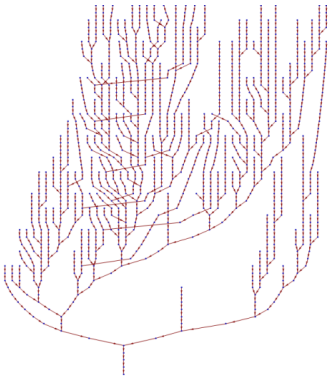
VOEIS Process Book

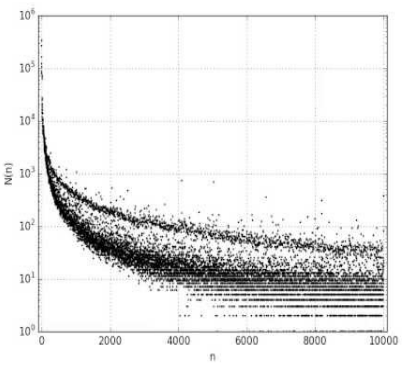
https://github.com/DavidJMiller/VOEIS

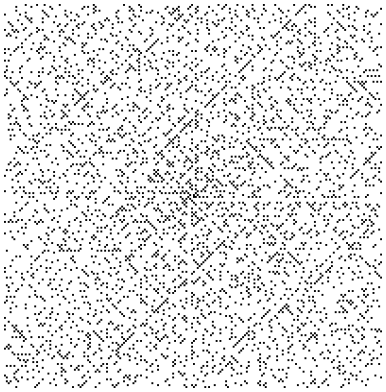
Qianlang Chen	David Miller	Jiawen Song
u1172983@utah.edu	u1312141@utah.edu	kevin.song.utah.edu
u1172983	u1312141	u1211977

Overview and Motivation

The importance of numbers is a core principle in Number Theory. Whether studied in an isolated manner or within the context of sequences, numbers contain patterns and significance. While Number Theory aims at uncovering these patterns and significance with mathematical tools, we propose to use visualization as our probing tool. It is no doubt that visualization can lead to interesting images that seem to show these hidden patterns. The Ulam Spiral, Sloane's Gap, and the Collatz Tree are a few that have been popularized through *Numberphile*, a YouTube channel that explains interesting patterns in math and a source of inspiration for us.



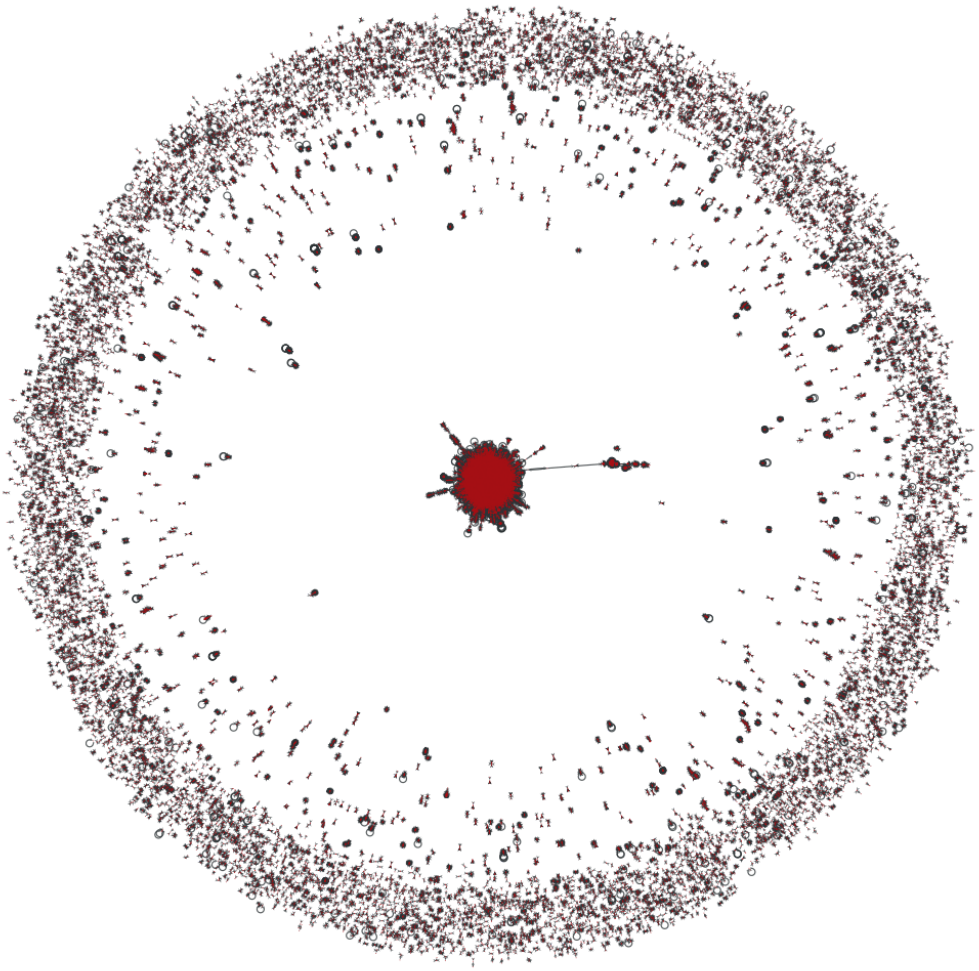




Since we are very interested in these patterns, naturally we asked ourselves what is the relationship among all patterns. We really want to determine why certain sequences spawn interesting visualizations while others do not or why a certain number contributes to many sequences. The goal of this project is to explore sequences and numbers within the context of sequences through visualization. Specifically, we are interested in visualizations that can convey meaning and are pleasing. Standards graphs and charts conveys the first part while the above visualizations conveys the second. We aim to create visualizations that convey both parts.

Related Work

There are many sequence visualizations that exist, as seen in the previous section. It is these interesting standalone visualizations that first sparked our interest. These first inspirations were encountered through *Numberphile* videos. Other inspirations are Dr. Katherine Stange's *Experimental Mathematics Lab*, in particular their project *Numberscope* (https://math.katestange.net/numberscope/), the discovery of new sequences through graph visualization in the paper *Integer Sequence Discovery from Small Graphs* (https://arxiv.org/pdf/1408.3644.pdf), and independent projects such as the one found at https://github.com/internaut/d3-int-sequences-visualizer. It is these visualizations which we drew inspiration from to create visualizations that are not only pleasing, but also informative. One thing we notice though is that most of the visualizations are a local viewpoint, that is the visualize the sequence within sequence within their own context. However, we are also interested in comparing sequences to each other as well as investigating the role of specific numbers within the context of sequences. One visualization, which can be found at https://sam.zhang.fyi/2018/12/01/oeis/, shows that there might be an inherent structure to how these sequences interact with each other. Sequences on OEIS have cross references to other sequences and the above graph shows sequences as nodes in which an edge connects two nodes if there is a cross reference between them. The graph below shows there is some inherent structure between sequences.



With the core idea of being able to visually investigate not only sequences, but multiple sequences and numbers, we plan to make visualizations from the mentioned inspirations.

Questions

What questions are you trying to answer?

- The questions we are curious to get answer from making this project is to explore the number sequence in the visualized way to find the patterns of each specific sequence and its relations to other sequences.

How did these questions evolve over the course of the project?

- Over the course of this project, we can see that there are a number of ways to visualize sequences that can be very intuitive for the people who are not exposed to number theory or math in general who can use this project to see the patterns.

What new questions did you consider in the course of your analysis?

- So the new questions we are considering now is how do we make users who are not just mathematicians and number theorists explore the number sequences with our VOEIS project with the help of interesting, interactive visualizations. So if the user can easily navigate to different number sequences and explore the differences and similarities among them.

Data

The Database

We've collected our data straight from the OEIS (https://oeis.org/wiki/Welcome). The raw data included two text files: one maps each sequence's unique ID to the sequence's name, and the other maps the IDs to the sequences' terms. We decided that we'll keep all relevant data in the Python server's local memory as dictionaries because that allows for faster lookup times. To make the process of building those dictionaries easier, we reconstructed the raw data from OEIS and called them our "database." Since our website has functions of showing information about sequences and numbers, we thought it'd be nice and friendly to our server if made the following databases:

- A sequence-database that maps a sequence's unique ID (officially called the "A-number" of a sequence) to its name and terms:

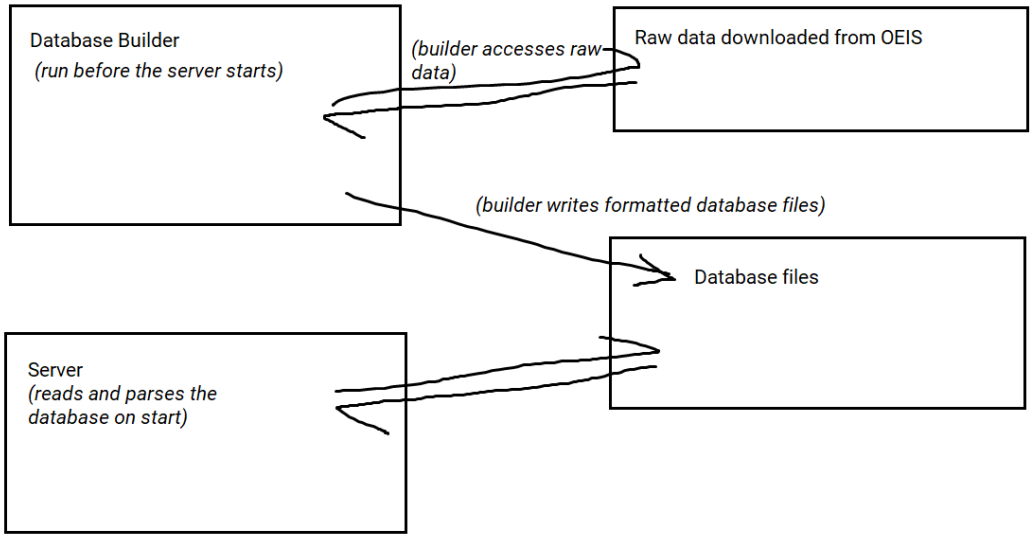
a_num	name	terms
"A000040"	"The prime numbers"	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...
...		

