THE *MACAULAY 2* PACKAGE EDGEIDEALS

ADAM VAN TUYL

Abstract. We introduce the *Macaulay 2* package `EdgeIdeals` as a tool to study edge and cover ideals. These tutorials complement the lectures given at *MONICA: MONomial Ideals, Computations and Applications*, at the CIEM, Castro Urdiales (Cantabria, Spain).

Introduction

Computer algebra systems, like *Macaulay 2* [7], Singular [3], and `CoCoA` [1], have become essential tools for many mathematicians in commutative algebra and algebraic geometry. These systems provide a "laboratory" in which we can experiment and play with new ideas. From these experiments, a researcher can formulate new conjectures, and hopefully, new theorems. Computer algebra systems are especially good at dealing with monomial ideals. As a consequence, the study of edge and cover ideals is well suited to experiments using computer algebra systems.

The purpose of this tutorial is to familiarize the user with the package `EdgeIdeals` that was written by C. Francisco, A. Hoefel, and myself [6]. This package, written for *Macaulay 2*, provides a suite of functions to experiment with edge and cover ideals. Many of the results discussed in the lectures have been implemented into this package. Hopefully, the tools introduced in this tutorial will be the basis of your own research results!

As a final note, although I primarily discuss the `EdgeIdeal` package, I would recommend that your also become familiar with the packages `SimplicialComplexes`, written by S. Popescu, G.G. Smith, and M. Stillman (see [9]), and `SimplicialDecomposability` by D.W. Cook II (see [2]). The first package contains a number of useful functions related to simplicial complexes. In fact, the `EdgeIdeals` package requires a number of functions from this package. The `SimplicialDecomposability` package of D.W. Cook II is useful if you wish to study the properties of the simplicial complex associated to the edge or cover ideal of a graph.

1. Getting Started

Obviously, the first thing you need to do is install the latest version[1] of *Macaulay 2* on your computer. The download page is here:

`http://www.math.uiuc.edu/Macaulay2/Downloads/`

Pick the appropriate operating system, and then follow the instructions. This may take some time and patience.

[1] At the time of writing this tutorial, the current version was 1.4

I am going to assume that you have installed *Macaulay 2* and now have it working. To familiarize yourself with the basic syntax and some simple examples, a good place to start is this web page:

http://www.math.uiuc.edu/Macaulay2/GettingStarted/

If you have never used *Macaulay 2*, take a couple of minutes to try a couple of the sample sessions.

2. The EdgeIdeals Package

Now that you have *Macaulay 2* installed, we want to load the EdgeIdeals package. If you are using a current version of *Macaulay 2* (i.e., a version ≥ 1.4), then this package should already be included with your installation of Macaulay 2, and it simply has to be installed.

Remark 2.1. If you have an older version, or if your version does not include this package, you should first download the source code from this link:

http://j-sag.org/Volume1/EdgeIdeals.m2

Save the code in a file named EdgeIdeals.m2, and save the file into your working directory. You can now return the directions below. Note that when you run the command installPackage ''EdgeIdeals'', *Macaulay 2* will install the package where it can always find it in the future.

Open Macaulay 2 and input the following command

```
Macaulay2, version 1.4
with packages: ConwayPolynomials, Elimination, IntegralClosure, LLLBases,
               PrimaryDecomposition, ReesAlgebra, TangentCone

i1 : installPackage "EdgeIdeals"
```

You will only need to enter this command the first time you use the package. In the background, this command is making all the help pages. Once you have installed the package, you do not need to use the command again, but instead, use the instructions below. If you wish, you can start a new session by typing restart.

When we first start *Macaulay 2*, we start with following screen:

```
Macaulay2, version 1.4
with packages: ConwayPolynomials, Elimination, IntegralClosure, LLLBases,
               PrimaryDecomposition, ReesAlgebra, TangentCone

i1 :
```

At the prompt, type the following command to load the package EdgeIdeals:

```
i1 : loadPackage "EdgeIdeals"

o1 = EdgeIdeals

o1 : Package
```

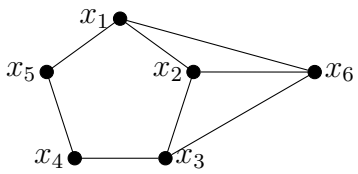
```
i2 : loadedPackages

o2 = {EdgeIdeals, SimplicialComplexes, SimpleDoc, Elimination, LLLBases,
     ------------------------------------------------------------------------
     IntegralClosure, PrimaryDecomposition, Classic, TangentCone, ReesAlgebra,
     ------------------------------------------------------------------------
     ConwayPolynomials, Core}

o2 : List
```

The second command returns all the packages currently loaded in *Macaulay 2*. Note that not only is the `EdgeIdeals` package loaded, but so is the `SimplicialComplexes` package. Many of the functions in `EdgeIdeals` run "on top" of `SimplicialComplexes`.

