Coding project: Chordal Graphs and P.E.O.

Group #4

Grader: Islam, Md. I.

Members:

1. Narla, Tejaswani
2. Nellore, Manvitha
3. Pammi, Sarika
4. Paruchuri, Prathima
5. Pentyala, Ravindra Babu
6. Podduturi, Manisha Reddy
7. Pourebadi Khotbesara, Maryam

Programs:

1. Create a random chordal graph (**Narla, Tejaswani**)
   Input: Interactively input number of vertices "n" and an integer "k" ($k < n/2$).
   Method: Create a complete graph with k vertices. Each next n-k iterations, identify a clique C of size k in the current graph and add a new vertex adjacent to all vertices of C.
   Output: a txt file giving an adjacency list of the chordal graph.

n, m
1: 4,6,7
2: 3,4,8,9
…

2. Create a random graph (**Nellore, Manvitha**)
   Input: Interactively input number of vertices "n" and a threshold number "t" between 0 and 1.
   Method: For every pair of vertices generate a random number between 0 and 1. If the number generated is larger than t, connect those vertices by an edge; otherwise, do not connect.
   Output: a txt file giving an adjacency list of the generated graph.

3. Naïve algorithm for checking if a graph is chordal (**Pammi, Sarika**)
   Input: a txt file giving an adjacency list of a graph.
   Method: Iteratively remove a simplicial vertex from the remaining graph if it exists until the graph is empty.
   Output: "No" if the process did not end with an empty graph. "Yes" if the process ended with an empty graph. In the latter case output also the obtained perfect elimination ordering.

4. MCS ordering of the vertices of a graph (**Paruchuri, Prathima**)
   Input: a txt file giving an adjacency list of a graph.
   Output: produce a vertex ordering using the MCS procedure.

5. Check in linear time if an ordering is a P.E.O. of a graph.(**Pentyala, Ravindra Babu**)
   Input: a txt file giving an adjacency list of a graph G on n vertices; ask the user to input a permutation of the numbers from 1 to n.
   Output: check, using a linear time algorithm, if that permutation gives a perfect elimination ordering of the vertices of G.

6. Fill-in of a graph along a vertex ordering (**Podduturi, Manisha Reddy**)
   Input: a txt file giving an adjacency list of a graph G on n vertices; ask the user to input a permutation of the numbers from 1 to n.
   Method: Treat the input permutation as a vertex ordering. For every vertex v in the ordering make vertices in $N^+[v]$ pairwise adjacent by adding into G new edges if necessary.

Output: a txt file giving an adjacency list of the filled graph and its perfect elimination ordering (= the input permutation).

7. LexBFS ordering of the vertices of a graph (**Pourebadi Khotbesara, Maryam**)
   Input: a txt file giving an adjacency list of a graph.
   Output: produce a vertex ordering using the LexBFS procedure.