GSoC 2024 Proposal

Open Science Labs: PyDataStructs: Add C++ Backend for all trees and port stack.py to C++ Backend

Organization: NumFOCUS
Sub-Organization: Open Science Labs
Project Partner: PyDataStructs

Kishan Ved

March 29, 2024

Contents

1 About Me

1.1 Personal Information

- **Name** - Kishan Ved
- **University** - Indian Institute of Technology, Gandhinagar
- **Program** - Bachelor of Technology (B.Tech.)
- **Major** - Computer Science and Engineering

- **Year** - Second Year Undergraduate
- **Expected Graduation Date** - August, 2026
- **Timezone** - IST (UTC + 5:30)
- **Email ID (University and GitHub)** - kishan.ved@iitgn.ac.in
- **Email ID (Personal, used for GSoC registration)** - kishanved123456@gmail.com
- **Contact Number** - +91 9619319866
- **GitHub** - Kishan-Ved
- **Resume** - Resume
- **Website** - kishanved.tech

1.2 Personal Background

I am a sophomore at IIT Gandhinagar, in the department of Computer Science and Engineering. I've always been fascinated by science and technology and I'm keen to implement solutions that benefit the society. I secured the **All India Rank 1348 in the JEE** Examination, to enter into my prestigious university, **IIT Gandhinagar**. I have been selected as a **Reliance Foundation Undergraduate Scholar**, which is offered to the top 5000 students among thousands of applicants. Please find my resume here. I would like to highlight that **I am among the top 5 performers of my batch (by CPI)**, with a CPI of 9.75/10. I am well versed with the languages: C++ and Python, and I actively take part in competitive programming contests on Codeforces. I have a **rating of 1350+ on Codeforces**, here is my handle. More information about me is available on my website. For communication, I prefer English.

1.3 Programming Background

I use Ubuntu 22.04.4 LTS as my operating system and Visual Studio Code as my editor. I began contributing to open source software by making changes to the documentation of **scikit-learn**. Here is the PR. I have also **contributed to PyDataStructs**, where I implemented a function: **Introsort**, in the

Python backend, which has been merged. I have done several projects using Python, C++ and many other popular languages. I **designed the website of Amalthea**, the technical summit of my college, IIT Gandhinagar. During my winter vacation, I made a resume generator, which can write LaTeX code for generating a professional resume, just by filling out a form. This project gained popularity in my college, and is currently being used by many. This encouraged me to open source the code, making this my first **(self owned) open source project: IIT Gandhinagar's Resume Generator**. I am interested in machine learning, and I've made a few projects, in the form of jupyter notebooks. I occasionally write machine learning blogs on my blog page. A complete list of my project can be found here. A few of my projects, particularly in Python and C++, are:

- **Implemented Introsort in PyDataStructs** - In this open source contribution, I implemented the function: Introsort, in the Python backend of PyDataStructs. All the parameters of this function have been taken into consideration and are carefully and correctly implemented. The code was properly documented (via comments), which give a detailed description of the function, the parameters, expected output and references. This **PR #549** has been successfully merged and this lead to the **Issue #545** being solved and closed.

- **Spacecraft Charge Distribution Modelling** - In this project, I worked with Dr. Soumyabrata Chakrabarty, a scientist at the Indian Space Research Organization (ISRO), who is also a visiting professor at IIT Gandhinagar. This project involved using the method of moments to derive equations for finding the charge density on a spacecraft's surface and employing numerical methods such as Gaussian elimination, Gauss-Jordan algorithm, and TDMA to solve this system of equations. Programming of all mathematical equations and visualizations are done using Python and it's libraries. Here is the link to the report and the poster.

- **Intelligent Game Engine** - This project involves making games where the user plays against an 'intelligent' algorithm, that doesn't allow the user to win, or delays it. This makes the game play by the computer logical. This has been achieved by constructing a graph of the game, and then performing a DFS or BFS search on all possibilities to find the best move. This project was focused on algorithmic optimization

and writing clean and reusable code. The repository contains games like The Game of Sim, Connect4, Up it Up game, a 2*2*2 Rubik's cube solver and a Sudoku solver. All the code has been written using C and C++. The GitHub repository can be found here.

