1. How I Implemented the Code

   At the beginning of my code, I need to include several libraries since I am using some STL functions. I then declared the vector of type int cur_best_clique and initialize it as the size of VERT_NUM+100, vector of vector of type bool conn and e with the size VERT_NUM+100, a multimap of type int named graph, and a set of type int named v. I the have a class MaxClique, and inside it there are 3 other classes: Vertices, ColorClass, and StepCount. Inside the class Vertices, there is a class Vertex to deal with each vertex of the graph.

   The class Vertex has several public functions namely set_i to set the number of the vertex, get_i to get the number of the vertex, set_deg to set the number of degrees, and get_deg to get the number of degrees of that vertex. The class Vertices has a private function desc_deg that will return true if the degree of the first parameter passed to that function is greater than the degree of the second parameter passed to that function. Then in the public section of the class, we first have the constructor that sets the size as 0 the first time, and allocate memory for the created array of Vertex named vertex. Then we also have the destructor, the functions del() to delete the created array vertex, sort_vertices() to sort the array vertex, initialize_colors() to initialize colors to the vertices, set_degree(MaxClique& m) to set the degree of a particular vertex, get_size() will return the size of the array vertex, push(const int val) will function like a stack STL's push, same thing as pop(), get_vertex() will return a vertex at a certain index, and top() will return the topmost of the array vertex.

   For the class ColorClass, we have a vector of integer type named arr and an int size in private, and in the public we have the constructor, destructor, the functions initialize(const int size) that will initialize the vector to be the size passed to the function as parameter, push(const int val) that will add an element to the end of the vector with the value val and increase the size, pop() that will pop the last element of the vector arr, clear() that will clear the vector, get_size() that will return arr's size, and get_arr(const int index) that will return the element of the requested index.

   As for the class StepCount, we have step1 and step2 in private, and in public we have the constructor that will initialize the value of step1 and step2 as 0, then we have the functions set_1(const int val) that will set the value of step1 as the value of the passed parameter val, get_1() that will return the value of step1, set_2(const int val) that will set the value of step2 as the value of the passed parameter val, get_2() to get the value of step2, and increment() to increment the value of step1 by 1.

   Then from the class Vertices we can create V, from class ColorClass we create an array of the ColorClass named C, and also clique_max and clique, and from the class StepCount we create an array of StepCount S.

   Then we also have several private functions for MaxClique namely:

   - bool connection(const int i, const int j) const
     - To return the value of e at position [i][j].
   - bool intersection_1(const int idx1, const ColorClass& idx2)
     - Calls the function connection and pass the parameter idx1 and idx2.get_arr(i) for I from 0 to the size of idx2 and returns true if the function connection returns true.
   - void intersection_2(const Vertices& A, Vertices& B)
     - Pushes the value of A.get_vertex(i).get_i() for i from 0 to the size of A minus 1, if connection(A.top().get_i(), A.get_vertex(i).get_i() is true.
   - void color_sort(Vertices& R)
     - At first, we have idx that is initially set as 0, max_num as 1, and min_k as the size of clique_max – the size of clique + 1. We then have to clear the first and second index of C,

then set k as 1. Then we have a for loop for I from 0 to the size of R, and inside the for loop we first set the value of index as R.get_vertex(i).get_i() and k as 1, then we increment k while the function intersection_1(index, C[k]) still returns true. Then, if the value of k is greater than the value of max_num, max_num will be assigned to the value of k since it has to have the maximum value, then we clear the C at index max_num+1. Next, we push the value of index at C[k], and if the value of k is smaller than the value of min_k, we get the vertex of idx++ from R and set the value as index. Outside of this for loop, if the value of idx is greater than 0, then we get the vertex at position idx-1 from R and set its degree as 0. Next, if the value of min_k is smaller or equal to 0, we set it back as 1. In the last part of this function, we have a for loop for k from min_k to max_num, and inside there is still another for loop for i from 0 to the size of C[k]. Here, we get the vertex from position idx from R and set its value to the value of C[k] at position i, also get the vertex at position idx++ from R and set its degree as k.

