ACTIVITY NAME
CS Unplugged

AGE LEVEL
8th graders, 13-14 years old

PREP TIME REQUIRED
About an 1 hour

TIME NEEDED TO COMPLETE ACTIVITY
This whole activity was completed during one block day (1 hour and 20 minutes). However, the activity is comprised of a bunch of smaller sub-activities, so sub-activities can be added and deleted to adjust to the school schedule.

MATERIALS NEEDED

Binary Magic Trick Activity
36 magnetic squares, with different colors on either side

I used blue and yellow pieces of construction paper, about 3x3 inches. I glued them together with a small magnet in between the two sheets.
Searching Activity
15 pieces of paper marked with numbers

I used pieces of construction paper with random numbers from 1-50 written on them. You can also just write numbers directly on the board. I would recommend using 15 pieces numbers to start with, but you can always add more pieces of paper for increased difficulty.

Sorting Activity Part 1
8 identical cups of varying weight

I used some extra snack cups I had and filled them with different amounts of rice. The 1st cup had about 1 tablespoon of rice and the 8th cup was completely full. I also marked the bottoms of the cups for easy reference.
Sorting Activity Part 2
A large floor pattern

I used different colors of masking tape to make the pattern about 8x15 feet.

Minimum Spanning Tree Activity Part 1
A simple spanning tree

I used the above images, but something similar can also be used. The goal is to go from a picture, to an image with the start of a spanning tree, to only a picture of a spanning tree with complete weights.
Minimum Spanning Tree Activity Part 2

A complicated spanning tree

I used the one pictured above, but any 10+ node spanning tree can be used. Make sure each path has a weight and that there are multiple paths through the tree.

DESCRIPTION OF ACTIVITY

Binary Magic Trick Activity

Have a student volunteer to set up 25 of the 36 squares in a 5x5 square pattern. Explain this is just like binary code on a computer. The pattern the student is setting up is their “code”, just like programmer’s code. They are programming in 0s and 1s (blue and yellow), but programmers will use words to code, not just numbers.

When the student is done setting up their grid, you can start to explain errors. Computers then get errors when one piece of the code is missing or changed. Flip one of the cards over to show how this can happen. Now the code that the student set up is no longer the same, so the computer won’t be able to read it the same way.

Now you will be acting as the error checker in the computer and the student will be a virus that whose intent is to mess up the code. Explain that the student’s code is a great one, but to make it more complicated you will be adding an extra row and column to make it a 6x6 grid. You will then close your eyes and the student will flip a square over (you can ask them to put a mark underneath the square to keep them honest) and you will then “guess” which square is flipped over.

The trick here is to add the extra squares specific pattern, so each row and column has an even number of 0s or 1s. In my activity I used the blue and yellow squares, so it didn’t really matter which was the 0 or 1, but when adding the extra squares I made sure there would be an even number of blue squares in each row/column. I drew an example below; the 1st row has to have a blue square added to make it even; the 2nd row already has an even number of blue squares, so I added a yellow; and the 3rd row also gets another blue square. Do this until the grid is full.
Now when the student flips the card over you’ll be able to tell which one has broken the “code” because it will be at the intersection of the odd row and column.

This activity is more of an intro activity than an educational one. It provides a simple explanation into binary code and error checking, but that it all, its a fun exercise to get the students interested in the day’s activities. I used different students as volunteers for each part and had them mess up the code and let me guess multiple times. I didn’t tell them the secret until the end of the day, and instead left it as an open question throughout the period. I left the squares on the board through the rest of the class to see if any students could figure out how I was correctly guessing the flipped over card each time.
Searching Activity

This activity is modified from the CS Unplugged activity and has been adapted to encourage participation and to better explain the core concepts.

Start with pieces of construction paper torn in half with random numbers from 1-50 written on them. Adhere these to a wall with the numbers facing the students. I chose to tape these with masking tape to the class’s whiteboard.

Ask for one student to pick a number but not to reveal it. Then ask a second student to try and guess what that number is. The first student should say higher and lower until the second student guesses the correct number. During the student’s guessing, count how many tries it takes for the second student to reach the correct answer and write it on the board. Repeat this activity another 2 or 3 times with different students so you have multiple guessing numbers.

Explain to the students that their guesses are like computations a computer would makes when running a program. We always want the fewest computations as possible, otherwise it would take a long time to get anything done on a computer. The next part depends on the class and how fast the students were able to guess the correct number. When I was doing this activity the average number of guesses the students had were in the 6-8 range, which is about double of the optimal solution. (If your students are guessing faster than this rate, you might want to increase the total numbers in the sample.)