- A number-database that records useful information about a number that has ever appeared in an OEIS sequence. A number-data item contains the following fields:
    - The total number of appearance of this number;
    - The total number of OEIS sequences this number has appeared in;
    - The total number of appearance of this number at a particular index of any sequence;

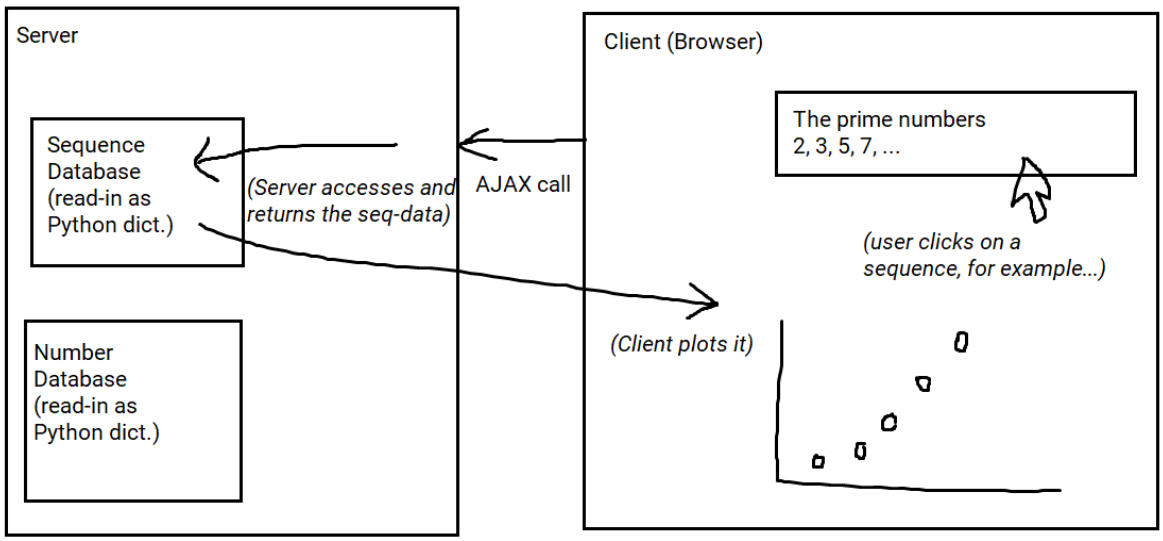
- The “popular neighbors” of this number. A neighbor is another number that likes to appear near this number in any sequence. Our database will include the neighbors of each number within six indices apart.

num	total_count	total_seqs	index_counts	neighbors	
				offset	neighbor_counts
2	312953	139205	{1: 4245, ...}	-6	{1: 3434, 3: 3565, ...}
				-5	{1: 3569, 2: 520, ...}
				...	
...					

These two database files are well-formatted text files that allows our server to quickly read and parse them as soon as it starts. This way, our server doesn’t need to read the files when the client’s already asking data, nor does it need to perform complex computations when certain data is asked as when the server only has the raw data at hand. We’ve included the Python program that does the database building process in our project, so we can download updated sequence-data from OEIS and rebuild our database at any time.

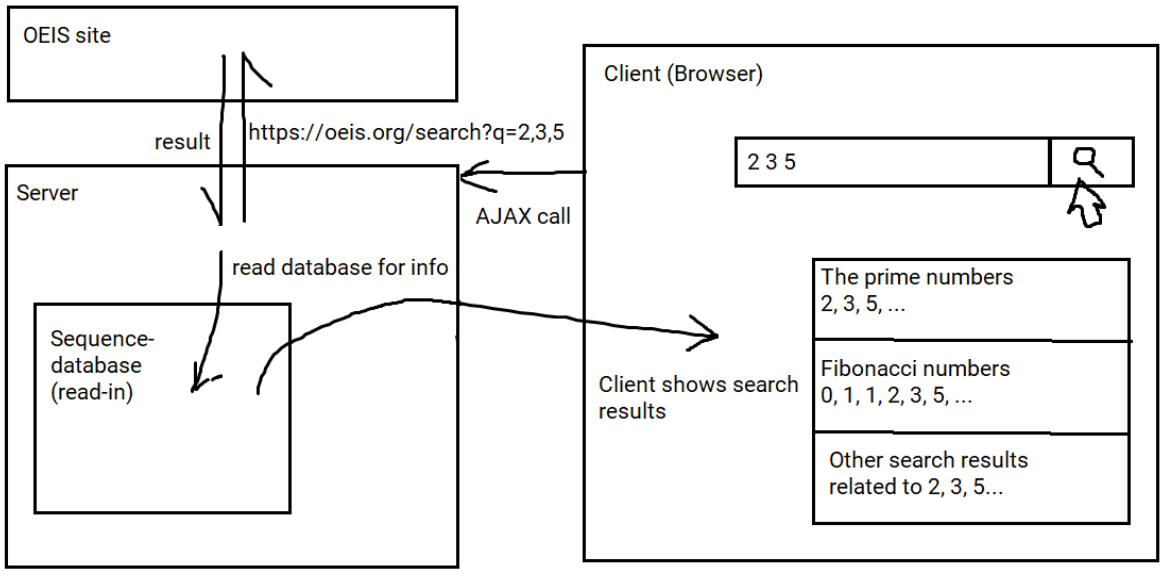


We use AJAX (Asynchronous JavaScript and XML) to communicate between the client (browser, JavaScript) and the server (Python). Every time the user chooses to lookup some sequence or number, the pertinent sequence's A-number or the number are sent as the input of the AJAX call. Our server will then lookup our read-in database, which are stored as Python dictionaries in the server's RAM and have high-performance lookup, and return the requested data.



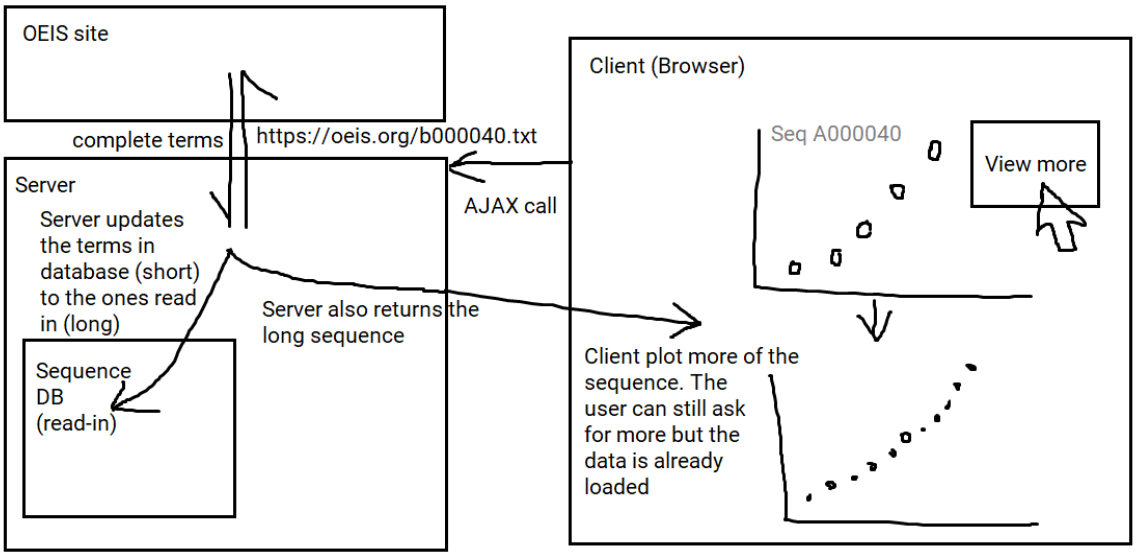
The Search Function

We've made use of the search function from the official OEIS site (https://oeis.org). The official OEIS' search function allows queries of sequences by both the names and the terms, which is very useful. Since sending HTTP requests to the OEIS doesn't take long, our server upon receiving a search query and internally accesses the OEIS for the search result data since OEIS has such an amazing search/ranking algorithm:

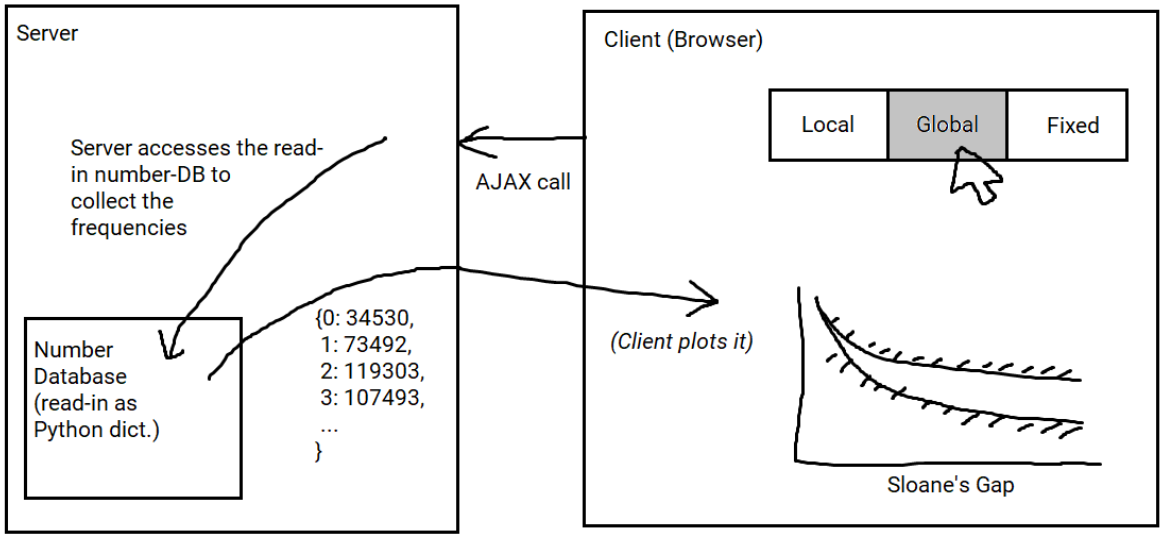


Other Related Data

The OEIS, by default, has only included the first few dozen terms (maximum) for each sequence. This is a nightmare for “prime-number enthusiasts” like us if we can only see up to the first sixty prime numbers. Actually, OEIS does have more terms recorded for each sequence (up to 10,000 terms) but they just stored them in separate text files. However, this allows the users to view more of a sequence if they want to: the client makes an AJAX call when the user’s requesting more terms of a sequence, and the server downloads the data from OEIS and updates the local database:



