International Institute of Information Technology, Bangalore

Basic Computational Topology
SM 402

BCT Implementation Assignment

May 10, 2022

Group 17
Rudransh Dixit (IMT2020056)
Maurya Patel (IMT2020517)
Mukul Gupta (IMT2020083)
Bhavjyot Singh(IMT2020116)

Problem Statement

Given an input simplicial complex (up to 3-dimensional), corresponding to each 2-hole (void) find a representative 2-cycle and visualize all the representative 2-cycles in the input simplicial complex.

Algorithm

In our program we take vertices, edges and faces (it is redundant) as input (from a .gts file). We then create a matrix corresponding to the linear transformation ∂_2 and then compute $dim(Im(\partial_2))$, i.e. the $rank(\partial_2)$ and also the kernel space of the matrix ∂_2.

As we know,

$$\partial_2(\overline{v_1v_2v_3}) = \overline{v_1v_2} + \overline{v_2v_3} - \overline{v_1v_3}$$

Now it is clear that $C_2(K)$ is the vector space of all the 2-chains $\implies$ $dim(C_2(K))$= number of faces.

The matrix ∂_2 is a mapping from $C_2(K)$ to $C_1(K)$. We take the input faces and accordingly compute the matrix ∂_2 with the faces making the columns and the edges making thew rows. Then according to the equation stated above, we fill the matrix with 1, -1 and 0 accordingly. Then a function is created to compute the RREF of the ∂_2 matrix. Computing the kernel space and the image space of this RREF will give us the 2-holes of all the 2D-simplicial complexes.

For the 3D data, we need to divide the spaces in the form of tetrahedrons instead of triangulation of the surfaces. In this case, the equivalence classes will be created from the computed kernel space of the matrix. Also in this case we will compute the boundary matrix ∂_3 which is computed by the formula,

$$\partial_3(\overline{v_1v_2v_3v_4}) = \overline{v_1v_2v_3} + \overline{v_2v_3v_4} + \overline{v_1v_2v_4} - \overline{v_1v_3v_4}$$

The boundary matrix ∂_3 will have the tetrahedrons representing the columns and the triangles representing the rows. Then the RREF, image space and the kernel space of this boundary matrix is calculated. Finally, the equivalence classes of the above calculated kernel is calculated. The vectors which are present in these equivalence classes will represent the 2-holes in the D-simplicial complex.

Implementation Steps

1. First we ask for filename as input from user (.gts file) (in main() function). We open the file corresponding to this filename. From here we get the number of vertices, edges and faces in the input simplicial complex.
2. Now in the next step, we read the coordinates of the input vertices, the vertices which share an edge between them, and the edges that constitute a face.
3. In the next step we create the boundary matrix corresponding to ∂_2 using the faces of the input.
4. In the next step, we calculate the RREF (Row Reduced Echelon Form) of the computed boundary matrix.
5. Next we compute the kernel and the image space of the RREF which was calculated above and also the rank of the aforementioned matrices.

Steps to run the code

1. Open the terminal.
2. Enter the command "g++ -Ofast main.cpp Matrix.cpp".
3. Enter the command "./a.out filename".
4. Enter the filename of the .gts file you want to take input from.
5. Press Enter to get the final result.

NOTE:- Here the code may take time to calculate the result for very large data (depending on the system hardware specifications).

