SimplicialSurfaces

A tutorial for the SimplicialSurfaces-package.

0.1

25 Oktober 2022

Reymond Akpanya

Alice Niemeyer

Wilhelm Plesken

Alice Niemeyer
Email: Alice.Niemeyer@Mathb.RWTH-Aachen.De
Homepage: http://www.math.rwth-aachen.de/~Alice.Niemeyer/
Address: Alice Niemeyer
Lehrstuhl für Algebra und Darstellungstheorie
RWTH Aachen
Pontdriesch 10/16
52062 Aachen
GERMANY

Reymond Akpanya
Email: akpanya@art.rwth-aachen.de
Address: –

Copyright

The primary sources for much of the covered material are:

The PhD-thesis "Regularity Aspects for Combinatorial Simplicial Surfaces" of Markus Baumeister

The book "Simplicial Surfaces of Congruent Triangles" by Alice C. Niemeyer, Wilhelm Plesken, Daniel Robertz, and Ansgar W. Strzelczyk (unpublished)

Chapter 1

Tutorial

This chapter deals with the targeted use of the `SimplicialSurfaces` package to solve certain problems of interest. The aim of this tutorial is to demonstrate the power of the package by combining several functions provided in the package. It should enable users to find routines for their tasks quickly without prior knowledge of the functions' names and also help users to familiarise themselves with the package without having to read the entire reference manual.

1.1 Parallelelepiped

Problems : Construction of the simplicial parallelepiped

1. from a cube
2. from an octahedron and two tetrahedra via vertices in faces
3. from an octahedron and two tetrahedra via surface constructions
4. from a double-6-gon using edge-turns
5. from a barycentric subdivision using edge-turns
6. using the powerset of [1,2,3] as vertices

Theoretical background

- Vertex-faithful surfaces and boolean operations

Frequently used commands

- ConnectedFaceSum() (**??**)
- EdgesOfFaces() (local version: EdgesOfFace()) (**??**)
- EdgesOfVertices() (local version: Edges) (**??**)
- EulerCharacteristic() (**??**)
- FaceDegreesOfVertices() (local version: FaceDegreeOfVertex)(**??**)
- FacesOfVertices() (local version: FacesOfVertex()) (**??**)

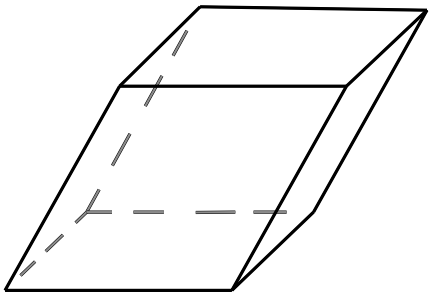
- Flags() (**??**)
- IsIsomorphic() (**??**)
- JoinBoundaries() (**??**)
- NeighbourVerticesOfVertex() (**??**),
- SimplicialSurfaceByVerticesInFaces() (works for vertex-faithful surfaces only) (**??**)
- SubcomplexByFaces() (**??**)
- VertexCounter() (local version: DegreeOfVertex()) (**??**)
- VerticesOfEdges() (local Version: VerticesOfEdge()) (**??**)
- VerticesOfFaces() (local Version: VerticesOfFace()) (**??**)
- UmbrellaPathsOfVertices() (local version: UmbrellaPathOfVertex()) (**??**)

Less frequently used commands

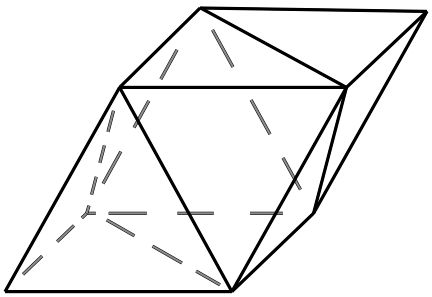
- Cube() (**??**)
- JanusHead() (**??**)

Mathematical details :

A parallelepiped is a three dimensional convex body. In this exercise we shall deal with its surface which we shall refer to as the ordinary parallelepiped, and with a triangulation of it, which we shall call the simplicial parallelepiped. Both are treated from a combinatorial point of view.

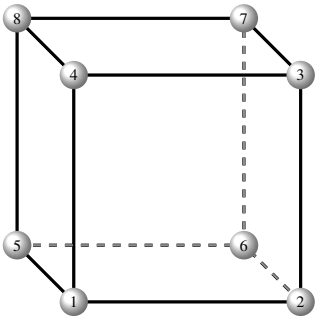


We use a tetrahedron to triangulate the surface of a ordinary parallelepiped. In this way each face is subdivided into 2 triangles and thus a simplicial parallelepiped with 12 faces is constructed. So after triangulating, the simplicial parallelepiped has more embeddings into three dimensional space, since the original faces might be bent along the new edges.



Given a set of three linearly independent vectors in real 3-space, we can see that they define a parallelepiped as well as a tetrahedron.

Note: Since we are only interested in surfaces, we shall refer to terms like tetrahedron, parallelepiped, octahedron etc. as the boundary surfaces of these three dimensional bodies, or rather to the combinatorial devices describing their combinatorial structure. So instead of working with an embedding of those structures, we see them as abstract surfaces represented by their incidence geometry.



Note: The `SimplicialSurfaces` package also contains functionalities to deal with polygonal complexes like the cube and the ordinary parallelepiped. But since their combinatorial structures do not differ, the simplicial parallelepiped is of greater interest to us.

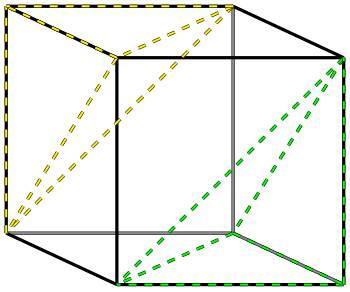
Example

```
gap> PE:=PolygonalComplexByVerticesInFaces([[1,2,3,4],[5,6,7,8],[1,2,5,6],
[2,3,6,7],[3,4,7,8],[1,4,5,8]]);
polygonal surface (8 vertices, 12 edges, and 6 faces)
gap> IsIsomorphic(PE,Cube());
true
gap> VerticesOfFaces(Cube());
[ [ 1, 2, 3, 4 ], [ 1, 2, 5, 6 ], [ 2, 3, 6, 7 ], [ 1, 4, 5, 8 ],
  [ 3, 4, 7, 8 ], [ 5, 6, 7, 8 ] ]
gap> VerticesOfFaces(PE);
[ [ 1, 2, 3, 4 ], [ 5, 6, 7, 8 ], [ 1, 2, 5, 6 ], [ 2, 3, 6, 7 ],
  [ 3, 4, 7, 8 ], [ 1, 4, 5, 8 ] ]
```

1.1.1 Construction from a cube

Idea behind the construcion

We construct a simplicial parallelepiped out of the combinatorial structure of a cube and two tetrahedra. Superimposing two disjoint tetrahedra onto the cube results in dividing each cube face into two triangles. This gives rise to new edges which were not edges of the cube beforehand. In that way we obtain the simplicial parallelepiped as a subdivision of the cube.



Example

```
gap> PE:=Cube()
polygonal surface ( 8 vertices, 12 edges, 6 faces)
gap> VerticesOfFaces(PE);
[ [ 1, 2, 3, 4 ], [ 1, 2, 5, 6 ], [ 2, 3, 6, 7 ], [ 1, 4, 5, 8 ]
[ 3, 4, 7, 8 ], [ 5, 6, 7, 8 ] ]
```

There are two disjoint tetrahedra contained in this cube with the following property: Each tetrahedron's faces subdivide three faces of the cube and the tetrahedron's vertices are also vertices of the cube.

How can we find them? First we list each vertex together with its neighbour vertices, so that we can find a pair of disjoint sets among them:

Example

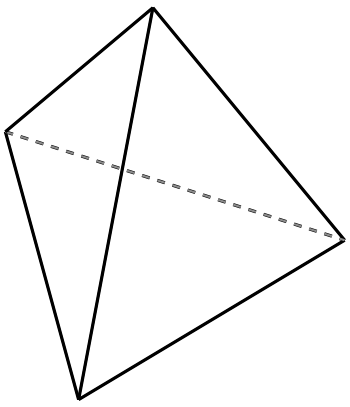
```
gap> List([1..8], i-> NeighbourVerticesOfVertex(PE,i));
[ [ 2, 4, 5 ], [ 1, 3, 6 ], [ 2, 4, 7 ], [ 3, 1, 8 ], [ 1, 8, 6 ],
 [ 2, 7, 5 ], [ 3, 6, 8 ], [ 4, 7, 5 ] ]
gap> SS:=List([1..8], i->Union([i],last[i]));
[ [ 1, 2, 4, 5 ], [ 1, 2, 3, 6 ], [ 2, 3, 4, 7 ], [ 1, 3, 4, 8 ],
 [ 1, 5, 6, 8 ], [ 2, 5, 6, 7 ], [ 3, 6, 7, 8 ], [ 4, 5, 7, 8 ] ]
gap> Filtered([1..8],i->Intersection(SS[1],SS[i])=[]);
[ 7 ]
gap> T1:=SS[1]; T2:=SS[7];
[ 1, 2, 4, 5 ]
[ 3, 6, 7, 8 ]
```

Construct the corresponding tetrahedra:

Example

```
gap> Combinations(T1,3);
[ [ 1, 2, 4 ], [ 1, 2, 5 ], [ 1, 4, 5 ],[ 2, 4, 5]];
gap> T1:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (4 vertices, 6 edges, 4  faces)

gap> Combinations(T2,3);
[ [ 3, 6, 7 ], [ 3, 6, 8 ], [ 3, 6, 8 ], [ 6, 7, 8 ] ]
gap> T1:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (4 vertices, 6 edges, 4  faces)
```



Before proceeding with further computations, verify that the two simplcial surfaces constructed are indeed isomorphic to the tetrahedron.

Example

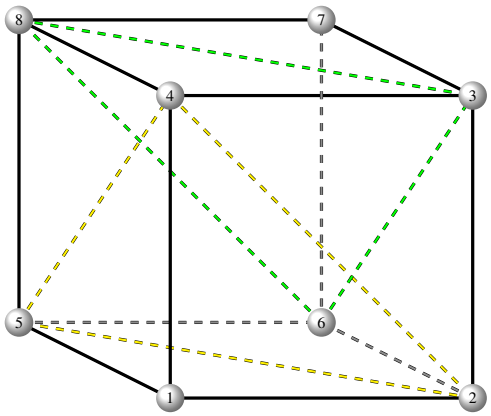
```
gap> IsIsomorphic(T1,Tetrahedron());
true
gap> IsIsomorphic(T2,Tetrahedron());
true
```

Now use the edges of the tetrahedra T1 and T2 to subdivide the cube's faces. More precisely take the edges of T1 and T2 not containing vertex 1 resp. 7 as diagonals of the faces of PE in which they lie to subdivide each of these faces into two triangular faces:

Example

```
gap>  VerticesOfFaces(PE);
[ [ 1, 2, 3, 4 ], [ 1, 2, 5, 6 ], [ 2, 3, 6, 7 ], [ 1, 4, 5, 8 ],
  [ 3, 4, 7, 8 ], [ 5, 6, 7, 8 ] ]
gap> VerticesOfEdges(T1);
[ [ 1, 2 ], [ 1, 4 ], [ 1, 5 ], [ 2, 4 ], [ 2, 5 ], [ 4, 5 ] ]
gap> VerticesOfEdges(T2);
[ [ 3, 6 ], [ 3, 7 ], [ 3, 8 ], [ 6, 7 ], [ 6, 8 ], [ 7, 8 ] ]
gap> neEd:= Union(Filtered(VerticesOfEdges(T1),r->not 1 in
> r),Filtered(VerticesOfEdges(T2),r->not 7 in r));
[ [ 2, 4 ], [ 2, 5 ], [ 3, 6 ], [ 3, 8 ], [ 4, 5 ], [ 6, 8 ] ]

gap> neEd:=List(VerticesOfFaces(PE), r->Filtered(neEd,s->IsSubset(r,s))[1]);
[ [ 2, 4 ], [ 2, 5 ], [ 3, 6 ], [ 4, 5 ], [ 3, 8 ], [ 6, 8 ] ]
gap> List([1..6], i->IsSubset(VerticesOfFaces(PE)[i],neEd[i]));
[ true, true, true, true, true, true ]
```



Now use the set neEd defined above to subdivide the cube's faces with the following function:

Example

```
gap> part:=function(Q,E)
# Q set of vertices of old faces
# E diagonal Edges
> local ne;
> ne:=Difference(Q,E);
> return [Union(E,[ne[1]]),Union(E,[ne[2]])];
> end;
function( Q, E ) ... end
```

Construct the simplicial parallelepiped by defining the faces represented by their sets of vertices. Note, this works only because the parallelepiped is vertex-faithful.

Example

```
gap> List([1..6],i->part(VerticesOfFaces(PE)[i],neEd[i]));
[ [ [ 1, 2, 4 ], [ 2 .. 4 ] ], [ [ 1, 2, 5 ], [ 2, 5, 6 ] ],
  [ [ 2, 3, 6 ], [ 3, 6, 7 ] ], [ [ 1, 4, 5 ], [ 4, 5, 8 ] ],
  [ [ 3, 4, 8 ], [ 3, 7, 8 ] ], [ [ 5, 6, 8 ], [ 6 .. 8 ] ] ]

gap> Union(last);
[ [ 1, 2, 4 ], [ 1, 2, 5 ], [ 1, 4, 5 ], [ 2 .. 4 ], [ 2, 3, 6 ],
  [ 2, 5, 6 ], [ 3, 4, 8 ], [ 3, 6, 7 ], [ 3, 7, 8 ], [ 4, 5, 8 ],
  [ 5, 6, 8 ], [ 6 .. 8 ] ]

gap> PE:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface ( 8 vertices, 18 edges, 12 faces )
```

We show that the surface has Euler characteristic 2 and vertex counter $v_3^2 v_5^6$.

Example

```
gap> EulerCharacteristic(PE);
2
gap> FacesOfVertices(PE);
[ [ 1, 2, 3 ], [ 1, 2, 4, 5, 6 ], [ 4, 5, 7, 8, 9 ], [ 1, 3, 4, 7, 10 ],
    [ 2, 3, 6, 10, 11 ], [ 5, 6, 8, 11, 12 ], [ 8, 9, 12 ],
  [ 7, 9, 10, 11, 12 ] ];
gap> VertexCounter(PE);
[ [ 3, 2 ], [ 5, 6 ] ]
```

Computing the set of vertices connected to a given vertex via an edge with the command `NeighbourVerticesOfVertex`

Example

```
gap> NeighbourVerticesOfVertex(PE,1);
[ 2, 4, 5 ]
gap> NeighbourVerticesOfVertex(PE,2);NeighbourVerticesOfVertex(PE,4);
[ 1, 3, 4, 5, 6 ]
[ 1, 2, 3, 5, 8 ]
gap> NeighbourVerticesOfVertex(PE,7);
[ 3, 6, 8 ]
```

Therefore we see that the simplicial surface `PE` is a sphere with the following umbrella descriptor.

