01

Undirected Graphs

02

Undirected Graphs

So, in this project, We are going to talk about a lot of topics related to the Undirected Graphs. And below are the topics that we have covered in this project:

03 Definition

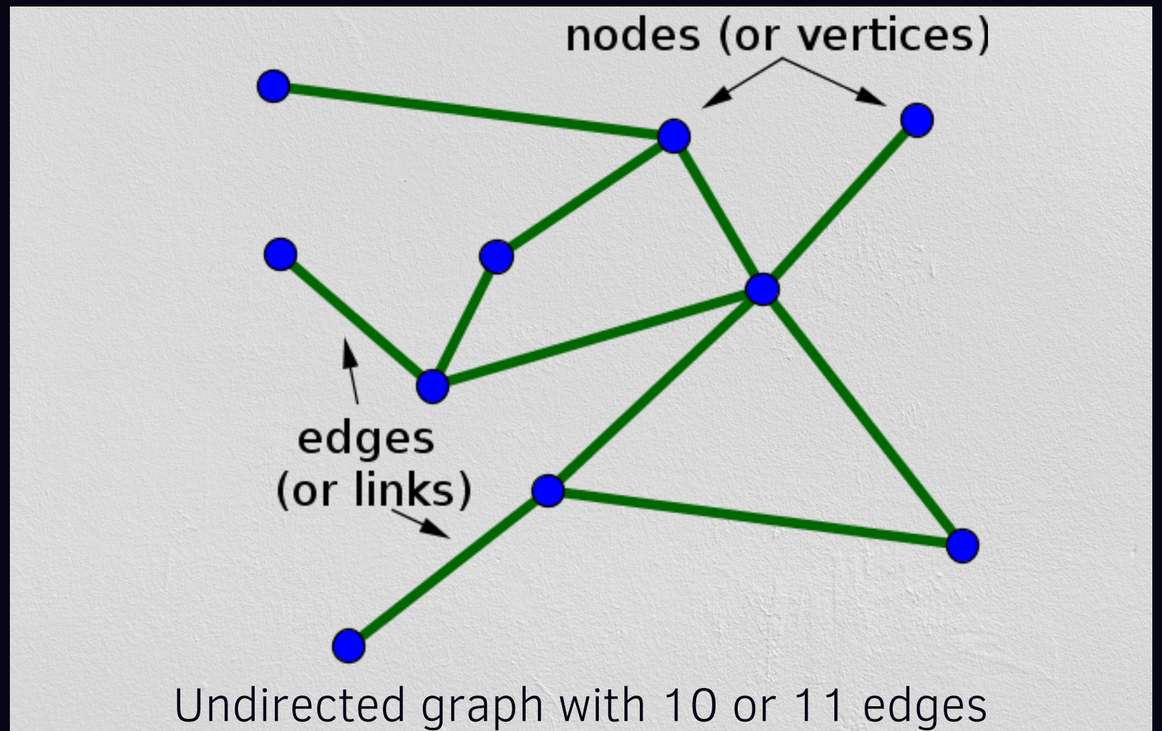
The definition of Undirected Graphs is pretty simple:

Set of vertices connected pairwise by edges.

Graph definition

Any shape that has 2 or more vertices/nodes connected together with a line/edge/path is called an undirected graph.

Below is the example of an undirected graph:



