Assignment 2
Algorithms for Big Data

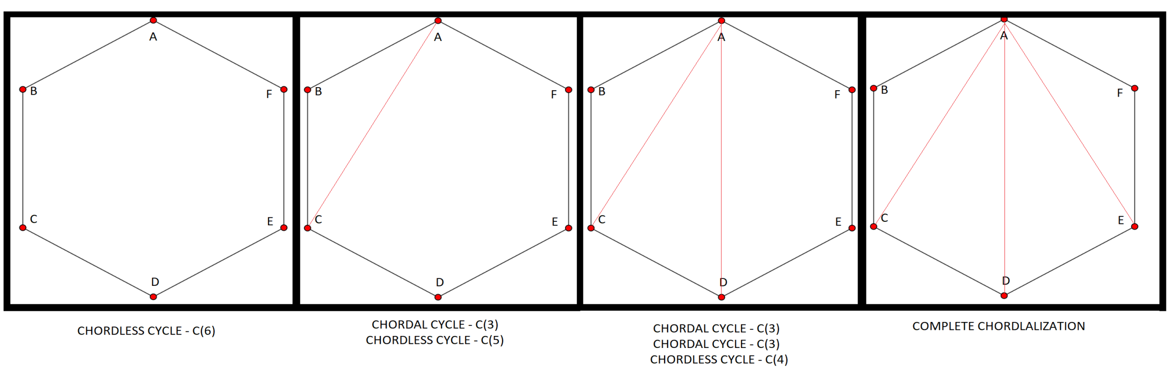
Lavinia Pulcinella
Student ID: i6233926

February 2020

1 Graded Exercise 2.5

Here we want to answer the question: is it possible to chordalize G by adding at most k new edges? We summarize this question as "(G, k)?".

1. **Let C_t be a chordless cycle on t vertices, $t \geq 4$. (So C_3 is a triangle, C_4 is a square, C_5 is a pentagon and so on).**
   There is a natural lower bound on the number of new edges required to chordalize C_t. What is it? You do not need to give a formal proof for all t, but you should argue convincingly why the bound holds for C_6.



Starting from vertex A, in order to chordalize the cycle we need to add enough new edges s.t. there are only cycles of 3 i.e. only chordal cycles.
Drawing the fist new edge between vertex A and its first non adjacent one C we end up having a C_3 chordal cycle and a C_5 chordeless one. Now, we draw another line between A and the next non-adjacent vertex D. Now we have two C_3 chordal cycles and a C_4 chordeless one. Finally, connecting A and F we have completely chordalized the C_6 cycle.

In general, the minimum number of k new edges required to chordalize a chordeless cycle C_t should be

$$min(k) = t - 3 \tag{1}$$

i.e. the total number of non-adjacent vertices.

2. **The answer to (1) suggests that, if G contains a certain structure, the answer to (G,k) will definitely be NO. What is it?**

   If graph G contains more than $k+3$ vertices then (G,k) is NO since we would need to add more than k edges. Thus:
   `if n > k+3 then , return NO`

3. **How many different edges can be drawn between non-adjacent vertices in C_t?**

   The maximum number of different edges that can be drawn is equal to the count of all possible diagonals that can be drawn within a cycle:
   $$max(k) = \frac{t(t-3)}{2} \tag{2}$$

4. **Describe an FPT branching algorithm to answer the question (G,k) with running time at most $O((k+3)^{2k} * poly(n))$ where n is the number of vertices in G.**
   In order to answer (G,k) we can employ the use of recursive branching algorithms.
   Having a graph $G = (V,E)$ we look for chordeless cycles C_t with $t \geq 4$. If no such cycle is found, then the answer to (G, k) would definitely be YES as there is nothing to chordalize.
   Once the chordeless cycle C_t is found if it has more than $k+3$ vertices then (G,k) is no as stated in point (2).
   Now, we can loop though all edges of the form $e = (u,v)$ then we call the problem for the graph G with the additional edge and we decrease k by one. Then if $(G+e,k-1)$ is YES then we return YES, otherwise we return NO.

   By branching off all the possibilities the treedepth is k (i.e. the number of new added edges) as it allows for the algorithm to stop when k is null. The branching factor, on the other hand, is captured by the maximum possible number of edges to be added in a chordeless cycle computed in point 3 which in terms of k can be written as $\frac{k(k+3)}{2}$.

Algorithm 1 BRANCHING _ CHORDALIZATION(G,k)

Look for a chordeless cycle C_t in $G = (V,E)$
If no C_t is found then return YES
If $t > k+3$ then return NO
for *all new edges i.e.* $e = \{u,v\}$ **do**
    If $(G+e,k-1)$ is YES then return YES
    Otherwise return NO
end

Thus with a treedepth of k and a branching factor of $\frac{k(k+3)}{2}$ the time complexity of the algorithm is $\approx O((k+3)^{2k} * poly(n))$