Personal Details

I am Hossam El-Deen El-Sabbagh from Cairo, Egypt. I am a 3rd-year student at Faculty of Computers and Information - Cairo University.

Phone: +20 1067991399
Email: hossameldeenfci@gmail.com

Years Completed: Both now and with the start of GSoC, I'll be in my 2nd semester in my 3rd year. It'll end some time in June.
Current Program: I'm CS-Major student, currently pursuing a BSc. degree. I'm currently a 3rd-year student.

Academic Experience

I'm a 3rd-year Computer Science Major. I'm the first on my class with regard to the GPA (3.95/4.0). Many courses had programming part. For example, In *Introduction to Decision Support* course, I implemented the *Simplex* algorithm.

Programming courses I've completed

Programming-1

Introductory programming course using C++.

Programming-2

Object-oriented programming using C++.

Data Structures

Learning and implementing many basic data structures and algorithms, asymptotic analysis of the different algorithms. Part of it was the independent study (and possibly implementation) of some data structures, such as: *Treap, Rootish Array and Dequeue*. Example graph algorithms studied were: *DFS, Prim Algorithm for Minimum Spanning Tree*. Many other data structures and algorithms were taught (and mostly implemented, required to solve problems with usually by making modifications to them).

Software Engineering-1

It was mainly about the idea of following a process during software development. The phases it revolved around were Requirements, Design, and Implementation software-development phases.

Multimedia

This course was about compressions algorithms, implemented various compression algorithms in Java, such as: *Standard Huffman, Adaptive Huffman, Arithmetic Coding, Lloyd's Optimal Quantizer, Vector Quantizer,* and a few others. And studied others in theory only such as *JPEG, MPEG*. Most algorithms were implemented using *Java Bitset*. to minimum the space complexity

of the algorithms.

Programming courses I'm taking this semester

Software Engineering-2

Its main focus is on Design, and Testing.

Algorithms

Following to some extent *Stanford*'s *Design and Analysis of Algorithms*.

Algorithms & Data Structures

My knowledge in Algorithms & Data Structures comes mainly either from my *Competitive-Programming* experience, or as part of faculty courses. Example algorithms and data structures I know or learned once, (and probably implemented) other than mentioned above: *Quick sort, k-way Merge-sort, B-Trees, Divide-and-conquer closest-pair solution, Strassen's algorithm for matrix multiplication, Simplex, Morris Algorithm for Inorder Traversal, Minmax, Hash-tables, Cichelli's Algorithm*, Minimax, and many others.
There are also techniques I'm very familiar with and use extensively such as: *Dynamic Programming*, *Greedy*, *Divide-and-conquer*.
I'm most comfortable with *recursion*, and it's often my favorite way of solving if performance wasn't an issues, if it's correct, it's so easy to prove, and usually no tracing is needed. I'm comfortable with simulating *recursion* with iterative solutions.

Graph-algorithm Experience

As part of my *Competitive Programming* participation, I implement some graph algorithms a lot on the fly and possible do some modifications to them according to problem at hand, such as: *DFS, BFS, Dijkstra's Shortest Path*. And there're others I've implement less often such as: *Floyd-Warshall*'s *All-Pairs-Shortest-Paths, Topological Sort*. I know in theory, but haven't tried to implement, such as: Bellman Ford.

Programming Experience

Programming Languages

C++: most comfortable with. Familiar with template programming, metaprogramming, done projects using them. Most comfortable with recursion.
Java: Made a good number of projects/assignments with it.

Operating Systems

Windows: Main OS.
Ubuntu: Started working on recently.

Libraries

A little of Boost.

Open-source experience

But worked with *Git* in SWE Project.

Competitive Programming

- 16th Rank 2013 ACM Programming Contest - Egypt.
- My best in a *TopCoder* online *SRM*: 1st on Egypt, and 137th world-wide.
- Facebook 2013 Hackercup: Advanced to Round-2, 290th in Round-1.
- TopCoder & Codeforces handles: hossameldeenfci.

Onlines Codes

ContactManager project, which includes implementation of Treap, Trie, and using Trie in a novel way: https://github.com/hossameldeenfci/ContactManager/tree/master/src .
Facebook 2013 Hackercup: (Round-1 codes are here: here, here, and here).
Codeforces Submissions: here.

Project Description "Search Trees and Priority Queues"

Mentor: Dr. Carsten Gutwenger.

Detailed Description

There seems to be a need for implementing more basic data structures in *OGDF*. The project consists mainly of 3 parts:

First Part

- Deciding on a good and generic design that separates between abstract data types, and implementing data structure. As stated in project description, the design is aiming at compile-time flexibility, so that's where research will be, especially trying to come up with a way that makes the abstract data type use the implementing data structure seeminglessly (That is, it doesn't differentiate between which data structure it's using in its code). A possible way for doing so is to make the data structures implement common class with same definition. More research will be made if a more elegant solution with no or little runtime overhead is available.
A problem that may arise for the above idea: If implementing data structures are used for different abstract data types, that may inflate their APIs to make them suitable for all. More research on this part shall be done, and solution should be come up with.

Part Two

Implementing a bunch of data structures: Which data structures to be implemented are still to be decided during the project.