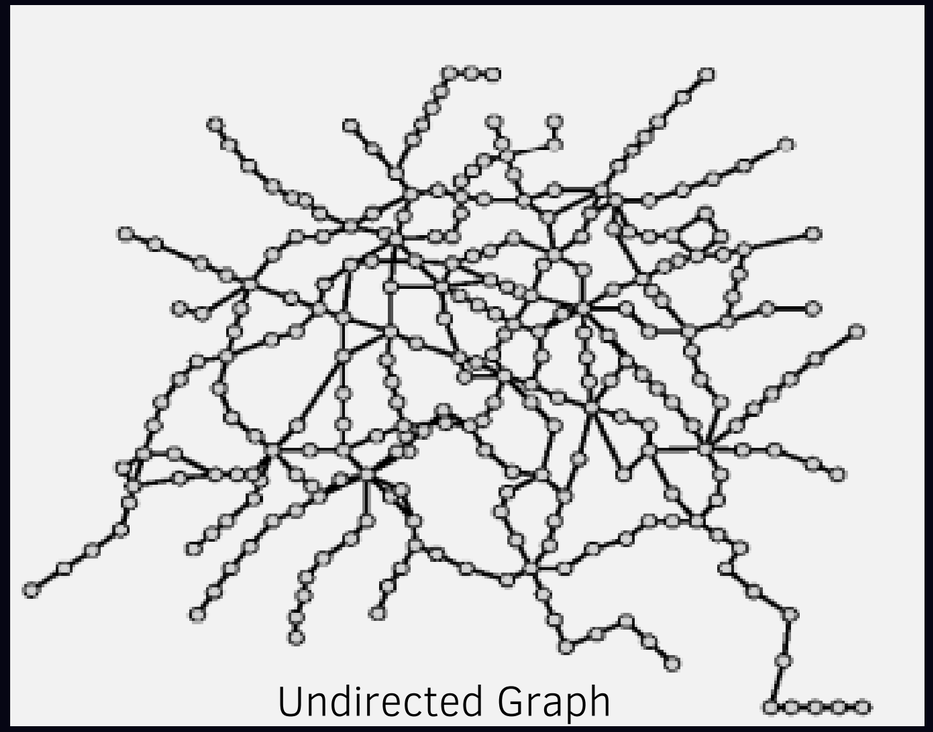
Undirected graph with 10 or 11 edges

04

Vertices are the result of two or more lines intersecting at a point. Edges or Links are the lines that intersect.

These graphs are pretty simple to explain but their application in the real world is immense. You will see that later in this presentation.

Here’s another example of an Undirected Graph:



Undirected Graph

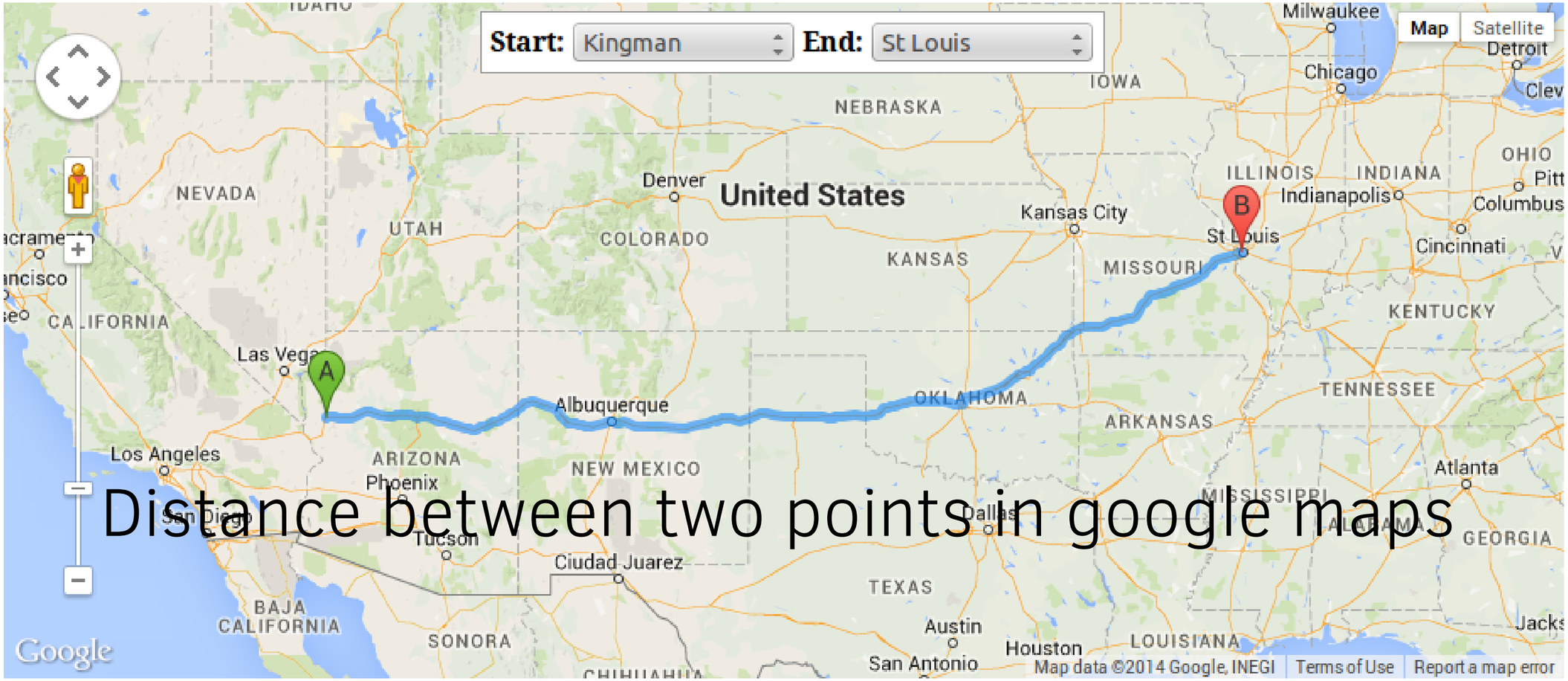
05

Why Study Graphs?
Application of given theory

As we said, there are thousands of practical applications of Undirected Graphs . And being an Information Systems student, it’s our job to study about these applications and try to make it more efficient.
As of today, there are thousands of graph algorithms that are used for different purposes. For example – Google map uses some of the graph algorithms to find the shortest distance between two points on Google Maps.