We are now ready to try out `EdgeIdeals`. To get going, we spend a little time discussing how to input a finite simple graph. As a concrete example, suppose that we want to study the graph



We enter this information in such a way that `Macaulay 2` recognizes it as a graph. There are a couple of ways to do this. The first way is to input a polynomial ring to denote the vertices, and then represent the edges as a list. For example

```
i3 : R = QQ[x_1..x_6]

o3 = R

o3 : PolynomialRing

i4 : E = {{x_1,x_2},{x_2,x_3},{x_3,x_4},{x_4,x_5},{x_5,x_1},{x_1,x_6},{x_2,x_6},{x_3,x_6}}

o4 = {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x },
       1   2     2   3     3   4     4   5     5   1     1   6     2   6     3   6
     ------------------------------------------------------------------------
     {x , x }}
       3   6

o4 : List

i5 : H = graph(R,E)

o5 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }}}
                      1   2     2   3     3   4     4   5     5   1     1   6     2   6     3   6
           ring => R
           vertices => {x , x , x , x , x , x }
                         1   2   3   4   5   6

o5 : Graph
```

Alternatively, the edges can be represented as the generators of a square-free quadratic monomial ideal. If no ring is passed to the command `graph`, it takes the variables of the current ring as the vertices of the graph. As an example, here is an alternative way to input the above graph into *Macaulay 2*:

```
i6 : e = monomialIdeal(x_1*x_2,x_2*x_3,x_3*x_4,x_4*x_5,x_5*x_1,x_1*x_6,x_2*x_6,x_3*x_6)

o6 = monomialIdeal (x x , x x , x x , x x , x x , x x , x x , x x )
                    1 2   2 3   3 4   1 5   4 5   1 6   2 6   3 6

o6 : MonomialIdeal of R

i7 : G = graph e

o7 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }}}
                      1   2     2   3     3   4     1   5     4   5     1   6     2   6     3   6
           ring => R
           vertices => {x , x , x , x , x , x }
                        1   2   3   4   5   6

o7 : Graph

i8 : G==H

o8 = true
```

Now that we have an object called a `Graph`, we can ask about its edge and cover ideals. Both of these ideals can be easily obtained using the following commands:

```
i9 : i = edgeIdeal G

o9 = monomialIdeal (x x , x x , x x , x x , x x , x x , x x , x x )
                    1 2   2 3   3 4   1 5   4 5   1 6   2 6   3 6

o9 : MonomialIdeal of R

i10 : j = coverIdeal G

o10 = monomialIdeal (x x x x , x x x x , x x x x , x x x x , x x x x ,
                     1 2 3 4   1 2 3 5   1 2 4 6   1 3 4 6   1 3 5 6
      ------------------------------------------------------------------------
      x x x x , x x x x )
       2 3 5 6   2 4 5 6

o10 : MonomialIdeal of R
```

The generators of $J(G)$ are the minimal vertex covers of G; convince yourself that the generators given in the above example are indeed the minimal vertex covers of the graph.

Recall that we showed that the Alexander dual of edge ideal $I(G)$ equals the cover ideal of $J(G)$. We can verify this for this ideal using a command from the `SimplicialComplexes` package (which is also loaded):

```
i11 : dual i == j

o11 = true
```

Once you have inputted your graph, you can now compute some of its graph theoretic invariants. For example, the chromatic number of the graph is computed as

```
i12 : chromaticNumber G

o12 = 3
```

To compute this number, we use of the fact that

$$\chi(G) = \min\{d \mid (x_1 \cdots x_n)^{d-1} \in J(G)^d\}$$

as proved in the first lecture. Similarly, Fröberg's Theorem gives us an algebraic characterization of chordal graphs. We can therefore check if G is chordal:

```
i13 : isChordal G

o13 = false
```

