Participation Portfolio Submission 2

## Example 1 (Receiver for second SRE)

* **What?**

I was a receiver in the second SRE. I think my partner was really good at synthesizing and going from abstract to concrete. This let me first understand the general idea of what we were doing (like what are the cases we should deal with), and from there we went on to the specifics of the algorithm and the proof-writing.

* **So what?**

Once we got to the proof however, I became quite confused about what we had to prove and in what order. She was explaining different key areas of the proof but I didn’t understand how all the fitted together. We overcame this by stoping and talking first about the general structure the proof should take in this case: Algorithm, Runtime Proof.

* **Now what?**

I think one improvement we could make in the future is talking about the general expectations of the exercise first, and then start talking about the algorithm/math content itself. Having structure particularly helps me understand what we are talking about and have a better sense of whether the explanation is going.

## Example 2 (Sender for third SRE)

* **What?**

I was a sender for the third SRE. We first went on the big level thinking of what we were doing with the BFS-CC algorithm, and then the specifics of the BFS-Label algorithm. As we were on the same page from BFS-CC with what we wanted to do within the label subroutine, it was relatively simple to write out.

* **So what?**

My partner did really well, and he indeed improved the my understanding of the concepts. He continuously question the runtime of the algorithm, which made me argue about the runtime in an intuitive fashion, so that it could be explained in the air without math. That is how I came to the conclusion that the runtime is O(n + m + n) = O(n + m) because you are checking every node once, and then an instance of BFS runs through a single node only once.

* **Now what?**

I had two important learnings that day:

* I think that when you are trying to get a big picture understanding of how many algorithms are working together, it is easier to start by the simplest one first, so that one can quickly begin to understand parts of the problem.
* Even if you initially are the one explaining a topic, that does not mean that you are not learning or improving your understanding through it. By the end I had intuition for the runtime that I did not have originally

## Example 3 (Helped a friend visualize the magic maze board game)

* **What?**

In PSet 4, a friend was having trouble understanding how to write a reduction that would transform the knowledge that we had developed in problems 3a and 3b into a reduction for 3c that was able to solve the MagicMazeProblem. I tried different alternative ways of explaining until I found one that struck the match.

* **So what?**

I explained to him that we could represent each player’s move though a different graph, that had edge according to what the rules of the game stated. One could only move in straight lines, one could only move in diagonals, and one could only move 2 steps in one axis and 1 step in the other. I then drew a simple version of each of the three graphs and layered them one on top of each other, with paths going down from one layer to the next.

* **Now what?**

I wanted to give him all the tools to come up with his own solution without knowing the final answer, because that is when one learns most. After going through a couple different explanations, it was seeing the movements of the players and the graph drawn out on paper the thing that made it all click for him. I learned from this that visualizations are way more quickly understood than prose or pseudocode explanations, and they allow for a deeper understanding. After seeing the graphs he was able to come up with the full reduction on his own.