Module III: Processes
Learning Goals (1/2)

• recognize and explain the five essential properties of an algorithm: input specified, output specified, definiteness, effectiveness, and finiteness

• explain how variables are used to ease data management and to describe actions on data, and be able to use variables for these purposes in familiar contexts

• recognize and explain the concept of sequences of instructions, variables, loops, functions, conditional statements, and arrays in short programs specified in a programming language such as JavaScript, or in other clearly expressed processes (which may or may not be computer related)
Learning Goals (2/2)

• determine, simulate and explain how short programs or program fragments work

• make small modifications to short program fragments to achieve clearly specified tasks

• explain and simulate simple algorithms for fundamental problems such as searching
Putting it all together

Data Organization

Interfaces

Processes
Okay, let’s design a process then!

• Let’s assume that we want to find when something appears in all of a set of lists
• This is useful in things like looking at DNA – we might look for repeating segments and see if they occur in each sample that you’re looking at.
• However, since looking at DNA is hard to visualize, let’s look at words instead:
Problem Example

Inputs:
- List 1: Socks, Box, Knox
- List 2: Socks, On, Knox
- List 3: Knox, In, Box

Desired output:
- Intersect List: Knox
**Intersect Unordered Lists**

**Assumption:** Input lists are in *any* lexicographic order

**Procedure** *Intersect Unordered Lists*:
1. Put a marker/arrow at the start of each list
2. If markers point to the same item, save it
3. Move marker of one list down by one
4. If the marker is at the end of the list, move the marker to the top and advance the marker on the next list by one
5. Repeat 2-4 until all markers are at the end of the list
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

At this point, we’ve only seen that the first thing in list 1 and the first thing in list 2 do not match with anything in List 3. So far we’ve done 1*3 steps.

We have to keep checking – next, we’ll try everything in List 3 with the second thing in List 2 and (still) the first thing in List 1.
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
Socks
Box
Knox

List 2
Socks
On
Knox

List 3
Knox
In
Box

Intersect List
At this point, we’ve seen that the first thing in list 1 and the first two things in list 2 do not match with anything in List 3. So far we’ve done 2*3 steps.

We have to keep checking – next, we’ll try everything in List 3 with the third thing in List 2 and (still) the first thing in List 1.
Intersect Unordered Lists

List 1: Socks, Box, Knox
List 2: Socks, On, Knox
List 3: Knox, In, Box

Intersect List
Intersect Unordered Lists

List 1: Socks, Box, Knox
List 2: Socks, On, Knox
List 3: Knox, In, Box

Intersect List
Intersect Unordered Lists

At this point, we’ve seen that the first thing in list 1 does not match with anything that appears in both lists 2 and 3. So far we’ve done 3*3 steps.

We have to keep checking – next, we’ll try everything in Lists 2 and 3 with the second thing in List 1.
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
Socks
Box
Knox

List 2
Socks
On
Knox

List 3
Knox
In
Box

Intersect List
Intersect Unordered Lists

List 1

- Socks
- Box
- Knox

List 2

- Socks
- On
- Knox

List 3

- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
At this point, we’ve seen that the first and second things in list 1 do not match with anything that appears in both lists 2 and 3. So far we’ve done 2*3*3 steps.

We have to keep checking — next, we’ll try everything in Lists 2 and 3 with the third thing in List 1.
Intersect Unordered Lists

List 1
Socks
Box
Knox

List 2
Socks
On
Knox

List 3
Knox
In
Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1: Socks
      Box
      Knox

List 2: Socks
      On
      Knox

List 3: Knox
      In
      Box

Intersect List
Intersect Unordered Lists

List 1

Socks
Box
Knox

List 2

Socks
On
Knox

List 3

Knox
In
Box

Intersect List

Connecting with Computer Science
www.ugrad.cs.ubc.ca/~cs101
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
Intersect Unordered Lists

List 1
Socks
Box
Knox

List 2
Socks
On
Knox

List 3
Knox
In
Box

Intersect List
Intersect Unordered Lists

List 1
- Socks
- Box
- Knox

List 2
- Socks
- On
- Knox

List 3
- Knox
- In
- Box

Intersect List
- Knox

MATCH!!
Intersect Unordered Lists

List 1
Socks
Box
Knox

List 2
Socks
On
Knox

List 3
Knox
In
Box

Intersect List
Knox
Intersect Unordered Lists

We are at the end of all the lists so we are done!