06

Directions service



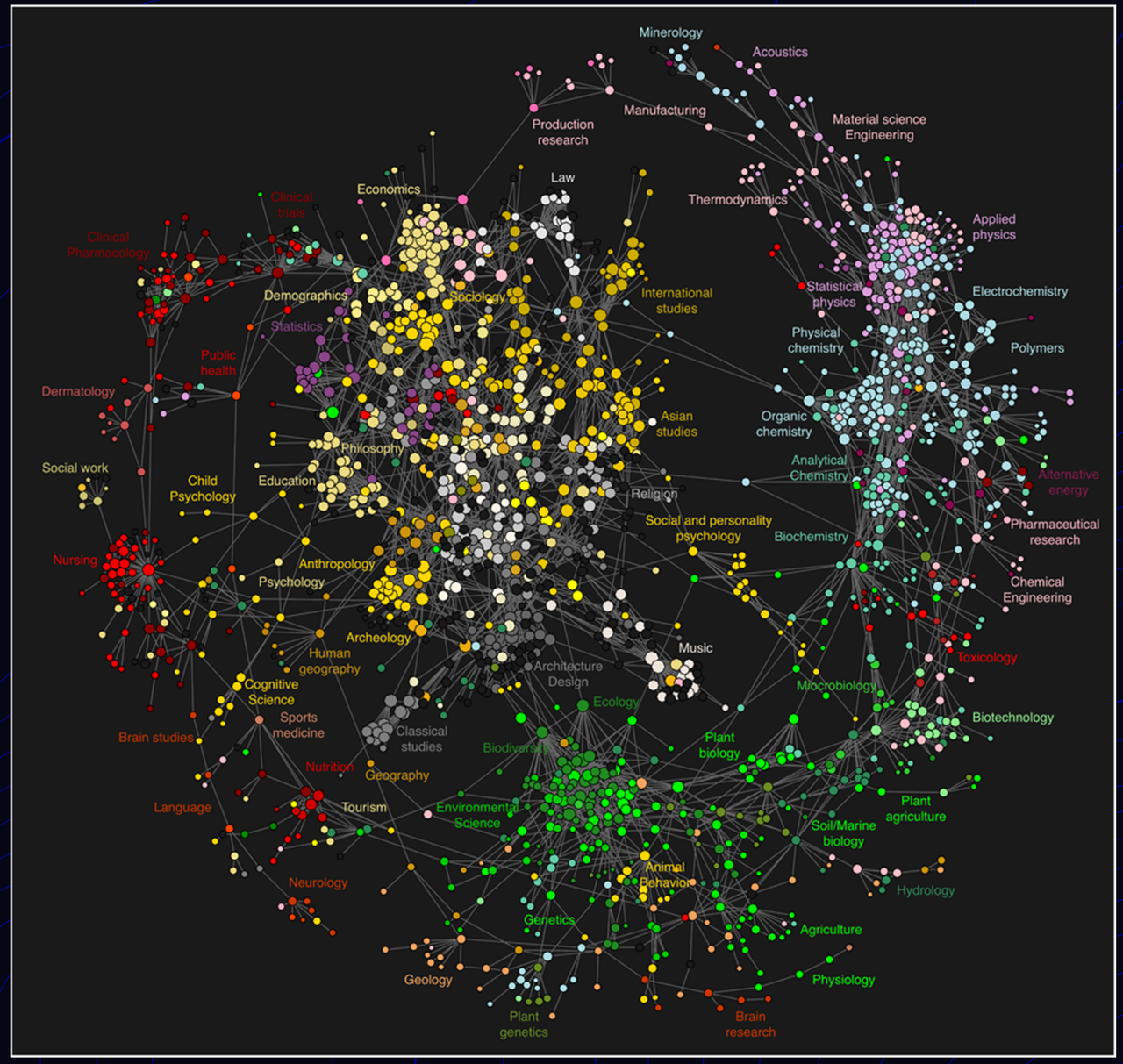
Distance between two points in google maps

07

Many businesses today such as Ola, Uber, Grab etc... relies on the accuracy of these maps. Their entire business model revolves around it. Smallest miscalculation would cause them thousands of bucks.
All of this is made possible just with the help of Graph Algorithms.
Well, let us share with you some more real-world applications of Graphs and the algorithms that made it possible.