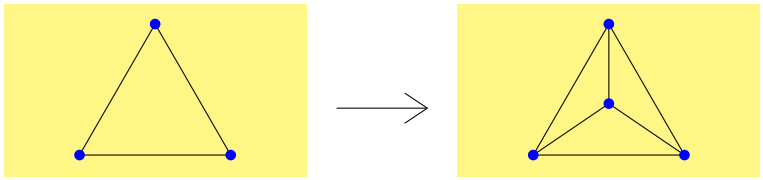
Example

```
gap> List(UmbrellaPathsOfVertices(PE),r->FacesAsPerm(r));
[ (1,3,2), (1,4,5,6,2), (4,7,9,8,5), (1,4,7,10,3), (2,6,11,10,3),
(5,8,12,11,6), (8,12,9), (7,10,11,12,9) ]
```

1.1.2 Construction from an octahedron

Idea behind the construcion

We want to construct the simplicial parallelepiped using tetrahedrial extensions. Starting from an octahedron we subdivide two disjoint faces into 3-gons, whereby the faces of the surfaces are represented by their sets of vertices. Subdividing a face of a simplicial surface gives rise to a new surface which can be seen as subdivision of the given surface.



Example

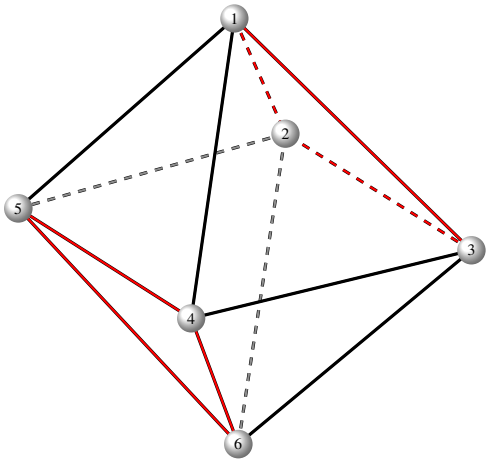
```
gap> O1:=Octahedron();
simplicial surface ( 6 vertices, 12 edges, 8 faces )

gap> VO1:=VerticesOfFaces(O1);
[ [ 1, 2, 3 ], [ 2, 5, 6 ], [ 1, 2, 5 ], [ 2, 3, 6 ], [ 1, 4, 5 ],
  [ 3, 4, 6 ], [ 1, 3, 4 ], [ 4, 5, 6 ] ]
```

Determine the faces which will be replaced by attaching the tetrahedra. Therefore search for two faces that share no common vertex.

Example

```
gap> Filtered(VO1,r->Intersection(VO1[1],r)=[]);
[ [ 4, 5, 6 ] ]
```



Compute the tetrahedra's vertices of faces so that the octahedron and each tetrahedron have exactly three vertices in common.

Example

```
gap> FT1:=Combinations([1,2,3,7],3);
[ [ 1, 2, 3 ], [ 1, 2, 7 ], [ 1, 3, 7 ], [ 2, 3, 7 ] ]
gap> FT2:=Combinations([4,5,6,8],3);
[ [ 4, 5, 6 ], [ 4, 5, 8 ], [ 4, 6, 8 ], [ 5, 6, 8 ] ]
```

Define the symmetric difference `sydif()` of sets A and B

Example

```
gap> symdif:=function(A,B)
> return Difference(Union(A,B),Intersection(A,B));
> end;
function( A, B ) ... end
```

Why is the symmetric difference helpful for the solution of this problem?

The symmetric difference of the set of faces of the octahedron and the set of faces of one of the above tetrahedra removes the face [1,2,3] resp. [4,5,6] which the octahedron and the tetrahedron have in common and replaces it with the three remaining faces of the tetrahedron. So by using the symmetric difference we obtain the vertices of faces of the desired simplicial parallelepiped.

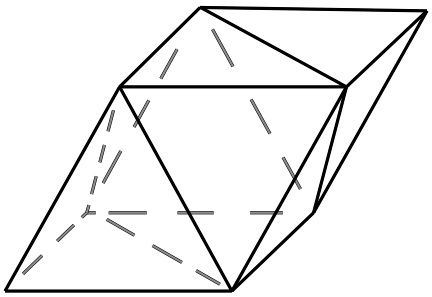
Example

```
gap> symdif(VO1,FT1);
[ [ 1, 2, 5 ], [ 1, 2, 7 ], [ 1, 3, 4 ], [ 1, 3, 7 ], [ 1, 4, 5 ],
[ 2, 3, 6 ], [ 2, 3, 7 ], [ 2, 5, 6 ], [ 3, 4, 6 ], [ 4, 5, 6 ] ]
gap> symdif(last,FT2);
[ [ 1, 2, 5 ], [ 1, 2, 7 ], [ 1, 3, 4 ], [ 1, 3, 7 ], [ 1, 4, 5 ],
 [ 2, 3, 6 ], [ 2, 3, 7 ], [ 2, 5, 6 ], [ 3, 4, 6 ], [ 4, 5, 8 ],
 [ 4, 6, 8 ], [ 5, 6, 8 ] ]
```

Finally construct the simplicial parallelepiped

Example

```
gap> PEn:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface ( vertices, 18 edges, 12 faces )
```



If the first and second construction were carried out correctly, the constructed simplicial surfaces must be isomorphic.

Example

```
gap> IsIsomorphic(PE,PEn);
true
```

1.1.3 Construction from an octahedron (2)

Idea behind the construcion

As seen in the previous construction two tetrahedra can be attached to an octahedron to construct the simplical parallelepiped. In this subsection we present an alternative construction. The required simplicial surfaces are computed directly instead of manipulating their vertices in faces to create the simplicial parallelepiped. This is achieved by using flags. A flag is a triple [V, E, F] satisfying the following conditions:

- the vertex V is incident to the edge E
- the edge E is incident to the face F
- the vertex V is incident to the face F.

We can compute flags by calling

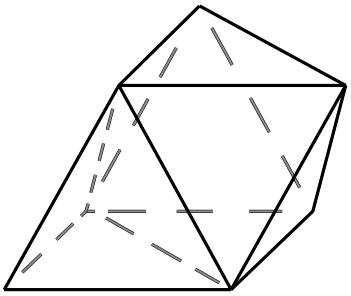
Example

```
gap> Flags(Tetrahedron());
[ [ 1, 1, 1 ], [ 1, 1, 2 ], [ 1, 2, 1 ], [ 1, 2, 4 ], [ 1, 3, 2 ],
  [ 1, 3, 4 ], [ 2, 1, 1 ], [ 2, 1, 2 ], [ 2, 4, 1 ], [ 2, 4, 3 ],
  [ 2, 5, 2 ], [ 2, 5, 3 ], [ 3, 2, 1 ], [ 3, 2, 4 ], [ 3, 4, 1 ],
  [ 3, 4, 3 ], [ 3, 6, 3 ], [ 3, 6, 4 ], [ 4, 3, 2 ], [ 4, 3, 4 ],
  [ 4, 5, 2 ], [ 4, 5, 3 ], [ 4, 6, 3 ], [ 4, 6, 4 ] ]
```

Use these flags to attach the tetrahedron to the octahedron with the function `ConnectedFaceSum()`

Example

```
gap> PE3:=ConnectedFaceSum(Tetrahedron(),[1,1,1],Octahedron(),[1,1,1]);
[simplicial surface (7 vertices, 15 edges, 10 faces)]
```



We have to find the face which has to be replaced by the second tetrahedron. How do we find it? The vertices of this face are not incident to the already attached tetrahedron, thus their face degree must be 4.

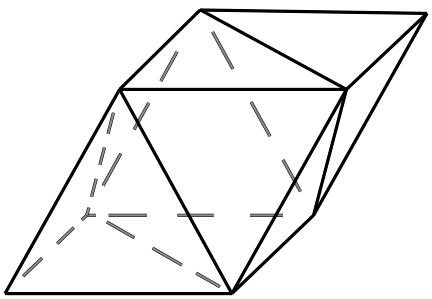
Example

```
gap> FaceDegreesOfVertices(PE3);
[ ,,, 3,,,,,, 4, 4, 4, 5, 5, 5 ]
gap> Vertices(PE3);
[ 4, 10, 11, 12, 13, 14, 15 ]
gap> Intersection(List([10,11,12],i->FacesOfVertex(PE3,i)));
[ 14 ]
```

Find the edge E of the corresponding flag `[10, E, 14]` to compute the simplicial parallelepiped

Example

```
gap> Intersection(EdgesOfFace(PE3,14),EdgesOfVertex(PE3,10));
[ 16, 17 ]
gap> PE3:=ConnectedFaceSum(Tetrahedron(),[1,1,1],PE3,[10,16,14]);
simplicial surface (8 vertices, 18 edges, 12 faces)
```



The vertex counter verifies that the constructed surface is indeed the simplicial parallelepiped

Example

```
gap> VertexCounter(PE3);
[ [ 3, 2 ], [ 5, 6 ] ]
```

The numbering of the vertices etc. can and should be improved in the sense that

- the vertices of the surface are given by the set [1..8]
- the edges of the surface are given by the set [1..18]
- the faces of the surface are given by the set [1..12]

This can be achieved by mapping each vertex to their position in `Vertices(PE3)`

Example

```
gap> VV:=Vertices(PE3);
[ 4, 10, 19, 20, 21, 22, 23, 24 ]
gap> LL:=VerticesOfFaces(PE3);
[ , [ 4, 22, 23 ], [ 4, 23, 24 ], [ 4, 22, 24 ],,,, [ 10, 19, 20 ],
 [ 10, 20, 21 ], [ 10, 19, 21 ],,,, [ 20, 23, 24 ], [ 19, 20, 23 ],
 [ 20, 21, 24 ], [ 19, 22, 23 ], [ 21, 22, 24 ], [ 19, 21, 22 ] ]
gap> LL:=Set(LL);
[ [ 4, 22, 23 ], [ 4, 22, 24 ], [ 4, 23, 24 ], [ 10, 19, 20 ],
 [ 10, 19, 21 ], [ 10, 20, 21 ], [ 19, 20, 23 ], [ 19, 21, 22 ],
 [ 19, 22, 23 ], [ 20, 21, 24 ], [ 20, 23, 24 ], [ 21, 22, 24 ] ]
gap> LL:=List(LL,r->List(r,i->Position(VV,i)));
[ [ 1, 6, 7 ], [ 1, 6, 8 ], [ 1, 7, 8 ], [ 2, 3, 4 ], [ 2, 3, 5 ],
 [ 2, 4, 5 ], [ 3, 4, 7 ], [ 3, 5, 6 ], [ 3, 6, 7 ], [ 4, 5, 8 ],
 [ 4, 7, 8 ], [ 5, 6, 8 ] ]
gap> PE3:=SimplicialSurfaceByVerticesInFaces(LL);
simplicial surface ( 8 vertices, 18 edges, 12 faces)
gap> Vertices(PE3);
[ 1, 2, 3, 4, 5, 6, 7, 8 ]
```

Verify that the resulting sphere is still isomorphic to the parallelepiped

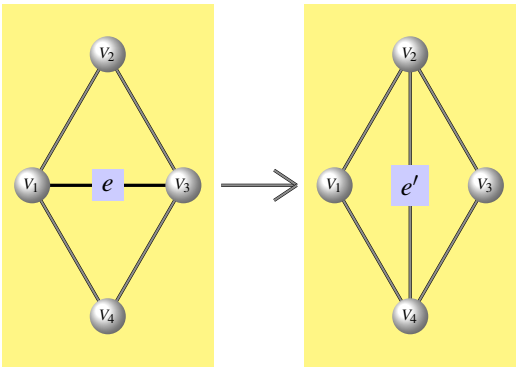
Example

```
gap> FaceDegreesOfVertices(PE3);
[ 3, 3, 5, 5, 5, 5, 5, 5 ]
gap> IsIsomorphic(PE3, PEn);
true
```

1.1.4 Construction from double-6-gon via butterfly turning (1)

idea behind the `construction`

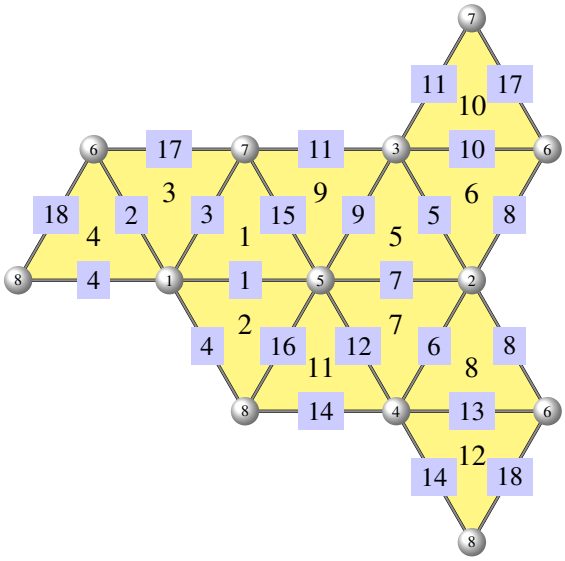
This construction uses the concept of edge-turns. If an inner edge e gives rise to a butterfly (e.g. if the surface is vertex faithful), a new surface can be created by turning the inner edge e. This amounts to replacing e by a new edge e' connecting the other two vertices of the butterfly. So it has the same number of faces, edges and vertices, but the vertex degrees in four positions will change by +-1 i.e. the vertex degrees of the vertices incidient to e decrease and the degrees of the vertices incident to e' increase by 1. We shall refer to the edge e' as the orthogonal edge. In this construction we perform edge-turns on the double-6-gon to obtain the simplicial parallelepiped.



Construct the double-6-gon by specifying its faces via its incident vertices

Example

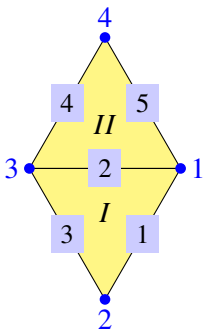
```
gap> VertInFacDouble6gon:= [ [ 1, 2, 7 ], [ 2, 7, 8 ], [ 1, 2, 3 ], [ 2, 3, 8 ],
> [ 1, 3, 4 ], [ 3, 4, 8 ], [ 1, 4, 5 ], [ 4, 5, 8 ], [ 1, 5, 6 ], [ 5, 6, 8 ],
> [ 1, 6, 7 ], [ 6, 7, 8 ] ];;
gap> PE:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
```



Construction of a butterfly

Example

```
gap> bufl:=function()
> return SimplicialSurfaceByVerticesInFaces([[1,2,3],[2,3,4]]);
> end;
function(  ) ... end
gap> bufl();
simplicial surface ( 4 vertices, 5 edges, 2 faces)
```



Define the function `checkbufl()` which checks whether a given edge is an inner edge giving rise to a butterfly in the surface.