- void expand_dyn(Vertices R)
  - We first need to set the step1 in class StepCount of S[level] to the value of S[level].get_1()+S[level-1].get_1()-S[level].get_2(), and set the value of step2 in StepCount class of S[level] to the value of S[level-1].get_1(). We then have a while loop that will keep looping while the size of R is not 0. Inside the while loop, we have a pair of if-else statement, if the size of clique + the degree of R.top() is greater than the size of the current clique_max, then we will push the value of R.top().get_i() to clique, then create R2 from class Vertices and initialize it with the size of R. Then we call the function intersection_2 and pass the parameters R and R2. If the size of R2 is greater than 0, and if the value of S[level].get_1/++pk is greater than time_limit which is 0.25, then we need to call he function sort_deg and pass the parameter R2. Next, we will have to call the function color_sort and pass R2 as the parameter, then increment step1 from S[level]. We also need to first increment the value of level by 1 then call the function expand_dyn itself and pass R2 as the parameter, then decrease the value of level by 1 again. Continuing from the if statement, we have else if the size of clique is greater than the size of clique_max, we need to assign clique_max to clique, but before that we need to initialize clique_max as the size of clique itself, then update the vector cur_best_clique by using the STL push_back and add the value of clique_max.get_size(), which is the size of the current clique_max. Then we need to call the function del() for R2 and pop the clique. If the size of clique_max is greater than the size of clique+the degree of R.top(), then we will return nothing. We also need to pop R at the end.
- void _mcq(int& size)
  - We first need to set the degree of V to the value of *this, and then call the function sort_vertices() and initialize_colors(). Then we have a for loop for I from 0 to the size of V to set step1 and step2 of S[i] to 0. Next we just need to call the function expand_dyn and pass V as the parameter, then set the value of size as the size of clique_max.

Also several public functions for MaxClique namely:

- Constructor
  - Set the value of pk as 0, level as 1, time_limit as dec which is 0.25, V, clique, and clique_max as the parameter size.
- int steps() const

- returns the value of pk.

- void initial(const int size)
  - Initializes V by setting all its value from 0 to the value of size as i for i from 0 to size. Sets e as conn, initializes C as the size of size+1 and calls the function initialize and pass the value of size+1 as the parameter for every C[i] for i from 0 to size+1. We then also initialize S as the size of size+1.
- maxclique(int& size)
  - calls the function _mcq and pass size as the parameter.
- destructor
  - deletes C, S, and V.

In the main function, I mostly followed the template given, except I just insert the value of the input from the file, A and B, to v, and use the make_pair STL to insert to graph. I then set the value of conn[it->first][it->second] and conn[it->second][it->first] to true using the iterator it from graph.begin() to graph.end(). I then call the function initial from the class MaxClique and pass graph_size as the parameter to initialize V, e, C, and S. I then call the function maxclique from class MaxClique and pass the parameter cur_best_clique_size for it to be stored as the value of the best clique size. To output the best clique size in order, we need to sort the vector cur_best_clique and get rid of the duplicates since I insert the best clique size to the vector each time a clique is found. After that, we can already output the best clique sizes. As for the K-core, I just need to store the index of the vertices and its degree to a vector called k_core_index and k_core_degree respectively, and I push back their values using STL and for loop in _mcq function after setting the degrees and sorting it, then I just need to print it to the kcore.txt file.

2. Challenge I Encountered

   I encountered quite a lot of challenging problems while doing this project. At first I need to figure out how to run a program using makefile since I have not done it for quite a long time. Then I also got stuck when trying to figure out the algorithm to find the maximum clique. I watched several YouTube videos, browsed countless articles on the internet and read some example codes regarding clique and k-core, even though there was a time I had a hard time understanding the code. But eventually I managed to understand and implement it on my own code. However, after the first trial of running my code, I encountered several errors such as bad_alloc, segmentation fault, etc. I decided to trace my code line per line, figuring out where it might go wrong, until I successfully compiled my code with no problem and is able to find the clique size. However, my program still need a long time to run under my laptop's specs, so I went to the computer lab and ran it there. It still exceeds 3 minutes 10 seconds, but it still ran faster than my laptop and the testing environment was not the same as mentioned in the PowerPoint, so I am assuming that my program will run way faster under the TA's testing environment.

3. References That Gave Me The Idea

   https://youtu.be/nBrFC0STApo, https://youtu.be/RJvy8lId6nw, https://youtu.be/3RBNPc0_Q6g, https://www.geeksforgeeks.org/graph-coloring-set-2-greedy-algorithm/, http://insilab.org/maxclique/