08

09



MAP OF SCIENCE CLICKSTREAMS

10

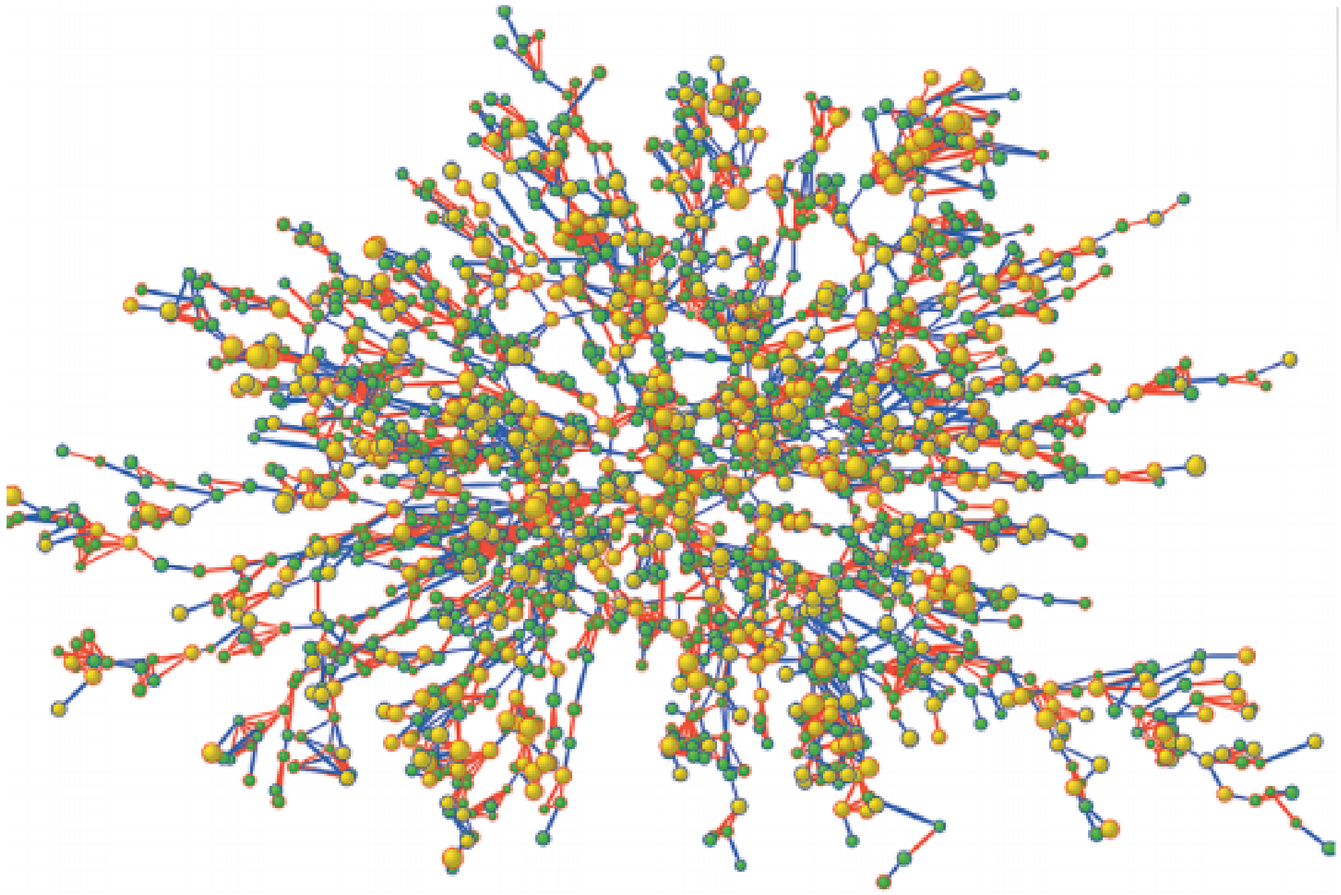