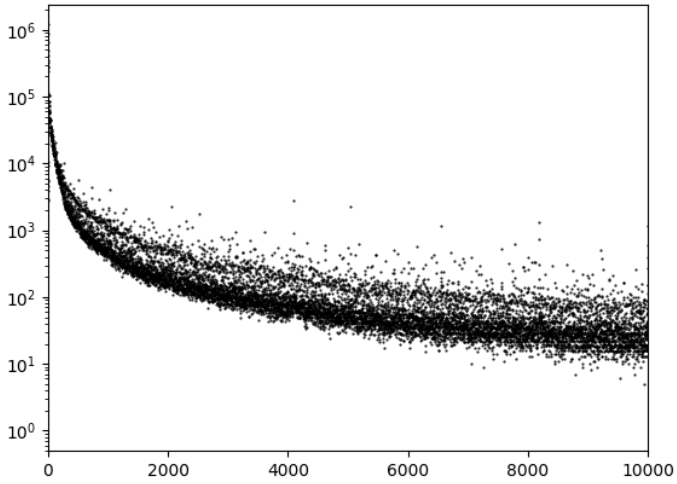
In our global-view, we have our main plot showing the Sloane's Gap, where every data point represents the number of sequences a number has appeared in; for example, the number 2 has appeared in 139205 sequences, and that'd correspond to the a dot located at (2, 139205). When the user switches to global-view, the client will just send an AJAX to request the frequency of each number, which the server will then return:



Exploratory Data Analysis

What visualizations did you use to initially look at your data?

- The Sloane's Gap is our initial way of looking at the data, which is also one of the inspirations for us into this project. It's a simple scatter plot that plots each number to the number of OEIS sequences that number has appeared in:



What insights did you gain?

- Clearly, this indicates that not all numbers are "born equally," or at least, they all get different amounts of attention from the mathematicians. In this sense, some numbers are more interesting than others.

How did these insights inform your design?

- This inspired us into visualizing the sequences in much more different ways because we, or anyone upon seeing the Sloane's Gap, would like to know what features numbers have to make them interesting. This also reminded us to come up with more statistics about the numbers to visualize to have a chance of understanding this pattern.

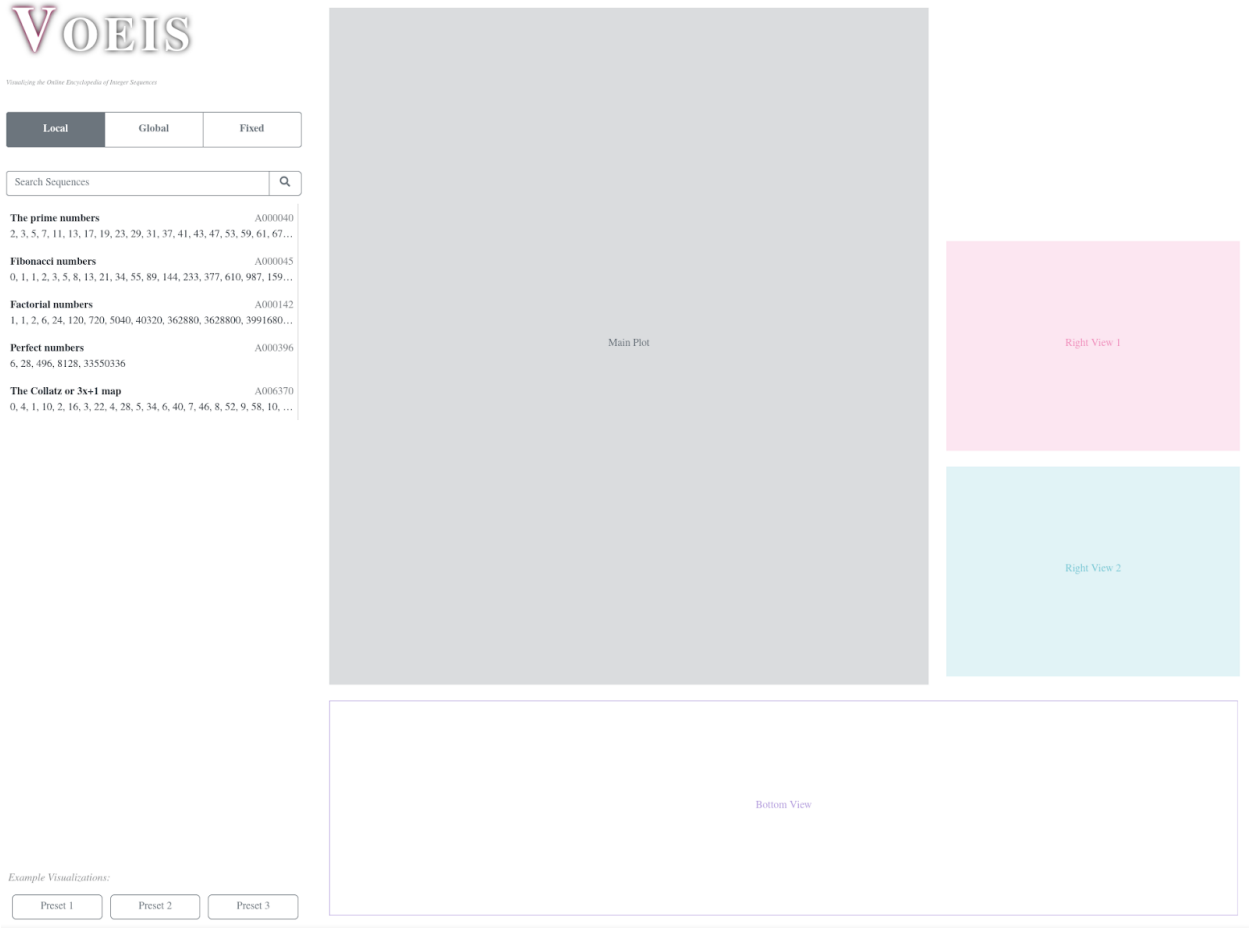
Design Evolution

Evolution 1.0

(view Iteration 1)

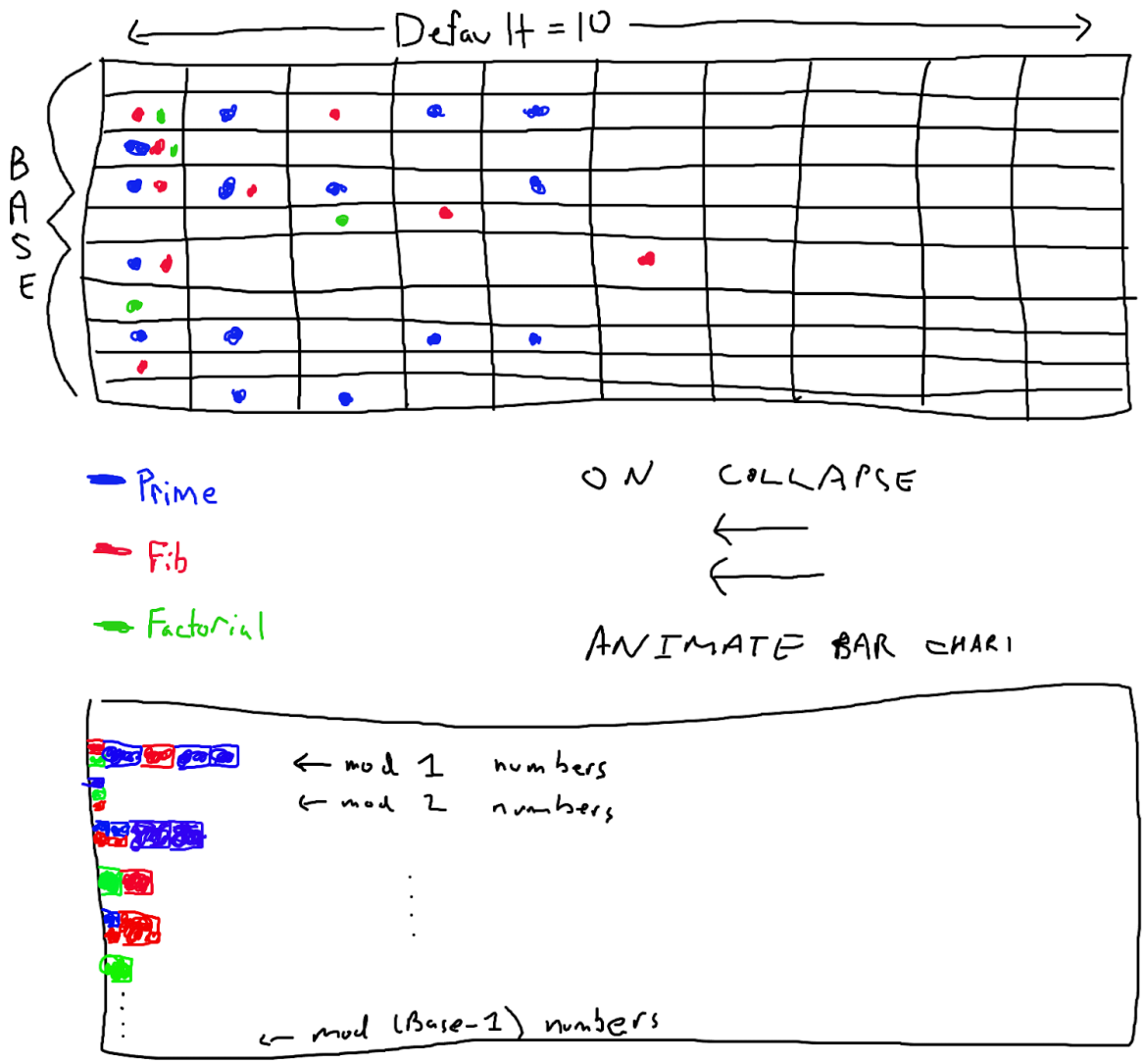
At first, we draw the basic layout of the three components of the VOEIS homepage, which consists of a search bar, main view, and secondary view. And then, we populate each component with a basic framework that requires no data such as the bottom to switch among local, global and fixed.

