Scenario One

Basic Data Structures

Linked List
Array
Hash Table
Binary Tree

Advanced Data Structures

Skip List (-> Balanced Binary Tree)
Rolling Hash (-> sed command too?)

A skip list in practise

A skip list provides several routes through a list. This enables searching for nodes to be somewhat streamlined by starting at each heads link (top first) and going through each until the desired node is found (tracing back one node and down a level when the comparison is over-reached).

****

A scenario for this could be intricate file editing by storing commands in each node to run on a file, these commands must be run in order on the files. The application that uses the skip list would allow a user to search for the command they need and run it. The command would be a maths function that would rely on running other maths “equations” before hand to get to the desired output.

You now have a in memory version of a file (long string). This can be loaded into the jump rope data structure. Perhaps to version of the file can be loaded and worked on to then compare against. Use a rolling hash to compare.

****

A second, and perhaps better and more relevant, scenario would be taking data (from a file or formatted data) and structure it for a faster search and display for a user.

The structure is a linked list of countries. Countries are a finite dataset meaning that you have a maximum defined. Knowing this manages exceptions for a search on countries in the list.

Each country node references one or many skip lists (named by it’s data type, like GDP, Price stuff etc).

Given a skip lists range of node heights the accuracy of graphs can be managed for the data. Level one for the most accurate, level 1+n for ever increasing estimates for graph shape (line graphs).

Scenario Two

Initial thoughts

Changes are made to results as the user makes selections. Can the interface keep up with rapid selections if making calls to retrieve data. They could be queued up but that would have to make sense.

```
onclick(send user selection state to queue)
While (command = queue.pop)
{
        command.run
        refresh data.
}
```

If the queue is measured to be too long cut x number out to avoid calls to datasource. That could mean using a different data structure to managed the commands.

Application State

Each component should hold its own state.

Application state is an aggregation of each components states.

Integration testing would be good for these components.

Application UI Behaviour

******

Modal window to deactivate the interface whilst the selection updates the UI (i'e. Request data from the server based on selections made).

******

A "heart beat" pattern to make requests to the server based on the current user selection at that time. (I.e every 5 seconds send a request to the server to update the UI with result estimates)

******

A cut down interface. 1. Data that can be selected is used in different tabs. 2. A user could have preferred options that they can select (not preferred options might have to be presumed to be always "unselected")

Final way. Update button, user updates their own selection once they have selected their criteria.

Scenario Three

There are four points of data that when combined are unique: **Country**, **Indicator**, **Year**, Amount. In **bold** are the points at which the data can be pivoted around.

From these ‘headers’ we can pivot and manipulate the display of the data.

In Memory. The data has to be stored in such a way it can be accessed and manipulated with ease. This may cause a greater use of memory however usability must be taken in to account and a compromise found.

- Immutable T uses a binary search tree in the background, good for comparison / LinkedList where every pivot moves through the data and sets groups based on each node/item.
- Prodedure
    A. Predicate to filter
    B. Grouping by
- Amount of data loaded into memory
    - Filter master data before pivot/transform.

Database

- Database technology
    - MSSQL indexable structured data.
    - Indexed document data, fast retrieval on search but higher level data. Data/results can be returned in chunks (paged) so limit usage.
- MSSQL
    - Pros
        - Structured data
        - Mature technology
        - Widely documented programming language (TSQL)
    - Cons
        - Structure is rigid and changing it can be laborious
        - Limit on database size or adding more DBs to a ‘cluster’ could be hard
- Considerations
    - Cost
        - Training
        - Set up
    - Maintenance
        - Technical debt, where is the tech going and what could it be used for in the future
    - Suitability
        - Making maximum (or near maximum) use of the features of a datastore. If not is the cost worth it.
    -

https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.