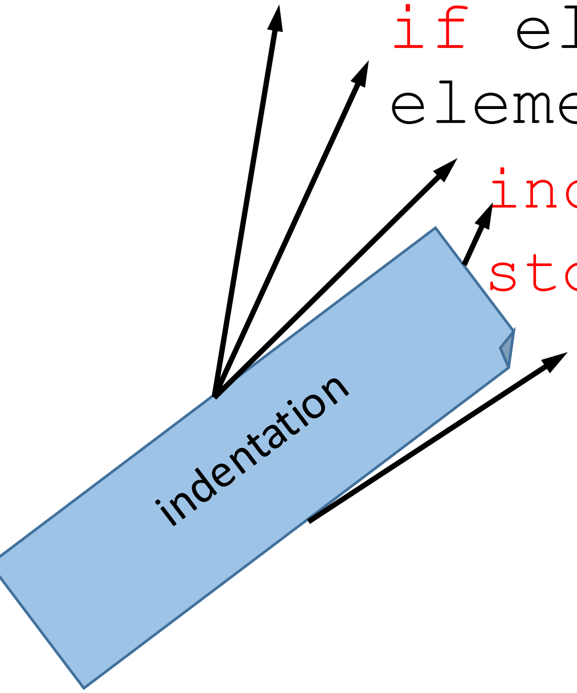
Intersecting 2 Lists

- equal: copy the element to the Intersection
- since the left list is at the end
  stop, because all the elements of the left list have been compared with all the elements of the right list

Friends of Mary	Friends of John	Intersection
Julie	Susie *	**Anne**
Anne	Charlie	**Susie**
Frank	Bob	
Susie *	Anne	

Intersecting two lists – algorithm in *pseudo-code*

```
prepare an empty variable intersection, set s to 0
repeat varying i from 1 to the length of list1
    repeat varying j from 1 to the length of list2
        if element in position i of list1 is equal to
        element in position j of list2
            increment s by 1
            store the element in position i of list1 into
                position s of intersection
```



Intersecting *two* *Alphabetized* Lists (IAL)

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne *	Anne *	
Frank	Bob	
Julie	Charlie	
Susie	Susie	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne *	Anne *	**Anne**
Frank	Bob	
Julie	Charlie	
Susie	Susie	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne	Anne	**Anne**
Frank*	Bob*	
Julie	Charlie	
Susie	Susie	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne	Anne	**Anne**
Frank*	Bob	
Julie	Charlie*	
Susie	Susie	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne	Anne	**Anne**
Frank*	Bob	
Julie	Charlie	
Susie	Susie*	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne	Anne	**Anne**
Frank	Bob	
Julie*	Charlie	
Susie	Susie*	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne	Anne	**Anne**
Frank	Bob	
Julie	Charlie	
Susie*	Susie*	

Intersecting *two Alphabetized* Lists (IAL)

Write the friends of Mary and John in *two* pieces of paper *in alphabetical order* and copy in a new piece of paper the *intersection*

Friends of Mary	Friends of John	Intersection
Anne	Anne	**Anne**
Frank	Bob	**Susie**
Julie	Charlie	
Susie*	Susie*	

Comparison

- IAL and IL are different algorithms
- IAL and IL accomplish the same thing, starting from the same data but in different order
- IL is much slower
- For two lists of 100 elements, worst case:
  - IAL: 100+100 = 200 comparisons
  - IL: 100×100 = 10,000 comparisons
- *how many steps do we need to put lists in alphabetical order?*
  - *you will learn it by the end of the course*

How Do We Know it Works?

- Algorithm solution is clear and simple and efficient
- Then, how do we know it works?
- If there is no loop, the program runs, gets to an end, and we can check the result
- What if there is a loop?
  - Programs with loops cannot be absolutely verified…there are too many possible cases

Then, what?

- *The way to know that an algorithm works is to know why it works…*

- Strategy for knowing why it works:
  - Find one or more properties that ensure the algorithm works
  - Explain, using the program, why they make it work.

Why IAL Works

- If a name appears in all lists, it forms a barrier
- Pointers that haven’t reached the barrier move sooner than ones that have
- When they all reach the barrier, the name is recorded
- None can pass the next barrier until they all reach it
- Etc.

Summary

- We use algorithms daily, and we continually create them as we instruct other people in how to do something
- Everyday algorithms can be sometimes be unclear because natural language is imprecise
- Algorithms have five fundamental properties

Summary

- Algorithms can be given at different levels of detail depending on the abilities of the agent
- Problems can be solved by different algorithms in different ways
- Algorithms always finish—either they give the answer, or say no answer is possible—and they are evaluated on their use of resources such as space and time

try to write the IAL algorithm in *pseudo-code*