Complex Algorithms: Applications and Designs

By: Angelo Camargo and Grant Swajian

An algorithm is a set of instructions used to execute a specific task, which both computers and humans use. These set of instructions must follow the rules of being finite, clear, and effective. Algorithms are not only found in modern day computer labs, they have actually been around since early Babylonian civilizations in 1600 BCE for finding square roots. This early method of calculation has thus blossomed into the basis of modern day algorithms. One of the most significant modern era algorithms is the one created by Alan Turing used to crack Enigma. Codebreaking is a science that can be used in multiple scenarios, whether it be solving a math equation, decrypting a password or simply solving a problem.

Algorithms are used by almost every botball programmer. When you are sorting puff balls that vary in colour, that is an algorithm. The wait for light command is also an algorithm. The function is given the input of how much light is seen and the output is whether to start the robot or not. In the example of the differently coloured puff balls, the robot will go through a process of elimination using if and else statements. Whenever a programmer uses if, else, and elseif statements, they have created an elimination algorithm. There are different algorithm methods but one of the most common is using the process of elimination.

Bruteforce, is attempting every possible combination until success. Bruteforce is a method mainly used by code and system architecture breakers but can also be applied to daily things. This uses the input of every character and gives every possible output in different orders and lengths. The generator function can be seen in the following code (*c++*) segment with its output when there is an input of “abc123”:

```
void makeCombinations(string prefix, string chars)
{
    int len = chars.length();
    int i=0;

    if (prefix.length() < len)
    {
        for (i=0; i<len; i++)
        {
            cout << prefix << chars[i] <<endl;
            makeCombinations(prefix+chars[i], chars);
            write << chars << endl;
        }
    }
}
```

```
bbc2c2
bbc2c3
bbc21
bbc21a
bbc21b
bbc21c
bbc211
bbc212
bbc213
bbc22
bbc22a
bbc22b
bbc22c
bbc221
bbc222
bbc223
bbc23
bbc23a
bbc23b
bbc23c
bbc231
bbc232
bbc233
bbc3
bbc3a
bbc3aa
bbc3ab
bbc3ac
bbc3a1
bbc3a2
bbc3a3
bbc3b
bbc3ba
bbc3bb
bbc3bc
bbc3b1
```

Another algorithm that is probably more common in botball is the the one used for sorting. Although these two algorithms are very different, they both fall in the category of finite sequence algorithms. This is because eventually, there will be no more possible combinations and everything that you were sorting will finish sorting.

As time has progressed, so has the complexity of the functions. It started with calculations that could only be processed by a human mind to those that we can't even imagine. We are currently developing our own algorithm that has a unique function that we plan to have ready by July. Most algorithms just have a single equation that process different inputs continuously. What we are trying to do is create an algorithm that does not stick to this standard.

There are some algorithms that do not require computers. Humans use mental algorithms to figure out problems. This is just a set of continuous mental if, else, and elseif statements. An example would be a student who is trying to figure out what the problem is with his robot he uses to compete in botball. Algorithms are great for problem solving so he goes by using the mental if and else statements. If the robot is veering in one direction, the student would probably first think "**if** my robots tire is slightly off the wheel that could be the problem". But it's not that, so they would move on to "**if** my robot has more weight on one side that could be the problem".

If that is also not the problem this student would say "**else:** i'm just going to change the speed of one motor to balance it out without changing the robot's design."

There are many applications and uses for algorithms, especially in a robotics competition where problem solving is ideal. Whether it be solving a rubix cube or hacking into a computer, these methods will work almost every time. The most useful algorithm is highly debatable and could range from Quicksort to Barnes-Hut treecode to the one humans use to decode body language. Something that makes algorithms interesting is when they are combined with artificial intelligence, which ultimately creates a system that solves its own problems. Algorithms will never cease to grow and only become more efficient with time, just imagine how much better our robotics program could be if we all knew the depths of algorithms and how to apply them to our designs.

Works Cited

http://www.theguardian.com/science/2013/jul/01/how-algorithms-rule-world-nsa

http://www.theatlantic.com/technology/archive/2015/01/the-cathedral-of-computation/384300/

http://www.quora.com/Why-and-How-algorithms-are-important-in-our-daily-life