Mission to Mars - USD 383 Summer STEM - Day 2

Summer 2017

Learning Objectives

- Students will explore different methods for sorting data.
- Students will see how different sorting algorithms perform differently
- Students will learn how to use a computer program to simulate a real-world scenario

Resources

- Slides: http://people.cs.ksu.edu/~russfeld/presentations/stem2017/day2.html
- Scratch Website: http://scratch.mit.edu
- Sorting Networks: http://csunplugged.org/sorting-networks/
- Potatoes on Mars Simulator: https://scratch.mit.edu/projects/112633885
- Hydrazine on Wikipedia: https://en.wikipedia.org/wiki/Hydrazine#Rocket_fuel
- Description of Chemical Reaction: https://www.quora.com/How-did-Mark-Watney-produce-water-by-burning-Hydrazine-in-a-closed-chamber-in-Mars

Lesson Setup Before Class

- Log on to computers using STEM accounts
- Create Sorting Network on the floor using tape
- Make sure **Potatoes on Mars** is available on Scratch website.

Schedule

- 9:00 - Welcome & Icebreaker
- 9:15 - Sorting Networks & Sorting Cards
- 9:45 - Sorting in Scratch
- 10:15 - Break
- 10:20 - Martian Video & Basic Chemistry Intro
- 10:30 - Creating Water Simulator
- 11:00 - Wrap-Up Discussion

Lecture Notes

1. [**Icebreaker**] Take a minute to review what was learned yesterday. Some good questions:
    a. How do we move a sprite around the stage? *Motion blocks, Move ___ Steps, etc.*
    b. What block can we use to make the sprite turn around if it hits a wall? *If on edge, bounce*
    c. What block can we use to get the sprite to do the same thing over and over again? *Repeat or Forever*
    d. What block can we use to see if one sprite touches another sprite? *Sensing*

blocks, Touching _____

e. What block can we use to change what a sprite is doing depending on the output of another block? *If block or If - Else block*

2. [**Sorting**] Today, we are going to learn about how computers can sort data into a certain order. Let's talk about that:
    a. What are some different orderings of data you can think of? *Alphabetical, Numerical, smaller to larger, by height, weight, size, color, etc. Basically, get them thinking about how to categorize and sort data*
    b. Why would it be important for a computer to sort data? *Think about a dictionary; what if all the words were in a random order? Or a phone book?*

3. [**Sorting Network**] Let's take a look at one way that you can sort data, using a sorting network. <gesture to the sorting network on the floor> This is a special design that will guarantee any data at one end will be sorted by the time you reach the other end. Think it will work? **<get feedback>** Let's give it a try!
    a. Assign some numerical value to the students and put 6 of them at the start end of the network. Height works well, or the number of letters in their name. Try to avoid duplicates at first, but later you can deal with those as well.
    b. Have the students follow their lines. At any place where the lines intersect, the student with the higher value will go one direction, and the lower value will go the other. It helps to establish some consistent direction beforehand. I've always used "higher goes right."
    c. At the end, check to see if the students are sorted. If they aren't, see if you can start over and diagnose the problem. Some students like to go faster than others and miss the point. You can challenge the students to create groups and see which group can do it the fastest but correctly.
    d. Lots of good resources are available here: http://csunplugged.org/sorting-networks/
    e. See **SortingNetwork.mp4** in the Google Drive folder for an example.

<<<These next slides come from my CIS 115 lectures on Algorithms and can be adapted to fit the audience as needed. I usually don't get all the way through it, but I at least like to do the first two algorithms: insertion sort and bubble sort and the quick discussion that they take the same amount of time. Beyond that, it is really difficult to relate to younger students. >>>

4. [**Sorting Cards**] Now that we've explored how to sort using a sorting network, let's look at ways a computer will sort data.
    a. **<ACTIVITY>** This is very similar to the PB&J activity http://www.mathcs.emory.edu/~valerie/courses/fall13/170/resources/pbj-algorithm.pdf
        i. I need a volunteer that thinks he or she is very good at bossing people

around.

ii. I also need a volunteer that is really good at following directions

iii. Have the first person give the steps to shuffle a deck of cards without looking at what the other person is doing. Encourage the second person to be as "literal" as possible; aka - offer a knife to help "cut" the cards, etc. Hopefully that person will not be specific enough and they'll end up with a mess.

iv. Maybe have them try again, but this time watch what each other does.

At the end, I usually relate the story of the PB&J activity so they understand what the intent was

5. [**How Shuffle**] As you can see, there is more to this than simply giving someone the steps of the process. They need to have the right ingredients, the right tools, the right skills, and the right prior knowledge before they can proceed.

6. [**al-Khwarizmi**] The origin of the word Algorithm comes from this man, Abu Abdallah Muhammad ibn Musa al-Khwarizmi (al - khwarithmi). In the 9th century AD, he wrote many important books covering the solutions to linear and quadratic equations. His solutions took the form of a series of steps, and over time, the word "algorithm," based on his name, became the term we use to describe such a series.