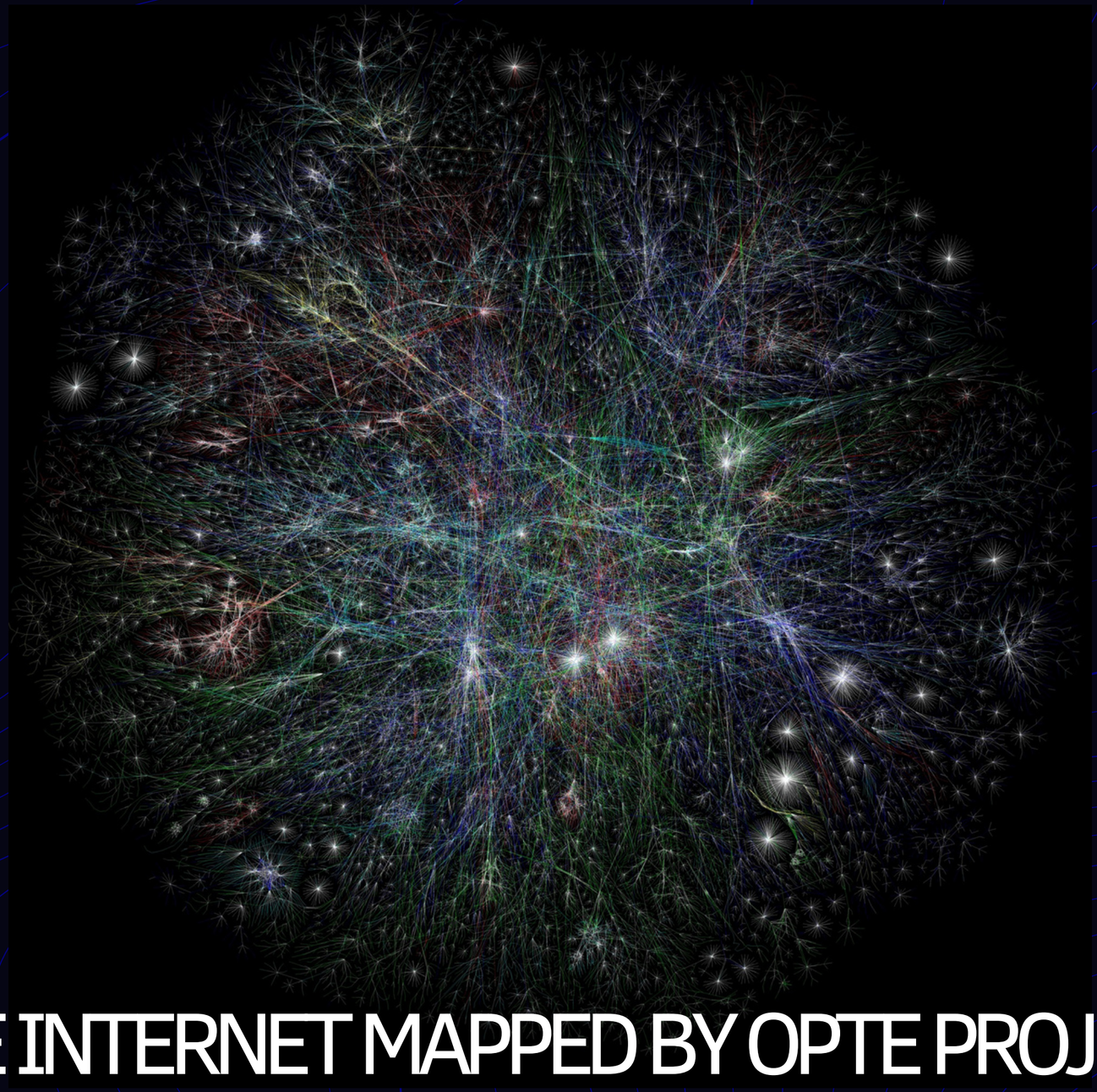