Example

```
gap> checkbufl:=function(S,e)
> local sS;
> if not (e in InnerEdges(S)) then Error("not inner");fi;
> sS:=FacesOfEdge(S,e);
> sS:=SubcomplexByFaces(S,sS);
> return IsIsomorphic(bufl(),sS);
> end;
function( S, e ) ... end

gap> checkbufl(PE,2);
true
```

If the function returns true, the butterfly of the edge e can be turned. Here the program to perform an edgeturn is introduced:

Example

```
gap> turnedge:=function(S,e)
> local sS,sB,v,ee;
> sB:=SubcomplexByFaces(S,FacesOfEdge(S,e));
> ee:=Difference(Edges(sB),[e]);
> v:=Intersection(VerticesOfEdge(sB,ee[1]),VerticesOfEdge(S,e))[1];
> sS:=Difference(Faces(S),FacesOfEdge(S,e));
> sS:=SubcomplexByFaces(S,sS);
> return JoinBoundaries(sS,[v,ee[1]],sB,
> [Difference(VerticesOfEdge(sB,ee[1]),[v])[1],ee[1]]);
> end;
function( S, e ) ... end
```

The vertex-counter of the simplicial parallelepiped is [[3,2],[5,6]]. So turn an edge incident to vertex 1 to create a vertex of degree 3 and vertices of degree 5.

Example

```
gap> VerticesOfEdges(PE);
[ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 1, 5 ], [ 1, 6 ], [ 1, 7 ], [ 2, 3 ], [ 2, 7 ],
  [ 2, 8 ], [ 3, 4 ], [ 3, 8 ], [ 4, 5 ], [ 4, 8 ], [ 5, 6 ], [ 5, 8 ], [ 6, 7 ],
  [ 6, 8 ], [ 7, 8 ] ]
gap> turnedge(PE,1);
[ simplicial surface (8 vertices, 18 edges, and 12 faces)
    , ( v26, E27, v27, E28, v28, E29, v29, E30, v26 )
```

```
    , 18 ]
gap> PE1:=last[1];
simplicial surface (8 vertices, 18 edges, and 12 faces)
```

Compute the face degrees of the vertices

Example

```
gap> FaceDegreesOfVertices(PE1);
[ ,,, 4, 4, 4,, 6,,,,,,,,,,,,,,,, 5, 5, 3, 5 ]
```

Since the face degrees of the vertices 4,5,6 and 8 is neither 3 or 5, we need to find an inner edge so that the resulting butterfly contains the vertices [4,5,6,8].

Example

```
gap> Filtered(Faces(PE1),f->IsSubset([4,5,6,8],VerticesOfFace(PE1,f)));
[ 8, 10 ]
gap> Intersection(EdgesOfFace(PE1,8),EdgesOfFace(PE1,10));
[ 15 ]
```

From this information we can compute the simplicial parallelepiped by turning edge 15

Example

```
gap> turnedge(PE1,15);
[ simplicial surface (8 vertices, 18 edges, and 12 faces)
    , ( v39, E48, v40, E49, v41, E50, v42, E51, v39 )
    , 30 ]
gap> PE3:=last[1];
simplicial surface (8 vertices, 18 edges, and 12 faces)
```

By calling the vertex counter we see that the sphere is indeed isomorphic to the simplicial parallelepiped.

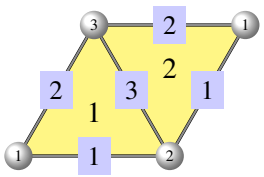
Example

```
gap> VertexCounter(PE3);
[ [ 3, 2 ], [ 5, 6 ] ]
```

1.1.5 Construction using the barycentric subdivision and edge-turns

Idea behind the construcion

The Janus Head is a closed simplicial surface with 2 faces whose barycentric subdivision is a sphere with 12 faces. From this information we can deduce that performing edgeturns on the resulting surface gives rise to a simplicial parallelepiped.



Compute the Janus Head

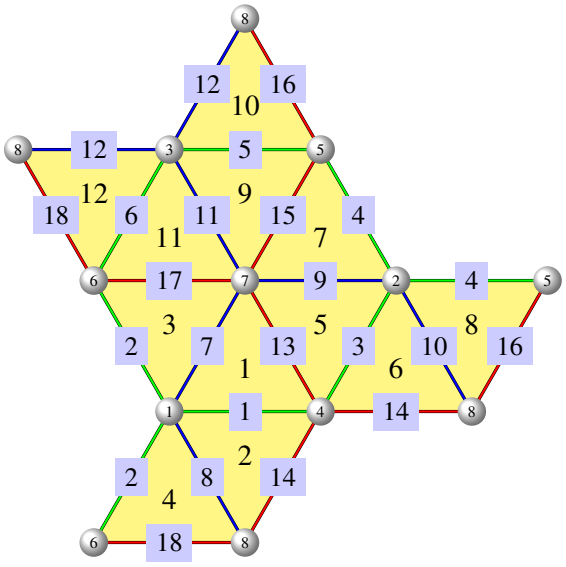
Example

```
gap> J:=JanusHead();
simplicial surface ( 3 vertices, 3 edges, 2 faces)
gap> Checkbufl(J,1);
false
```

Constructing the Janus-Head's barycentric subdivision

Example

```
gap> BJ:=FlagSurface(J);
coloured surface (MMM with 8 vertices, 18 edges and 12 faces)
gap> FaceDegreesOfVertices(BJ);
[ 4, 4, 4, 4, 4, 4, 6, 6 ]
```



Which edges have to be turned to obtain the parallelepiped? The vertex counter of the surface `BJ` is `[[4,6],[6,4]]`. To create a sphere with the vertex counter `[[3,2],[5,6]]`, we first have to find an edge incident to vertices with degree 6 and 4. Their degree will decrease by 1 and the vertex degrees of the vertices incident to the orthogonal edge will increase by 1.

Example

```
VerticesOfEdges(BJ);
[ [ 1, 4 ], [ 1, 5 ], [ 2, 4 ], [ 2, 6 ], [ 3, 5 ], [ 3, 6 ], [ 1, 7 ],
  [ 1, 8 ], [ 2, 7 ], [ 2, 8 ], [ 3, 7 ], [ 3, 8 ], [ 4, 7 ], [ 4, 8 ],
  [ 5, 7 ], [ 5, 8 ], [ 6, 7 ], [ 6, 8 ] ]
gap> BJ1:=turnedge(BJ,18)[1];
simplicial surface (8 vertices, 18 edges, 12 faces)
```

Computing face degrees of the vertices

Example

```
gap> FaceDegreesOfVertices(BJ1);
[ 4,,, 4, 4,, 6,,,,,,,,,,,,,,,,,, 3, 5, 5, 5 ]
```

Find an edge incident to a vertex with face degree 4 and another vertex with degree 6. For that we may use

Example

```
gap> VerticesOfEdges(BJ1);
[ [ 1, 4 ], [ 1, 5 ], [ 4, 27 ],, [ 5, 29 ],, [ 1, 7 ], [ 1, 28 ], [ 7, 27 ],
 , [ 7, 29 ],, [ 4, 7 ], [ 4, 28 ], [ 5, 7 ], [ 5, 28 ], [ 7, 26 ],,,,,,,,,,
 ,,,,,,,, [ 27, 29 ], [ 26, 27 ], [ 27, 28 ], [ 28, 29 ], [ 26, 29 ] ]
gap> Position(last,[1,7]);
7
gap> JB2:=turnedge(BJ1,7)[1];
simplicial surface (8 vertices, 18 edges, 12 faces)
gap> FaceDegreesOfVertices(JB2);
[ ,,,,,,,,,,,,,,,,,,,,,,, 3, 5, 5, 5,,,,,,,,,,,,,,,,, 3, 5, 5, 5 ]
```

Check whether the constructed surface is isomorphic to the simplicial parallelepiped

Example

```
gap> IsIsomorphic(JB2,PEs);
true
```

The surface is vertex-faithful. Hence we easily can clean up the sets of vertices, edges and faces

Example

```
gap> V:=Vertices(JB2);
[ 26, 27, 28, 29, 47, 48, 49, 50 ]
gap> FF:=VerticesOfFaces(JB2);
[ , [ 28, 47, 48 ],, [ 28, 47, 50 ], [ 27, 48, 49 ], [ 27, 28, 48 ],
 [ 26, 27, 49 ],, [ 29, 49, 50 ], [ 28, 29, 50 ], [ 26, 29, 49 ],,,,,,,,,,,
 ,, [ 26, 27, 29 ],,,, [ 27, 28, 29 ],,,,,,,,,,, [ 47, 48, 50 ],,
 [ 48, 49, 50 ] ]
gap> FF:=Set(FF);
[ [ 26, 27, 29 ], [ 26, 27, 49 ], [ 26, 29, 49 ], [ 27, 28, 29 ],
 [ 27, 28, 48 ], [ 27, 48, 49 ], [ 28, 29, 50 ], [ 28, 47, 48 ],
 [ 28, 47, 50 ], [ 29, 49, 50 ], [ 47, 48, 50 ], [ 48, 49, 50 ] ]
gap> FF:=List(FF,r->List(r,i->Position(V,i)));
[ [ 1, 2, 4 ], [ 1, 2, 7 ], [ 1, 4, 7 ], [ 2, 3, 4 ], [ 2, 3, 6 ],
 [ 2, 6, 7 ], [ 3, 4, 8 ], [ 3, 5, 6 ], [ 3, 5, 8 ], [ 4, 7, 8 ],
 [ 5, 6, 8 ], [ 6, 7, 8 ] ]
gap> PEt:=SimplicialSurfaceByVerticesInFaces(FF);
simplicial surface (8 vertices, 18 edges, and 12 faces)
gap> FaceDegreesOfVertices(PEt);
[ 3, 5, 5, 5, 3, 5, 5, 5 ]
gap> Vertices(PEt);
[ 1, 2, 3, 4, 5, 6, 7, 8 ]
```

We want to understand the barycentric subdivision of the Janus head better. Indeed we shall see that it is isomorphic to the double 6-gon.

Define a function that creates a n-gon

Example

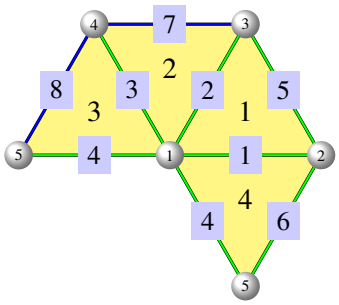
```
gap> ngon:=function(n)
> local c,L;
> c:=CycleFromList([1..n]);
> L:=List([1..n],i->[i,i^c,n+1]);
> return SimplicialSurfaceByVerticesInFaces(L);
```

```
> end;
function( n ) ... end
```

Compute the 4-gon

Example

```
gap> ngon(4);
simplicial surface (5 vertices, 8 edges, and 4 faces)
```



Define a function to create a double-n-gon

Example

```
gap> Doublengon:=function(n)
> local c,L1,L2;
> c:=CycleFromList([1..n]);
> L1:=List([1..n],i->[i,i^c,n+1]);
> L2:=List([1..n],i->[i,i^c,n+2]);
> L1:=Union(L1,L2);
> return SimplicialSurfaceByVerticesInFaces(L1);
> end;
function( n ) ... end
```

Verify that the barycentric subdivision of the Janus Head is isomorphic to the double-6-gon

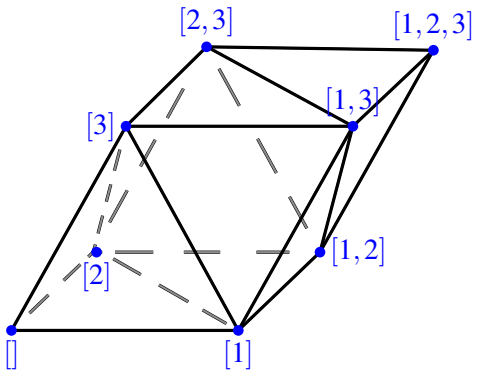
Example

```
gap> Doublengon(4);
simplicial surface (6 vertices, 12 edges, and 8 faces)
gap> IsIsomorphic(Doublengon(4),Octahedron());
true
gap> IsIsomorphic(Doublengon(6),BJ);
true
gap> FaceDegreesOfVertices(Doublengon(7));
[ 4, 4, 4, 4, 4, 4, 4, 7, 7 ]
```

1.1.6 Construction using power sets

Idea behind the construcion

The construction of the simplicial parallelepiped can be achieved by using the power set of `[1,2,3]`. The power set forms the set of vertices of the parallelepiped. So we refer to the different subsets of [1,2,3] as vertices. Since the surface is vertex-faithful, we see that constructing the surface can be achieved by determining the vertices of faces. The set of vertices of a face can be written as [V1,V2,V3] where the subsets V1 and V2 have the same cardinalty and V3 is either the intersection or the union of V1 and V2.



Compute the ordered power set such that we can easily access subsets of a given size.