Evolution 2.0



(view Iteration 2)

In the iteration 2, it has the search bar view being implemented with title, and the buttons for local, global and fixed. And it is ready to implement the various different visualizations in the main view and the secondary view.

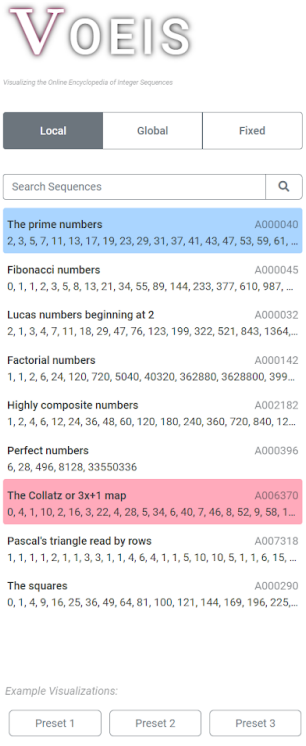


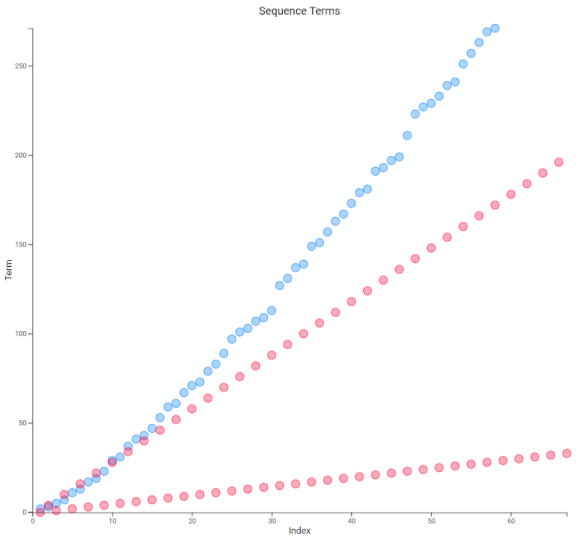
One of the ideas for the visualization of the local view is to have a matrix table, and the number of cells is the base. For instance, if we have a base-10 sequence, we would construct a matrix table with 10 rows. And each cell will be populated with different number sequences in that value differentiated by the different colors. And to make the view more interactive, we also offer the option to collapse the matrix table by turning the matrix into a horizontal bar chart so we can observe the pattern of the different number sequence with the help of colors.

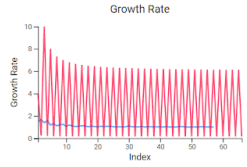
Evolution 3.0

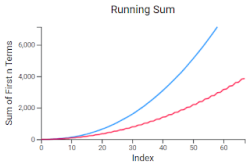
In the iteration 3, we have significantly implemented various different designs into our project across the three views.

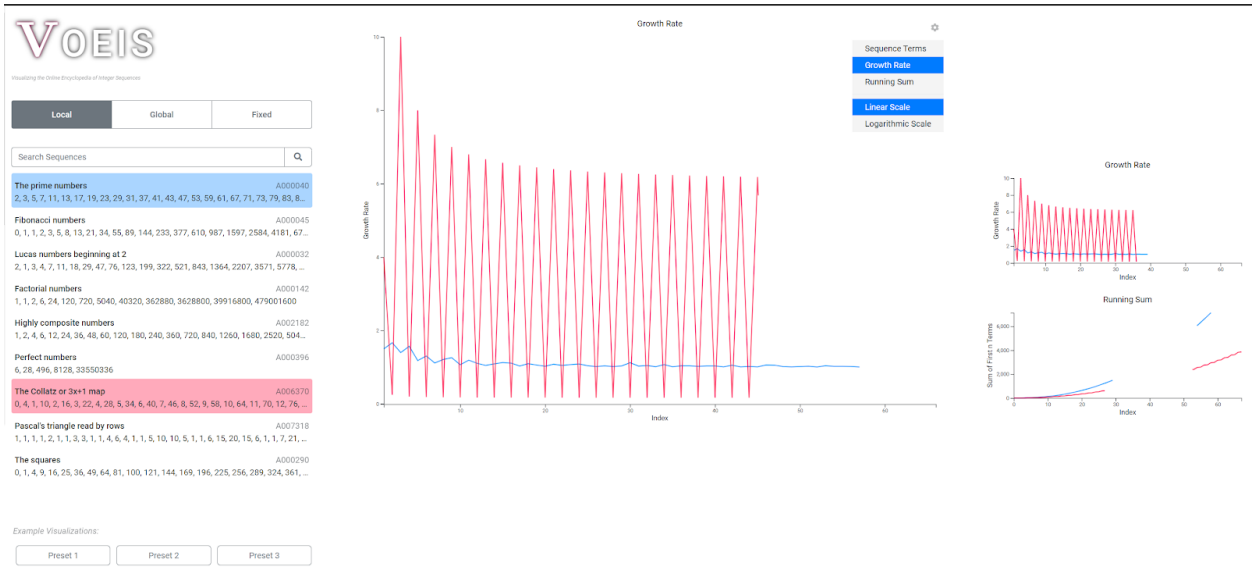
In the local view, we have implemented growth rate to the right view 1, and running sum to the right view 2. These two both have the similar 2D layout as the main view, and the data can be derived from information contained within existing ones. In this way, we can observe the pattern by transforming the data into a form more useful for the task at hand. At the bottom of the local view, we plan to implement a unique matrix representation (the matrix table described above) to minimize clutter for large and dense graphs of sequence into more dense and abstract representation of the data. We will go into the details by the next iteration of our design evolution.





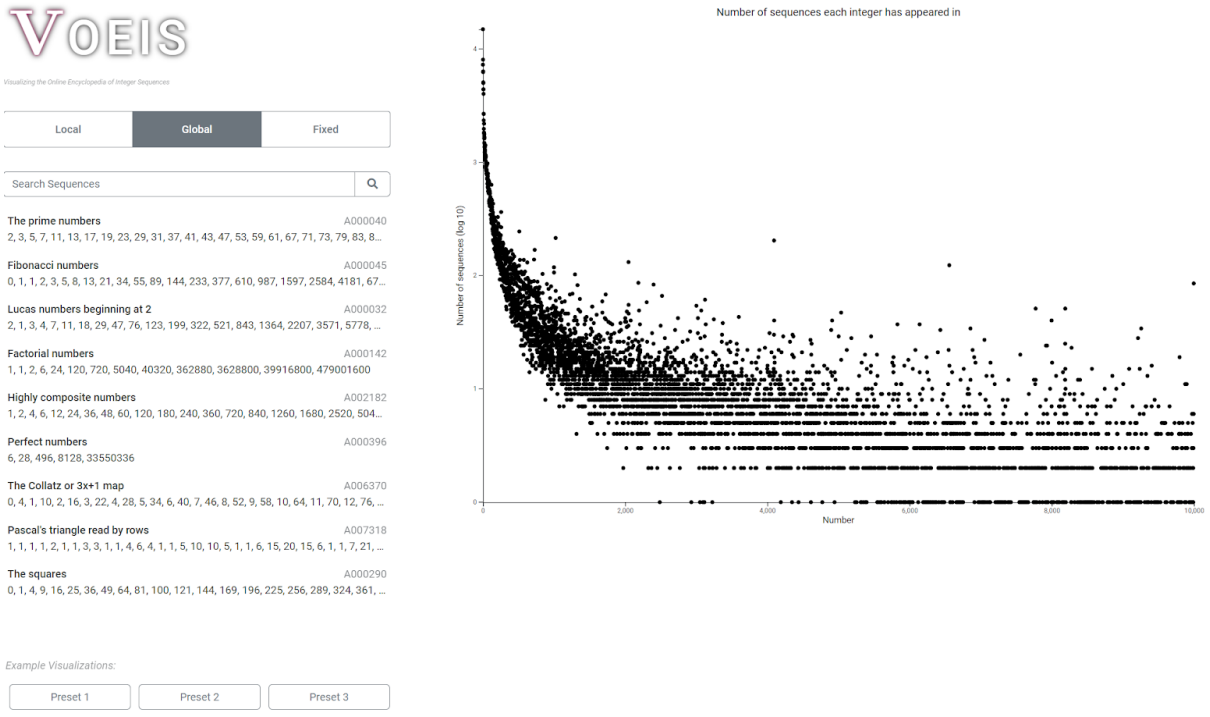




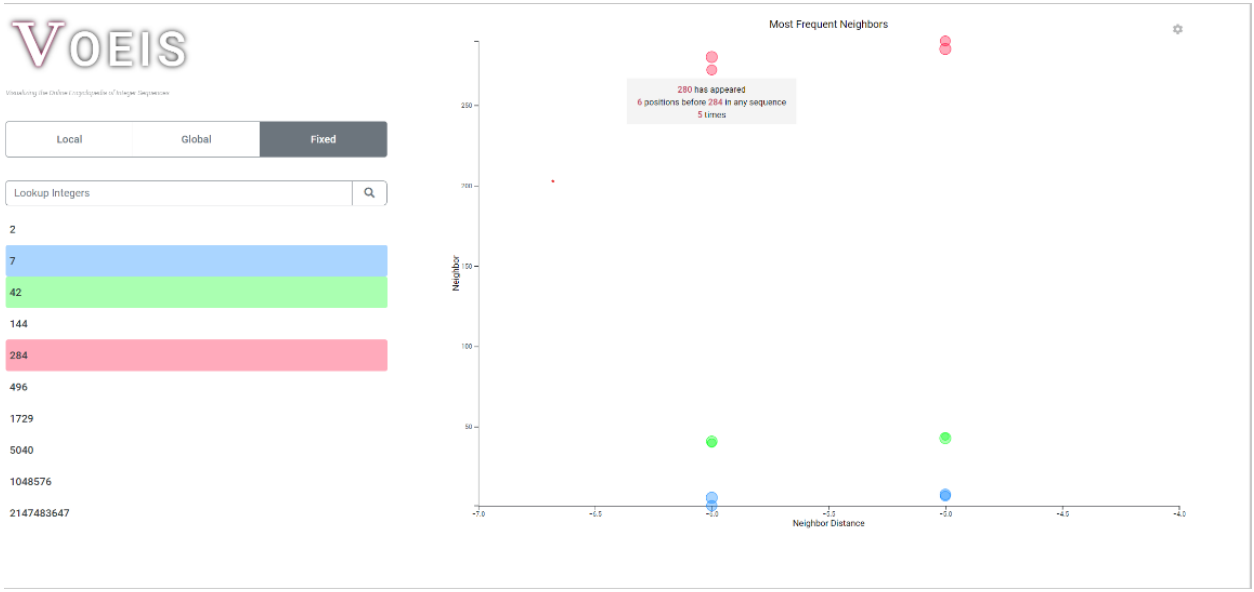


We also have the option to switch the main view to the growth rate or running sum so the user can focus on either the raw data, the growth rate, and the running sum with more precise visualization representation.