Figure 1. **Largest Connected Subcomponent of the Social Network in the Framingham Heart Study in the Year 2000.**
Each circle (node) represents one person in the data set. There are 2200 persons in this subcomponent of the social network. Circles with red borders denote women, and circles with blue borders denote men. The size of each circle is proportional to the person's body-mass index. The interior color of the circles indicates the person's obesity status: yellow denotes an obese person (body-mass index, ≥30) and green denotes a nonobese person. The colors of the ties between the nodes indicate the relationship between them: purple denotes a friendship or marital tie and orange denotes a familial tie.

FRAMINGHAM HEART STUDY

11



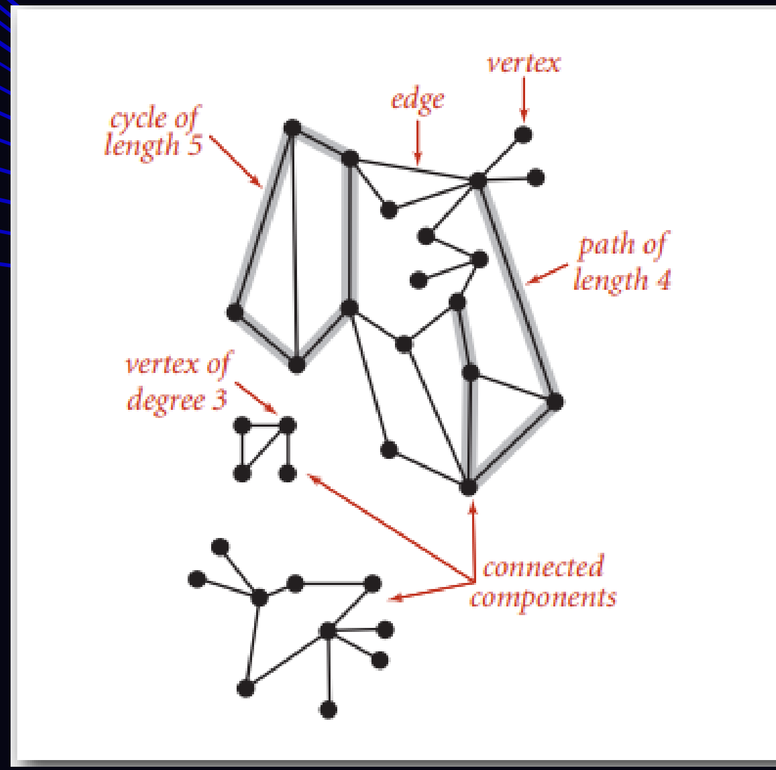
THE INTERNET MAPPED BY OPTE PROJECT

12

These are some of the live applications made possible by studying graphs. Almost all the maps that you and we use is the consequence of studying graphs.

Meaning-out-of-chaos is determined with the help of graph. And if you are going to do anything amazing, you must study the graph algorithms. Mark Zuckerberg is a genius and that’s why he was able to connect everyone around the entire globe. And we are sure he was also a really good student of algorithms.

Graph Terminology



The graph terminology is pretty simple and easy.

- Path
- Cycle

Path

Sequence of vertices connected by edge.

Cycle

Path whose first and last vertices are the same.

You can say that the two vertices are connected if there is a path between them.

Say you want to go to point B from some point A. It is only possible for you to go to that point if there is a path between the two. The path that connects both the point.

Here is the image that depicts the same:

14

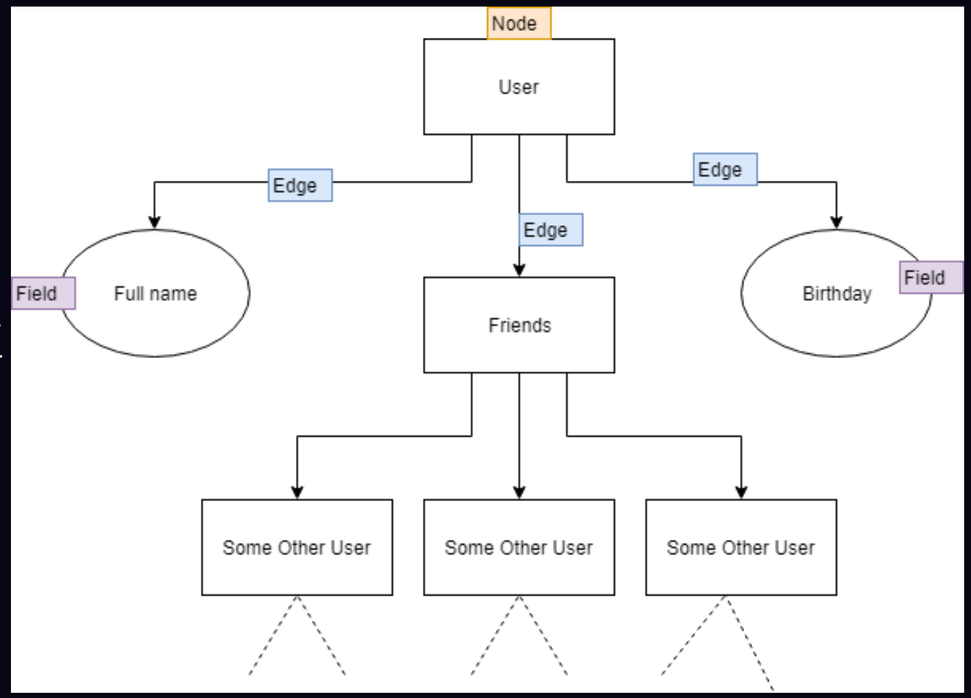
Graph API

We thought of using the Facebook Graph API concept to illustrate it. Because most of you are aware of Facebook and how it works.