- **Human Activity Recognition using ML** - In this project, I made a machine learning model that uses only decision trees to recognize 6 different human activities by utilizing time series data of acceleration involved. Dimensionality reduction and hyperparameter tuning have been employed. This project is done in Python. The GitHub repository can be found here.

- **Extracting communities from the Facebook Graph** - In this project, I have implemented a greedy algorithm to extract the densest sub-graph from Facebook's graph (with pages being nodes and mutual links being edges). Here, I learned to employ data clustering techniques on large real-world datasets, involving logic and algorithmic optimization in terms of both space and time complexity. This project is developed using Python and its relevant libraries like NumPy and Pandas. The GitHub repository can be found here.

Details on the above and more projects of mine are available on my GitHub profile and on my website.

1.4 Previous Contributions to Open Source

1.4.1 PyDataStructs

Merged PR
#549: Implemented Introsort in Python backend

Closed Issue
PyDataStructs Issue #545

In this open source contribution, I implemented the function: Introsort, in the Python backend of PyDataStructs. All the parameters of this function have been taken into consideration and are carefully and correctly implemented. The code was properly documented (via comments), which give a detailed description of the function, the parameters, expected output and references.

1.4.2 Other Open Source Contributions

I have submitted more than **10 merged PRs to the LFortran repository** and more than **3 merged PRs to the LPython repository**. Please find my contributions here:

- LFortran contributions link
- LPython contributions link

1.5 Why me?

In this section, I have listed down some strong points that I believe make me the best candidate for this GSoC project:

- I possess **experience in contributing to PyDataStructs codebase**, affording me a deep understanding of their intricacies.
- I am **experienced as an open source contributor**, I have been contributing to open source organizations like LPython and LFortran for about 3 months now.
- I possess **good knowledge about tree and graph algorithms**, which is necessary for this project. I regularly participate in competitive programming contests and I have a **Codeforces rating of 1350+**.
- I have established a **strong communication based relationship with the current team members** who are set to mentor me during the GSoC program.
- I am **well versed with Python, C++, Git and GitHub**. I use Ubuntu 22.04.4 LTS and VS code as my code editor.
- My summer vacation aligns perfectly with the GSoC timeline, guaranteeing my **full availability for GSoC**, without any conflicting professional commitments. This ensures dedicated work.
- I have a very **high level of motivation** for GSoC, and I am **consistent** in my work, this is reflected by my academic results.
- I am enthusiastic about the success of PyDataStructs and I am **willing to continue contributing post GSoC** too.

1.6 My Availability

I ensure full availability during the months of May, June and July. I have my summer vacation during this period. This perfectly aligns with GSoC's timeline. I plan to do a project of 350 hours, and I will start coding from the community bonding period itself. I will start coding from 2^{nd} May, as my examination ends on 1^{st} May. I will work for 20 hours a week in May, 40 hours a week in June and 20 hours a week in July. In August, I will complete the remaining time, which is around 10 hours a week. This is because my college resumes in August. The detailed timeline is in Section 3 ahead.

2 The Project

Sub-Org name: Open Science Labs
Partner Organization: PyDataStructs
Duration: 350 hours

I am interested in the project: Add C++ Backend for all trees and port stack.py to C++ Backend. I wish to do a project of 350 hours. The detailed timeline, along with division of work (by hours) is described in the next section.

Project Abstract:
There are 2 broad goals of my project:

- The first is to add C++ backend for all trees present in binary_trees.py file, which currently supports Python backend for the following trees:
  - AVL Tree
  - Binary Tree
  - Binary Search Tree
  - Binary Indexed Tree
  - Cartesian Tree
  - Treap
  - Splay Tree
  - Red Black Tree

I will implement different types of tree traversals, like:

  - Pre Order traversal
  - Post Order traversal
  - In Order traversal
  - Out Order traversal
  - Depth First Search (DFS)
  - Breadth First Search (BFS)

- The second goal is to port stack.py to the C++ backend. Currently, the Python backend supports the following classes:

  - Stack
  - Array Stack
  - Linked List Stack

I will add exhaustive tests for all my implementations. Adding C++ backend for these will make the algorithms run faster. I will ensure that the implementations are efficient in terms of time and space complexity. This will make PyDataStructs extremely useful.