In the global view, so far, we have only implemented to draw the sloane's gap. But the end goal would be to let the user select various sequences and the corresponding scatters will be highlighted with different color hues to highlight specific sequences among other sequences in the sloane's gap graph.



In the fixed view, we are interested in exploring any given number and its neighbors in the sequences. For instance, here we have three numbers: 7, 42, 284. And the graph will show up the most frequent neighbors. The x-axis indicates how far the neighbors are and the y-axis indicates how frequently the neighbors show up.

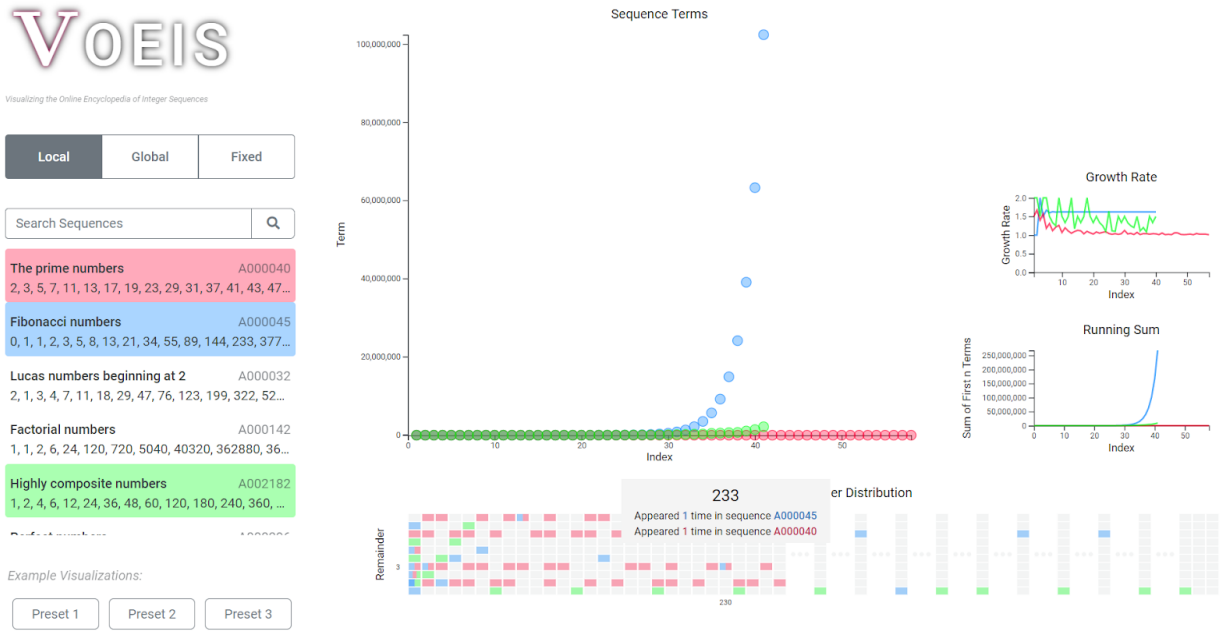


And when we hover onto a scatter circle, it shows the number that has appeared 6 positions before the number we picked in any sequence n times.

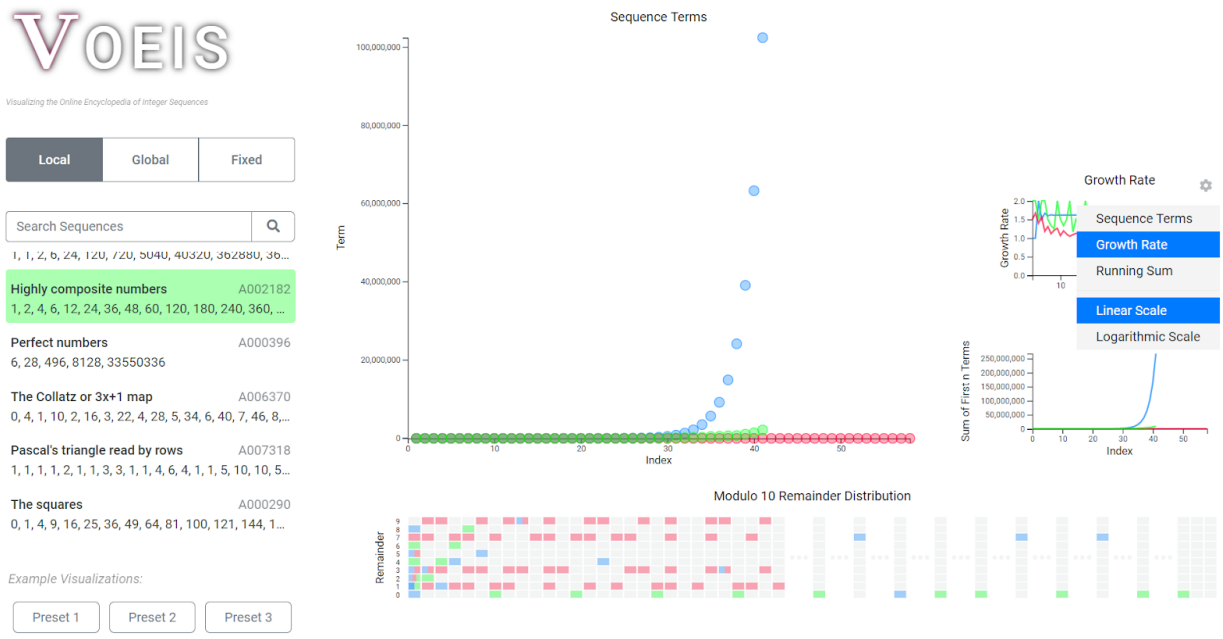
Evolution 4.0

In this iteration, we have significantly increased the interactivity of our visualization.

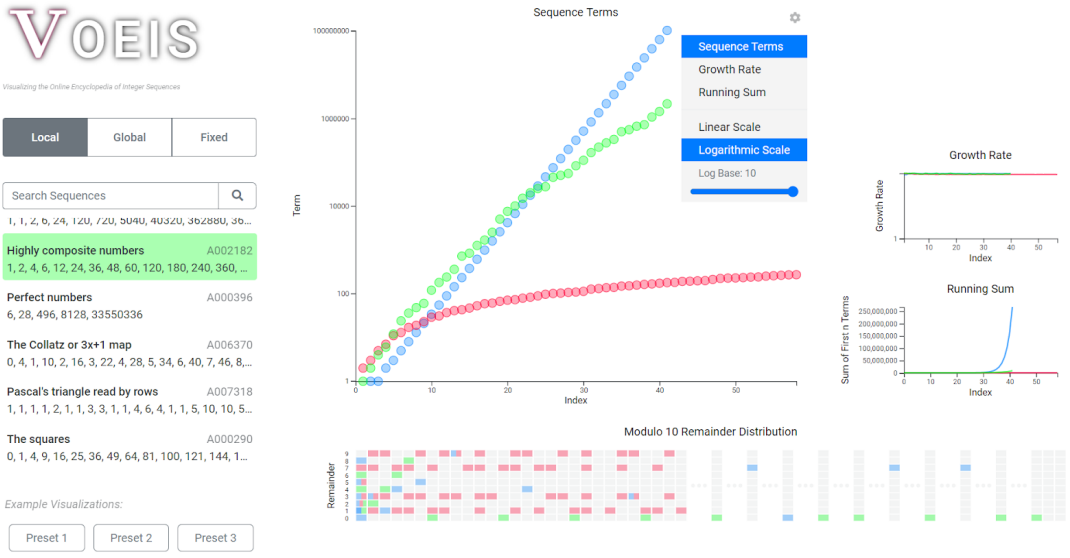
For the local view, we have implemented the matrix table idea mentioned earlier.



On top of that, we have implemented animated transitions across all the views so the user has better experience viewing the visualizations. When the hovers onto a blockin the matrix table, it indicates the exact value, and how many times it shows in the sequence. If there are multiple sequences in a single block of rectangle, it will be separated by the color hue, which is encoded to that specific sequence across the search bar, the main view, the secondary views, and the matrix table.



Also, we have implemented menus for the two secondary views so that the user can customize how they want to access the plot as they see fit. And we also implemented a scale slider for the user to better see the pattern of different sequences in various scales.



For the global and fixed view, we didn't implement a significant proportion. The major update for the global view will be to implement the story telling part, where the user can

scroll up and down to view the different narratives. And for fixed views, we will be focusing on implementing the secondary views and the bottom view to reflect the neighboring integer values for a fixed value.