Demo Results

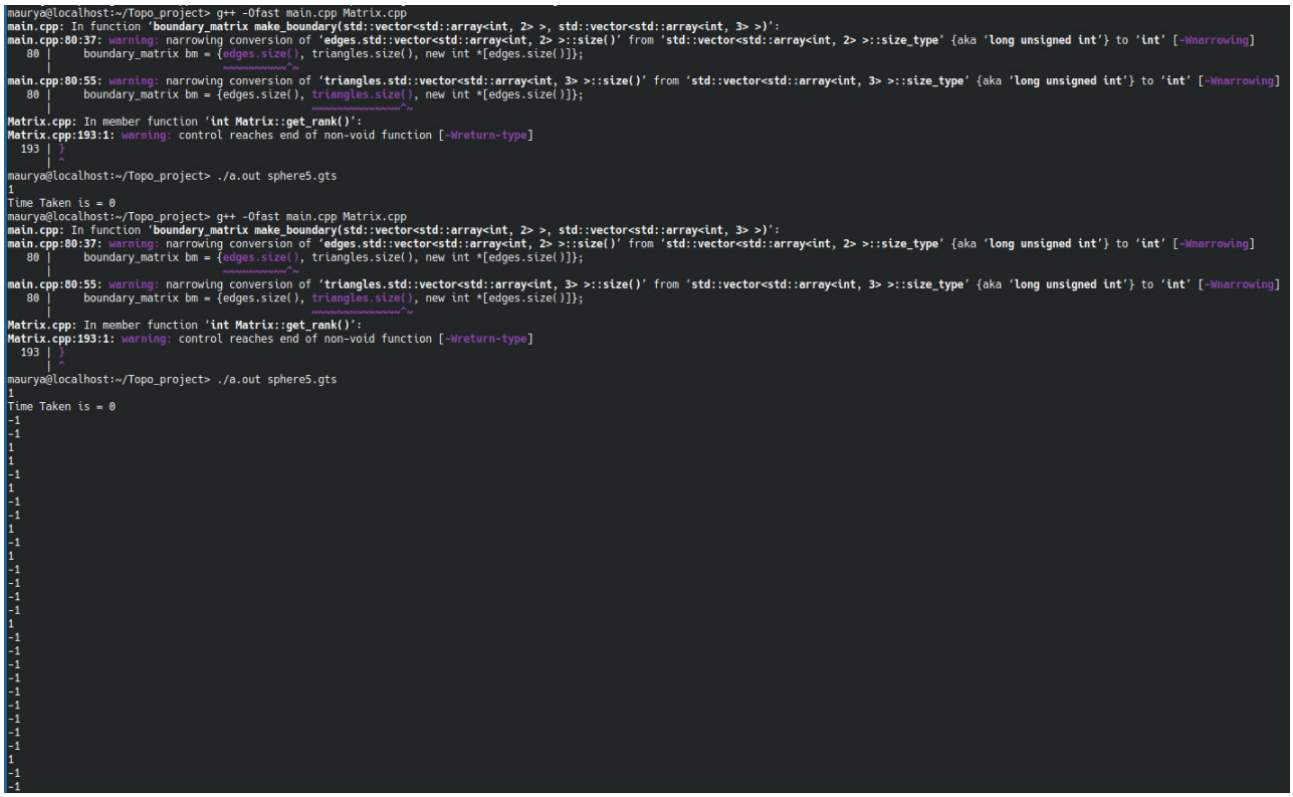

Figure 1: Testcase 1:- Sphere

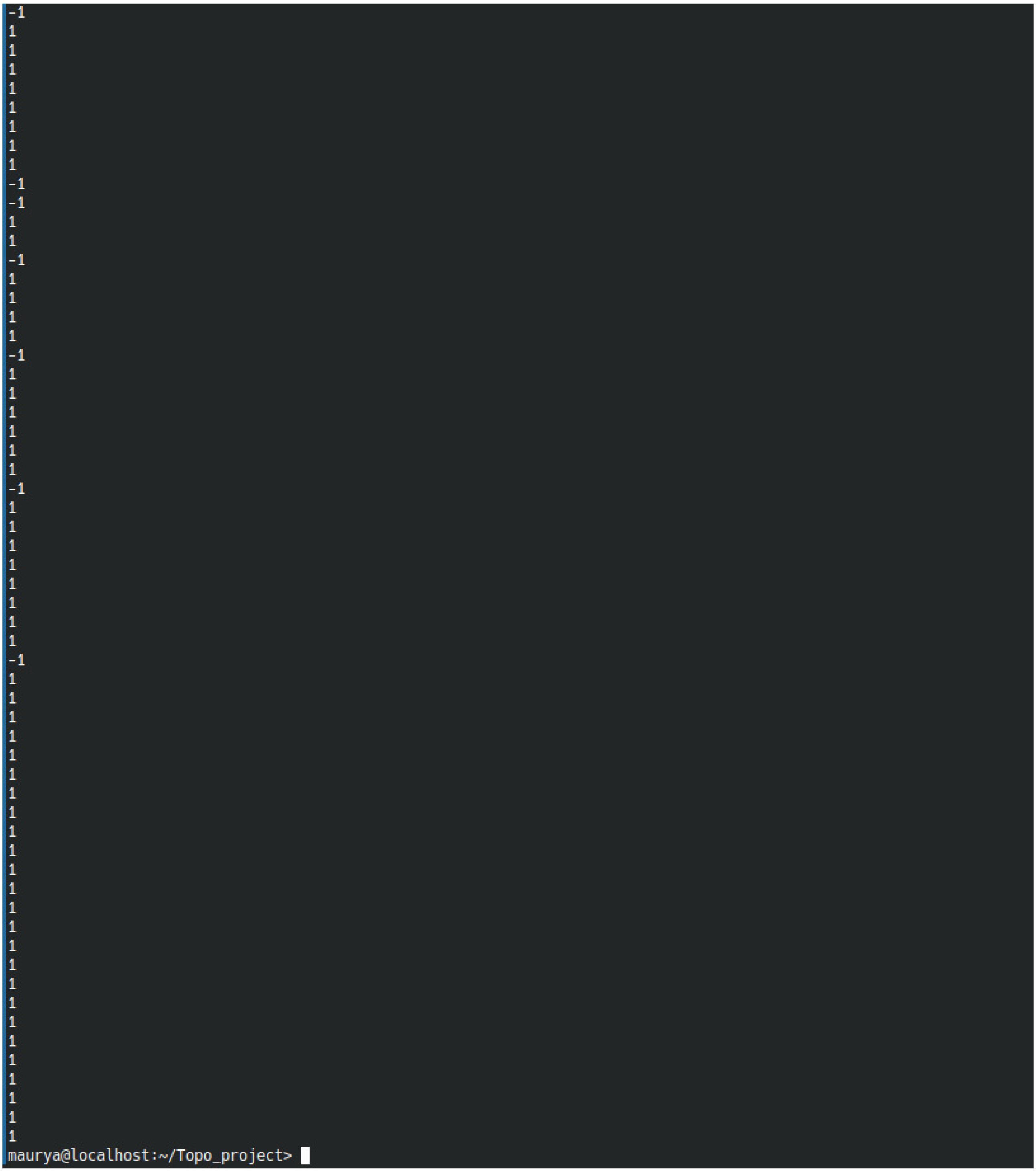

Figure 2: Testcase 1:- Sphere