7. **[al-Khwarizmi Video**] Here is a quick video from Hank Green on the Science Show about the impact of al-Khwarizmi on the world of mathematics.

   <play video; start at 0:00, stop at 0:35 when the show intro begins> *Encourage students to finish video later*

8. [**Algorithm**] In general terms, an algorithm is simply a finite list of specific instructions for carrying out a procedure or solving a problem. Let's spend some time looking at examples of algorithms.

9. [**Euclid**] ***<<<This part can be skipped if needed; depends on the age of the students>>>*** One of the oldest algorithms still used today is called Euclid's Algorithm. As you might know, Euclid was greek mathematician from around 300 BC. One of the things he discovered was a simple and easy way to calculate the greatest common divisor of two numbers. If you remember from algebra, this is needed to help reduce fractions.

10. [**Euclid's Algorithm**] His algorithm is as follows

    a. Go over slide

11. [**Euclid Example**] **<Slide has 2 positions>** Let's do an example and find the GCD of 1071 and 462.

a. Go through the example
    i. 1071, 462
    ii. 609, 462
    iii. 147, 462
    iv. 147, 315
    v. 147, 168
    vi. 147, 21
    vii. 126, 21
    viii. 105, 21
    ix. 84, 21
    x. 63, 21
    xi. 42, 21
    xii. 21, 21
    xiii. 21, 0

12. [**Euclid Example**] Here is the full process

13. [**Sorting Algorithms**] To help us really understand algorithms, let's try a couple out ourselves. We're going to look at some different sorting algorithms.

    a. <optional> First, everyone stand up and move to a different table. Try to sit at a table that doesn't have any of your other teammates. We'd like to be in groups of no more than four.
    b. Each table needs a deck of cards.
    c. First, split the deck into the 4 suits, and each person needs a single suit.
    d. Shuffle the suit so that it is random.

14. [**Sorting**] So, to begin, sort the decks of cards. As you do so, try to think of what the steps are that you follow and how you would describe that to others.

    a. Discuss: How did you do it? (Try to link back to common algorithms such as selection or insertion sort)

15. [**Insertion Sort**] First, let's implement insertion sort.

    a. Follow Slide
    b. Have students keep track of the number of times they have to ask "does this card go before this card?" starting from the beginning each time. (A crude measure of running time)
    c. Record their results on the board and average them for later
    d. As you can see, this is very similar to the way that most humans would sort something.

16. [**Bubble Sort**]

    a. *Now, lets try bubble sort.*
    b. *As you are doing bubble sort, have students count the number of times that they swap one card for another (not how many times you compare two cards).*
    c. *Have the students record their counts on the board.*

17. [**Big O Notation**] So, now that we've played with a couple of algorithms, let's talk about how we can decide which one to use. Can we tell from our data we've recorded which one is faster? **<discuss & speculate>** Obviously we need a better way to do this. The answer lies within Big O (notation)! (No, I'm not talking about the giant robot anime from the early 2000's)

18. [**Big O Notation**] Big O notation is simply a way to express the complexity of an algorithm. It approximates the number of steps needed to complete the algorithm based on the size of the input. Finally, Big O notation assumes that the input is "worst case" for that particular algorithm.

19. [**Worst case**] So, what would you say is the worst case input for Bubble Sort?

    a. **Discuss the options**

20. [**Worst case**] As you can see, the worst case input for Bubble sort is a list that is sorted in reverse order. In the case of a suit of cards, it takes 78 swaps to get it back to the sorted order.

21. [**Graph**] If we graph the number of swaps against the number of cards, we get the following graph. What kind of a function does this look like?

    a. **Discuss. Hopefully should get x^2 or quadratic or something similar**

22. [**Sorting Algorithms**] As you can see, the complexity of Insertion Sort and Bubble Sort is in Big O of n^2. So, what about some algorithms that are faster?

23. [**Merge Sort**]

    a. *Combine the deck back together and shuffle it*
    b. *Do merge sort at the table (split into smaller and smaller piles)*
    c. *After completing, try to quickly walk them through the calculation that leads to Big O of n lg n. I usually draw a tree of piles starting with 13 at the top, then count the layers of the tree (should be lg n) and then the max number of swaps (n/2) and then they layers up again (should be lg n).*

24. [**Quicksort**]

    a. *Do the same with Quicksort*
    b. *Again, try to explain the running time of Quicksort by showing the worst-case example (already sorted) and describe how it can't be calculated absolutely but in*

practice it is very fast

25. [**Sorting Algorithms**] As you hopefully can tell, these algorithms are much quicker to implement, but they are much more complex. They run in Big O of n lg n time.