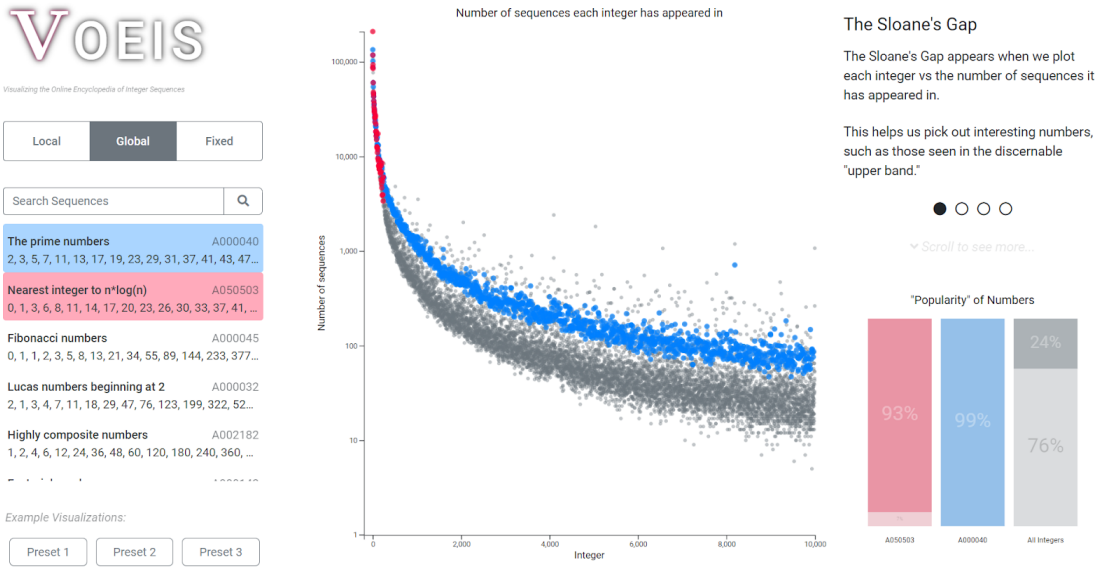
Evolution 5.0 (final iteration)

Overview -

In the fifth and final iteration, we went in detail on how we can leverage the various different kinds of visualizations other than bars and lines to visualize the number sequence. We argue that the quantitative data such as numbers should be encoded other than the two attributes using a line mark with the vertical spatial position channel for the quantitative attribute and the horizontal spatial position channel for the categorical attribute. We have also implemented animated transitions to stimulate the user's attention and to make sure the user is maintaining the context. In our testing of the project, we found out that the transitions are useful for the people to track everything that occurs if many items change in different ways all over the frame.
And we also managed to save the selected sequence even when switching the views.
We think features like this are *NIFTY*.

Global View -

we have mostly stayed with the same plan as the project proposal, where we would have the main view at the center, and the storytelling on the right hand side. We have also decided to shift our default main page from the local to global. This is because we think the narrative storytelling aspect of the global view attracts the viewer's eyes the most. And we think a dense overview with scripted storytelling can provide orientation and context to the users with or without related backgrounds.



And above is our final iteration of the global view. We have managed to integrate the story telling aspect into the user's interaction with the visualization. We decided to let the user play around with different sequences while introducing the user to some of the most iconoclast among the number sequence.

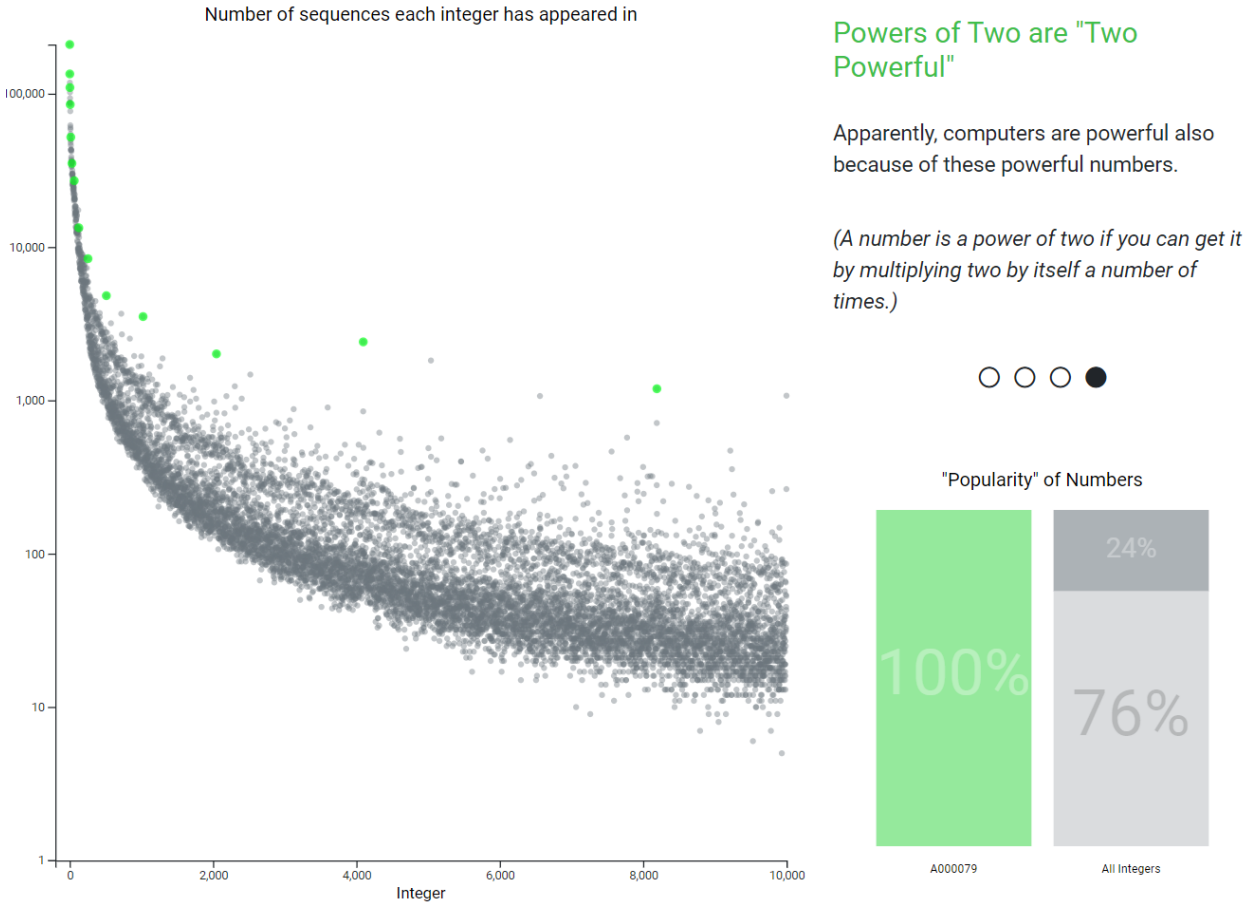
For the Sloane's Gap, our initial plan was to only have static visual encoding. But after implementing the static version, we thought that the user probably would not be very interested, so we decided to let the user have the ability to select the sequence, and the selected sequence in the scatter plot will be highlighted with proper color hues. On top of that, the 'popularity' of numbers will now not only show the distribution of all the integers, but also specifically the distribution of that sequence with the same color hues. The views are linked together with color highlighting.

And the user has two ways to switch to the second slide: either clicking the slide dots,

or simply scrolling down via a mouse.

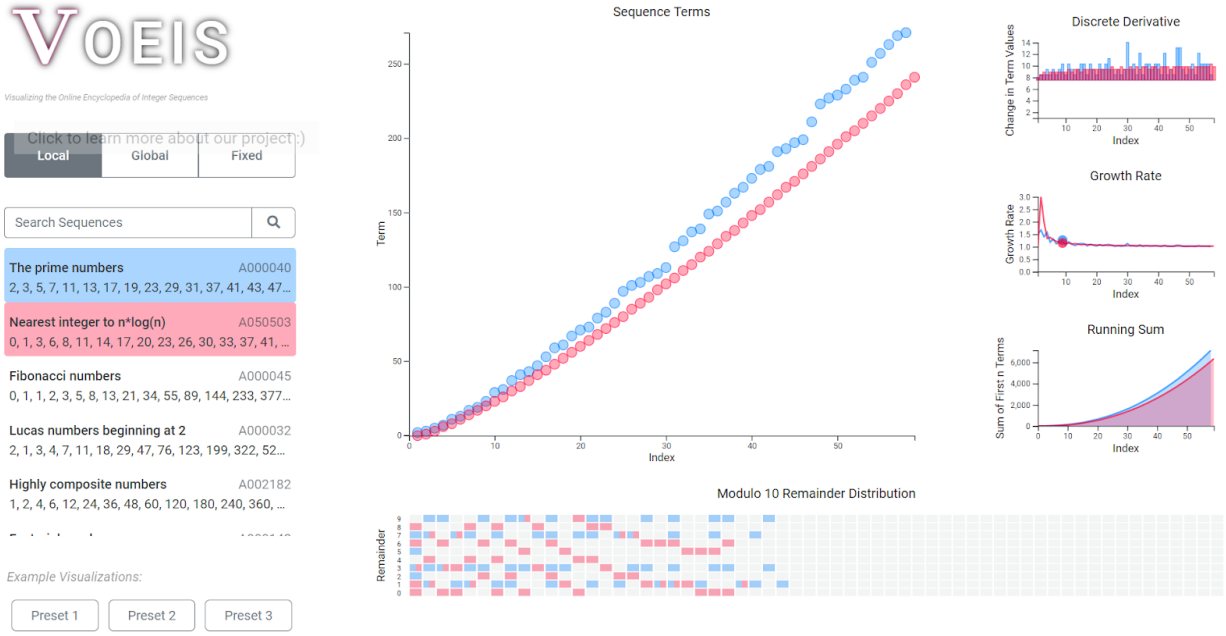
For the rest of the story telling, we focus on each individual number sequence: prime numbers, highly composites, and power of two. The scatterplots, the "popularity" bar, as

well as the title of the sequence are all encoded with the same color hue to be linked together as mentioned above.