To facilitate experimentation, we have built a number of functions to create commonly occurring graphs, like cycles and cliques. Here are some examples:

```
i14 : C6 = cycle R

o14 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }}}
                       1   2     2   3     3   4     4   5     5   6     1   6
            ring => R
            vertices => {x , x , x , x , x , x }
                         1   2   3   4   5   6

o14 : Graph

i15 : C5 = cycle(R,5)

o15 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }}}
                       1   2     2   3     3   4     4   5     1   5
            ring => R
            vertices => {x , x , x , x , x , x }
                         1   2   3   4   5   6

o15 : Graph
```

The command `cycle` will return a cycle of length equal to the number of variables in the ring `R` as a default. If a number n is given, it will make a cycle of that length using the first n variables. Cliques of size n are defined similarly:

```
i16 : K4 = completeGraph(R,4)

o16 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }}}
                       1   2     1   3     1   4     2   3     2   4     3   4
            ring => R
            vertices => {x , x , x , x , x , x }
                         1   2   3   4   5   6
```

```
o16 : Graph
```

The command antiCycle is similar in that it returns the graph of the complement of a cycle.

Also built into the EdgeIdeals package is a number of commands to construct subgraphs. For example, suppose that we wish to look at the induced subgraph of G on the vertices $P = \{x_1, x_2, x_6, x_5\}$. This can be done as follows:

```
i17 : P = {x_1,x_2,x_6,x_5}

o17 = {x , x , x , x }
       1   2   6   5

o17 : List

i18 : GP = inducedGraph(G,P)

o18 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }}}
                      1   2     1   5     1   6     2   6
           ring => QQ[x , x , x , x ]
                      1   2   6   5
           vertices => {x , x , x , x }
                        1   2   6   5

o18 : Graph
```

Another similar command that may prove helpful is deleteEdges which removes a collection of edges from a graph.

To facilitate research, the EdgeIdeals package includes a function called randomGraph. This function allows you to generate a random graph on defined number of vertices and edges, and is useful when creating conjectures. Here is an example of the this function in action:

```
i19 : randomGraph(R,8)

o19 = Graph{edges => {{x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }, {x , x }}}
                      1   2     2   3     2   4     5   6     4   6     3   6     2   5     2   6
           ring => R
           vertices => {x , x , x , x , x , x }
                        1   2   3   4   5   6

o19 : Graph
```

In this case, we are asking for a random graph on 6 vertices (the number of variables in the polynomial ring R) with 8 edges. This function can be used to test a large number of examples quickly.

As a final note, the documentation of the EdgeIdeals package can be found here:

http://www.math.uiuc.edu/Macaulay2/doc/Macaulay2-1.4/share/doc/Macaulay2/EdgeIdeals/html/index.html

All the commands given in the package are listed on this page. Detailed documentation and example can be found by clicking on the approriate linkgs.

3. Tutorials

The afternoon tutorials give you a chance to play around and experiment with *Macaulay 2*. When required, the tutorials provides need definitions, results, and references. Some of the initial problems ask you to prove some simple results in order to give you a feeling for the material, while other problems ask you to program some simple procedures using *Macaulay 2* in order to help you develop your *Macaulay 2* skills. The last batch of questions for each tutorial is a series of open questions. These questions are denoted by an asterisk. (If you come up with any ideas, I would love to hear them!)

3.1. **Tutorial 1: Splitting Monomial Ideals.**

In this tutorial, we explore some of the properties of splitting monomial ideals.

Exercise 3.1.1. Suppose $I = J + K$ is a Betti splitting. Prove that

$$\operatorname{reg}(I) = \max\{\operatorname{reg}(J), \operatorname{reg}(K), \operatorname{reg}(J \cap K) - 1\}.$$

Here, $\operatorname{reg}(-)$ denotes the regularity of the given ideal.

Remark. This result can be quite useful when doing induction. For example, this fact was used to give a new proof for the regularity of the edge ideal of a tree [8].

Exercise 3.1.2. Write a `Macaulay 2` program that takes as input two monomial ideals J and K, and will return `true` or `false` depending upon whether $J + K$ is a Betti splitting.