<<<Sorting Activity in Scratch>>>

26. [**Sorting in Scratch**] Now that we know how bubble sort works, let's see if we can try to write it in Scratch!
    a. Have the students create a list of numbers in Scratch. At the start of the program it should clear the list. Then, instead of assigning random numbers which would change each time, have them manually add 8 - 10 numbers in an unsorted order to the list.
    b. Discuss how bubble sort works. Hopefully they should realize that the first step is to compare the first number and the second number in the list. Show them how to access those numbers and use an If block to test for that situation.
    c. Talk about what they need to do to swap those numbers. I usually use 2 different items held in my hand. They should realize that they'll need a "3rd hand" to make it work. That leads to the idea of creating a Temp variable to store that item. Show them how to do the 3 step swap operation.
    d. Once they have that, talk about the next step. It should be to compare items 2 and 3. Show them how to duplicate the block of code you've created to handle that (tell them to watch and not follow along, since you'll undo it later), and discuss how they would do the next steps (3 and 4, 4 and 5, etc.). They should remark that it is very inefficient. Talk about how to use a repeat block to repeat steps. It should do one fewer comparison than there are number of items. Help them understand by looking at their hands. There are one fewer gaps (pairs) than there are fingers.
    e. Now that we are repeating steps, we need to change the two items it is looking at each time. Discuss how they could do that. Lead them towards using a variable to keep track of the current position, and show how to include that. They will also have to update it after each repeat.
    f. Finally, discuss how bubble sort repeats all of those steps from the beginning each time until it is sorted. Add one more forever loop and a block to reset the position to 1 each time.
    g. For more information, watch the **BubbleSort.mp4** video in the Google Drive for a demonstration from 2016.
    h. There is also a solution file included, named **BubbleSortSoln.sb2** from that demo.

27. [**Martian Video**] Sorting is just one way that computer programming can help us deal with the real world. We can also use it to simulate what would happen in a particular situation. Let's check out this video from *The Martian* to see what astronaut Mark Watney is dealing with. <**watch video**>

28. [**Chemistry**] So, Mark Watney needs to figure out how to grow food on the surface of Mars. Thankfully, in the food for the mission were actual, real potatoes. Most of the food was freeze-dried, but since they would be on Mars over Thanksgiving, they sent real potatoes so they could have a little treat. So, we have access to plants that can grow. What else do we need to grow plants? <**discuss - Water, sunlight, soil, nutrients, etc.**>.

    He could make soil by mixing the dirt on Mars with actual human waste (just like we use animal waste as natural fertilizer on farms), and there was plenty of sunlight available. The one thing he was missing was water. Does anyone know what water is made of? <**discuss - Hydrogen & Oxygen**>.

    Humans have to breathe oxygen, so he had a way to get a continuous supply of that. The only thing he was missing was hydrogen. Thankfully, hydrogen is a major component of almost all fuels, including rocket fuel like Hydrazine <**see slide**>. So, all he needs to do is break down the Hydrazine into hydrogen and nitrogen, then mix it with oxygen using fire to create useable water. It was a dangerous idea (and he about blew himself up at least once), but thankfully we can simulate it in a computer program on Scratch.

29. [**Potatoes on Mars**]
    a. Direct the students to load the Potatoes on Mars scratch project: https://scratch.mit.edu/projects/112633885
    b. The students should click the “See Inside” button to see the code. If they have a Scratch account, they can also click the “Remix” button to save it to their account.
    c. For each of the major sprites, we need to simulate what it will do. There are variables created for many different things, including the hydrogen, nitrogen, oxygen and water available, as well as the number of plants grown and the number of calories available.
        i. Hydrazine - Forever change N2H4 by 1, wait some amount of time, usually 1 second to start (can be adjusted later)
        ii. O2 Reclaimer - Forever change O by 1 if it is less than a certain value, then wait some time before checking again (can be adjusted later). 18% - 22% is the optimal range for real life, so I usually have it add more oxygen if the value is less than 20.
        iii. Ionizer - Forever if N2H4 is greater than 0, reduce it by 1 and increase N by 2 and H by 4 (using the chemical formula). It doesn’t have to wait, as it can do this anytime.
        iv. Flame - Forever if H is greater than 2 and O is greater than 18 (to keep a safe value), reduce each by the appropriate amount and release 1 water into the air.
        v. Once the water is released, the plants will start growing and multiplying.

As they grow, they release more oxygen and the simulation will run faster. However, if at anytime the amount of hydrogen in the air exceeds 100, the simulation will explode, just like it would in real life if there is too much hydrogen in the atmosphere.

vi. Students will have to adjust the wait times in the Hydrazine sprite to optimize the simulation. They can also make the O2 reclaimer work faster. Encourage them to see how fast they can make the simulation run without it blowing up.

vii. See **MarsPotatoes.mp4** in Google Drive for an example from 2016.

viii. See **PotatoesonMars.sb2** for a starter example (it will still explode and needs adjustment).

30. [**Reflections**]
    a. Why do computers want to sort data?
    b. What are some ways we can sort information?
    c. Can we use computers to simulate real world situations such as a chemical reaction?
    d. What else could we simulate with computers? What have you observed computers doing in the real world?

- 112633885