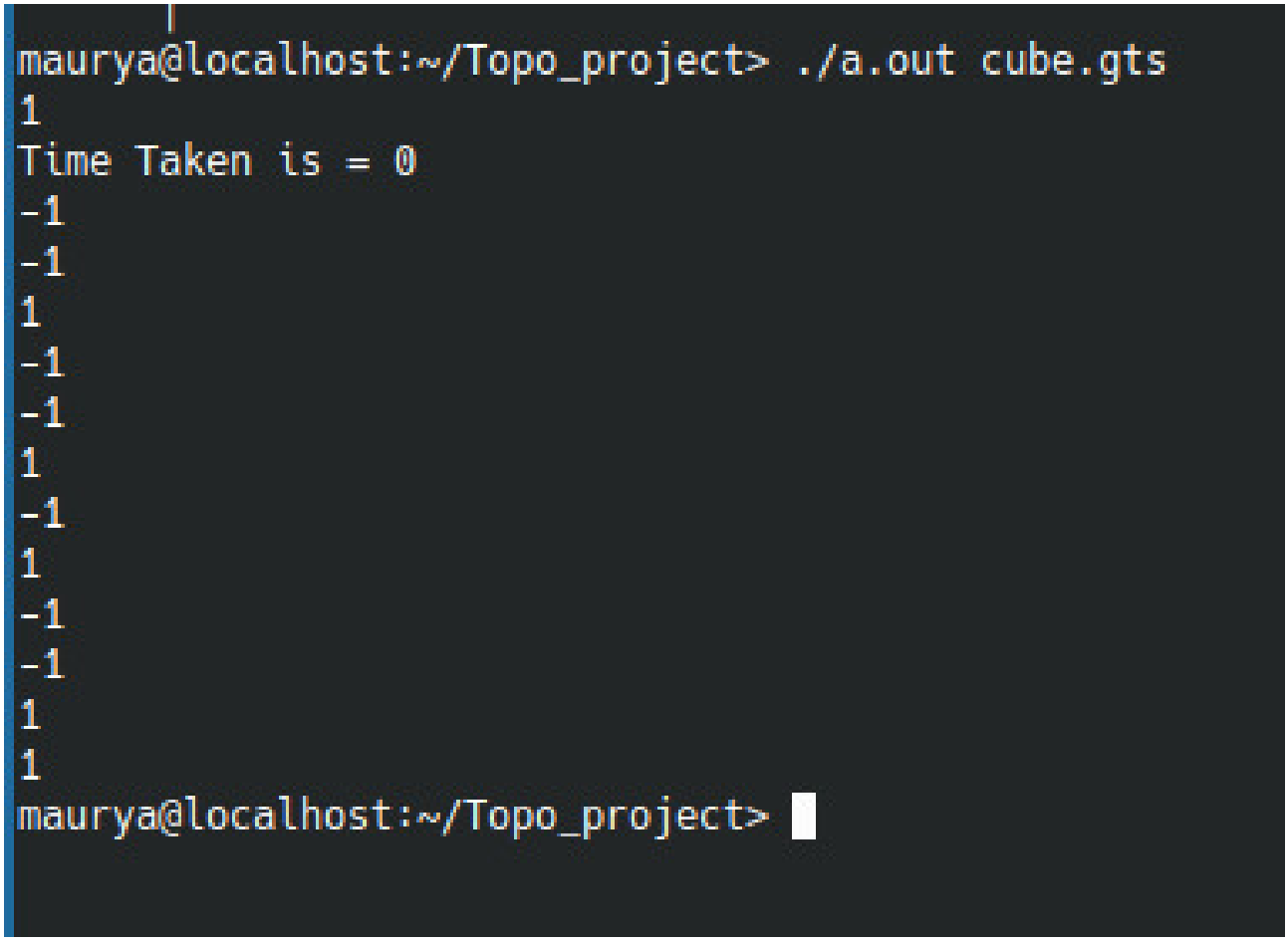

Figure 3: Testcase 2:- Cube

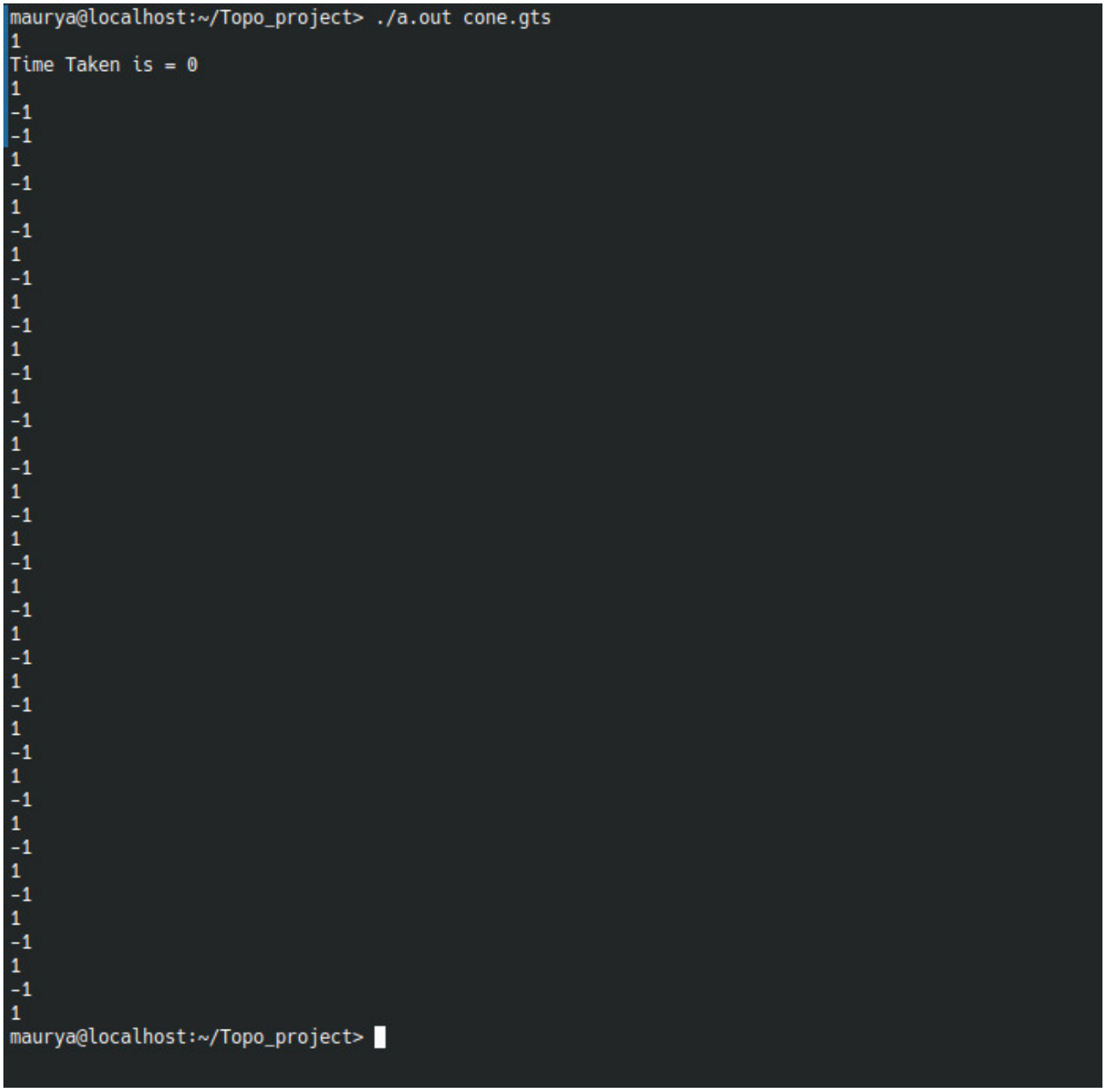

Figure 4: Testcase 3:- Cone

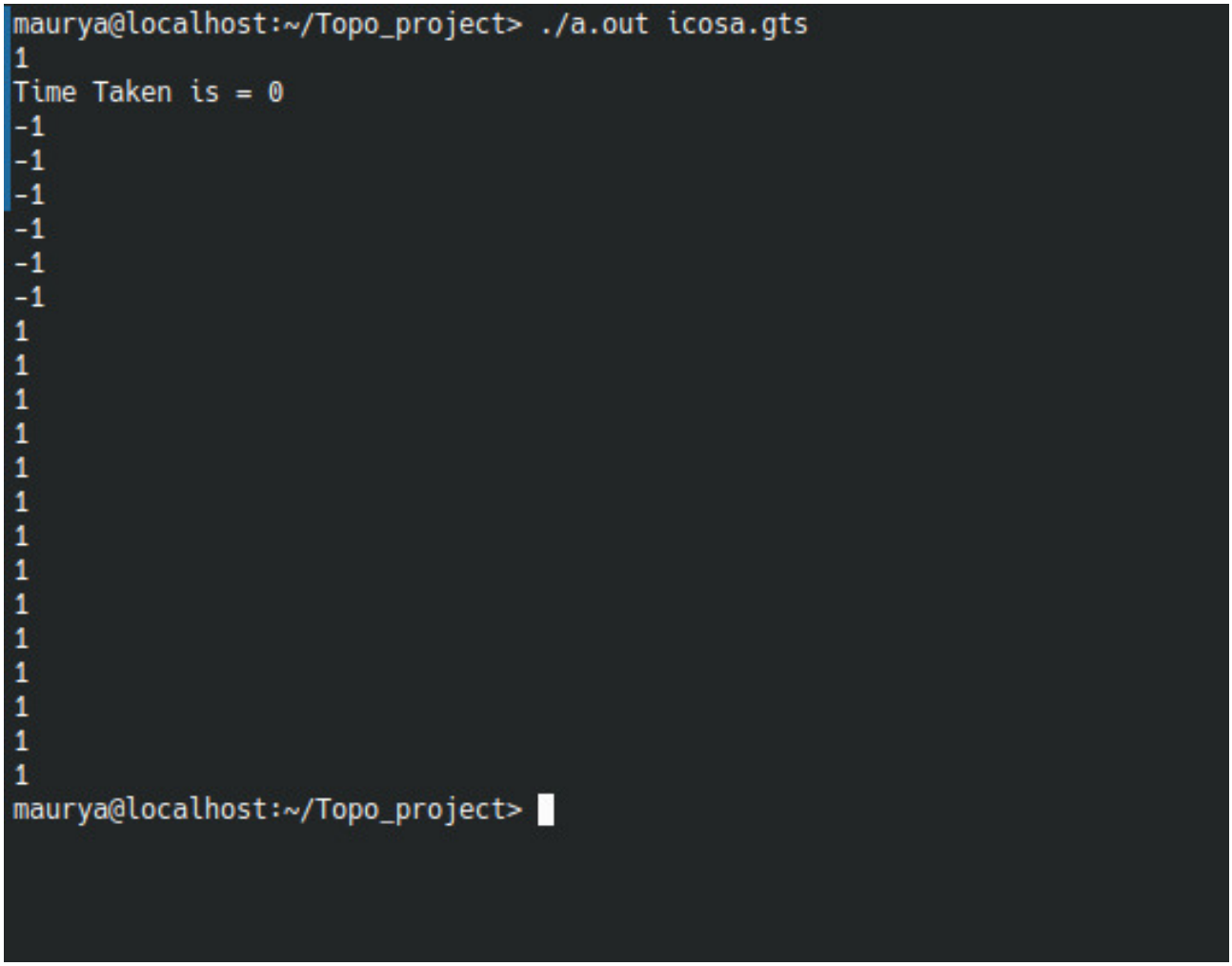

Figure 5: Testcase 4:- Icosahedron

Figure 6: Testcase 5:- Tetrahedron

GitHub Link

Please visit this for the source code.
https://github.com/raddi1972/topo_project.git

References

1. https://jeremykun.com/2013/04/10/computing-homology/
2. https://en.wikipedia.org/wiki/Quotient_space_(linear_algebra
3. https://en.wikipedia.org/wiki/Rank%E2%80%93nullity_theorem
4. https://en.wikipedia.org/wiki/Rank_(linear_algebra)
5. https://en.wikipedia.org/wiki/Simplicial_complex
6. http://gts.sourceforge.net/samples.html (for sample testcases)