Three programming team members from the University of Central Florida traveled across the world to compete in the ACM International Collegiate Programming Contest World Finals, ultimately earning the bronze medal. Team member Timothy Buzzelli reflects on his World Finals journey and the training and guidance that led to victory.

Timothy Buzzelli, University of Central Florida

The long flight home from Beijing was the perfect time to slow down and reflect on what we as a team had just accomplished and what it took to get there. My two teammates, Eric Ly and Alexander Coleman, and I traveled over 7,500 miles to compete among 140 of the best competitive programming teams from universities all over the world at the ACM International Collegiate Programming Contest (ICPC) World Finals. Defying expectations, we placed 10th in the world out of over 16,000 teams, earning a bronze medal and establishing the University of Central Florida (UCF) as the top competitive programming team in the US for two years in a row and now as number one in North America.

CONTEST BACKGROUND
ACM has sponsored the International Collegiate Programming
Contest (ICPC) for more than 40 years, though the contest only became truly international in the 1990s. The contest is two-tiered, where teams compete regionally in the fall semester, and the top teams from the regional contests go on to the World Finals in the following spring semester.

ACM ICPC contests present teams of 3 with a problem set consisting of 10–13 programming problems covering a wide variety of areas such as number theory, computational geometry, and dynamic programming. In the early years, there was more emphasis on software engineering, but over the course of the last 15 years, the contest has focused more heavily on problem solving via algorithms and data structures. Each 3-person team uses one computer to solve as many problems as possible in the 5-hour time period. Students may use a variety of programming languages, but nearly every team that qualifies for the World Finals level of competition solves the problems posed in either C++ or Java. Students are allowed to use many of the pre-written libraries in both languages. Solutions to problems are judged by data based on test cases made in advance by the judges. For a solution to be judged as correct, it has to correctly solve every test case (typically, the judges create anywhere from 20 to 150 test cases per problem), each within the time limit specified. Even if only one test case is solved incorrectly or takes a second too long, teams get no credit for the solution. Thus, heuristic solutions or inefficient solutions are discouraged, even if they are sufficient to solve an average test case.

**2018 WORLD FINALS**

This year’s World Finals included teams from more than 3,000 universities that competed in the many regional competitions. The top scoring team in the finals, Moscow State University, solved 9 of the 11 problems in the 5-hour contest. Along with three other teams, they earned gold medals in the competition. Four teams earned the silver medal, and the bronze medal was awarded to 5 teams, including UCF. This year, UCF placed narrowly behind St. Petersburg ITMO University, which had placed first in 4 out of the past 7 World Finals.

**Example Questions**

In one problem, “Catch a Plane,” the contestant is attempting to take a series of buses to get to the airport. The input data involved schedules for up to one million buses and up to one million bus stations. Each bus connects two stations and has some probability of running at all. The goal of the problem is to determine the probability of reaching the airport on time, assuming you use the optimal strategy.

A naive brute-force solution to Catch a Plane would run in exponential time. A better, straight-forward, dynamic programming solution that attempts to build the probability of reaching each station from every other station would run in \(O(n^2)\) time, where \(n\) is the number of stations. Since \(n\) could be as large as one million, this strategy takes too long on the larger input cases. The key observation for solving the problem is to realize that the vast majority of pairs of stations in the larger cases aren’t connected, and if the data is processed backward in time with the appropriate information stored, the run time can be brought to be close to linear to the number of buses. A sort is required on all the bus times, which adds a multiplicative log factor to the run time. In addition, the times, which are integers that range up to \(10^{18}\), have to be coordinate-compressed to more easily process the dynamic programming (DP).

Another problem, “Panda Preserve,” gives the description of a polygonal enclosure (of up to 2,000 line segments) and a list of locations where wireless transmitters will be placed. The goal of the problem is to determine the minimal radius of transmission for each transmitter (each must be set to the same distance) such that every point within the polygon is within the radius of at least one of the wireless transmitters. An efficient solution that runs in \(O(n)\) time, where \(n\) is the number of vertices, can be obtained by first sorting the vertices of the polygon and then running the algorithm for a simpler problem, such as the one described in the problem statement. The key observation for solving the problem is to realize that the vast majority of pairs of stations in the larger cases aren’t connected, and if the data is processed backward in time with the appropriate information stored, the run time can be brought to be close to linear to the number of buses. A sort is required on all the bus times, which adds a multiplicative log factor to the run time. In addition, the times, which are integers that range up to \(10^{18}\), have to be coordinate-compressed to more easily process the dynamic programming (DP).

