



# Six Degrees of Separation

GitHub Repository: [github.com/KrishGoel/sixdegreesofseparation](https://github.com/KrishGoel/sixdegreesofseparation)

<https://github.com/KrishGoel/sixdegreesofseparation>

Submission of **Krish Goel (#210310342)**, Section C of CSE with AI & ML

## The Project

A Python ~~and SQL based~~ software application that helps you visualize the Six Degrees of Separation Theory through sophisticated implementations of Graph Theory and shortest path algorithms.

## Math and Theory

 <https://math.mit.edu/research/highschool/primes/circle/documents/2021/Heikkinen.pdf>

A mathematical paper on Graph Theory and the Six Degrees of Separation Theory

### Shortest Path Algorithms | Brilliant Math & Science Wiki

Shortest path algorithms are a family of algorithms designed to solve the shortest path problem. The shortest path problem is something most people have some intuitive familiarity with: given two points, A and B, what is the shortest path between them?

 <https://brilliant.org/wiki/shortest-path-algorithms/>



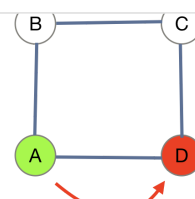
BRILLIANT

Notes on Shortest Path Algorithms

### Graph Shortest Path Algorithms

Let's make some change, say \$0.47 cents. What are we going to do? Well, we start with \$0.47, and then we take out a quarter. Then, we subtract \$0.25 from the original, and we are left with \$0.22. Then, we take a dime, and subtract that from what is left over, for a total of \$0.12 cents

 <https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1206/lectures/dijkstra/>




Shortest Path: A-D

Notes on Shortest Path Algorithms

## Tech Stack

### Application Layer Programming

Welcome to Python.org  
The official home of the Python  
Programming Language


 <https://www.python.org/>




### (Abandoned) Front End

Svelte.JS (for front-end) (with SvelteKit). The basics of Svelte boil down to HTML, CSS and JS (follows a single page template format).

Svelte  
Cybernetically enhanced web  
apps

 <https://svelte.dev/>



SvelteKit  
SvelteKit is the official Svelte  
application framework


 <https://kit.svelte.dev/>




### (Abandoned) Authentication and Realtime Database

Merge signup and login features for simplicity (renamed → check-in).

Firebase  
Join us in person and online for  
Firebase Summit on October  
18, 2022. Learn how Firebase

 <https://firebase.google.com/>




### Back End (

Primarily using the course material and learnings from Relational Database Management Systems Lab.

#### MySQL

MySQL HeatWave is a fully managed service that enables customers to run OLTP, OLAP, and machine learning workloads directly from their MySQL Database. HeatWave boosts MySQL

 <https://www.mysql.com/>

<https://apex.oracle.com/>

### Hosting, APIs and Other Services

#### (Abandoned) Hosting

Develop. Preview. Ship. For the best frontend teams - Vercel

Vercel combines the best developer experience with an obsessive focus on end-user performance. Our platform enables frontend teams to do their best work. Start with the developer Vercel is the

<https://vercel.com/>

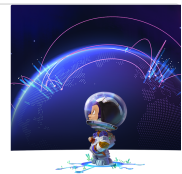
### Version Control

Using GitHub, repository link shared.

GitHub: Where the world builds software

GitHub is where over 83 million developers shape the future of software, together. Contribute to the open source community,

 <https://github.com>



## Functionality and Future Rollouts

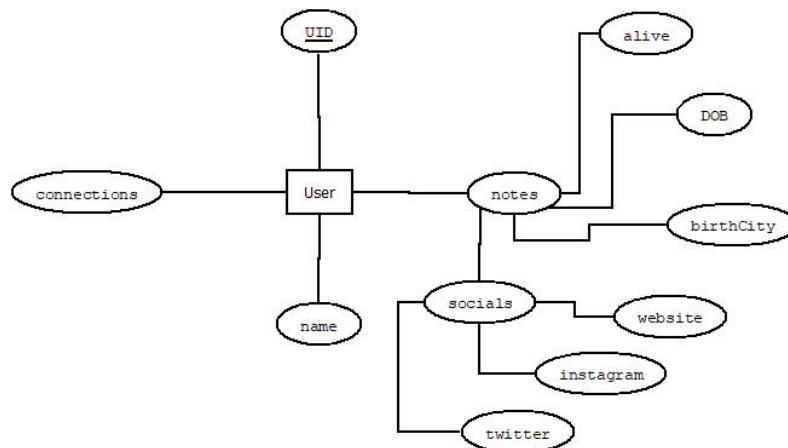
### Current Functionality

- Enlist all users on the database
- Find the social distance (if the connection exists) between any 2 people

## Future Rollouts

- Ability to find the strength of the bond (by adding weights to each binary-bond)
  - Ability to find paths beyond the ones entered directly into the database through social media profiles
- This will be achieved using Instagram, Twitter and Facebook APIs. Upon being granted the permission, the software will import all overlapping entries which will then be used to expand the network. This would however only be limited to people the user would know online and more in-real-life connections would still require to be entered manually. This concept may be extended on to the contacts section of the user's phone.
- Ability create users autonomously (without being the database admin)

## Designing the Database



ER Diagram for the project, made in DIA.