Example

```
gap> powset:=List([0..3],i->Combinations([1,2,3],i));
[ [ [  ] ], [ [ 1 ], [ 2 ], [ 3 ] ], [ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ] ],
  [ [ 1, 2, 3 ] ] ]
```

To compute the faces incident to exactly two vertices of cardinality 1, we have to tell GAP to compute the union and intersection of those sets

Example

```
gap> t1:=List(Combinations(C[2],2),r->Union(r, [Intersection(r)]));
[ [ [  ], [ 1 ], [ 2 ] ], [ [  ], [ 1 ], [ 3 ] ], [ [  ], [ 2 ], [ 3 ] ] ]
gap> t2:=List(Combinations(C[2],2),r->Union(r, [Union(r)]));
[ [ [ 1 ], [ 1, 2 ], [ 2 ] ], [ [ 1 ], [ 1, 3 ], [ 3 ] ],
  [ [ 2 ], [ 2, 3 ], [ 3 ] ] ]
```

Conpute the faces incident to exactly two vertices of cardinality 2

Example

```
gap> t3:=List(Combinations(C[3],2),r->Union(r, [Intersection(r)]));
[ [ [ 1 ], [ 1, 2 ], [ 1, 3 ] ], [ [ 1, 2 ], [ 2 ], [ 2, 3 ] ],
  [ [ 1, 3 ], [ 2, 3 ], [ 3 ] ] ]
gap> t4:=List(Combinations(C[3],2),r->Union(r, [Union(r)]));
[ [ [ 1, 2 ], [ 1 .. 3 ], [ 1, 3 ] ], [ [ 1, 2 ], [ 1 .. 3 ], [ 2, 3 ] ],
  [ [ 1 .. 3 ], [ 1, 3 ], [ 2, 3 ] ] ]
```

Compute the set of faces represented by their set of vertices

Example

```
gap> t:=Union([t1,t2,t3,t4]);
[ [ [  ], [ 1 ], [ 2 ] ], [ [  ], [ 1 ], [ 3 ] ], [ [  ], [ 2 ], [ 3 ] ],
  [ [ 1 ], [ 1, 2 ], [ 1, 3 ] ], [ [ 1 ], [ 1, 2 ], [ 2 ] ],
  [ [ 1 ], [ 1, 3 ], [ 3 ] ], [ [ 1, 2 ], [ 1 .. 3 ], [ 1, 3 ] ],
  [ [ 1, 2 ], [ 1 .. 3 ], [ 2, 3 ] ], [ [ 1, 2 ], [ 2 ], [ 2, 3 ] ],
  [ [ 1 .. 3 ], [ 1, 3 ], [ 2, 3 ] ], [ [ 1, 3 ], [ 2, 3 ], [ 3 ] ],
  [ [ 2 ], [ 2, 3 ], [ 3 ] ] ]
```

From this information we can already reclaim the simplicial parallelepiped, but we notice that the vertices' labelling can be improved by calling

Example

```
gap> powset:=C[1];
[ [  ] ]
gap> Append(powset,C[2]);
gap> Append(powset,C[3]);
gap> Append(powset,C[4]);
gap> powset; #powset is the power set
[ [  ], [ 1 ], [ 2 ], [ 3 ], [ 1, 2 ], [ 1, 3 ], [ 2, 3 ], [ 1, 2, 3 ] ]
gap> verticesinfaces:=List(t,r->List(r,i->Position(powset,i)));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 5, 6 ], [ 2, 5, 3 ],
  [ 2, 6, 4 ], [ 5, 8, 6 ], [ 5, 8, 7 ], [ 5, 3, 7 ], [ 8, 6, 7 ],
  [ 6, 7, 4 ], [ 3, 7, 4 ] ]
```

Construct the simplicial parallelepiped

Example

```
gap> PEs:=SimplicialSurfaceByVerticesInFaces(verticesinfaces);
simplicial surface (8 vertices, 18 edges, and 12 faces)
gap> VertexCounter(PEs);
[ [ 3, 2 ], [ 5, 6 ] ]
```

Note, this construction can be generalised to other sets. For the powerset of [1,2], the resulting surface is isomorphic to the butterfly.

1.2 Facegraph of simplicial surfaces

Problems :

- Analyse the face graph of

1. a tetrahedron
2. an octahedron

Theoretical background

- Vertex-faithful surfaces and boolean operations

Frequently used commands

- AutomorphismGroup() (**??**)
- AutomorphismGroupOnEdges() (**??**)
- AutomorphismGroupOnFaces() (**??**)
- EdgesOfFaces() (local version: EdgesOfFace()) (**??**)
- EdgesOfVertices() (local version: Edges) (**??**)
- EulerCharacteristic() (**??**)
- FacesOfEdges() (local version: FacesOfEdge()) (**??**)

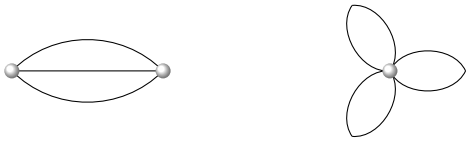
- ImageOfVertex() (**??**)
- IsOrientable() (**??**)
- PolygonalComplexByDownwardIncidence() (**??**)
- UmbrellaPathsOfVertices() (**??**)
- VertexCounter() (local version: DegreeOfVertex()) (**??**),
- VerticesOfEdges() (local Version: VerticesOfEdge()) (**??**)
- VerticesOfFaces() (local Version: VerticesOfFace()) (**??**)

Less frequently used commands

- Tetrahedron() (**??**)
- Octahedron() (**??**)

Mathematical details :

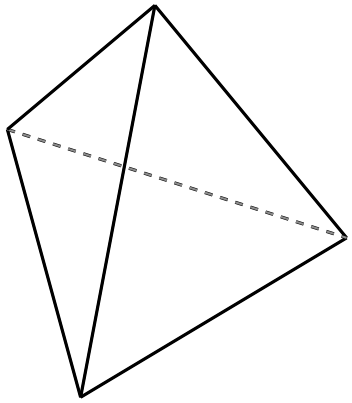
In this chapter we shall familiarize ourselves with the face graph of a simplicial surface. The face graph has the set of faces of the simplicial surface as it's set of vertices and the set of edges of the simplicial surface as it's edges. Two vertices F,F' of the face graph are connected through an edge if and only if the corresponding faces form a set [F,F'] which is an element of the set containing the faces incident to an edge.



For example the face graph of the one-face is given by a graph with one vertex and three loops and the face graph of the Janus-Head is a graph with two vertices which are connected through three edges. Note, the map resulting through mapping a simplicial surface on it's face graph is not bijective, since non-isomorphic surfaces can have isomorphic face graphs. But restricting the map to spheres only delivers a bijection. We shall refer to two faces of a simplicial surface as opposite if they share no common vertex. By looking at the cube as an example, we see that the faces 1 and 6, 2 and 5, 3,4 form pairs of opposite faces.

1.2.1 Face graph of a tetrahedron

For the purpose of handling face graphs of simplicial we start with the tetrahedron as an example.



Compute a tetrahedron

Example

```
gap> T:=Tetrahedron();
simplicial surface (4 vertices, 6 edges, and 4 faces)
```

Define a function to compute the ordinal symbol of a simplicial surface

Example

```
gap> Symbol:=function(S)
> return [Size(Vertices(S)),Size(Edges(S)),Size(Faces(S))
,VerticesOfEdges(S),EdgesOfFaces(S)];
> end;
function( S ) ... end
```

Compute the symbol of the Tetrahedron

Example

```
gap> Symbol(T);
[ 4, 6, 4, [ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ],
  [ [ 1, 2, 4 ], [ 1, 3, 5 ], [ 4, 5, 6 ], [ 2, 3, 6 ] ] ]
```

Note, we can always replace a simplicial surface with n vertices, k edges and m faces by an isomorphic surface where the set of vertices is given by [1..n], the set of edges is given by [1..k] and the set of faces is given by [1..m] by using the function CanonicalRepresentativeOfPolygonalSurface()

Example

```
gap> XX:=CanonicalRepresentativeOfPolygonalSurface(T);
[ simplicial surface (4 vertices, 6 edges, and 4 faces)
   , <polygonal morphism> ]
gap> Symbol(XX[1]);
[ 4, 6, 4, [ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ], [ 1, 4 ], [ 2, 4 ], [ 3, 4 ] ],
  [ [ 1, 2, 3 ], [ 1, 4, 5 ], [ 2, 4, 6 ], [ 3, 5, 6 ] ] ]
```

Compute the images of the vertices of T under the isomorphism mapping T on XX

Example

```
gap> List(Vertices(XX[1]),i->ImageOfVertex(XX[2],i));
[ 2, 1, 3, 4 ]
```

We could do the same for the edges and faces, but since the vertices, etc are already given by [1..4] etc. we shall continue our examinations with T. The face graph can be constructed from the ordinal symbol of T We already know that the set of vertices is given by the set of faces [1,2,3,4] and we can compute the edges of the face graph by calling

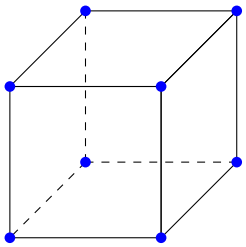
Example

```
gap> FacesOfEdges(T);
[ [ 1, 2 ], [ 1, 4 ], [ 2, 4 ], [ 1, 3 ], [ 2, 3 ], [ 3, 4 ] ]
```

Note, the edges of the face graph are also determined by the following list:

Example

```
gap> EdgesOfFaces(T);
[ [ 1, 2, 4 ], [ 1, 3, 5 ], [ 4, 5, 6 ], [ 2, 3, 6 ] ]
```



Both lists are partual information of the ordinal symbol of the sphere. We simply omitted the information for the vertices. But we can obtain the vertices from the face graph closed edge-face-paths containing exactly three faces and three edges.

Example

```
gap> VerticesOfEdges(T);
[ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ]
gap> UmbrellaPathsOfVertices(T);
[ ( e1, F1, e2, F4, e3, F2, e1 )
   , ( e1, F1, e4, F3, e5, F2, e1 )
   , ( e2, F1, e4, F3, e6, F4, e2 )
   , ( e3, F2, e5, F3, e6, F4, e3 )
]
```

Note, here the vertices are given in terms of edges and faces, i.e. only in terms of the face graph. Compute the corresponding closed edge-paths

Example

```
gap> List(last,r->EdgesAsList(r));
[ [ 1, 2, 3, 1 ], [ 1, 4, 5, 1 ], [ 2, 4, 6, 2 ], [ 3, 5, 6, 3 ] ]
```

How can we reconstruct the VerticesOfEdges with knowledge of the face-graph? Let the number of the vertices be encoded in the position of the closed paths in the above list. So for example vertex 1 is incident to the edges 1, 2 and 3. This helps us to compute the desired set.

Furthermore define the function `Umbre2VeOfEd` which has a list of closed edge-face-paths as input and returns the corresponding vertices of edges of a given surface.

Example

```
gap> Umbre2VeOfEd:=function(R)
> local L,ne,nv,i,j,rr;
> ne:=Size(Union(R));#Number of edges
> nv:=Size(R);#Number of vertices
> L:=List([1..ne],i->[]);
> for i in [1..nv] do
> for j in R[i] do
> L[j]:=Union(L[j],[i]);
> od;
> od;
> return L;
> end;
function( R ) ... end
```

Now compute the vertices of edges of the tetrahedron T by using the function defined above

Example

```
gap> UmbrellaPathsOfVertices(T);
[ ( e1, F1, e2, F4, e3, F2, e1 )
   , ( e1, F1, e4, F3, e5, F2, e1 )
   , ( e2, F1, e4, F3, e6, F4, e2 )
   , ( e3, F2, e5, F3, e6, F4, e3 )
]
gap> List(last,r->EdgesAsList(r));
[ [ 1, 2, 3, 1 ], [ 1, 4, 5, 1 ], [ 2, 4, 6, 2 ], [ 3, 5, 6, 3 ] ]
gap> Umbre2VeOfEd(last);
[ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ]
```

Let us verify, that this set is indeed the vertices of edges of T

Example

```
gap> VerticesOfEdges(T);
[ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ]
```

Let us see whether there exists another simplicial surface X so that X and T have the same face graph. gap> #T has 6 edges and hence contains 6 butterflies. The butterflies come gap> #in pairs which have the same boundary. This gives us a net of gap> #three vertex defining paths (why?) We want to examine the face that two non isomorphic simplicial surfaces can have isomorphic face graphs. So we will construct a simplicial surface which has the same face graph as our tetrahedron T.

Idea behind the construction

Since there exist 6 edges belonging to the sphere T, there are 6 butterflies contained in the tetrahedron. The butterflies can be sorted into three pairs so that the two butterflies of the corresponding pair of edges share the same boundary vertex-edge path.

By interpreting the vertices of those paths as faces, we obtain three vertex defining paths.

Example

```
gap> L:=List(Edges(T),r->Difference(Union(List(FacesOfEdge(T,r),f->EdgesOfFace(T,f))),[r]));
[ [ 2, 3, 4, 5 ], [ 1, 3, 4, 6 ], [ 1, 2, 5, 6 ], [ 1, 2, 5, 6 ],
```

```
 [ 1, 3, 4, 6 ], [ 2, 3, 4, 5 ] ]
gap> L:=Set(L);
[ [ 1, 2, 5, 6 ], [ 1, 3, 4, 6 ], [ 2, 3, 4, 5 ] ]
gap> Umbre2VeOfEd(last);
[ [ 1, 2 ], [ 1, 3 ], [ 2, 3 ], [ 2, 3 ], [ 1, 3 ], [ 1, 2 ] ]
gap> nn:=last;;
```

Note that the elements of L are not stricly paths but the sets of edges that occur in a relevant path. But that is enough for our purpose of constructing the desired simplicial surface.

Example

```
gap> TT:=PolygonalComplexByDownwardIncidence(nn,EdgesOfFaces(T));
simplicial surface (3 vertices, 6 edges, and 4 faces)
gap> EulerCharacteristic(TT);
1
gap> IsOrientable(TT);
false
```

Since the Euler-Characteristic of the surfaces differ and the constructed surface is not orientable it cannot be isomorphic to an tetrahedron

Of course it would be desiralbe to compute all simplicial surfaces having the same face graph up to isomorphism.

1.2.2 Face graph of an octahedron

Let us analyse the face graph of an octahedron as second example.

Example

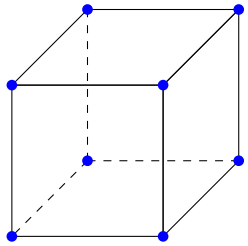
```
gap> ok:=Octahedron();
simplicial surface (6 vertices, 12 edges, and 8 faces)
```

Compute the octahedron's symbol

Example

```
gap> Symbol(ok);
[ 6, 12, 8,
 [ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 1, 5 ], [ 2, 3 ], [ 2, 5 ], [ 2, 6 ],
     [ 3, 4 ], [ 3, 6 ], [ 4, 5 ], [ 4, 6 ], [ 5, 6 ] ],
 [ [ 1, 2, 5 ], [ 6, 7, 12 ], [ 1, 4, 6 ], [ 5, 7, 9 ], [ 3, 4, 10 ],
     [ 8, 9, 11 ], [ 2, 3, 8 ], [ 10, 11, 12 ] ] ]
gap> okc:=FacesOfEdges(ok);
[ [ 1, 3 ], [ 1, 7 ], [ 5, 7 ], [ 3, 5 ], [ 1, 4 ], [ 2, 3 ], [ 2, 4 ],
 [ 6, 7 ], [ 4, 6 ], [ 5, 8 ], [ 6, 8 ], [ 2, 8 ] ]
```

Note the edge graph of a simplicial surface is the graph resulting from the vertices and edges of the surfaces being the vertices and edges of the graph. Two vertices V1,V2 connected through an edge in the graph if they are incident vertices in the surface. The face graph of an octahedron is isomorphic to the edge graph of a cube.



The vertices of the octahedron are determined by the closed paths containing exactly four faces and edges.