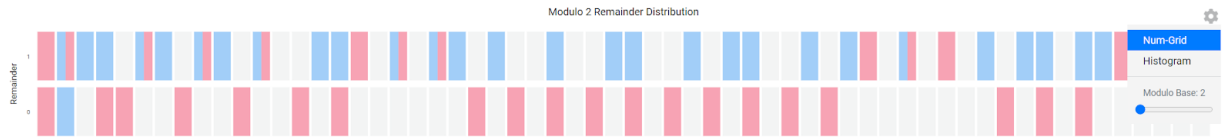


Local View -

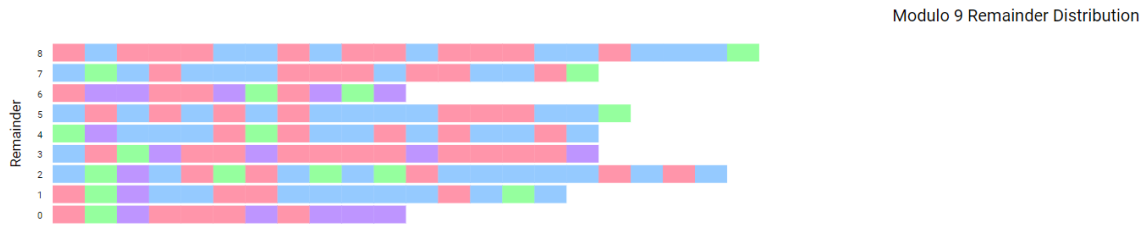
We have added a new view to the right: discrete derivative. In this local view, we have now included a scatter plot, bar chart, line chart, and area chart. Area is telling us the sum, which is perfect for the running sum of a sequence, the line chart and the bar chart are telling us the discrete value, which are suitable for the growth rate, and discrete derivative, and scatter plot is disclosing the cluster discovery in detail. Therefore, we think this comprehensive coverage of the same data (again using the same color hue to connect the different views).



And the last major update we implemented besides the discrete derivative, we also bolstered our matrix table by allowing the user to edit the modulo number and histogram.

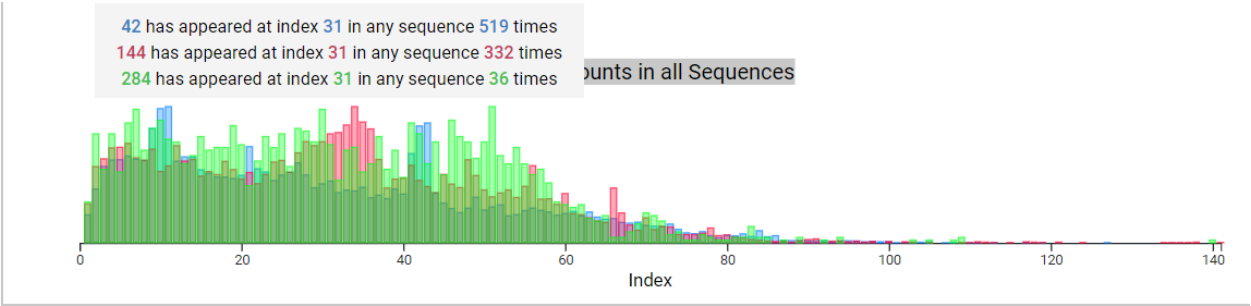


On top of that, we also implemented an option to “collapse” the matrix table into a histogram so the user can see the pattern of the trends and distributions of different sequences in a macro-view.



Fixed View -

At last, we also made some major changes to our fixed view.
At the bottom, we added a bar chart of index counts in all sequences of an integer. It shows the index of that number in a sequence the number of times. We think a bar chart is a decent choice because it indicates the distance of the index, and the vertical spatial position shows the number of times.

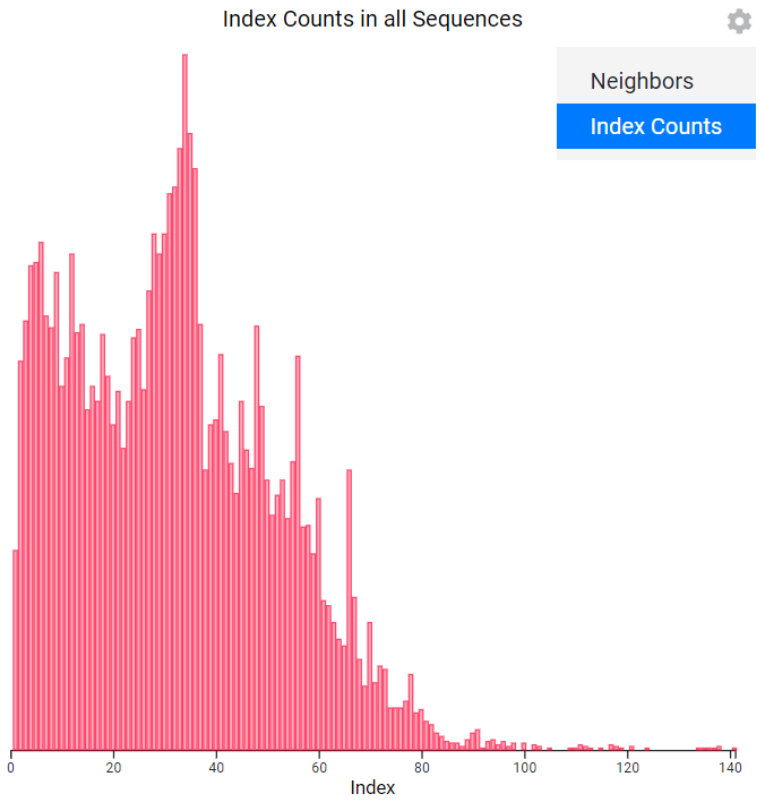


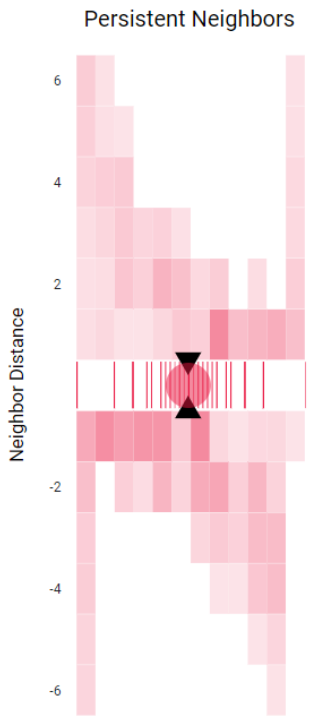
However, we think this choice of bar chart is probably not the best way. When the user selects more than two, the bars overlap each other. But we think the goal of this visualization is not to compare and contrast among various integers, but to observe each value across different indexes. Therefore, we are willing to tolerate these overlapping bars.

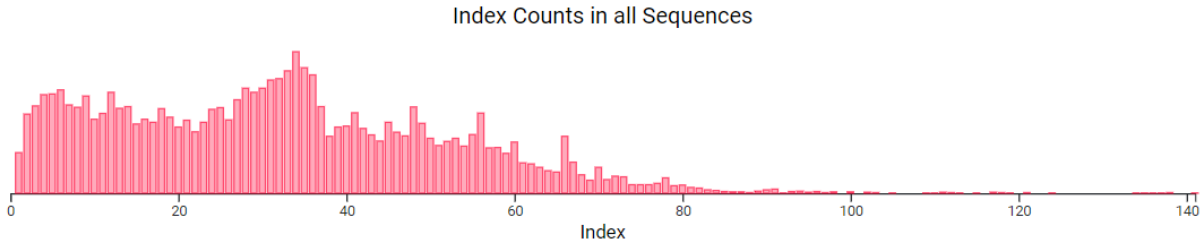
At the right side, we came up with a unique idea to demonstrate the persistent neighbors,

The horizontal spatial position indicates the nearby integer values, the vertical spatial position indicates the persistence of the neighboring integers. And since the saturation interacts strongly with the size channel, we decided to use it to reflect the number of times the persistent neighbors occur.

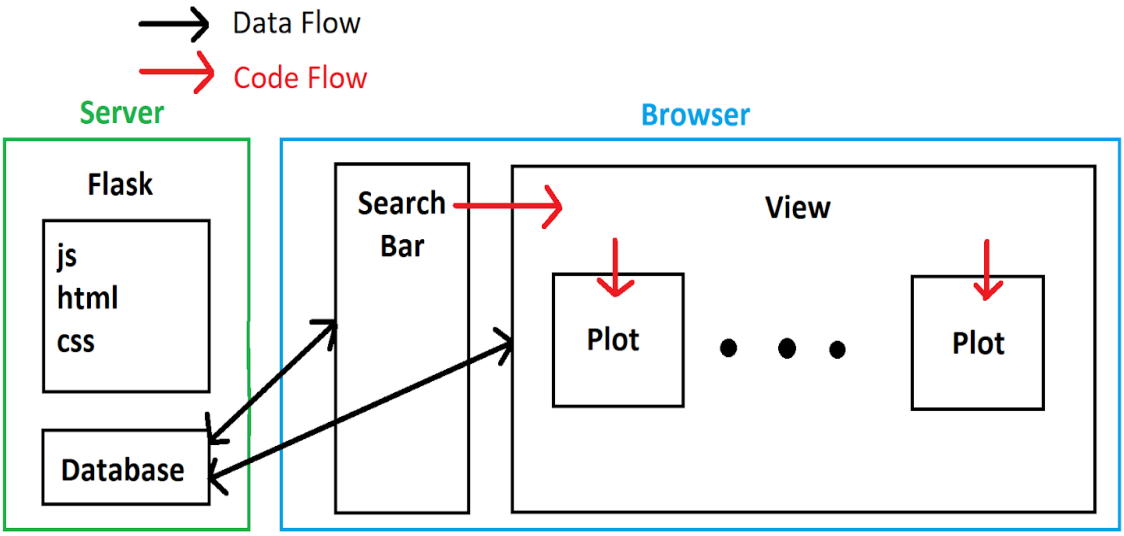
For the main view of the fixed view, we also added a way to switch the *most frequent neighbors* to a bar chart of *index counts in all sequences*. Again, we want the user to observe the frequency of the index counts with less dense vertical spatial position.