And just to be clear, facebook graph API is a real thing. The developers built Facebook this way. Graph API records every action that the user makes on the page.

Facebook’s Graph API is composed of:

- Nodes: Individual objects such as a User, a Photo, a Page or a Comment.
- Edges: they represent connections between different objects such as Photos on a Page or Comments on a Photo.
- Fields: Actual data associated with the object, like User’s birthday or Page’s name.



So in order to access any information, typically you use nodes to get data about a specific object, use edges to get collections of objects on a single object and use fields to get data about a single object or each object in a collection.

Undirected graphs representation

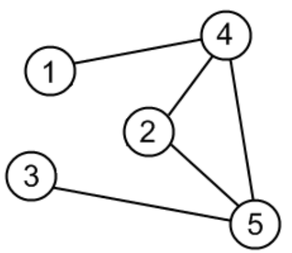
Adjacency matrix

There are several possible ways to represent a graph inside the computer. We will discuss two of them: adjacency matrix and adjacency list.

Adjacency matrix

Each cell aij of an adjacency matrix contains 0, if there is an edge between i-th and j-th vertices, and 1 otherwise. Before discussing the advantages and disadvantages of this kind of representation, let us see an example.

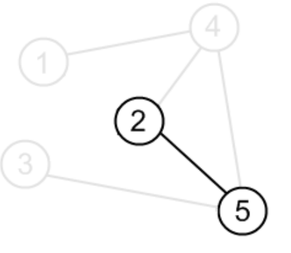
The graph presented by example is undirected. It means that its adjacency matrix is symmetric. Indeed, in undirected graph, if there is an edge (2, 5) then there is also an edge (5, 2). This is also the reason, why there are two cells for every edge in the sample. Loops, if they are allowed in a graph, correspond to the diagonal elements of an adjacency matrix.



Graph

	1	2	3	4	5
1	0	0	0	1	0
2	0	0	0	1	1
3	0	0	0	0	1
4	1	1	0	0	1
5	0	1	1	1	0

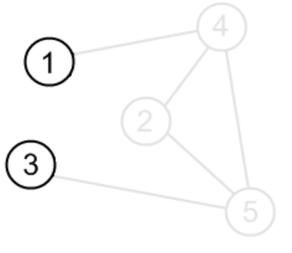
Adjacency matrix



Edge (2, 5)

	1	2	3	4	5
1	0	0	0	1	0
2	0	0	0	1	1
3	0	0	0	0	1
4	1	1	0	0	1
5	0	1	1	1	0

Cells for the edge (2, 5)



Edge (1, 3)

	1	2	3	4	5
1	0	0	0	1	0
2	0	0	0	1	1
3	0	0	0	0	1
4	1	1	0	0	1
5	0	1	1	1	0

Cells for the edge (1, 3)

Advantages & Disadvantages

Advantages. Adjacency matrix is very convenient to work with. Add (remove) an edge can be done in O(1) time, the same time is required to check, if there is an edge between two vertices. Also it is very simple to program and in all our graph tutorials we are going to work with this kind of representation.

Disadvantages.

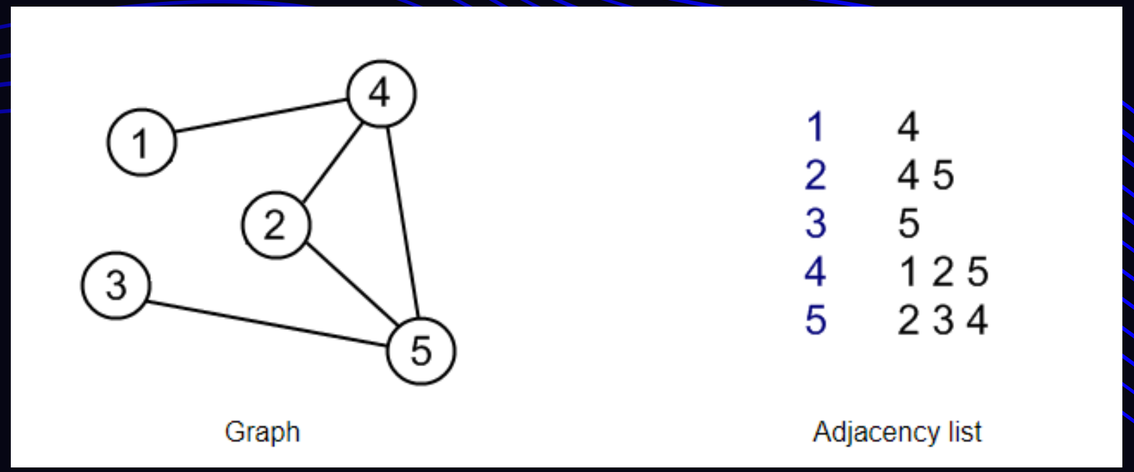
- Adjacency matrix consumes huge amount of memory for storing big graphs. All graphs can be divided into two categories, sparse and dense graphs. Sparse ones contain not much edges (number of edges is much less, that square of number of vertices, |E| << |V|2). On the other hand, dense graphs contain number of edges comparable with square of number of vertices. Adjacency matrix is optimal for dense graphs, but for sparse ones it is superfluous.
- Next drawback of the adjacency matrix is that in many algorithms you need to know the edges, adjacent to the current vertex. To draw out such an information from the adjacency matrix you have to scan over the corresponding row, which results in O(|V|) complexity. For the algorithms like DFS or based on it, use of the adjacency matrix results in overall complexity of O(|V|2), while it can be reduced to O(|V| + |E|), when using adjacency list.
- The last disadvantage, we want to draw you attention to, is that adjacency matrix requires huge efforts for adding/removing a vertex. In case, a graph is used for analysis only, it is not necessary, but if you want to construct fully dynamic structure, using of adjacency matrix make it quite slow for big graphs.