Example

```
gap> UmbrellaPathsOfVertices(ok);
[ ( e1, F1, e2, F7, e3, F5, e4, F3, e1 )
   , ( 1, F1, e5, F4, e7, F2, e6, F3, e1 )
   , ( e2, F1, e5, F4, e9, F6, e8, F7, e2 )
   , ( e3, F5, e10, F8, e11, F6, e8, F7, e3 )
   , ( e4, F3, e6, F2, e12, F8, e10, F5, e4 )
   ,  ( e7, F2, e12, F8, e11, F6, e9, F4, e7)
]
gap> List(last,r->EdgesAsList(r));
[ [ 1, 2, 3, 4, 1 ], [ 1, 5, 7, 6, 1 ], [ 2, 5, 9, 8, 2 ],
 [ 3, 10, 11, 8, 3 ], [ 4, 6, 12, 10, 4 ], [ 7, 12, 11, 9, 7 ] ]
gap> List(last2,r->FacesAsList(r));
[ [ 1, 7, 5, 3 ], [ 1, 4, 2, 3 ], [ 1, 4, 6, 7 ], [ 5, 8, 6, 7 ],
  [ 3, 2, 8, 5 ], [ 2, 8, 6, 4 ] ]
```

Compute the opposite faces of the octahedron ok

Example

```
gap> VV:=VerticesOfFaces(ok);
[ [ 1, 2, 3 ], [ 2, 5, 6 ], [ 1, 2, 5 ], [ 2, 3, 6 ], [ 1, 4, 5 ],
 [ 3, 4, 6 ], [ 1, 3, 4 ], [ 4, 5, 6 ] ]
gap> List([1..8],i->Filtered([1..8],r->Intersection(VV[i],VV[r])=[]));
[ [ 8 ], [ 7 ], [ 6 ], [ 5 ], [ 4 ], [ 3 ], [ 2 ], [ 1 ] ]
```

Hence the pairs of opposite faces are [1,8], [2,7], [3,6] and [4,5]. gap> # Those vertex-edge-paths avoiding the vertices of such a diagonal gap> # are all in one dihedral class. Taken together they correspond to the vertices of gap> # of a surface in the same edge-face-class as ok:

Example

```
gap> List([1,8],i->EdgesOfFace(ok,i));
[ [ 1, 2, 5 ], [ 10, 11, 12 ] ]
gap> z1:=Difference(Edges(ok),Union(last));
[ 3, 4, 6, 7, 8, 9 ]
gap> List([2,7],i->EdgesOfFace(ok,i));
[ [ 6, 7, 12 ], [ 2, 3, 8 ] ]
gap> z2:=Difference(Edges(ok),Union(last));
[ 1, 4, 5, 9, 10, 11 ]
gap> List([3,6],i->EdgesOfFace(ok,i));
[ [ 1, 4, 6 ], [ 8, 9, 11 ] ]
gap> z3:=Difference(Edges(ok),Union(last));
[ 2, 3, 5, 7, 10, 12 ]
```

```
gap> List([4,5],i->EdgesOfFace(ok,i));
[ [ 5, 7, 9 ], [ 3, 4, 10 ] ]
gap> z4:=Difference(Edges(ok),Union(last));
[ 1, 2, 6, 8, 11, 12 ]
gap> Umbre2VeOfEd([z1,z2,z3,z4]);
[ [ 2, 4 ], [ 3, 4 ], [ 1, 3 ], [ 1, 2 ], [ 2, 3 ], [ 1, 4 ], [ 1, 3 ],
  [ 1, 4 ], [ 1, 2 ], [ 2, 3 ], [ 2, 4 ], [ 3, 4 ] ]
gap> nok:=PolygonalComplexByDownwardIncidence(last,EdgesOfFaces(ok));
simplicial surface (4 vertices, 12 edges, and 8 faces)
```

Compute elementary to verify that nok is not isomorphic to the octahedron ok

Example

```
gap> EulerCharacteristic(nok);
0
gap> VertexCounter(nok);
[ [ 6, 4 ] ]
gap> IsOrientable(nok);
true
```

gap> #Hence it is a torus. One can easily recover it from a hexagonal gap> #in the plane by cutting out a rhombus of appropriate size and gap> #identify opposite sides. A nice challenge in paperfolding: gap> #Embed the barycentric subdivision of this gadget into 3-space in gap> #such a way that all the smaller triangles lie in one plane. gap> gap> #We finish this worksheet by visalizing TT, the projective plane above.

Example

```
gap> TT;
simplicial surface (3 vertices, 6 edges, and 4 faces)
gap> IsOrientable(TT);
false
gap> EulerCharacteristic(TT);
1
```

gap> #We shall construct a 2-fold covering of TT by the octahedron

Example

```
gap> VertexCounter (TT);
[ [ 4, 3 ] ]
gap> AutomorphismGroup(TT);
Group([ (1,2)(5,6)(7,8)(10,11), (1,2)(5,7)(6,8)(12,13), (2,3)(4,5)(8,9)
(11,13) ])
gap> StructureDescription(last);
"S4"
gap> G:=AutomorphismGroup(ok);
Group([ (1,2)(4,6)(8,11)(9,13)(10,12)(14,15)(16,18)(20,23)(22,25), (3,5)
(8,10)(11,12)(14,16)(15,18)(19,21)(20,22)(23,25)(24,26), (2,3)(4,5)(7,8)
(9,10)(12,14)(13,15)(17,18)(20,24)(21,25) ])
gap> StructureDescription(G);
"C2 x S4"
gap> C:=Center(G);
Group([ (1,6)(2,4)(3,5)(7,17)(8,18)(9,13)(10,15)(11,16)(12,14)(19,26)(20,25)
(21,24)(22,23) ])
```

```
gap> Orbits(C,Vertices(ok));
[ [ 1, 6 ], [ 2, 4 ], [ 3, 5 ] ]
gap> Orbits(C,Edges(ok));
[ [ 1, 6 ], [ 2, 4 ], [ 3, 5 ], [ 7, 17 ], [ 8, 18 ], [ 9, 13 ], [ 10, 15 ],
  [ 11, 16 ], [ 12, 14 ] ]
```

gap> #The numbering gets confusing. We do edges and faces by themselves each

Example

```
gap> Ge:=AutomorphismGroupOnEdges(ok);
Group([ (2,5)(3,7)(4,6)(8,9)(10,12), (2,4)(5,6)(8,10)(9,12), (1,2)(3,4)(6,8)
(7,9)(11,12) ])
gap> Ce:=Center(Ge);
Group([ (1,11)(2,12)(3,7)(4,9)(5,10)(6,8) ])
gap> Orbits(Ge,Edges(ok));
[ [ 1, 2, 5, 4, 6, 3, 8, 7, 9, 10, 12, 11 ] ]
gap> Orbits(Ce,Edges(ok));
[ [ 1, 11 ], [ 2, 12 ], [ 3, 7 ], [ 4, 9 ], [ 5, 10 ], [ 6, 8 ] ]
gap> Gf:=AutomorphismGroupOnFaces(ok);
Group([ (2,5)(4,7), (1,3)(2,4)(5,7)(6,8), (2,6)(3,7) ])
gap> Cf:=Center(Gf);
Group([ (1,8)(2,7)(3,6)(4,5) ])
gap> Fo:=Orbits(Cf,Faces(ok));
[ [ 1, 8 ], [ 2, 7 ], [ 3, 6 ], [ 4, 5 ] ]
gap> Eo:=Orbits(Ce,Edges(ok));
[ [ 1, 11 ], [ 2, 12 ], [ 3, 7 ], [ 4, 9 ], [ 5, 10 ], [ 6, 8 ] ]
gap> Vo:=Orbits(C,Vertices(ok));
[ [ 1, 6 ], [ 2, 4 ], [ 3, 5 ] ]
gap> List([1,2,3],i->EdgesOfFace(ok,i));
[ [ 1, 2, 5 ], [ 6, 7, 12 ], [ 1, 4, 6 ] ]
gap> List([6,4,5],i->EdgesOfFace(ok,i));
[ [ 8, 9, 11 ], [ 5, 7, 9 ], [ 3, 4, 10 ] ]
gap> Eo;
[ [ 1, 11 ], [ 2, 12 ], [ 3, 7 ], [ 4, 9 ], [ 5, 10 ], [ 6, 8 ] ]
gap> Fo;
[ [ 1, 8 ], [ 2, 7 ], [ 3, 6 ], [ 4, 5 ] ]
gap> List([1,2,3,4],i->EdgesOfFace(ok,i));
[ [ 1, 2, 5 ], [ 6, 7, 12 ], [ 1, 4, 6 ], [ 5, 7, 9 ] ]
gap> List([8,7,6,5],i->EdgesOfFace(ok,i));
[ [ 10, 11, 12 ], [ 2, 3, 8 ], [ 8, 9, 11 ], [ 3, 4, 10 ] ]
gap> Eo;
[ [ 1, 11 ], [ 2, 12 ], [ 3, 7 ], [ 4, 9 ], [ 5, 10 ], [ 6, 8 ] ]
gap> EoF:=[[1,2,5],[2,3,6],[1,4,6],[3,4,5]];
[ [ 1, 2, 5 ], [ 2, 3, 6 ], [ 1, 4, 6 ], [ 3, 4, 5 ] ]
```

gap> #Now the vertices of Edges:

Example

```
gap> Eo;
[ [ 1, 11 ], [ 2, 12 ], [ 3, 7 ], [ 4, 9 ], [ 5, 10 ], [ 6, 8 ] ]
gap> Eo1:=List(Eo,i->i[1]);
[ 1, 2, 3, 4, 5, 6 ]
gap> Eo2:=List(Eo,i->i[2]);
```

```
[ 11, 12, 7, 9, 10, 8 ]
gap> List(Eo1,i->VerticesOfEdge(ok,i));
[ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 1, 5 ], [ 2, 3 ], [ 2, 5 ] ]
gap> List(Eo2,i->VerticesOfEdge(ok,i));
[ [ 4, 6 ], [ 5, 6 ], [ 2, 6 ], [ 3, 6 ], [ 4, 5 ], [ 3, 4 ] ]
gap> Vo;
[ [ 1, 6 ], [ 2, 4 ], [ 3, 5 ] ]
gap> VoE:=[[1,2],[1,3],[1,2],[1,3],[2,3],[2,3]];
[ [ 1, 2 ], [ 1, 3 ], [ 1, 2 ], [ 1, 3 ], [ 2, 3 ], [ 2, 3 ] ]
gap> Okmod:=PolygonalComplexByDownwardIncidence(VoE,EoF);
simplicial surface (3 vertices, 6 edges, and 4 faces)
gap> IsIsomorphic(Okmod,TT);
true
```

1.3 Vertex-faithful surfaces

Problems :

- Construction of multi-tetrahedral spheres

Theoretical background

- Vertex-faithful surfaces and boolean operations

Frequently used commands

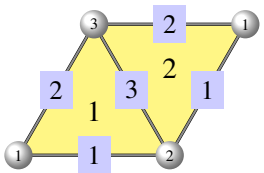
- AllSimplicialSpheres() (**??**)
- AutomorphismGroupOnEdges() (**??**)
- AutomorphismGroupOnFaces() (**??**)
- EdgesOfFaces() (local version: EdgesOfFace()) (**??**)
- EdgesOfVertices() (local version: Edges) (**??**)
- FaceDegreesOfVertices() (local version: FaceDegreeOfVertex)(**??**)
- FacesOfEdges() (local version: FacesOfEdge()) (**??**)
- IsIsomorphic() (**??**)
- SimplicialSurfaceByVerticesInFaces() (works for vertex-faithful surfaces only) (**??**)
- VertexCounter() (local version: DegreeOfVertex()) (**??**),
- VerticesOfFaces() (local Version: VerticesOfFace()) (**??**)

Less frequently used commands

- Tetrahedron() (**??**)

Mathematical details :

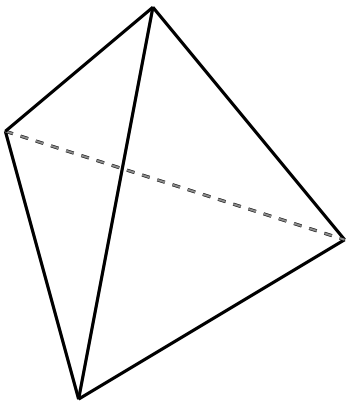
This section deals with vertex-faithful surfaces. A simplicial surface is vertex-faithful if and only if the map resulting by mapping an edge resp. face on it's set of incident vertices is injective. An example of a non vertex-faithful sphere is the Janus-Head.



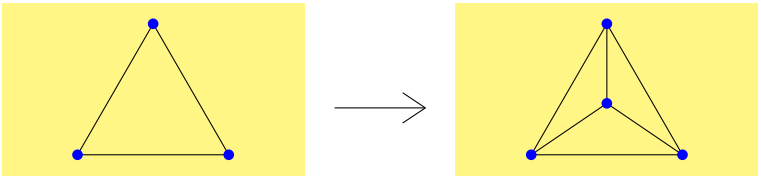
Example

```
gap> JanusHead();
simplicial surface (3 vertices, 3 edges, and 2 faces)
gap> VerticesOfFaces(last);
[ [ 1, 2, 3 ], [ 1, 2, 3 ] ]
```

In this exercise we construct multi-tetrahedral spheres. A multi-tetrahedral sphere is a vertex-faithful sphere constructed through tetrahedral extensions. Starting from a tetrahedron we iteratively replace faces by 3-gons to obtain the desired spheres. We have already seen examples of multi-tetrahedral spheres, namely the tetrahedron and the double-tetrahedron.



In this exercise we shall refer to a vertex of face degree 3 together with it's incident faces and edges as tetrahedron. Furthermore we say that a tetrahedron is attached resp. a tetrahedron is removed, if we replace a face by a 3-gon resp. a 3-gon by a face.



We can construct a new sphere out of a vertex-faithful sphere by removing all attached tetrahedra. So in other words, a vertex-faithful sphere is a multi-tetrahedral sphere if and only if iteratively removing all tetrahedra from the constructed spheres leads to the tetrahedron or the double-tetrahedron. Since we are only interested in surfaces we shall refer to terms like tetrahedron, double- tetrahedron, etc. as the combinatorial devices describing their combinatorial structure. So we will work with their incidence geometry and view them as abstract surfaces. Note, constructing a new sphere by replacing a face by a tetrahedron can be seen as a subdivision of a surface.

1.3.1 Multi-tetrahedral spheres

Idea behind the construcion

We construct multi-tetrahedral spheres with up to 12 faces by using their sets of vertices in faces. Tetrahedral extensions can be achieved by computing the symmetric difference of a given multi-tetrahedral sphere and a tetrahedron, both represented by their vertices in faces. Note, although spheres are constructed by replacing different faces of a sphere by tetrahedra, the resulting spheres can still be isomorphic.