Index Counts in all Sequences
Neighbors
Index Counts
0
20
40
60
80
100
120
140
Index


Persistent Neighbors
Neighbor Distance
6
4
2
-2
-4
-6


Index Counts in all Sequences
0
20
40
60
80
100
120
140
Index

Implementation



For the browser part, our project is divided into search bar, view, and plot demonstrated above. So far, we've implemented a prototype with just scatter-plots. But we have made communication protocols for the server and the browser fully functional. The search bar is fully functional. After the milestone, we will be working on sending the data dynamically from the search bar to the view, and from the view to the plot. And we will be implementing the various kinds of visualizations that have been discussed so far.

Code Refactor

After a very lengthy discussion, we settled on doing an overhaul of the code so that it is generic and easy to add visualizations. In the end, we settled for the following code structure:

db-handler.js

This is a global static class that allows us to query the database. It has the following functions:

1. getNumber(number): gets information of number. This includes how many times it appears in OEIS, how many sequences it appears in, and its top neighbors with their counts.
2. getSeuquence(aNum): gets information on a sequence. This includes A Number, the title and description of the sequence, and the actual sequences itself.
3. searchSequences(query): query the OEIS database. This is usually used when a query error happens on our end or we don't have the sequence stored. This is called by the search bar.
4. getMoreofSequence(aNum): used to get more integers within a sequence. We query the OEIS database since we only store a max of 100 integers per sequence on our end.
5. getSloanes(): load the Sloane's Gap data located on our local server.
6. contactDB(url, input): the main accessor to our database.

plot.js

A class created by the View class to represent a Plot and all information that goes along with it. This contains expected and obvious code for a Plot class, such as clearing the plot, drawing the plot, and updating the plot.

plottable.js

A class for the actual item(s) being plotted. This class just describes each plottable statistics with the following fields:

- rawData: the rawData being plotted
- xExtent/yExtent: the domain values for X and Y
- x/y: the data for x and y. Sometimes the x is used to transport complicated data and has more to it than just x points
- magnitudes: magnitudes for each data point/item. This can be used to scale circle radii, box widths, etc.
- Length: length of data, i.e. how many data items there are

plot-option.js

A class for possible options. The plottable class is a class for the current data being plotted whereas a plot-option is the framework that is setup if we want to create a plottable instance for a plot option. In essence, plot-option contains raw metadata and pointers on how a plottable can be constructed. For example A histogram chart and a discrete density chart may both be plot options of BAR, but they will have to own metadata and function pointers on how to compute data. However, they share the same underlying structure that they are both plotted with BAR. This class contains the following fields:

- menuTitle: display text of plot to be displayed on drawdown menu
- plotTitle: the title of the plot
- plotXLabel: x Label of the plot
- plotYLabel: y Label of the plot
- function pointer 1: the function used to computer the data for the plot to use explicitly from the raw data
- function pointer 2: the function to plot the axis
- function pointer 3: the function to generate the text of the info panel for each plot item
- boolean: this is set to true if the plot needs to be redrawn when the window is resized. This is used by plots that explicitly use pixel values rather than percentages.

view.js

A class that was essentially the leader for the plots in the leader/follower setup. The view would instantiate the Plots, pass it data , assign Plot metadata (such as title, labels, and menus), and give the plots pointers to functions on how to compute everything needed: Plot axis, data, and visualization. The view was also responsible for saving history (items selected) on a context switch and loading whatever may have been present on the newly selected view.

functions.js

This housed all the Functions used to compute almost everything in our visualizations. Any other functions, such as utility functions, are kept in util.js. Most of the programming that is similar to our homeworks is within this file. There are two types of functions. One is a calculating function, which converts a sequence- or number-data item into “plottable” format, such as the direct X- and Y-values. The other is a plotting function, which takes the formatted plottable data and plots it with different visualizations.

Our current version contains these calculating functions:

- sequence: directly translates the sequence’s terms into X- and Y-values.
- derivative: calculates the discrete derivative (differences) between the sequence terms.
- growth-rate: calculates the growth between terms.
- running-sum: calculates the running sums of the first terms.
- index-counts: directly translates the number of occurrences at each index for an integer into coordinates.
- neighbors: directly translates an integer’s popular neighbors into coordinates.
- connections: calculates the persistent neighbors an integer has.

Our current version contains these plotting functions:

- area chart.
- bar chart.
- bar chart (for number-popularity).
- num-grid (and its histogram visualization).
- heat map.
- line chart.
- scatter chart.

search-bar.js

The class that hosts anything related to the search bar. This includes contacting the database, storing history items and presets, and loading search results.

The search bar is the “root” of our website, as in our search bar is the most active element in the webpage, where the user’s actions here drives the rest of the webpage. There are two major functions our search bar provides: search and history. A search is triggered when a query, and any search result that the user selected will be added into the history for the user to easily compare. There are a handful of default history-items involving both sequences and numbers, which we included because we thought they were significant and representative of the OEIS.

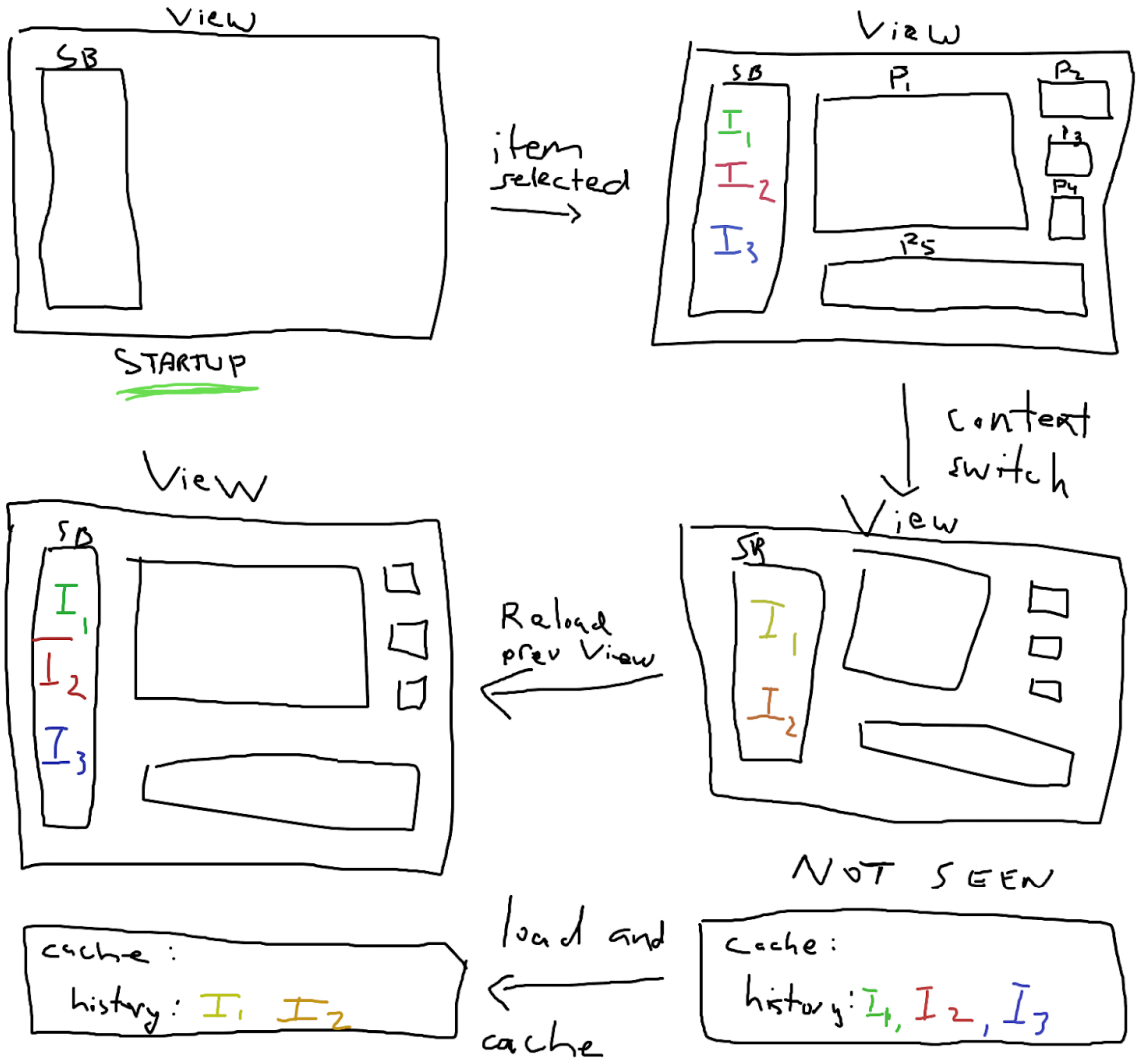
info-panel.js

The class that hosts anything related to the info panel used whenever an item is hovered over. This includes where to display the info panel, what to display, and to hide when a mouseout event occurs.

script.js

The start of the js code. This spawns the leader View class and a search bar.

The following is a diagram taken from one of our Discord discussions on how the leader/follower situation and caching happen:



As seen above, only the view and search bar are spawned at start up. It is not until the user selects something that the plots are instantiated. From there, we always cache user selected items on context switch and load any for the appropriate view if a cache for that view exists.

Evaluation

From our visualization, the user can find patterns within certain sequences by using our local-view, or see the relationship between the sequence and all sequences in OEIS by using our global-view. We think our visualizations are reasonably organized and informative so that anyone could take a thing or two away from it. More specifically, we believe our visualization achieves the following objectives:

- Clearly and aesthetically visualize single or multiple sequences or numbers
- Highlight the importance of some number or sequences, especially when done against Sloane's gap
- Help the user understand why integer sequences are more interesting than just a list of numbers
- Highlight local and global statistic of sequences which allows the user to determine some sort of significance to each sequences
- Allows the user to investigate any recorded integer sequence that has been peer reviewed and passed as "mathematically interesting". This is done by OEIS.
- Lets the user enjoy math in a fun and interactive way!

However, due to time constraints, we were not able to

- Allow the user access to more visualizations
- Allow the user to change the speed of the animations
- Allow the user to input their own sequence (this is important when the sequence they care about has not been accepted by OEIS).
- Let the integers default to more integers. This would require a larger database and therefore more overhead.

Overall, we believe the project to be a success and plan to improve on it in the future.