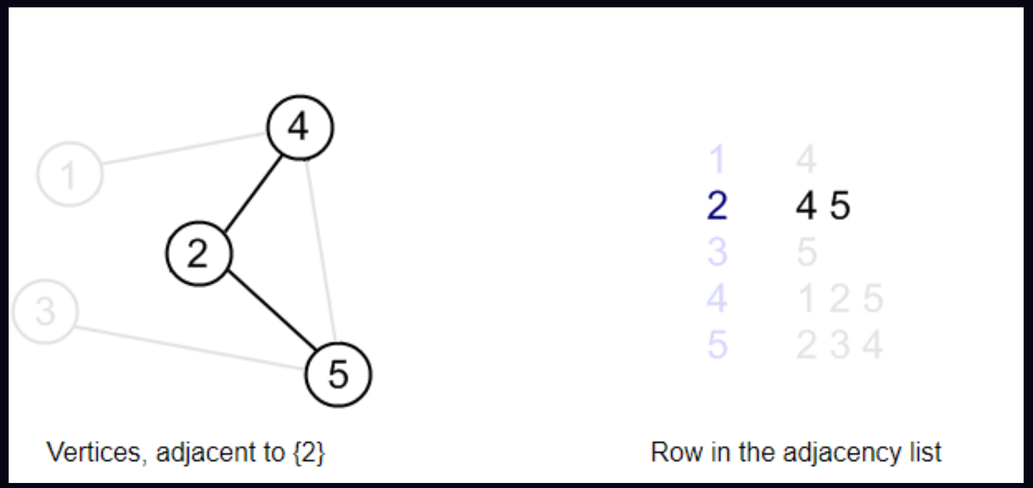
To sum up, adjacency matrix is a good solution for dense graphs, which implies having constant number of vertices.

Undirected graphs representation

Adjacency list

This kind of the graph representation is one of the alternatives to adjacency matrix. It requires less amount of memory and, in particular situations even can outperform adjacency matrix. For every vertex adjacency list stores a list of vertices, which are adjacent to current one. Let us see an example.





Advantages & Disadvantages

Advantages. Adjacent list allows us to store graph in more compact form, than adjacency matrix, but the difference decreasing as a graph becomes denser. Next advantage is that adjacent list allows to get the list of adjacent vertices in O(1) time, which is a big advantage for some algorithms.

Disadvantages.

- Adding/removing an edge to/from adjacent list is not so easy as for adjacency matrix. It requires, on the average, O(|E| / |V|) time, which may result in cubical complexity for dense graphs to add all edges.
- Check, if there is an edge between two vertices can be done in O(|E| / |V|) when list of adjacent vertices is unordered or O(log2(|E| / |V|)) when it is sorted. This operation stays quite cheap.
- Adjacent list doesn't allow us to make an efficient implementation, if dynamically change of vertices number is required. Adding new vertex can be done in O(V), but removal results in O(E) complexity.

To sum up, adjacency list is a good solution for sparse graphs and lets us changing number of vertices more efficiently, than if using an adjacent matrix. But still there are better solutions to store fully dynamic graphs.

OUR TEAM:

NURGOZHAYEVA KAMILLA

URMANOVA NAZERKE

MUKHAMBETZHANOVA LAILA

DUISENBAYEV NURLYKHAN

MYRATBEKOV DAMIR

THANKS!