Hint. The command `betti res I` will return the Betti diagram of the ideal I. Read through the `betti` documentation in order to extract out the information you are looking for. If you are interested in a particular graded Betti number, you may wish to first define the function:

```
beta = (i,j,I) -> (betti res I)#(i,{j},j)
```

Exercise 3.1.3. (Importance of $\operatorname{char}(k)$) Consider the following ideal in $R = k[x_1, \ldots, x_6]$:

$$I = (x_1x_2x_4, x_1x_2x_6, x_1x_3x_5, x_1x_3x_4, x_1x_5x_6, x_2x_4x_5, x_2x_3x_6, x_2x_3x_5, x_3x_4x_6, x_4x_5x_6).$$

Fix a variable x_i, and form an x_i-partition of I, i.e., let J be the ideal generated by all the generators of I divisible by x_i, and let K be the ideal generated by the remaining generators. Use *Macaulay 2* to show $I = J + K$ is a Betti splitting in $\operatorname{char}(k) = 2$, but not a Betti splitting if $\operatorname{char}(k) \neq 2$.

Hint. One way to input a ring a characteristic two is

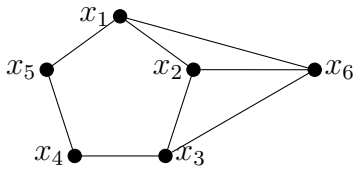
```
i1 : S = ZZ/(2)[a,b,c]
```

Definition 3.1. Let $I(G)$ be the edge ideal of a graph. For any edge $e = \{x_i, x_j\}$, we have the partition

$$I(G) = \langle x_ix_j \rangle + I(G \setminus e)$$

where $G \setminus e$ is the graph G with the edge e removed. We call e a *splitting edge* if this partition is a Betti splitting.

Exercise 3.1.4. Consider the graph



Determine which edges of this graph are splitting edges.

Exercise 3.1.5. Write a program in *Macaulay 2* that inputs a graph, and returns all the edges in the graph that are splitting edges.

Exercise 3.1.6. Let $G = C_n$ be a cycle of length $n \geq 4$. Prove that G has no splitting edge.

Exercise 3.1.7. Let $G = K_n$ be the clique of size $n \geq 3$. Prove that every edge of G is a splitting edge.

Exercise 3.1.8. Find a graph G that is not a cycle, but no edge is a splitting edge. Then, find a graph G that is not a clique, but every edge is a splitting edge.

Exercise 3.1.9. A vertex v is called a leaf if $\deg v = 1$. Suppose that v is a leaf, and $e = \{v, u\}$ is the only edge that contains v. Prove that e is a splitting edge.

Exercise 3.1.10. Let $N(x) = \{y \mid \{x, y\} \in E(G)\}$ be the neighbours of x. Make a conjecture about $\{x, y\}$ being a splitting edge in terms of $N(x) \cup N(y)$. Compare your answer to [8].

⋆**Exercise 3.1.11.** Is the number of splitting edges related to any invariants of G or $I(G)$?

⋆**Exercise 3.1.12.** Find other ways to split $I(G)$.

⋆**Exercise 3.1.13.** Are they any nice ways to construct Betti splittings of the cover ideal $J(G)$? (I am only aware of how to split $J(G)$ in the case that $R/J(G)$ is Cohen-Macaulay and G is bipartite [5].)

⋆**Exercise 3.1.14.** Are there Betti splittings of the ideals $I(G)^s$ and $J(G)^s$, for some integer s?

3.2. **Tutorial 2: Regularity.** In this tutorial, we look at the regularity of an edge and cover ideals.

Exercise 3.2.1. A *tree* is graph without any induced cycles. If T is tree, what is the regularity of $I(T^c)$, where T^c is the complement of T?

Exercise 3.2.2. Describe all trees T with the property that $\operatorname{reg}(I(T)) = 2$.

Exercise 3.2.3. Create any graph G where the smallest induced cycle of G^c has length 4. Use `Macaulay 2` to compute the resolution. Now repeat for a graph G whose smallest induced cycle in G^c has length $5, 6, 7, \ldots$ until you observe your pattern. Compare your answer to Eisenbud, et al. [4].