Part Three

Study of *OGDF* libraries and trying to extract out the already-existing data structures, and encapsulate them in generic classes of their own.
[Optional] If the resulting design is suitable for the classes from which we extracted some data structures, perhaps refactor these classes to make use of the new classes. But that will be according to time constraints. Another possible extension is to add more abstract data types, or

data structures [To be decided according to time constraints during the project], for example, *SortedSequence* abstract data type.

Timeline

There are 2 main parts of my work:

Research Period

The one before during *Coding Period* is about researching the idea more, learning more about *OGDF*, learning needed knowledge such as C++11 relevant features, more on C++ metaprogramming.
Probably more idea research will be done starting from March 25th.

Coding Period

Explained in the following timeline.

Work hours per week

May 19 - June 23: 25-35 hours
June 23 - August 18: 40-50 hours

Other Commitments

There'll be probably 6 days where my working hours will be less than usual some time around June (for final exams).
I'll work some days full-time to compensate for other days.

Timeline and Plan

Week No	GSoC Milestone	Work	Deliverables
Week 1 - 2	Students begin coding for their Google Summer of Code projects	Researching the possible design decisions. Coming up with a generic design	Generic design should be ready
Week 3 (Work hours are less than usual)		Documentation and testing of design on small modules	Design documentation should be ready - C++ tutorials should be ready
Week 4 - 5	Mentors and student can begin submitting midterm evaluations.	Deciding on which data structures to implement, and possibly implement one of them.	Implementation of the abstract data types *PriorityQueue and Dictionary* (and perhaps *SortedSequence)*

Week 6 - 7 - 8 - 9		Implement Data Structures (or extract them from *OGDF*), with a rough rate of 2 data structures per week. Unit testing is included	About 12 - 16 data structures should be implemented and tested.
Week 10 - 11.5		Testing the integration	At the very least, *PriorityQueue* and *Dictionary* should tested with all data structures and be ready to use.
Week 11.5 - 12		Documentation for everything not documented	Documentation of everything
Week 13	Suggested 'pencils down' date.	Project Buffer for anything - any enhancement	
Week 14	Firm 'pencils down' date. Final submission	Google Summer of Code Ends	

Anticipated Challenges

- Some Data Structures' implementation take more time than needed.
- Something wrong comes up with the design at the very end. To try to prevent that, small modules for testing the design will be made, and *PriorityQueue* design may start as soon as one data structure for it is done. Same for *Dictionary*.
- The complexity of the design is a little high.
- There's no universal way for communicating design based on compile-time polymorphism.

Why OGDF?

Previous Experience with OGDF

No, I haven't used *OGDF* before GSoC.

Interest in Graph Algorithms

In *Competitive Programming*, one of my main types of problems are those related to graphs, or can be solved easily if put modeled as a graph. They're just so much fun to me!
Although I haven't built interest yet in applications of graph other than problems faced at college or Competitive Programming: When attending a crash course on *Foundations of Distributed Computing* for Prof.. Amr El-Abbadi (from UC, Berkeley), and seminar about *Representing and Reasoning about uncertain knowledge* for Dr. Hala Mostafa, it just seemed that graphs and graph algorithms are an essential tool of thinking for a computer scientist. In fact, in pretty much all my study, graphs almost always came up, and knowing a good number of algorithms could make my life easier, or accelerate some thinking phases.

Interest in Graph Drawing

I still haven't built enough interest in *Graph Drawing* algorithms, but until now, there're mainly 2 driving forces for my interest in it:

- I've always had problems with graphs and visualizing the problem/solution/test case, probably could make life easier. But till before GSoC, it didn't come to my mind that such thing existed of an acceptable quality.

My vision of involvement with OGDF after GSoC

What I'm pretty sure about is that I'll continue contributing through implementing (and possibly enhancing) more basic data structures and algorithms, especially next two semesters as their load is less than usual. And I have the intention of being a regular OGDF contributor.
There's very high probability that I implement *Basic Linear Algebra Support* as well, if it will not have been implemented yet.
For graph-drawing related algorithms: I have no experience in it yet, but they seem interesting and fun, though. Learning them on my own with small guide of *OGDF* community will probably add a lot to me, especially in the part of reading research papers, as I'm not so good at it yet. So, I hope to do so, but not so sure if I can, especially next year. Probably after graduation, it'll be my free-time technical hobby.
I'm currently thinking about giving sessions about *OGDF* after GSoC, to get students started with open source, as I think skills of my colleagues are suitable for such organization, and contributing to open source could be both fun and very useful.
If I come to really like graph drawing, perhaps my Masters Thesis would be about it (if I become a Masters student, which I'm not yet, and not sure if I will).

Additional Notes

Extracurricular Activities

- I was a teaching assistant in a few *Introduction to Programming using C++*, and *Object-Oriented Programming*, and *Data Structures* courses. I'm also a very active member on

my class online groups.
- I was on the Cairo University Delegate to Germany for celebrating the cooperation between *Ulm University* and *Cairo University*.
- I've given one or a few technical sessions.
- I'm planning to start my blog very soon.

Programming Exercise(s)

- While implementing the programming exercise(s), coding standards seemed very natural to me and I was most comfortable coding with them.