Name: ____________________________

Algorithms and Searching

Professor Bob Brown – Kennesaw State University
Bob.Brown@Kennesaw.edu

Algorithms

An *algorithm* is a set of steps that describe a method of solving a problem.

More formally, an algorithm is:

- A well-ordered sequence of
- Unambiguous and effectively computable operations that
- Produces a result, and
- Halts in a finite amount of time.

It's not the Code, it's the Algorithm

An *algorithm* is a set of steps that describe a method of solving a problem.

A *computer program* (the "code") is a concrete implementation of one or more algorithms in a specific programming language for a specific computer and operating system.

You have to have the algorithm before you can write the code!

Find the Pokémon Game

The object of the game is to find your partner's hidden Pokémon in as few guesses as possible. Pokémon have numbers, and their hiding places are indicated by letters.

You will work in pairs. One of you will have an A worksheet and one a B worksheet. They are different. **Do not let your partner see your worksheet!**

Pick one of your Pokémon, mark it on your sheet, and tell your partner the number, but ***not*** the hiding place letter. Your partner will do the same. You each make a note of the number the other gave you.

Now, take turns trying to guess the hiding place (letter) of your partner's Pokémon. Keep track of the number of guesses, and keep going until each of you find the position of the other's Pokémon.

We will play the game two times with up to three sets of worksheets. The second time, the numbers are in order. There is a shortcut to finding the position of the Pokémon when the numbers are in order. Can you work out what the shortcut is?

A Little About Searching

If we don't know anything about the order of the items to be searched, the best search for a computer program is a ***linear search***. We "go down the line," starting with the first item, checking each one, until we find the target or reach the end of the list. If we reach the end, the item we're looking for isn't in the list. On average, a linear search required checking about half the items. With a linear search, doubling the number of items doubles the search time.

A ***binary search*** only works if the items are in order. We start by checking the *middle* item in the list. If the middle item is the one we're looking for, we're done! If the middle item is less than what we're looking for, then the target item can only be in the second half of the list. If the middle item is greater than the target, the target must be in the first half of the list. Either way, we've cut the number if items in half. We perform the same procedure, starting in the middle, in whichever half of the list our target must be. Repeat that until the target is found or we know it's not in the list of items. For a list of n items, the maximum number of checks is $\log_2(n)$. That means the smallest power of two greater than or equal to the number of items in our list. For a list of 26 items, like the Pokémon game, we need at most five checks because 2^5 is 32, and 32 is the smallest power of two greater than or equal to 26, and 32 is two to the fifth power.

There are other searching algorithms.

If a list must be searched many times, it may pay to do some pre-work, such as sorting the list, so that a faster algorithm can be used.

Choosing the Right Algorithm

As we saw with the Pokémon game, choosing the right algorithm depends a lot on the data we have. We also need to consider the circumstances. If we're going to search the same list many times, it might pay to spend the effort to sort the list so that we can use a binary search.

There are other algorithms for searching, and other tasks, such as sorting, have several different algorithms. Choosing the right algorithms is crucial to writing effective programs. No matter how good your code is, a bad algorithm will make it look bad!

Scooping the Loop Snooper

You can find Professor Pullum's poem here: http://www.lel.ed.ac.uk/~gpullum/loopsnoop.html