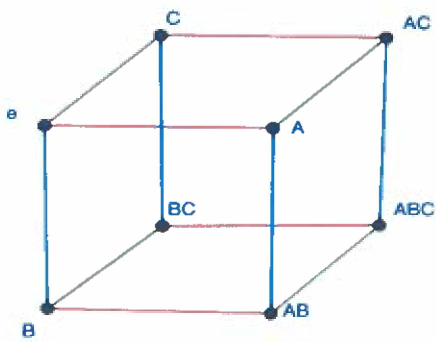
CSCI.761 Assignment 04

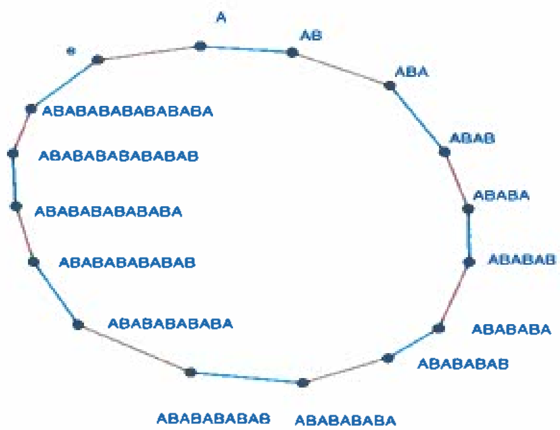
Xiaoyi Yang (*xy3371@rit.edu*)

March 16, 2021

1. Draw Cayley graphs of the two automorphism groups of Graph 2 and Graph 3 on this page, using the generators as listed there. Since in these cases all generators are involutions, your Cayley graphs can be shown as undirected graphs (example: Cayley graphs of the group of automorphisms of C5 for three different pairs of generators).



(a) Graph2



(b) Graph3

2. Let $J_n = K_n - e$ denote the complete graph K_n with one edge dropped. Trivially, $R(J_4, J_3) = 5$. It is known that $R(J_4, J_4) = 10$, $R(J_4, J_5) = 13$, $R(J_4, J_6) = 17$ and $R(J_4, J_7) = 28$. The first open case for the Ramsey numbers of this type is for J_4 versus J_8, for which the best known bounds are $30 \leq R(J_4, J_8) \leq 32$ (see pages 12 and 13 of the survey SRN for more details and references).

   (a) Generate and describe all graphs in $(J_4, J_4; 9)$ and $(J_4, J_5; 12)$.

   Number of $(J_4, J_4; 9)$: 1

   ```
   H{dQXgj
   ```

   All vertices have degree of 4, each triangle in this graph is adjacent to 3 quadrilaterals.

   Number of $(J_4, J_5; 12)$: 14

   ```
   K?rDTHXT`qes
   K@h[rPRdRKSF
   KQQDRQTT`IbT
   KTXKI`@_hPax
   KTXYCDA_XPax
   K`_iOmXpPsks
   K`j@aaIDYFDb
   KoCOZBWl@NHY
   KoCQPJSmAMd[
   KoStE?VEqTCj
   KtaHGthYaiis
   ```

```
KwC[SLPKiTCj
K{cAGgeBQSeK
K{cAHGUBQSeK
```

(b) Generate and describe all graphs in $(J_4, J_6; 16)$.

Number of $(J_4, J_6; 16)$: 4

```
OTXbCUESapIgaL`UCPwz`
Os_GagjLASkohqpkFHipk
OsaBA`GP@`dIHWEcas_]O
O{aSO@RRQqCxKsJWGpXpg
```

(c) Generate and describe all graphs in $(J_4, J_7; 27)$. Hint: the famous Schläfli graph, which remarkably has 51840 automorphisms, is involved. See the bounds listed in Table IIIa in the survey paper SRN.

Description:

The basic idea is starting from K_3, use the 0-1 matrix and gradually add vertices and filter those graphs with J_4, or other J-patterns based on the requirement.

Taking J_5 as example, since the graph is 2-colored, the J_5 could be either color, which in the 0-1 matrix, the J_5 may be represented as all 0s or all 1s. An J_5 has totally 10 edges. Therefore, if the sum of all the edges in 0-1 matrix higher than 9(1-edges) or lower than 1(0-edges), then we find a J_5 in the graph.

The searching algorithm is shown as below(using J5 as example):

```
1  int hasJ5(int color){
2      int edges = 9;
3  
4      if(n >= 5){
5          int i, j, k, l, m;
6          boolean red, blue;
7          for (i = 0;i < n - 4; i++){
8              for (j = i + 1; j < n - 3; j++){
9                  for (k = j + 1; k < n - 2; k++){
10                     // Prune if there are 2 less edges already
11                     red = (gmat[i][j] + gmat[i][k] + gmat[j][k] >= 2);
12                     blue = (gmat[i][j] + gmat[i][k] + gmat[j][k] <= 1);
13                     if(color ? red : blue){
14                         for(l = k + 1; l < n - 1; l++){
15                             // Only check the last added vertex
16                             m = n - 1;
17                             red = (gmat[i][j] + gmat[i][k] + gmat[j][k] + gmat[i][l] + gmat[j][l] + gmat[k][l] + gmat[i][m] + gmat[j][m] + gmat[k][m] + gmat[l][m] >= edges);
18                             blue = (gmat[i][j] + gmat[i][k] + gmat[j][k] + gmat[i][l] + gmat[j][l] + gmat[k][l] + gmat[i][m] + gmat[j][m] + gmat[k][m] + gmat[l][m] <= 1);
19                             if(color ? red : blue){
20                                 return(1);  /* J5 found */
21                             }
22                         }
23                     }
24                 }
25             }
26         }
27     }
28     return(0);  /* J5 not found */
29 }
30 
```

In order to prune the search to reduce the computation complexity. There are 2 tricks in the algorithm:

1. As the comment shows at Line 10, we check the number of edges with the vertices that we have, if there are 2 less edges already, then these vertices will not be part of J_5 no matter how they will connect to the rest of vertices. Then we stop the search and proceed to next vertex.

2. As the comment shows at Line 15, we only consider the situation that the last vertex is the latest added one. Since we have checked all the combinations of existing vertices already, we will not check them again.

// full source code