**PTA Report:**

1. Abstract
2. Introduction
   1. Hoisting Rules
3. Algorithm of the analysis
   1. Mention the Jalangi call backs we have used
   2. Pseudocode
4. Handling of corner cases with example and a single output screen shot for reference
   * 1. Function expression
     2. Function declaration
     3. Eval (direct and indirect)
     4. Recursive function
5. Limitation of Jalangi
6. Evaluation
   1. First we will run the test suits of each node.js library (lodash, q, underscore) on the original version of their repositories and note down the execution time.
   2. Then we will run our analysis on the same repositories which will mark all the hoistable and non-hoistable functions.
   3. We will then do manual hoisting as per the results of our analysis.
   4. And we will run the test suits again of each library but this time on the updated hoisted version of the repositories and check if there is any different in the execution time.
   5. Lodash
   6. Q
   7. (third library)

**PTA Presentation:**

1. What is hoistability?
2. Motivation behind Hoisting Nested Function?
3. List of conditions that need to be checked to determine hoistability?
4. List of Jalangi call backs we used
5. Algorithm (block diagram with each block listing the jalangi call back)
6. Handling of corner cases:
   1. Function Expression
   2. Function declaration
   3. Eval()
   4. Recursive
7. Evaluation Results (IF)

|  |  |  |
| --- | --- | --- |
| Library | Execution Time with Hoisted functions | Execution Time with non- Hoisted functions |
|  |  |  |

1. Limitations and Future work (e.g. we are not telling the Level at which the function can be hoisted, nested functions defined with in an expression can be tested as well, jalangi2 is not able to analyze nested functions in an indirect eval)
2. Conclusion