The primary objective was to minimize dynamicity of each record entry

This was accomplished by reducing the number of attributes a person may have to only the essential ones requiring close to none maintenance. Examples of attributes eliminated are `cityOfResidence`, `age`, `stateOfLiving` (i.e. still alive) etc.

## How the `distanceCalculator()` function works

Effectively, the data structure formed in this program is a tree with unweighted edges (at least for the initial stages, will be upgraded to weight edges in the future rollouts). Naturally, the most effective way to go about this traversal would be a Breadth First Search (BFS).

However, conventional BFS assumes that all vertices are reachable from the starting vertex. This will not be necessarily true in our case, therefore a slight modification is required. The BFS function performed would have to perform BFS from each unvisited node of the graph post the first iteration. The output of this function will be conclusive in deciding the length of the social path or if it exists at all.

This will be implemented by creating an adjacency matrix through the `connections` attribute of each data entry. The code for the same will be as follows (pseudo code assuming a 5 point system, can be generalized, covered in EM-3 MA2102) -

```
# Python3 implementation of modified BFS
import queue

# A utility function to add an edge
# in an undirected graph.
def addEdge(adj, u, v):
```

```

adj[u].append(v)

# A utility function to do BFS of
# graph from a given vertex u.
def BFSUtil(u, adj, visited):

    # Create a queue for BFS
    q = queue.Queue()

    # Mark the current node as visited
    # and enqueue it
    visited[u] = True
    q.put(u)

    # 'i' will be used to get all adjacent
    # vertices of a vertex list<int>::iterator i

    while(not q.empty()):

        # Dequeue a vertex from queue
        # and print it
        u = q.queue[0]
        print(u, end = " ")
        q.get()

        # Get all adjacent vertices of the
        # dequeued vertex s. If an adjacent
        # has not been visited, then mark
        # it visited and enqueue it
        i = 0
        while i != len(adj[u]):
            if (not visited[adj[u][i]]):
                visited[adj[u][i]] = True
                q.put(adj[u][i])
            i += 1

# This function does BFSUtil() for all
# unvisited vertices.
def BFS(adj, V):
    visited = [False] * V
    for u in range(V):
        if (visited[u] == False):
            BFSUtil(u, adj, visited)

# Driver code
if __name__ == '__main__':

    V = 5
    adj = [[] for i in range(V)]

    addEdge(adj, 0, 4)
    addEdge(adj, 1, 2)
    addEdge(adj, 1, 3)
    addEdge(adj, 1, 4)
    addEdge(adj, 2, 3)
    addEdge(adj, 3, 4)
    BFS(adj, V)

# This code is contributed by Pranchalk

```

## Roadmap

### Progress of Things

| Aa Task   | ≡ Type                                     | ☀ Status | 🕒 Timeline | ≡ Notes                                    |
|---|--|----------|------------|--|
| <a href="#">Understand shortest path<br/>algs and the theory.<br/>paper</a> | <div>Learning</div> <div>Programming</div> | Done     | Week 1     | Notes added to the math and theory section |

| Aa Task  | Type                                       | Status      | Timeline | Notes   |
|--|--|-------------|----------|---|
| <u>Make the progress deck for the first week</u>                         | Documentation                              | Done        | Week 1   | Presented   |
| <u>Initiate the Git repository and share it with @aishshub on Github</u> | Deployment<br>Documentation                | Done        | Week 1   | <a href="https://github.com/KrishGoel/sixdegreesofseparation">https://github.com/KrishGoel/sixdegreesofseparation</a>   |
| <u>Make the Progress Deck for the second week</u>                        | Documentation                              | Done        | Week 2   |   |
| <u>Make the ER Diagram on-paper and paste in the RDBMS File</u>          | Development                                | Done        | Week 2   |   |
| <u>Write the project excerpt</u>   | Documentation                              | Done        | Week 2   |   |
| <u>Start the Oracle Course</u>   | Learning                                   | Done        | Week 3   | Running late due to home visit for a week   |
| <u>Learn the conversion of ER Model to a SQL Table(s).</u>               | Learning                                   | Done        | Week 3   | From Korth's Book on Database Systems   |
| <u>Convert the ER Diagram from the RDBMS File to a SQL Schema</u>        | Development                                | Done        | Week 4   |   |
| <u>Change the tech-stack to Python and drop the Svelte based webapp</u>  | Development                                | Done        | Week 5   | Overambitiousness   |
| <u>Learn to and create the database in MySQL</u>                         | Development<br>Learning                    | Done        | Week 6   |   |
| <u>Write the Python Script for traversal and shortest-path functions</u> | Development<br>Programming                 | Done        | Week 7   |   |
| <u>Complete connecting the database with Python interface</u>            | Development<br>Integration                 | In progress | Week 7   | Failed, tech stack changed  |
| <u>Final updates</u>   | Deployment<br>Development<br>Documentation | Done        | Week 8   | Project fully deployed, SQL was an overambitious addition in the given time-frame and consequentially was dropped. The Python program now uses JSON for the database. |
| <u>Make the Progress Deck for the whole duration</u>                     | Documentation                              | Done        | Week 8   | Done  |