- **Detailed** instructions on how we can run your code. We should be able to reproduce your results. If you use different algorithms on different files, tell us. If you hand-solved some files, tell us. If you use a random number generator, tell us how you seed it. Tell us all the details to be able to reproduce your results. Make sure to explain what libraries you used and how we should set them up.

Commmand : python3 combined_solver_2.py --all inputs outputs
There are 3 random algorithms. In line 712, 723, 733. Range means the repetition time. You can set that as 100 each. Also, we wanted try many greedy algorithms. So we set range of randoms to 0 and run it.

Do it manually : 561 562 563(Ours)
(others) 195 2 708 710 446 218 272 271 270 216 591 68

- Your approach -- what worked well, what didn't work well. Did you try to reduce the problem to some other problem? What libraries worked well? Did you find some inputs to be particularly troublesome?

We changed our major algorithm from ILP to greedy and Christofide.
At first, we made greedy algorithm from phase 1. It has current points. In each current points, it put the name of cities and the sum of cost to conquer and to go back current city. And we choose to go minimum cost cites. --Greedy Explanation

Since there are a lot of inputs that fakes greedy, we got around 250 points. So we redesign our algorithm to mix more than two algorithm.
It is Christofide and Bellman Ford. We used library to make this algorithm. We run this two algorithm in one file and choose the better one. Even after that we found some inputs that has terrible outcome. We sort the input by size of the Kingdoms. Then If it is smaller than 10, and our algorithm can't solve, we calculate manually. Some of inputs had a kind of "unicode" problem, so we fixed it manually.
At this time, we add every path that we searched in Christofide, so we fixed the bug. We wanted to improve our algorithm, so we put randomness for Greedy and Christofide. It returned better outputs but it took too many time to run. We set up randomness to 10 -> 30 -> 100.
After that we download the Rankings.csv and find the worst ranking input. Then rerun these inputs by increasing random range in last day.

- How work was distributed among your teammates.

Won Jun Son made our major solver algorithm, greedy and its random part. He almost designed reduction project problem to ILP algorithm, but we decided not to make it. He wrote Bellman Ford part from Christofide. He wrote the cost comparator to choose better outputs from two

algorithms. He fixed the error made from greedy algorithm and its random parts. He wrote phase 2 document.
Andrew Chen made our another solver major algorithm, Christofide and MST and its random part. He wrote output generator to make random output for phase 1. He wrote the document for phase 1. He made a github to submit. He wrote the program to write our inputs. He wrote the files to combine the algorithms.
Hong Seok Jang calculate the cost manually for our inputs and small size inputs. He wrote input generator to make random input to test our algorithm. He made output Cost calculator and find bug in greedy. He analyzed the ranking.csv to find the outputs files that we got terrible score. He designed our inputs which fakes greedy. He managed the files to focus on bad output. He wrote phase 2 document.