Example

```
gap> t1:=Combinations([1,2,3,4],3);
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 4 ] ]
gap> T:=SimplicialSurfaceByVerticesInFaces(t1);
simplicial surface (4 vertices, 6 edges, and 4 faces)
```

Replace face [2,3,4] by a tetrahedron

Example

```
gap> t11:=Combinations([2,3,4,5],3);
[ [ 2, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ], [ 3, 4, 5 ] ]
gap> Sydi(t1,t11);
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ],
  [ 2, 4, 5 ], [ 3, 4, 5 ] ]
gap> T11:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (5 vertices, 9 edges, and 6 faces)
```

Replace face [1,2,3] by a tetrahedron

Example

```
gap> t12:=Combinations([1,2,3,5],3);
[ [ 1, 2, 3 ], [ 1, 2, 5 ], [ 1, 3, 5 ], [ 2, 3, 5 ] ]
gap> Sydi(t1,t12);
[ [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 4 ], [ 1, 2, 5 ],
  [ 1, 3, 5 ], [ 2, 3, 5 ] ]
gap> T12:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (5 vertices, 9 edges, and 6 faces)
```

Replace face [1,2,4] by a tetrahedron

Example

```
gap> t13:=Combinations([1,2,4,5],3);
[ [ 1, 2, 4 ], [ 1, 2, 5 ], [ 1, 4, 5 ], [ 2, 4, 5 ] ]
gap> Sydi(t1,t13);
[ [ 1, 2, 3 ], [ 1, 3, 4 ], [ 2, 3, 4 ], [ 1, 2, 5 ],
  [ 1, 4, 5 ], [ 2, 4, 5 ] ]
gap> T13:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (5 vertices, 9 edges, and 6 faces)
```

Replace face [1,3,4] by a tetrahedron

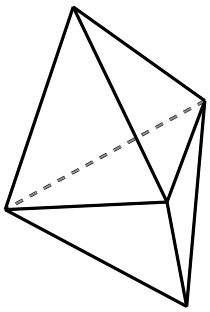
Example

```
gap> t14:=Combinations([1,3,4,5],3);
[ [ 1, 3, 4 ], [ 1, 3, 5 ], [ 1, 4, 5 ], [ 3, 4, 5 ] ]
gap> Sydi(t1,t14);
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 2, 3, 4 ], [ 1, 3, 5 ],
  [ 1, 4, 5 ], [ 3, 4, 5 ] ]
gap> T14:=SimplicialSurfaceByVerticesInFaces(last);
simplicial surface (5 vertices, 9 edges, and 6 faces)
```

Check whether the constructed surfaces are isomorphic

Example

```
gap> List([T12,T13,T14],S->IsIsomorphic(T11,S));
[ true, true, true ]
```



So special care has to be taken to the choice of faces. Furthermore we introduce some terminology for the purpose of having an easy way to refer to the constructed spheres. Let X be a multi-tetrahedral sphere. We shall refer to surfaces obtained through a tetrahedral extension on X as children of X. So the double-tetrahedron is a child of the tetrahedron.

Start by defining the function `Te()` which returns the set of vertices in faces of a tetrahedron whereby the four vertices are given by `a,b,c,d`

Example

```
gap> Te:=function(a,b,c,d)
> return Combinations([a,b,c,d],3);
> end;
function( a, b, c, d ) ... end
```

Up to isomorphism there is exactly one multi-tetrahedral sphere with 4 faces. Compute the vertices in faces of the first tetrahedron T1:

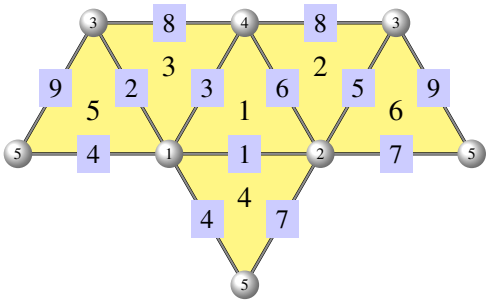
Example

```
gap> T1:=Te(1,2,3,4);
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 4 ] ]
gap> VerticesOfFaces(Tetrahedron());
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 2, 3, 4 ], [ 1, 3, 4 ] ]
gap> Te(2,3,4,5);
[ [ 2, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ], [ 3, 4, 5 ] ]
```

As seen in previous computations, there is exactly one multi-tetrahedral sphere with 6 faces up to isomorphism, namely the double-tetrahedron.

Example

```
gap> T2:=Sydi(T1,Te(2,3,4,5));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 5 ] ]
gap> t2:=SimplicialSurfaceByVerticesInFaces(T2);
simplicial surface (5 vertices, 9 edges, and 6 faces)
```

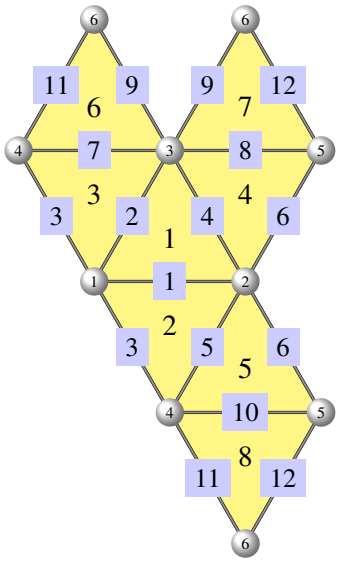


How do we find all children of t2? We could simply replace every possible face by a tetrahedron and gather the constructed spheres up to isomorphism. But we want to keep the number of tetrahedral extensions thus the expenses to a minimum. Therefore we will use the automorphism group on the faces of our spheres to determine the minimum number of faces and also the faces which have to be replaced to obtain all children of t2.

Example

```
gap> a2:=AutomorphismGroupOnFaces(t2);
Group([ (1,2)(4,5), (2,3)(5,6), (1,4)(2,5)(3,6) ])
gap> Orbits(a2);
[ [ 1, 2, 4, 3, 5, 6 ] ]
gap> T3:=Sydi(T2,Te(3,4,5,6));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 6 ] ]
```

So there is exactly one multi-tetrahedral sphere with 8 faces.



Note, we can find the automorphism group of a simplicial surface with n faces with the help of the symmetric group of degree n. It is easy to see that the symmetric group acts on the set M, whereby an element A of M has a cardinality of n. The elements of A are subsets of the set of vertices of our surface containing exactly three vertices. Then the automorphism group on faces can be identified as the stabilizer of the set of vertices in faces of our multi-tetrahedral sphere under the described group action.

Example

```
gap> A3:=Stabilizer(SymmetricGroup(6),T3,OnSetsSets);
Group([ (3,4), (1,6)(2,5) ])
gap> Orbits(A3,T3,OnSets);
[ [ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 3, 5, 6 ], [ 4, 5, 6 ] ],
  [ [ 1, 3, 4 ], [ 3, 4, 6 ] ], [ [ 2, 3, 5 ], [ 2, 4, 5 ] ] ]
```

So there are exactly three multi-tetrahedral spheres with 10 faces which at the same time are all children of the same multi-tetrehedral sphere. They are represented by the following set of faces represented by their vertices in faces.

Example

```
gap> O3:=last;
[ [ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 3, 5, 6 ], [ 4, 5, 6 ] ],
  [ [ 1, 3, 4 ], [ 3, 4, 6 ] ], [ [ 2, 3, 5 ], [ 2, 4, 5 ] ] ]
gap> T4x1:=Sydi(T3,Te(4,5,6,7));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
  [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 7 ], [ 4, 6, 7 ], [ 5, 6, 7 ] ]
gap> T4x2:=Sydi(T3,Te(3,4,6,7));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
  [ 3, 4, 7 ], [ 3, 5, 6 ], [ 3, 6, 7 ], [ 4, 5, 6 ], [ 4, 6, 7 ] ]
gap> T4x3:=Sydi(T3,Te(2,4,5,7));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 7 ],
  [ 2, 5, 7 ], [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 6 ], [ 4, 5, 7 ] ]
```

Comparing their vertex counters shows that they are indeed not isomorpic.

Example

```
gap> SimplicialSurfaceByVerticesInFaces(T4x1); VertexCounter(last);
simplicial surface (7 vertices, 15 edges, and 10 faces)
[ [ 3, 2 ], [ 4, 2 ], [ 5, 2 ], [ 6, 1 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T4x2); VertexCounter(last);
simplicial surface (7 vertices, 15 edges, and 10 faces)
[ [ 3, 2 ], [ 4, 3 ], [ 6, 2 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T4x3); VertexCounter(last);
simplicial surface (7 vertices, 15 edges, and 10 faces)
[ [ 3, 3 ], [ 5, 3 ], [ 6, 1 ] ]
```

Computing the children of t4x1:

Example

```
gap> A4x1:=Stabilizer(SymmetricGroup(7),T4x1,OnSetsSets);
Group([ (1,7)(2,6)(3,5) ])
gap> O4x1:=Orbits(A4x1,T4x1,OnSets);
[ [ [ 1, 2, 3 ], [ 5, 6, 7 ] ], [ [ 1, 2, 4 ], [ 4, 6, 7 ] ],
  [ [ 1, 3, 4 ], [ 4, 5, 7 ] ], [ [ 2, 3, 5 ], [ 3, 5, 6 ] ],
```

```
 [ [ 2, 4, 5 ], [ 3, 4, 6 ] ] ]
gap> T5x1:=Sydi(T4x1,Te(5,6,7,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 7 ], [ 4, 6, 7 ], [ 5, 6, 8 ],
 [ 5, 7, 8 ], [ 6, 7, 8 ] ]
gap> T5x2:=Sydi(T4x1,Te(4,6,7,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 7 ], [ 4, 6, 8 ], [ 4, 7, 8 ],
 [ 5, 6, 7 ], [ 6, 7, 8 ] ]
gap> T5x3:=Sydi(T4x1,Te(4,5,7,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 8 ], [ 4, 6, 7 ], [ 4, 7, 8 ],
 [ 5, 6, 7 ], [ 5, 7, 8 ] ]
gap> T5x4:=Sydi(T4x1,Te(3,5,6,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 6 ], [ 3, 5, 8 ], [ 3, 6, 8 ], [ 4, 5, 7 ], [ 4, 6, 7 ],
 [ 5, 6, 7 ], [ 5, 6, 8 ] ]
gap> T5x5:=Sydi(T4x1,Te(3,4,6,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 8 ], [ 3, 5, 6 ], [ 3, 6, 8 ], [ 4, 5, 7 ], [ 4, 6, 7 ],
 [ 4, 6, 8 ], [ 5, 6, 7 ] ]
```

Hence we have 5 non isomorphic descendents of t4x1.

Computing their vertex counter:

Example

```
gap> SimplicialSurfaceByVerticesInFaces(T5x1); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 2 ], [ 4, 2 ], [ 5, 2 ], [ 6, 2 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x2); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 2 ], [ 4, 2 ], [ 5, 3 ], [ 7, 1 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x3); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 2 ], [ 4, 3 ], [ 5, 1 ], [ 6, 1 ], [ 7, 1 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x4); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 3 ], [ 4, 1 ], [ 5, 1 ], [ 6, 3 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x5); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 3 ], [ 4, 1 ], [ 5, 2 ], [ 6, 1 ], [ 7, 1 ] ]
```

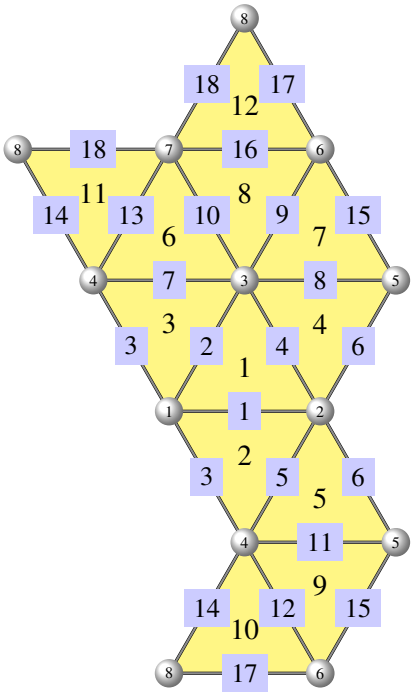
Note, non isomorphic multi-tetrahedral spheres can still have isomorphic children. But the vertex counter indicates whether two spheres are non isomorphic. This is the case, if their vertex counters do differ. If they have the same vertex counters we have to check whether there exists an isomorphism mapping one to the other.

Compute the children of t4x2:

Example

```
gap> A4x2:=Stabilizer(SymmetricGroup(7),T4x2,OnSetsSets);
Group([ (3,4), (1,7)(2,6) ])
gap> O4x2:=Orbits(A4x2,T4x2,OnSets);
[ [ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 3, 6, 7 ], [ 4, 6, 7 ] ],
```

```
 [ [ 1, 3, 4 ], [ 3, 4, 7 ] ],
 [ [ 2, 3, 5 ], [ 2, 4, 5 ], [ 3, 5, 6 ], [ 4, 5, 6 ] ] ]
gap> T5x6:=Sydi(T4x2,Te(4,6,7,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
 [ 3, 4, 7 ], [ 3, 5, 6 ], [ 3, 6, 7 ], [ 4, 5, 6 ], [ 4, 6, 8 ],
 [ 4, 7, 8 ], [ 6, 7, 8 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x6); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 2 ], [ 4, 3 ], [ 5, 1 ], [ 6, 1 ], [ 7, 1 ] ]
gap> T5x6 in Orbit(SymmetricGroup(8),T5x3,OnSetsSets);
true
```



So the sphere represented by T5x6 is a child of t4x1 and t4x2. Alternatively the isomorphism check can be done by constructing the spheres and using `IsIsomorphic()`

Compute the remaining children of t4x2:

Example

```
gap> T5x6:=Sydi(T4x2,Te(3,4,7,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
  [ 3, 4, 8 ], [ 3, 5, 6 ], [ 3, 6, 7 ], [ 3, 7, 8 ], [ 4, 5, 6 ],
  [ 4, 6, 7 ], [ 4, 7, 8 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x6); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 2 ], [ 4, 4 ], [ 7, 2 ] ]
gap> T5x7:=Sydi(T4x2,Te(4,5,6,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ],
  [ 3, 4, 7 ], [ 3, 5, 6 ], [ 3, 6, 7 ], [ 4, 5, 8 ], [ 4, 6, 7 ],
  [ 4, 6, 8 ], [ 5, 6, 8 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x6); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
```

```
[ [ 3, 2 ], [ 4, 4 ], [ 7, 2 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x7); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 3 ], [ 4, 1 ], [ 5, 2 ], [ 6, 1 ], [ 7, 1 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x5); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 3 ], [ 4, 1 ], [ 5, 2 ], [ 6, 1 ], [ 7, 1 ] ]
gap> T5x7 in Orbit(SymmetricGroup(8),T5x5,OnSetsSets);
true
```

So computing the children of t4x2 gives rise to three non isomorphic multi- tetrahedral spheres of which two are also children of t4x1.

Constructing the children of t4x3:

Example

```
gap> A4x3:=Stabilizer(SymmetricGroup(7),T4x3,OnSetsSets);
Group([ (2,3)(6,7), (1,7)(3,5) ])
gap> O4x3:=Orbits(A4x3,T4x3,OnSets);
[ [ [ 1, 2, 3 ], [ 3, 5, 6 ], [ 2, 5, 7 ] ],
  [ [ 1, 2, 4 ], [ 3, 4, 6 ], [ 4, 5, 7 ], [ 1, 3, 4 ], [ 2, 4, 7 ],
      [ 4, 5, 6 ] ], [ [ 2, 3, 5 ] ] ]
```

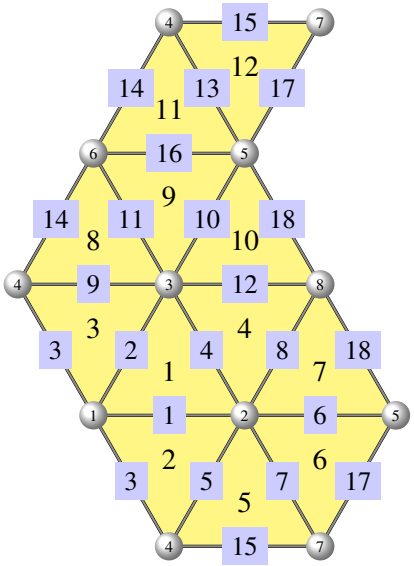
Hence the multitetradral-sphere T4x3 has exactly three children.

