Build the A-bruijn graph

Assignment #2

Ans to Problem 1: Add vertex weights

I made two primary changes to my previous code to assign weight to each each vertex. First, in the data structure where I required to add new field to save vertex weight. Second, I required to update data structure building process so as to calculate weight for each vertex. Weight of a vertex is calculated by counting the number of occurrences for (k-1)mers corresponding to the vertex.

```
class kmerStore {
public:
        std::string kmers;
        unsigned int weight;
        kmerStore *right;
        kmerStore *left;
};
```

Ans to Problem 3: Algorithm for traversing the graph

As we already know optimally traversing the de Bruijn Graph is NP-hard problem. Thus we need to rely on some heuristic techniques.

Before traversing the De Bruijn Graph we can further reduce it by

- Removing all the low coverage vertices [Low coverage vertices are most probably present because of sequencing or assembly error]

For traversing the De Bruijn Graph, we can use Eulerian cycle algorithm in which,

- We start at any vertex v, traverse unused edges until returning to v.
- Every time the vertex is traversed decrease the weight by 1
- Repeat following steps until the cycle is not Eulerian,
    - Pick any vertex w along the cycle for which there are untraversed outgoing edges
    - Traverse unused edges until ending up back at w
    - Join two cycles into one cycle

As we know K-mers can be sequence from both the ends. So we can sequence one end, then turn it around and sequence the other end. The two sequences we get are 'paired end reads'. This technique can help in reducing the ambiguity in assembly.

Reference: Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Daniel R. Zerbino and Ewan Birney.