Arpit Agarwal | Google Summer of Code 2017, Gene Genomes Variations (Ensembl, EBI)

 May 27, 2017

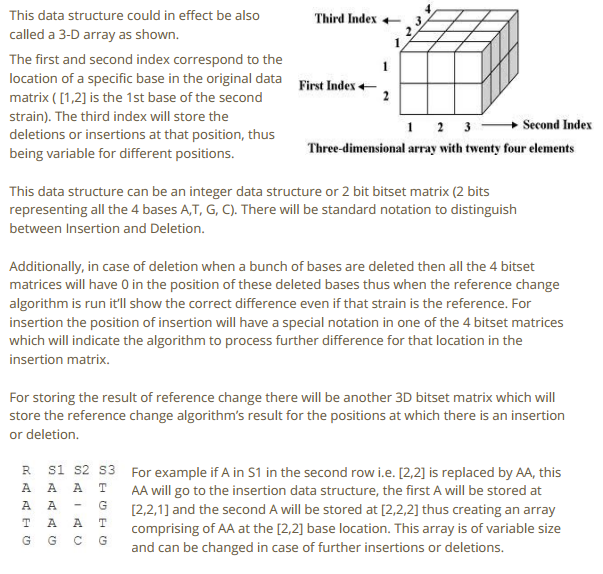
Handling Insertions/Deletion

Proposing a shifting algorithm

# Assumptions:

1. All Insertions and Deletions are made on one strain (i.e. vertically) and not on multiple strains (i.e. horizontally) in one go.
2. There will be limit to the data added or deleted in one go due to the limitations in processing and the limitations in writing and reading the data from disk (this will be tested on the servers and the limit will be decided/reported, also the limit should be more prominent in adding data as compared to deleting data)

# What we have till now –



This is what was proposed, this stands valid but has shortcomings –

## SHortcommings in the proposed algorithm:

1. This increases the processing and hence limits the extent of the dataset that can be processed on the go.
2. Requires more storage.
3. The data stored in the 3D matrix won’t be of any use because the algorithm used for SNP’s cannot be used directly on this kind of data structure.
4. This stored 3D data structure cannot be directly converted back to VCF format which is required for - i.) Storing the processed data on the server. ii.) For converting the data to gene tracks for visualization compatible with Genoverse.

## Solutions to the shortcommings:

* For points 1. & 2. - The increase in processing and storage space might render the whole algorithm incompatible (i.e. by increasing the time complexity from O(N) to something more time consuming) for larger datasets (i.e. complete genome of a plant) but that does not hinder with our primary goal of generating visualization tracks for gene variations. Moreover, the process will still be valid for generating tracks of only SNP’s for large datasets.
* For points 3. & 4. - The algorithm used in SNP’s is based on the formatting of the gene matrix we get from the VCF file formatting and the gene tracks are also generated from VCF file formatting. Thus, all the data needs to be kept in such a format that it is either directly a VCF format data or can be converted to VCF format. The data structure (bitset matrices) can be converted to and from VCF file format but this 3D matrix as of now has no way of being converted to VCF file format.  
    
  This can be handled in two ways –
* Think of a method of converting the indel data structure to VCF format compatible data structure.
* Convert the 3D matrix data structure to SNP format bitset matrix.

The second method of converting the indel data structure to bitset matrix seems the most logical choice – because this will conserve the continuity of the process (the data could be further subjected to SNP/INDEL processing without any extra processing required), this will also solve the data storage problem as this indel data structure could be deleted just after the whole process is completed.

The proposed method of converting the indel data structure to bitset matrix is given below – it is named as “Shifting Algorithm”.

# Shifting Algorithm:

This algorithm mainly deals with translating the indel data structure to bitset matrices.

After we are done with making the indel data structure the data needs to be added/deleted from the bitset data structure at the right position and the matrix needs to be adjusted accordingly.

Basically, the algorithm will shift the rows of data per the insertion or deletions.

The 3D indel data structure will have the insertion details in the z row of [x,y,z], this data will be translated to the y column and all the other strains having insertions at those positions will be translated accordingly. For all the strains having no data in these newly added rows, simply a 0 (as was the convention for bitset matrix) will be displayed.

For insertions, all the data below will be **shifted** by the number of index equal to the number of insertions, this can be easily done in a single pass algorithm which is O(N).

All this change in index will need to reflected at all the required positions, not sure of all the places where the index numbers might be required because I haven’t started writing the code till now but I am quite sure the data will be required, thus, the data of the index change due to insertion will be kept in a separate file system, this will be a 2 column table where the first column will contain the starting index where the insertion started and the second column will have the number of bases added. (this can be any simple data structure – SQL works).

For deletions, all the rows of data deleted will simply show a 0 in place per the bitset matrix convention defined earlier.

As with insertions the changes will be saved in a SQL table for further use.

For deletions, we can either **shift** the data upwards or let it remain there (with 0 inplace of the deleted data) – this can be decided at the time of application.

As we can see that we are mainly shifting data according to insertions or deletions in the main genome data matrix, hence, the name “Shifting Algorithm”

Note: The index change of all the data suggested in case of insertions has a known algorithm I read some time back, that algorithm is proven to be lower bound (most efficient). I am trying to find the algorithm but am unable to recall it, I’ll send you the details of it once I find/figure it out.

# Conclusion:

This is a rather simple process to handle insertions and the deletions, there are still be some short comings/things which can be thought of better which I am working on. Mostly I wanted to deal with the 3d Matrix dataset itself but soon I realized that it would be better to translate all the data to bitset matrix.

This is a descriptive detail of the work I did, I have been working with pen and paper on creating a example for indel, the reason why I don’t give it here is because It’s not full proof and it was taking a lot of time to chalk out that example here, we can discuss the example over a call.