Exercise 3.2.4. If you would like to see the code of a *Macaulay 2* function, you can use

`code methods use` ⟨ function name ⟩

Look at the code for `smallestCycleSize`. Try to figure out how *Macaulay 2* finds the smallest induced cycle in a graph.

Exercise 3.2.5. Write a *Macaulay 2* function that checks if graph has an induced 4 cycle.

Hint. Use the fact that

$$\beta_{1,4}(I(G)) = c_4(G^c)$$

where $c_4(H)$ denotes the number of induced four cycles in the graph H. (see [13]).

Exercise 3.2.6. Write a *Macaulay 2* function that tests whether an ideal has linear resolution.

Exercise 3.2.7. Nevo and Peeva [11] have made the following conjecture:

Conjecture 3.2. *For all graphs G, if G^c has no induced four cycles, then there exists a integer s such that $I(G)^s$ has a linear resolution.*

Using the command `randomGraph`, find 10 graphs where the conjecture is true, and for each graph, find the smallest integer s where $I(G)^s$ has a linear resolution.

Exercise 3.2.8. The *path* of length n, denoted P_n is the graph with vertex set $\{x_1, x_2, \ldots, x_n\}$ and edge set

$$\{\{x_1, x_2\}, \{x_2, x_3\}, \ldots, \{x_{n-1}, x_n\}\}.$$

Compute the regularity of $I(P_n)$ for some n until you find a pattern. Compare your result to Jacques [10].

⋆**Exercise 3.2.9.** Let T be a tree. Find a formula for $\operatorname{reg}(I(T)^s)$ as s varies.

Hint. You may wish to start with the case that $T = P_n$ first.

⋆**Exercise 3.2.10.** Find a formula for $\operatorname{reg}(J(G))$ and $\operatorname{reg}(I(G))$ for any graph.

Hint. This problem is probably too open ended. I am not aware of many results on the regularity of $J(G)$. For edge ideals, more is known (do a Google search on "regularity edge ideals"). For bipartite graphs, we almost have a complete story. See [12] for more.

References

[1] CoCoATeam, CoCoA: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`

[2] D.W. Cook II, Simplicial Decomposability. The Journal of Software for Algebra and Geometry **2** (2010) 20-23.

[3] W. Decker, G.-M. Greuel, G. Pfister, H. Schönemann, SINGULAR 3-1-3 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de (2011).

[4] D. Eisenbud, M. Green, K. Hulek, S. Popescu, Restricting linear syzygies: algebra and geometry. Compos. Math. **141** (2005), no. 6, 1460–1478.

[5] C.A. Francisco, H.T. Hà, A. Van Tuyl, Splittings of monomial ideals. Proc. Amer. Math. Soc. **137** (2009), no. 10, 3271–3282.

[6] C.A. Francisco, A. Hoefel, A. Van Tuyl, `EdgeIdeals`: a package for (hyper)graphs. The Journal of Software for Algebra and Geometry **1** (2009) 1-4.

[7] D. R. Grayson and M. E. Stillman, Macaulay 2, a software system for research in algebraic geometry. `http://www.math.uiuc.edu/Macaulay2/`

[8] H.T. Hà, A. Van Tuyl, Splittable ideals and the resolutions of monomial ideals. J. Algebra **309** (2007), no. 1, 405-425.

[9] S. Ho sten, G.G. Smith, Monomial Ideals. In *Computations in algebraic geometry with Macaulay 2*, Algorithms and Computations in Mathematics 8, pp. 73–100, Springer-Verlag, New York, 2001.

[10] S. Jacques, Betti Numbers of Graph Ideals. PhD Thesis, University of Sheffield, (2004). `arXiv:math/0410107v1`

[11] E. Nevo, I. Peeva, Linear resolutions of powers of edge ideals. Preprint (2010).

[12] A. Van Tuyl, Sequentially Cohen-Macaulay bipartite graphs: vertex decomposability and regularity. Archiv der Mathematik **93** (2009), 451–459.

[13] R.H. Villarreal, *Monomial algebras.* Monographs and Textbooks in Pure and Applied Mathematics, **238**. Marcel Dekker, Inc., New York, 2001.

Department of Mathematical Sciences, Lakehead University, Thunder Bay, ON P7B 5E1, Canada

E-mail address: `avantuyl@lakeheadu.ca`

URL: `http://flash.lakeheadu.ca/∼avantuyl/`