Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 11 **Spring 2019, CU-Boulder**

A startup has hired you as the chief technology officer (i.e., the only one who knows how to program). After the founders (all MBAs) finish explaining their vision for changing the world, you realize that what they describe can be reduced to the traveling salesman problem. No worries, you develop a solution that is a 1.5 approximation. The founders are devastated that they cannot use the word "optimal" in their next VC pitch, and wonder out loud if they need to get a new CTO who can do better. Convince them that an efficient optimal solution is unlikely (i.e., P probably does not equal NP) and that your solution is quite good.

Dear Founders,

First off, I want to clear the ground by saying that I can see why you are concerned. From your point of view, it kind of sounds like I'm not good at my job. But before you make any rash decisions, can I defend myself for a little bit? Because, believe it or not, to get a 1.5 approximation is actually rather good!

To start my defense, I want to explain some things about the runtimes of algorithms. Each algorithm can be categorized into a specific 'class' based on how fast the run, in proportion to the input. These classes consist of liner, or polynomial, or exponential, among others. But that's getting ahead of ourselves.

Let's take, for example, sorting a list of numbers from higher to lower has two different variables. One variable is the size of the input (ie, how many numbers you are sorting), and the other variable is the approach that your algorithm takes to sort those numbers. So when I say that an algorithm is has a polynomial runtime, it means that for an input of size 'n' the algorithm takes 'X * n' time to complete. Well one of the worst runtimes we know about is the class of algorithms that take exponential time. It's like exponential growth in business, but in this case, 'growth' is the length of our runtime, and is therefore bad. To continue the example, if your sorting algorithm took exponential time, that would mean that to sort a thousand or so numbers would take millions of clock ticks. Because these exponential runtimes are so damaging, they teach us a lot about how to avoid these in school. In fact, they even give us a list of the most commonly used exponential algorithms.

So why does this matter? Well the algorithm that you all were coming up with is essentially one of those algorithms they warned us about. It's called the Traveling Salesman Problem. You can look it up if you want, but basically, it takes exponential amounts of time to complete. Meaning that the more input you give it, the runtime increases exponentially. Which brings me to why my solution is so good. I created an algorithm that is in the polynomial class! This means that the runtime of my algorithm only grows by a polynomial amount, or a constant rate. So as our input increases into the thousands, our runtime stays in the thousands.

Now like I said, I understand that not being able to use word optimal sounds like a bad thing, there are much worse cases. I know that a 1.5 approximation is worse than a one-to-one ratio, but those are nearly impossible to come by. Especially for the algorithm that you were describing.

Please, if you have any questions, call me. Or write me back. But I can promise you that getting a 1.5 approximation in polynomial time, is a lot better than any exponential time algorithm that most would come up with!

Your CTO
Trevor

Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 11 **Spring 2019, CU-Boulder**