Example

```
gap> T5x7:=Sydi(T4x3,Te(2,5,7,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 7 ],
  [ 2, 5, 8 ], [ 2, 7, 8 ], [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 6 ],
  [ 4, 5, 7 ], [ 5, 7, 8 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x7); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 3 ], [ 4, 1 ], [ 5, 1 ], [ 6, 3 ] ]
gap> T5x7 in Orbit(SymmetricGroup(8),T5x4,OnSetsSets);
true
gap> T5x7:=Sydi(T4x3,Te(4,5,6,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 7 ],
  [ 2, 5, 7 ], [ 3, 4, 6 ], [ 3, 5, 6 ], [ 4, 5, 7 ], [ 4, 5, 8 ],
  [ 4, 6, 8 ], [ 5, 6, 8 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x7); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 3 ], [ 4, 1 ], [ 5, 2 ], [ 6, 1 ], [ 7, 1 ] ]
gap> T5x7 in Orbit(SymmetricGroup(8),T5x5,OnSetsSets);
true
gap> T5x7:=Sydi(T4x3,Te(2,3,5,8));
[ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 3, 4 ], [ 2, 3, 8 ], [ 2, 4, 7 ],
  [ 2, 5, 7 ], [ 2, 5, 8 ], [ 3, 4, 6 ], [ 3, 5, 6 ], [ 3, 5, 8 ],
  [ 4, 5, 6 ], [ 4, 5, 7 ] ]
gap> SimplicialSurfaceByVerticesInFaces(T5x7); VertexCounter(last);
simplicial surface (8 vertices, 18 edges, and 12 faces)
[ [ 3, 4 ], [ 6, 4 ] ]
gap> VC5:=List([T5x1,T5x2,T5x3,T5x4,T5x5,T5x6,T5x7],r-
>VertexCounter(SimplicialSurfaceByVerticesInFaces(r)));
[ [ [ 3, 2 ], [ 4, 2 ], [ 5, 2 ], [ 6, 2 ] ],
```

```
    [ [ 3, 2 ], [ 4, 2 ], [ 5, 3 ], [ 7, 1 ] ],
    [ [ 3, 2 ], [ 4, 3 ], [ 5, 1 ], [ 6, 1 ], [ 7, 1 ] ],
    [ [ 3, 3 ], [ 4, 1 ], [ 5, 1 ], [ 6, 3 ] ],
    [ [ 3, 3 ], [ 4, 1 ], [ 5, 2 ], [ 6, 1 ], [ 7, 1 ] ],
    [ [ 3, 2 ], [ 4, 4 ], [ 7, 2 ] ], [ [ 3, 4 ], [ 6, 4 ] ] ]
```

So out of the three children two are also children of t4x2. So in total there are exactly seven multi-tetrahedral spheres with 12 faces up to isomorphism. The last sphere with vertex counter `[[3,4],[6,4]]` is of greater interest to us.



Example

```
gap> t5x7:=SimplicialSurfaceByVerticesInFaces(T5x7);
simplicial surface (8 vertices, 18 edges, and 12 faces)
gap> FaceCounter(t5x7);
[ [ [ 3, 6, 6 ], 12 ] ]
```

One could get the idea that this is a multi-tetrahedral sphere obtained from a tetrahedron where every face is subdivided due to tetrahedral extension

Example

```
gap> Filtered(Vertices(t5x7),r->FaceDegreeOfVertex(t5x7,r)=3);
[ 1, 6, 7, 8 ]
gap> List(last,i->NeighbourVerticesOfVertex(t5x7,i));
[ [ 2, 3, 4 ], [ 3, 4, 5 ], [ 2, 4, 5 ], [ 2, 3, 5 ] ]
gap> Te(2,3,4,5);
[ [ 2, 3, 4 ], [ 2, 3, 5 ], [ 2, 4, 5 ], [ 3, 4, 5 ] ]
gap> Set(last2)=last;
true
```

1.4 Isosceles coloured Surfaces

Problems : Analysing Isosceles coloured surfaces

Theoretical background

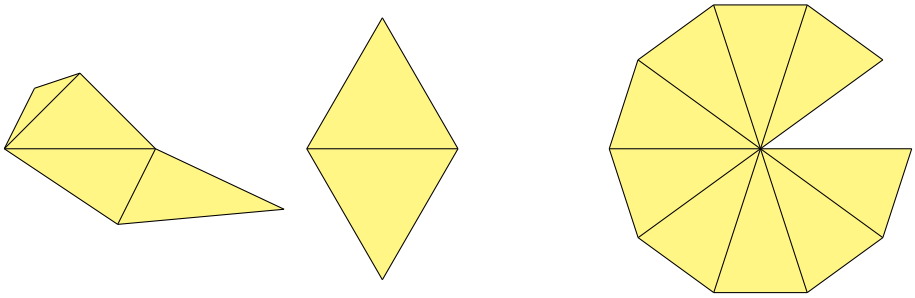
- isosceles coloured surfaces and boolean operations

Frequently used commands

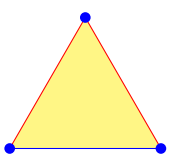
- AllIsoscelesColouredSurfaces() (**??**)
- AllSimplicialSpheres() (**??**)
- EdgesOfFaces() (local version: EdgesOfFace()) (**??**)
- FaceDegreesOfVertices() (local version: FaceDegreeOfVertex) (**??**)
- NeighbourVerticesOfVertex() (**??**),
- SubcomplexByFaces() (**??**)
- VertexCounter() (local version: DegreeOfVertex()) (**??**),
- VerticesOfFaces() (local Version: VerticesOfFace()) (**??**)
- UmbrellaPathsOfVertices() (local version: UmbrellaPathOfVertex()) (**??**)

Mathematical details :

Most of what has been said in the previous exercises applies for arbitrary simplicial surfaces and simplicial surfaces of equiangular triangles. This chapter deals with the combinatorial side of simplicial surfaces of congruent isoscles triangules. More precisely we want to investigate those simplicial surfaces whose isosceles triangules are not equiangular.



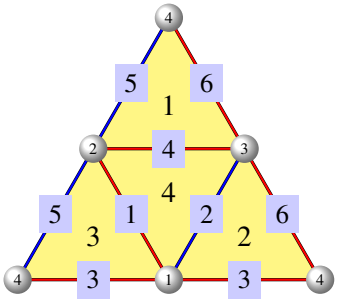
Therefore we define the base selection or simply base of a simplicial surface as a subset of the edges underlying the condition that every face of the surface is incident to exactly one edge of the base. We shall refer to an edge as base edge if it is contained in the base and as a leg if it is not. This gives rise to two types of vertices of a face. If a vertex of a face is incident to the two legs of the face, then we call it apex vertex or simply apex. So each face is incident to exactly one apex.



We handle simplical surfaces of isosceles triangles in the package by introducing a colouring for edges and call the resulting structures isosceles coloured simplicial surfaces. For example up to isomorphism there is exactly one isosceles coloured surface resulting from the tetrahedron.

Example

```
gap> pr:=rec(edgeColourClassColours:=["red","blue"]);;
gap> AllIsoscelesColouredSurfaces(T);
[ isosceles coloured surface (4 vertices, 6 edges and 4 faces) ]
gap> DrawSurfaceToTikz(last[1],"Image_IsoscelesColouredTetrahedron1.tex",pr);
Picture written in TikZ.
```

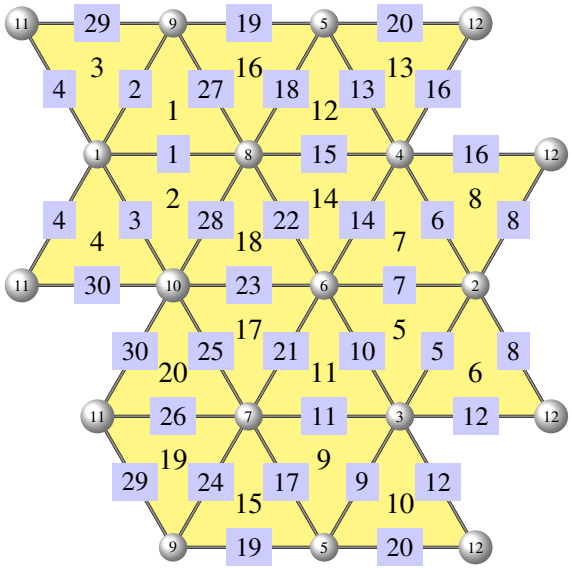


In this exercise we want to examine the apex vertices of a surface. We want to check in how many isosceles coloured surfaces a given vertex can be an apex vertex, whereby the coloured surfaces all result from a given surface. We gather those coloured surfaces up to isomorphism

For the purpose of this exercise we choose the following surface.

Example

```
gap> S:=AllSimplicialSpheres(20)[5];
simplicial surface (12 vertices, 30 edges, and 20
faces)
```



Compute elementary properties of S

Example
```
gap> VertexCounter(S);
[ [ 4, 4 ], [ 5, 4 ], [ 6, 4 ] ]
gap> FaceDegreesOfVertices(S);
[ 4, 4, 5, 5, 6, 6, 6, 6, 5, 5, 4, 4 ]
```

We choose vertex 1 for our examinations and compute the neighbour vertices of this vertex

Example
```
gap> NeighbourVerticesOfVertex(S,1);
[ 8, 9, 10, 11 ]
```

Compute all isosceles coloured surfaces of S

Example
```
gap> ss:=AllIsoscelesColouredSurfaces(S);
[ isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
    , isosceles coloured surface (12 vertices, 30 edges and 20 faces)
 ]
```

So introducing the isosceles colouring results in 14 isosceles coloured surfaces. Display all elementary properties of the first surface in the list ss

Example
```
gap> Display(ss[1]);
Isosceles coloured surface ( closed, orientable, Euler-characteristic 2)
    Vertices (12): [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ]
    Edges (30): [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 ]
    Faces (20): [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20 ]
    VerticesOfEdges: [ [ 1, 8 ], [ 1, 9 ], [ 1, 10 ], [ 1, 11 ], [ 2, 3 ],
[ 2, 4 ], [ 2, 6 ], [ 2, 12 ], [ 3, 5 ],
[ 3, 6 ], [ 3, 7 ], [ 3, 12 ],
[ 4, 5 ], [ 4, 6 ], [ 4, 8 ], [ 4, 12 ],
[ 5, 7 ], [ 5, 8 ], [ 5, 9 ], [ 5, 12 ], [ 6, 7 ], [ 6, 8 ], [ 6, 10 ],
[ 7, 9 ], [ 7, 10 ], [ 7, 11 ], [ 8, 9 ], [ 8, 10 ], [ 9, 11 ], [ 10, 11 ] ]
   VerticesOfFaces: [ [ 1, 8, 9 ], [ 1, 8, 10 ], [ 1, 9, 11 ], [ 1, 10, 11 ],
[ 2, 3, 6 ], [ 2, 3, 12 ], [ 2, 4, 6 ], [ 2, 4, 12 ], [ 3, 5, 7 ], [ 3, 5, 12 ],
[ 3, 6, 7 ], [ 4, 5, 8 ], [ 4, 5, 12 ], [ 4, 6, 8 ], [ 5, 7, 9 ], [ 5, 8, 9 ],
[ 6, 7, 10 ], [ 6, 8, 10 ], [ 7, 9, 11 ], [ 7, 10, 11 ] ]
```

```
    EdgesOfFaces: [ [ 1, 2, 27 ], [ 1, 3, 28 ], [ 2, 4, 29 ],
[ 3, 4, 30 ], [ 5, 7, 10 ], [ 5, 8, 12 ], [ 6, 7, 14 ],
[ 6, 8, 16 ], [ 9, 11, 17 ], [ 9, 12, 20 ], [ 10, 11, 21 ], [ 13, 15, 18 ],
[ 13, 16, 20 ], [ 14, 15, 22 ], [ 17, 19, 24 ], [ 18, 19, 27 ], [ 21, 23, 25 ],
[ 22, 23, 28 ], [ 24, 26, 29 ], [ 25, 26, 30  ]
   Umbrella-paths: [ ( e1, F1, e2, F3, e4, F4, e3, F2, e1 ),
( e5, F5, e7, F7, e6, F8, e8, F6, e5 ),
( e5, F5, e10, F11, e11, F9, e9, F10, e12, F6, e5 ),
( e6, F7, e14, F14, e15, F12, e13, F13, e16, F8, e6 ),
( e9, F9, e17, F15, e19, F16, e18, F12, e13, F13, e20, F10, e9 ),
( e7, F5, e10, F11, e21, F17, e23, F18, e22, F14, e14, F7, e7 ),
( e11, F9, e17, F15, e24, F19, e26, F20, e25, F17, e21, F11, e11 ),
( e1, F1, e27, F16, e18, F12, e15, F14, e22, F18, e28, F2, e1 ),
( e2, F1, e27, F16, e19, F15, e24, F19, e29, F3, e2 ),
( e3, F2, e28, F18, e23, F17, e25, F20, e30, F4, e3 ),
( e4, F3, e29, F19, e26, F20, e30, F4, e4 ),
( e8, F6, e12, F10, e20, F13, e16, F8, e8 ) ]
   EdgesOfColours: [  [ 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14, 18, 19, 20, 22,
23, 24, 25, 26 ], [ 7, 12, 15, 16, 17, 21, 27, 28, 29, 30 ] ]
   LocalSymmetry: [ mirror, mirror, mirror, mirror, rotation, rotation, mirror,
mirror, rotation, mirror, mirror, mirror, mirror, rotation, mirror, mirror,
mirror, mirror, rotation, mirror, mirror, mirror, rotation, rotation, rotation,
mirror, mirror, mirror, mirror, mirror ]
```

Compute the automorphism group permutating the edges of S

Example

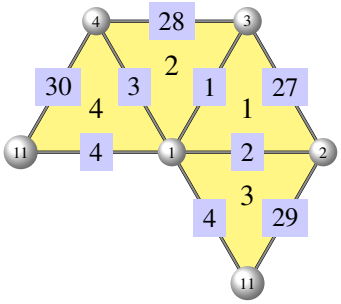
```
gap> A:=AutomorphismGroupOnEdges(S);
Group([ (1,26)(2,29)(3,30)(5,6)(9,13)(10,14)(11,15)(12,16)(17,18)(21,22)
  (24,27)(25,28), (2,3)(5,12)(6,16)(7,20)(9,10)(13,14)(17,21)(18,22)(19,23)
  (24,25)(27,28)(29,30), (1,7)(2,5)(3,6)(4,8)(9,24)(10,27)(11,19)(12,29)
  (13,25)(14,28)(15,23)(16,30)(18,21)(20,26) ])
gap> Size(A);
8
```

For vertex 1 being an apex, all edges incident to the vertex 1 have to be legs and the remaining edges of the faces incident to our vertex have to be base edges.