This year’s World Finals included teams from more than 3,000 universities that competed in the many regional competitions.
grid is given where different square buildings are adjacent to each other, in both rows and columns, with no streets in between. Each building has a different height. Robin, a superhero, starts at the center of the roof of one building and can leap with some velocity. The problem is as follows: assuming ideal physics (no wind resistance, etc.) and the usual force of gravity, determine the fewest number of leaps it will take Robin to get from his starting point to the center of all the other buildings. Robin only attempts to leap precisely to the center of other buildings and can not make a jump unless he clears all building edges he is supposed to leap over to get to his destination. If Robin ever jumps precisely over the intersection of four buildings, he must clear the tallest of them. The key to solve the problem is to use mathematics to determine the perfect angle for Robin to jump between each pair of buildings so that he lands precisely at the center of his target. Typically there are two possible angles that are valid solutions. Of these, Robin always wants the higher jump as it gives him a better chance to clear buildings that might be in his way. Once this angle is determined, the altitude Robin reaches as he crosses each possible building right when they get in his way must be calculated. If any are as tall as his current altitude at that point in time, then the jump can't be made. Once all of the information is calculated to determine which pairs of buildings Robin can and cannot jump between in a single jump, a breadth-first search allows us to calculate the fewest number of jumps necessary to reach every other building from Robin's starting point. Our team only had about 40 minutes to solve this problem after submitting a correct solution to our seventh problem. During the time we had, we wrote the code for the breadth-first search, but didn’t have time to fix the physics calculations. If any are as tall as his current altitude at that point in time, then the jump can't be made. Once all of the information is calculated to determine which pairs of buildings Robin can and cannot jump between in a single jump, a breadth-first search allows us to calculate the fewest number of jumps necessary to reach every other building from Robin's starting point. Our team only had about 40 minutes to solve this problem after submitting a correct solution to our seventh problem. During the time we had, we wrote the code for the breadth-first search, but didn’t have time to fix the physics calculations. With more time, we believe that we would have been able to solve this problem correctly.

Many of these problems are beyond what is typically seen in an undergraduate classroom. What separates these problems from typical homework questions is that many of the problems require some sort of insight/proof to determine an efficient solution that solves all cases. In addition, students had to correctly deal with many thorny implementation details in an extremely short amount of time without access to any comprehensive test cases. Teams had to create their own test cases to discover where their own bugs might be. The contest provided some sample input cases, but these tended to be fairly easy and didn’t test the limits of the problem being posed. The contest in general makes students better at creating their own test cases, better at breaking down complicated problems, and better at debugging — all critical skills for software engineers.

Although our performance as three amazing competitive programmers working together allowed us to solve the 7 problems, which gave us our 10th place finish, it was the years of coaching and constant practice that truly won us the medal.

**COACHES’ GUIDANCE**

This year our team had two primary coaches assigned to us, Arup Guha and Shahidul Islam. Guha, an associate instructor of computer science at UCF, has been my coach throughout my time on the programming team. He has guided my practice over the years and helped me evolve into the competitor I am today. Arup took on a much larger coaching role than in years past, and he gave us our new motto: “Discipline and properly directed effort maximize talent.” Arup encouraged the three of us to work hard in our final sprint to the finish as we prepared for the contest that was, in many ways, a culmination of our combined efforts at UCF.

Shahidul “Sumon” Islam, a PhD candidate in computer science at UCF, took the specific role of being our strategy coach. He regularly recorded us during practice and kept detailed notes on our performance. He kept us informed about the other teams we would be competing against in Beijing. Sumon met with Alex, Eric, and I individually and as a team nearly every week to formulate a team strategy that would maximize our chance of earning a medal.

I wholeheartedly believe that without Arup’s and Sumon’s guidance throughout the year, and without...
all of the work put in from the other coaches, we would not have done as well as we did. It was the amazing coaching staff at UCF that directed our efforts properly and showed us what we could accomplish that enabled us to make UCF proud in Beijing.

THE DREAM OF MEDALING
Medaling at World Finals has been the dream of the UCF programming team since the competition went global. Programming team members dedicate nearly all of their time to the team, practicing a minimum of 20 hours a week in addition to a 5-hour practice session every Saturday. With the careful but tough guidance of our coaches, we trained hard enough to succeed at the Southeast Regionals competition, which was just the first step to achieving our ultimate goal of representing UCF and competing in the World Finals.

JOIN US!
Although many universities compete in the ACM-ICPC, there are many excellent universities that don’t. Schools that wish to participate should contact the contest’s regional director for information about competing in their region. Each of the regions within North America can be found here at the ACM-ICPC website: https://icpc.baylor.edu/regionals/finder/northamerica-2018. With so many websites that provide problems and automatically check solutions, it’s relatively easy to practice these types of problems without too much overhead for coaches. By participating, students can gain the camaraderie of others who love algorithms and competition; a leg up on both their classes and the job interview process; the opportunity to represent their university; and most importantly, the opportunity to improve in problem-solving, implementation, testing, and debugging.

ACKNOWLEDGMENTS
We would like to thank UCF for providing the funds for us and all of the coaches to travel to the contest.

TIMOTHY BUZZELLI is a computer science undergraduate student at the University of Central Florida. Contact him at timothybuzzelli@knights.ucf.edu.