These have been described in detail below:

2.1 Adding C++ backend for Binary Trees

The current version of PyDataStructs supports a Python backend for all trees in binary_trees.py file. My project involves implementing a C++ backend for all these trees. The implementations will be optimized in terms of time and space complexity. I will add exhaustive tests for all my implementations. The details of individual trees are described below:

2.1.1 Binary Trees:

The following functions will be supported:

- Creating a new binary tree
- insert

- delete
- search
- Printing functions

2.1.2 Binary Search Trees:

The following functions will be supported:

- insert
- delete
- search
- rank
- lowest common ancestor
- lower bound
- upper bound

2.1.3 Self Balancing Binary Trees:

The following functions will be supported:

- rotate left
- rotate right
- rotate left right
- rotate right left

2.1.4 Cartesian Trees:

The following functions will be supported:

- bubble up
- trickle down

- delete
- insert
- search

2.1.5 Treap:

The following functions will be supported:

- delete
- insert
- search

2.1.6 AVL Trees:

The following functions will be supported:

- delete
- insert
- search
- balance insert
- balance delete
- rotate left
- rotate right
- rotate left right
- rotate right left

2.1.7 Splay Trees:

The following functions will be supported:

- delete
- insert
- search
- join
- split
- zig zag

2.1.8 Red Black Trees:

The following functions will be supported:

- delete
- insert
- search
- get parent
- is leaf
- has red child
- other functions specific to Red Black Trees

2.1.9 Binary Indexed Trees:

The following functions will be supported:

- get prefix sum
- get sum

2.1.10 Binary Tree Traversals:

Different forms of traversals will be implemented, these include:

- Pre Order traversal
- In Order traversal
- Post Order traversal
- Out Order traversal
- Depth First Search (DFS)
- Breadth First Search (BFS)

2.2 Porting Stack.py to C++ Backend

This part of the project involves implementing stacks in the C++ backend. I will also add exhaustive tests for my implementations. There are three classes of stacks in PyDataStructs, their details are:

2.2.1 Stack:

The following functions will be implemented:

- Creating a new stack
- push
- pop
- is empty
- peek

2.2.2 Array Stack:

The following functions will be implemented:

- Creating a new stack
- push

- pop
- is empty
- peek
- length/size
- printing functions

2.2.3 Linked List Stack:

As the name suggested, linked list data structure is used in this implementation of stacks. The following functions will be implemented:

- Creating a new stack
- push
- pop
- is empty
- peek
- length/size
- printing functions

The detailed timeline of my project is described in the section below.

3 Timeline

I would like to do a project of 350 hours. I will start working from the community bonding period itself, which is in May. I will work for 20 hours a week in May, 40 hours a week in June and 20 hours a week in July. In August, I will complete the remaining time, which is around 10 hours a week. This is because my college resumes after summer break in August.

Plan for Feedback and Code Review: I have planned out my project as a set of multiple weekly sub goals, each having it's own PR. This will make it

easier to review my code. As I have weekly PRs, I will use the last 1-2 days of the week to polish my implementations based on the feedback received from mentors.

Here is the tentative plan for my project:

3.1 Pre-GSoC Period

This period includes the time before the beginning of community bonding phase i.e., 1^{st} May, 2024. I will address existing issues, and solve them with excellent PRs. I will spend time reading about the trees that I plan to implement in the C++ backend, specifically AVL trees and Red Black Trees.

3.2 GSoC Period

According to the official dates, the complete program can be divided into three parts, which include: **Community Bonding Period**, **Phase - 1** and **Phase - 2**. The details of each of these is discussed in the following subsections.