Tell the students they can decrease the number of computations if we follow an algorithm. Do this activity one more time, but guess the number yourself, you should be able to guess the number significantly faster than the students. Explain to the students that this problem can always be solved in 4 guesses or less, they only need to complete a binary search. This is a fancy term, but is a very simple approach to solving a problem like this. The first step is to always guess the center number, with this method it doesn’t matter if the answer is higher or lower, the dataset will always be cut in half. Then keep guessing the center number and cuttin the dataset in half until the correct number is reached. With this method you only need 4 guesses to get to the correct number.

The explanation of binary search might take a while to explain to the students, it is a very abstract concept and difficult for them to grasp at this age. I would recommend showing them how the algorithm works by doing an example, then explaining the finer points of the algorithm to them, then doing another example. If there are students who still don’t understand it, volunteer them to be the “guesser” in another activity, but guide them through the algorithm step by step, so they gain a better understanding on how the algorithm works in practice.

As stated before, this activity runs the risk of the students correctly guessing the number in 2-3 tries. If this is the case, try to get a larger data set, this way the students will have a harder time correctly guessing the right number, but the minimum number of guesses needed will always be 1-log2(total). I would recommend moving up to 31 possible numbers, this doubles the dataset, but it only requires one more guess in the optimal case (so it can always be solved in 5 guesses or less).
Sorting Activity Part 1

This activity was a great demonstration to show how hard it is for a computer to sort objects in a correct order. First show the students the 8 cups, if they are filled with a substance, shake them so the students can hear the difference in the weights. Then ask for a volunteer to help sort the cups in the correct order.

When the student is explaining how to sort the cups, don't do anything to help them. your job should only be to implement their algorithm. If they use ambiguous steps, ask them to clarify and keep up all of their instructions. Like the last activity, keep count of the number of operations the students use to sort the cups (each comparison or movement is one operation). Once the students are done, flip the cups over to see if their algorithm is correct. Asked more students to volunteer to see if they can beat the previous student’s number or operations.

The object of this activity should to show the students quick sort, which is the quickest way to sort any set of objects; however, it is a fairly complicated algorithm to learn, even for first year computer science majors. The best way to show the students the algorithm is to show by example, so at every step in algorithm, you should also be sorting the cups of rice. Some of the students didn’t fully grasp the whole algorithm, which was okay, the point of the activity was to show that there could be a faster way, not to make the students memorize the exact algorithm.

To do quick sort, first pick a cup at random, then compare it to the remainder of the cups, setting the lighter cups to the left and the heavier cups to the right. Repeat these steps until all the cups are sorted. I made an example of this below, where the yellow blocks (cups) are unsorted and the blue blocks are sorted.

10 random cups

Pick 1

We now know the position of this cup

That cup was the lightest. Now we know the position of these two cups

That cup was the second lightest. Now we know the position of these three cups

Sort lighter cups to the left and heavier cups to the right

Pick another random cup, sort lighter and heavier cups to left and right again (remember you don’t have to compare all cups, only those in the section to the left or right of the known cup).

Pick another random cup, sort lighter cups to the left and heavier cups to the right

We can also tell the third lightest cup’s position, since no other cups can be lighter or heavier than it.

Repeat this algorithm on the 2nd half of the cups until all cups are sorted
This activity is fun, but takes some time to set up. It requires making a large pattern on the floor like the picture on the right. The CS Unplugged activity has this pattern taped on a large blanket, which makes prep time easier and can work just as well. I opted to tape the pattern on the classroom floor which gave the students more room to move around.

You need 6 volunteer students for this activity. Give them the pieces of paper with numbers on them (shown at the right) and tell them to stand on the bottom blue squares in random order.

Tell the students to step along the black lines to the next blue square so they are standing in that space with another student. The students should compare their numbers with one another, if they are holding a smaller number, they should move along the left black line; and if their number is larger, they should move along the right black line to the next blue square. The students should then be standing on the third row of blue boxes, each paired up with a different student. Tell them to repeat the previous step over and over until they all have moved forward to the final row of blue boxes. The students should now be in numerical order.

When the students are moving along the pattern, remind them that they cannot leave any of their fellow classmates behind, all students must move along each row of blue boxes together, or else they will have to go back and start over. Once the first group of students completes the pattern and the class understands what to do, have a new group of students complete the activity. For an added incentive, you can start timing the students to see how fast they can move through the pattern. Have 3 or 4 more groups of students complete this activity, encouraging them to try and beat the previous group’s time. For an extra challenge you can have the students hold pieces of paper with larger numbers on them, so it is harder to compare and sort themselves.

With this activity, the students are learning another way to sort elements quickly. Though this provides an algorithm that is difficult to replicate, the most interesting takeaway is multiple comparisons the students are doing when traveling through the pattern. Explain to the student that this is called parallel processing and it is how modern computers work. Ask the students if they have tried buying a computer lately and seen advertisements for dual-core or quad-core processors. The processors are similar to the lines the students were walking along. When a computer has multiple processors they can complete multiple operations at a time, just like the students were doing when three groups would compare numbers at a time and move along the black lines.
Minimum Spanning Tree Activity Part 1

This activity requires exposition and time explaining the concept to the students, so it is best placed after Sorting Activity Part 2 (when the students are able to be physically involved in the activity), since this activity requires more focus and concentration.