We have checked everything in list 1 with everything against list 2 and everything in list 3 – we have done 3*3*3 steps
Can we do any better?
Intersect Alphabetized Lists

**Assumption:**
All input lists are in alphabetic (= lexicographic) order

**Procedure** *Intersect Alphabetized Lists:
1. Put a marker/arrow at the start of each list
2. If all markers point to the same item, save it, because they are the same item
3. Move the marker(s) to the next position for whichever item is the earliest in the alphabet
4. Repeat Step 2-3 until some marker reaches the end of the list
Intersect Alphabetized Lists

Which list should we move down? Well, can “Elephants” ever match with anything? Yes (and in fact it does). But because “Amy’s” is the thing that we are checking that is alphabetically first, it can never match anything.
Intersect Alphabetized Lists

List 1
Elephants
Have
Humongous
Noses
Today

List 2
Amy’s
Elephants
Have
No
Noses

List 3
Elephants
Fix
Flying
Noses
Surrounding
The
Zoo

We *never* have to check “Amy’s” again! Since everything in lists 1 and 3 comes later in the alphabet, “Amy’s” can *never* match with anything!

Total number of steps with “Amy’s”? 1!
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy’s
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

Intersect List
- Elephants

MATCH!!
Which marker do we move now? The algorithm tells us to move the marker for the **alphabetically first item**. In this case, that’s all of them! So we might as well move them all. Note: this is tricky if we’re looking for duplicates (but we’re not, so carry on!).
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy’s
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

Intersect List
- Elephants
Intersect Alphabetized Lists

List 1: Elephants Have Humongous Noses Today
List 2: Amy’s Elephants Have No Noses
List 3: Elephants Fix Flying Noses Surrounding The Zoo

Intersect List: Elephants
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy's
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

Intersect List
- Elephants
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy’s
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

Intersect List
- Elephants

Connecting with Computer Science
www.ugrad.cs.ubc.ca/~cs101
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy’s
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

Intersect List
- Elephants
Intersect Alphabetized Lists

List 1
- Elephants Have Humongous Noses Today

List 2
- Amy's Elephants Have No Noses

List 3
- Elephants Fix Flying Noses Surrounding The Zoo

Intersect List
- Elephants
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy's
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

Intersect List
- Elephants
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy’s Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding The Zoo

Intersect List
- Elephants
- Noses

MATCH!!
Intersect Alphabetized Lists

Now what? Well, there’s **nothing** left that can possibly match anything else in List 2 – everything before “noses” has been checked, and there’s nothing afterwards, so we’re done!
**Intersect Alphabetized Lists**

**List 1**
- Elephants
- Have
- Humongous
- Noses
- Today

**List 2**
- Amy’s
- Elephants
- Have
- No
- Noses

**List 3**
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

How many steps did we take over all? We only moved the marker *one* time for each element in each list, so $5+5+7 = 17$
Intersect Alphabetized Lists

List 1
- Elephants
- Have
- Humongous
- Noses
- Today

List 2
- Amy’s
- Elephants
- Have
- No
- Noses

List 3
- Elephants
- Fix
- Flying
- Noses
- Surrounding
- The
- Zoo

- Steps for Intersect Any Order List: $5 \times 5 \times 7 = 175$
- Steps for Intersect Alphabetized List: $5 + 5 + 7 = 17$
RQ: IAL vs NAL

Despite NAL taking more time than IAL, do you think that there will be a point in time that the difference in the amount of time they take would be negligible? Or do you think that algorithms will continuously become more complex and therefore there will still be a difference resulting in a preferred algorithm?
Intersect Alphabetized Lists vs Intersect Unordered Lists

- **Intersect Alphabetized List:**
  - Requires input lists to be in alphabetical order
  - Requires much fewer steps than Intersect Any Order List

- **Intersect Any Order List:**
  - Does not require the lists to be in alphabetical order
  - Requires many more steps than Intersect Alphabetized List

**Lesson learned:** Organizing data in certain ways can make computations much more efficient!
But let’s talk a bit about programming

You’ve all done the scratch lab. Let’s take a look at something that’s a teeny part of what you did, but makes a big change…
A variable is a named quantity.
Playing with variables

• The following examples work pretty much the same no matter which programming language you use
• That being said, it’s helpful to have something to test it out for you.
• So we’re going to be using Javascript
The Assignment Statement

<variable> <assignment symbol> <expression>;

• <variable> is any declared variable in the program

• <assignment symbol> is the language’s notation for the assignment operation

• JavaScript’s <assignment symbol> is the equal sign (=)

• <expression> is a kind of formula telling the computer how to compute the new value

• Like any other statement, an assignment statement is terminated by a semi colon.
In computer science, the symbol “=” can be used in two ways:

1) to express equality (as in math),
   e.g., 3=5 (false) or 3=5-2 (true)
   (I read this as “equals” – as usual)

2) to assign values to variables,
   e.g., X=5 (after doing this, variable X
      has value 5, and the equality X=5 is true)
   (I like to read this as “gets”)

```
age = 38;
days_in_year = 365.25;
age_in_days = age * days_in_year;
```
Tips and Tricks

- In HTML a statement like $X=5$ always refers to a variable (actually, attribute) assignment.

- Likewise, in JavaScript, a statement like $X=5$ always denotes variable assignment, while equality is written differently ($X==5$).

- In other languages, ‘=‘ is used to denote equality, and ‘:=‘ or ‘<-‘ to denote variable assignment.