3.2.1 Community Bonding Period

This part of the program starts from 1^{st} May, 2024 and ends on 26^{th} May, 2024 consisting of nearly 4 weeks. I plan to start coding from 2^{nd} May itself, as my examination ends on 1^{st} May. During this period, I will work for 20 hours a week. By the end of the community bonding period, I will have completed 80 hours of work.
The weekly plan is as follows:

- In the first week, I will work on defining a C++ backend for my implementations. I will start working on the Binary Trees class, add support for different parameters and implement the functions that it contains.
- In the second week, I will work on Binary Search trees. Once done, I will start implementing Self Balancing Binary Trees.
- In the third week, I will finish Self Balancing Binary Trees and then implement Cartesian trees and all the functions that it supports.

- In the fourth week, I will implement Treap. In this week, I will also take out time for getting my code reviewed. I will address any changes needed or add more functionality if suggested by my mentors.

By the end of the community bonding period, I will have successfully implemented C++ backend for Binary Trees, Binary Search Trees, Self Balancing Binary Trees, Cartesian Trees and Treap, with all the PRs reviewed and the suggestions addressed.

3.2.2 Phase 1

This phase starts from 27^{th} May, 2024 and ends on 12^{th} July, 2024, consisting of around 6 weeks. I plan to work for 40 hours a week in June (ie; the first 5 weeks of this phase) and 20 hours a week in July. Hence, in phase 1, I will work for 240 hours.
The weekly plan for phase 1 is as follows:

- In the first week, I will implement AVL Trees, with all the functions and parameters handled properly.
- In the second week, I will add C++ backend for Splay Trees.
- In the third week, I will work on implementing Red Black Trees.
- In the fourth week, I will implement Binary Indexed Trees.
- In the fifth and the sixth week, I will implement Binary Tree traversals, like BFS, DFS, preorder, postorder and inorder. In these 2 weeks, I will also address reviews and suggestions provided by my mentors.

Hence, by the end of Phase 1, I will successfully implement a C++ backend for all trees present in binary_trees.py file. I will ensure correctness of my implementations by adding exhaustive tests. This will make the algorithms run very fast and consequently, PyDataStructs will become more useful and efficient.

3.2.3 Phase 2

This would be the final phase of the program starting from 12^{th} July, 2024 and ending on 26^{th} August, 2024 consisting of around 6 weeks. I will work for 20 hours a week in July and complete the remaining project in August (by working for around 10 hours a week). This is because my college resumes in August.
The weekly plan for this duration is as follows:

- In the first week, I will work on correctly defining a C++ backend for Stacks and implement the Stack class, with all the member methods properly defined.
- In the second week, I will implement the Array Stack class.
- In the third and the fourth week, I will add C++ backend for the Linked List Stack class.
- In the last 3 weeks, I will address reviews and add more functionality as suggested by my mentors. I will also spend this time in documenting my implementations and adding tests. These weeks might also be used as buffer time for any hurdles that might arise or for any missing feature that I need to implement.

By the end of Phase 2, I will have ported Stack.py file to the C++ backend. This will make the Stack implementation of PyDataStructs fast and efficient. This will complete my GSoC project.

3.3 Post-GSoC Period

I would like to continue working with the PyDataStructs team to implement other classes and add data structures in both Python and C++ backend. I'll also contribute to exciting future goals, which lead to bringing PyDataStructs closer to everyday use.

4 Acknowledgements

I would like to thank Gagandeep Sir for helping me get started with open source contributions, giving valuable feedback on my Pull Requests, guiding

me whenever requested and most importantly, providing valuable advice to help me draft this proposal.

5 Preference

I have submitted 2 proposals for GSoC 2024. My order of preference is:

1. Python Software Foundation: LPython
2. NumFOCUS: Open Science Labs: PyDataStructs

My primary objective for participating in the Google Summer of Code (GSoC) program is to make substantive contributions to open-source projects. Consequently, I am open and willing to contribute to any organization, as I am genuinely enthusiastic about contributing to a project during my summer vacation.