I began this activity with a story, I have found this is the best way to engage my students, but this lengthy introduction can be replaced with a shorter introduction, it is up to your discretion.

My story starts with a muddy village (pictured on the right). Farmers built their houses next to each other, but fenced off their entire properties. After a little bit, the farmers started talking to their neighbors and became great friends with one another. However, when they wanted to visit one another, they would have to travel through the mud, the dirt, hop each other’s fences, and travel through more mud and dirt just to get to one another’s house. They soon got tired of this plan and decided to build roads connecting everyone’s farms. However, they don’t want to spend too much money, so the roads they build have to connect everyone together, but be the shortest paths possible.

Though this story may look juvenile in the surface, it was extremely effective in engaging the students in the activity that would otherwise be boring to them, gave them real world context for this type of computer science concept, and gave them a reason to care about the outcome of the problem. If you chose not to tell the story, make sure to highlight the problem, which is connecting all the houses with the shortest path possible.

After the introduction ask the students how they would solve the problem, and if there was an easy way to represent it. Then show them the picture on the right, which has the muddy farm, but a spanning tree overlaid on top. Explain to them that this is how we can visually represent this type of problem; each blue block is a farmhouse and the grey lines connecting them are the possible paths the farmers can build on.

Then show them the spanning tree without the farming village and explain to them that this is the same problem, but without pretty picture. Then explain numbers on the lines connecting the “farms” are the number of bags of cement that are needed to build the road. In my first class I tried explaining that these numbers are called weights, but the scientific terminology was confusing to the students. I soon learned it became important to always relate the problem back to the muddy farm story; the blue blocks aren’t nodes, they are farmhouses; the numbers aren’t weights, they are bags of cement needed.

The final part of this problem is asking the students how they can connect all the farmhouses to each other with the shortest path possible. This first example is extremely easy and many students might give the correct answer on the first try. Remember to ask the students why the path they chose is the best and not a different path. Do not focus on what spanning trees are or what they represent, those concepts will be addressed in the next activity, instead focus on the reasoning behind why the students choose the paths they do.
This activity is a direct extension of the previous activity, but with a more complicated spanning tree. Showed the students the spanning tree to the right and ask them how they would go about solving it. They might become intimidated at such an overwhelming problem, which is fine. The goal of this activity is to show the students a problem that looks too big or even unsolvable, but by making them apply computational thinking and a simple algorithm, the students will see they can solve the problem with ease.

Ask the students what their strategy for solving the last spanning tree was and if they can apply the same strategy (or algorithm) in this problem. In my experience the students will settle on one of two possibilities: shortest first, where the shortest paths are selected until all nodes are connected; or longest elimination, where the heaviest paths are deleted until the minimum path is reached. Both of these strategies are just as effective in finding the minimum path, the important thing is that the students think of a strategy and follow it through.

I found it useful to display the spanning trees with a powerpoint presentation, but without with projector screen down, so the image was displayed directly onto the whiteboard. This allowed me to draw along the paths the students chose to take, so they could visually see how their spanning tree was growing or shrinking.

Despite the spanning tree’s initial daunting appearance, the students should find the shortest fairly quickly (under 10 minutes). The purpose of this last activity is multi-faceted. I first want to introduce the topic of recursion, which encapsulates the idea that just because a problem is large doesn’t mean that there isn’t a simple solution to it; a lot of problems can be solved with a few steps repeating themselves over time. The second goal of this activity is to show the students that they can solve large, complicated problems. Quite a few of my students have large misconceptions about what computer science or engineering is and a large portion of them think they aren’t smart enough to go into these fields. This problem is designed to show them that they can think like engineers, they only need to approach the problem logically and think things through. It was great seeing the student’s gain this slight confidence boost, since they were all able to solve a complicated in an extremely short amount of time.
ADDITIONAL RESOURCES
This is the link to my presentation I used to guide and present these activities.
https://docs.google.com/presentation/d/1ss_kNGn9M8HuPPCAs16dwdWRheMFX8PMZJYm4xcURSs/edit?usp=sharing

This entire activity was adapted from multiple exercises from the CS Unplugged website.

Binary Magic Trick Activity
Taken directly from: http://csunplugged.org/error-detection

Search Activity
Adapted from: http://csunplugged.org/searching-algorithms

Sorting Activity Part 1
Adapted from: http://csunplugged.org/sorting-algorithms

Sorting Activity Part 2
Taken directly from: http://csunplugged.org/sorting-networks

Minimum Spanning Tree Activity Part 1
Adapted from: http://csunplugged.org/minimal-spanning-trees

Minimum Spanning Tree Activity Part 1
An extension from Minimum Spanning Tree Activity Part 1