Compute the the edges which have to be base edges

Example

```
gap> S1:=SubcomplexByFaces(S,FacesOfVertex(S,1));
simplicial surface ( 5 vertices, 8 edges, and 4 faces)
gap> bb:=BoundaryEdges(S1);
[ 27, 28, 29, 30 ]
```



Hence we have to find all base-selections containing bb or the set resulting from manipulating bb with the help of an isomorphism of the automorphism group on the edges.

First we have to compute the base-selections of the sufaces in the ss.

Example

```
gap> ss:=List(ss,r->Set(List(Faces(r),f->BaseEdgeOfFace(r,f))));
[ [ 7, 12, 15, 16, 17, 21, 27, 28, 29, 30 ],
  [ 7, 8, 15, 17, 20, 21, 27, 28, 29, 30 ],
  [ 3, 6, 10, 12, 13, 17, 22, 25, 27, 29 ],
  [ 4, 8, 9, 10, 13, 14, 24, 25, 27, 28 ],
  [ 4, 7, 12, 15, 16, 17, 21, 26, 27, 28 ],
  [ 4, 7, 11, 12, 15, 16, 24, 25, 27, 28 ],
  [ 4, 7, 8, 15, 17, 20, 21, 26, 27, 28 ],
  [ 4, 7, 8, 11, 15, 20, 24, 25, 27, 28 ],
  [ 2, 3, 7, 12, 16, 17, 18, 21, 22, 26 ],
  [ 2, 3, 7, 11, 12, 15, 16, 19, 23, 26 ],
  [ 2, 3, 7, 8, 17, 18, 20, 21, 22, 26 ],
  [ 2, 3, 7, 8, 11, 15, 19, 20, 23, 26 ],
  [ 1, 4, 7, 8, 17, 18, 20, 21, 22, 26 ],
  [ 1, 4, 7, 8, 11, 15, 19, 20, 23, 26 ] ]
gap> Size(ss);
14
```

Check whether bb is contained in one of the base selections

Example

```
gap> obb:=Orbit(A,bb,OnSets);
[ [ 27, 28, 29, 30 ], [ 2, 3, 24, 25 ], [ 10, 12, 14, 16 ], [ 5, 6, 9, 13 ] ]
gap> Filtered([1..14],i->IsSubset(ss[i],obb[1]));
[ 1, 2 ]
gap> Filtered([1..14],i->IsSubset(ss[i],obb[2]));
[  ]
gap> Filtered([1..14],i->IsSubset(ss[i],obb[3]));
[  ]
gap> Filtered([1..14],i->IsSubset(ss[i],obb[4]));
[  ]
```

So at first glance it looks like there are only two possibilities. But there are more. We obtain this observatin by manipulating the base selection of the first isoscoles coloured surface in the list ss.

Example

```
gap> ss1:=Orbit(A,ss[1],OnSets);
[ [ 7, 12, 15, 16, 17, 21, 27, 28, 29, 30 ],
  [ 2, 3, 7, 11, 12, 16, 18, 22, 24, 25 ],
  [ 5, 6, 15, 17, 20, 21, 27, 28, 29, 30 ],
  [ 1, 10, 12, 14, 16, 17, 18, 23, 29, 30 ],
  [ 2, 3, 5, 6, 11, 18, 20, 22, 24, 25 ],
  [ 1, 5, 6, 9, 13, 19, 21, 22, 29, 30 ],
  [ 2, 3, 10, 12, 14, 16, 17, 18, 23, 26 ],
  [ 2, 3, 5, 6, 9, 13, 19, 21, 22, 26 ] ]
gap> Filtered(ss1,r->IsSubset(r,bb));
[ [ 7, 12, 15, 16, 17, 21, 27, 28, 29, 30 ],
  [ 5, 6, 15, 17, 20, 21, 27, 28, 29, 30 ] ]
gap> ss2:=Orbit(A,ss[2],OnSets);
```

```
[ [ 7, 8, 15, 17, 20, 21, 27, 28, 29, 30 ],
  [ 2, 3, 7, 8, 11, 18, 20, 22, 24, 25 ],
  [ 1, 4, 10, 12, 14, 16, 17, 18, 23, 26 ],
  [ 1, 4, 5, 6, 9, 13, 19, 21, 22, 26 ] ]
gap> Filtered(ss2,r->IsSubset(r,bb));
[ [ 7, 8, 15, 17, 20, 21, 27, 28, 29, 30 ] ]
```

Hence we have at least three possibilities for an isosceles coloured simplicial suface resulting from S and having vertex 1 as apex. Indeed there exactly three such surfaces. Let us now visualise these edge coloured surfaces.

Compute the isosceles coloured surfaces with the constructed base selections

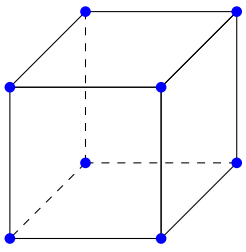
Example

```
gap> SS1:=List([1..30],i->1);
[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1 ]
gap> for i in ss1[1] do SS1[i]:=2;od;
gap> SS1:=EdgeColouredPolygonalComplex(S,SS1);
isosceles coloured surface (12 vertices, 30 edges and 20 faces)
gap> SS2:=List([1..30],i->1);
[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1 ]
gap> for i in ss1[3] do SS2[i]:=2;od;
gap> SS2:=EdgeColouredPolygonalComplex(S,SS2);
isosceles coloured surface (12 vertices, 30 edges and 20 faces)
gap> SS3:=List([1..30],i->1);
[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1 ]
gap> for i in ss2[1] do SS3[i]:=2;od;
gap> SS3:=EdgeColouredPolygonalComplex(S,SS3);
isosceles coloured surface (12 vertices, 30 edges and 20 faces)
```

Draw the net of the surface SS1 into a tex-file using TikZ

Example

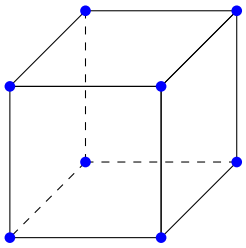
```
gap> pr:=rec(edgeColourClassColours:=["red","blue"],
> edgeColourClassLengths:=[1.2, 0.8]);;
gap> pr.compileLaTeX:=true;
true
gap> pr:=DrawSurfaceToTikz(SS1,"Zwa1",pr);;
DrawSurfaceToTikz: Could not find intersection-free continuation. Draw face
17 via edge 21 instead.
Picture written in TikZ.Start LaTeX-compilation (type 'x' and press ENTER to
abort).
Picture rendered (with pdflatex).
```



Draw the net of the surface SS2 into a tex-file using TikZ

Example

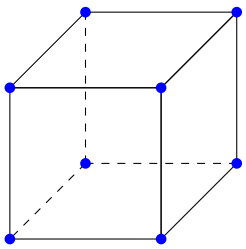
```
gap> pr:=rec(edgeColourClassColours:=["red","blue"],
> edgeColourClassLengths:=[1.2, 0.8], compileLaTeX:=true);;
gap> pr:=DrawSurfaceToTikz(SS2,"Zwa2",pr);;
DrawSurfaceToTikz: Could not find intersection-free continuation. Draw face
18 via edge 22 instead.
Picture written in TikZ.Start LaTeX-compilation (type 'x' and press ENTER to
abort).
Picture rendered (with pdflatex).
```



Draw the net of the surface SS3 into a tex-file using TikZ

Example

```
gap> pr:=rec(edgeColourClassColours:=["red","blue"],
> edgeColourClassLengths:=[1.2, 0.8], compileLaTeX:=true);;
gap> pr:=DrawSurfaceToTikz(SS3,"Zwa3",pr);;
DrawSurfaceToTikz: Could not find intersection-free continuation. Draw face
17 via edge 21 instead.
Picture written in TikZ.Start LaTeX-compilation (type 'x' and press ENTER to
abort).
Picture rendered (with pdflatex).
```



1.5 Butterfly-Deletion

Problems : Construction of spheres via butterfly-deletions on the icosahedron

Theoretical background

- Vertex-faithful surfaces and boolean operations
- Umbrella descriptor of simplicial surfaces

Frequently used commands

- AutomorphismGroupOnFaces() (**??**)

- UmbrellaDescriptorOfSurface() (**??**)

Less frequently used commands

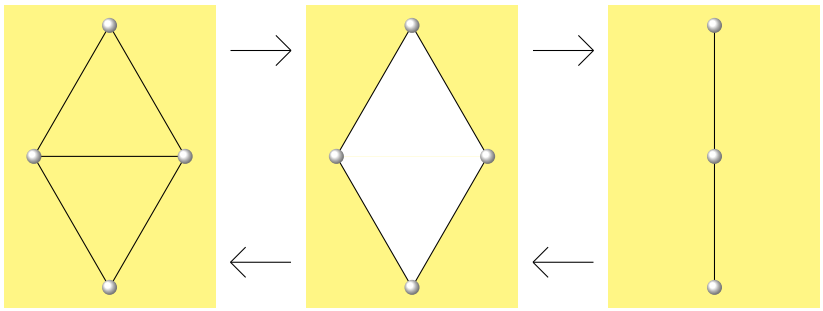
- Icosahedron() (**??**)
- DistanceOfFaces() (**??**)

Mathematical details : The butterfly deletion constructs a simplicial surface by removing a butterfly of a given simplicial surface. To apply the butterfly deletion we need an inner edge of the surface not belonging to a 2-Waist. The resulting simplicial surface can be defined in two steps.

1. The two faces incident to the given inner edge are removed from the surface by applying cuts along the four edges incident to exactly one of the two faces.
2. Step 1 gives rise to four new boundary edges and two boundary vertices which were incident to exactly one of the removed faces. By applying mender operations so that each pair of edges incident to the same boundary vertex get identified, the simplicial surfac resulting from the butterfly deletion is constructed.

Clearly the butterfly deletion can be reversed, namely by using the butterfly insertion.

Butterfly deletion

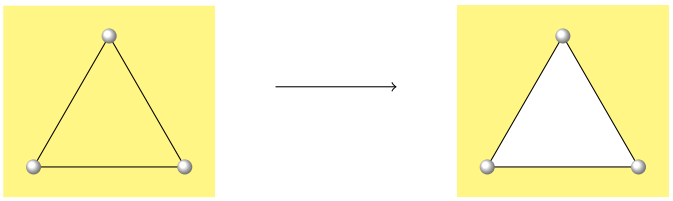


Butterfly insertion

In this chapter we will familliarize ourselves with the butterfly deletion. We want to compute the simplicial surfaces constructed up to isomorphism by applying at most two butterfly deletions to the icosahedron. Note, applying the butterfly deletion to a simplicial surface does not affect the Euler-Characteristic.

Since a given simplicial surface can be uniquly reconstructed from it's umbrella descriptor, we shall use this structure to examine the butterfly deletion in this exercise. Note, it is also possible to define this construction by manipulating the incidence structure, namely the edges of faces and the vertices of edges.

Let us start by defining a function that omits a given face from a simplicial surface represented by its umbrella descriptor.



Example

```
gap> Omit:=function ( Z, i )
   local a, b;
   if i ^ Z = i then
       return Z;
   fi;
   a := Orbit( Group( Z ), i );
   b := List( [ 2 .. Size( a ) ], function ( i )
           return a[i];
       end );
   return CycleFromList( b );
end
```

Given the umbrella descriptor of a simplicial surface, the butterfly deletion can be applied by using the folllowing function.

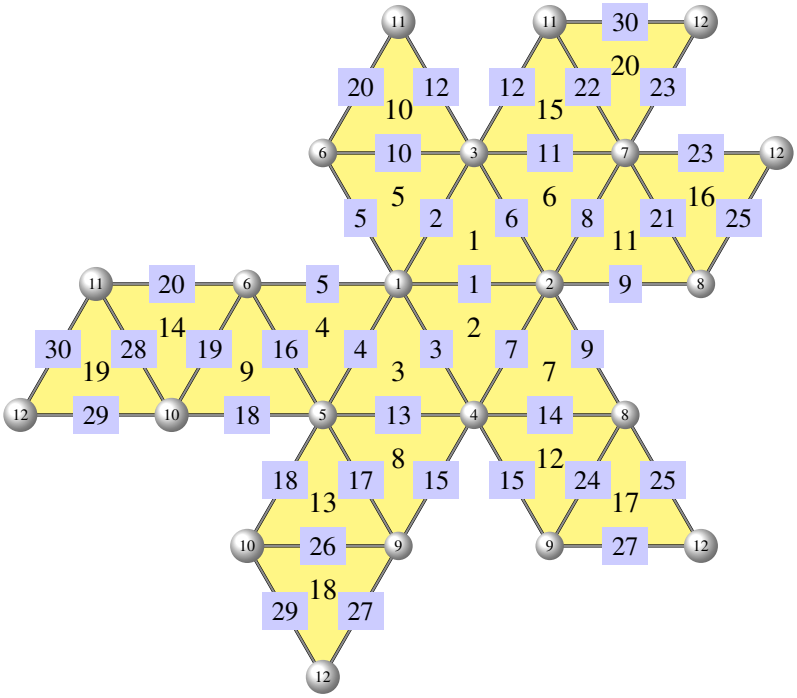
Example

```
gap>Zus:=function ( ZZ, i, j )
    local ZZ1, r, k;
    ZZ1 := Filtered( ZZ, function ( r )
            return not (i ^ r = j or j ^ r = i);
       end );
    ZZ1 := List( ZZ1, function ( r )
           return Omit( r, i );
       end );
   ZZ1 := List( ZZ1, function ( r )
           return Omit( r, j );
        end );
    r := Filtered( ZZ, function ( r )
            return i ^ r = j or j ^ r = i;
        end );
    if i ^ r[1] <> j then
        r[1] := r[1] ^ -1;
    fi;
    if j ^ r[2] <> i then
        r[2] := r[2] ^ -1;
    fi;
    r[1] := Orbit( Group( r[1] ), i );
    r[2] := Orbit( Group( r[2] ), j );
    r[1] := List( [ 3 .. Size( r[1] ) ], function ( k )
            return r[1][k];
        end );
    r[2] := List( [ 3 .. Size( r[2] ) ], function ( k )
            return r[2][k];
        end );
    for k in r[2] do
        Add( r[1], k );
    od;
    Add( ZZ1, CycleFromList( r[1] ) );
    return ZZ1;
end
```

Let us compute the icosahedron and it's umbrella descriptor.

Example

```
gap> Ik:=Icosahedron();
simplicial surface (12 vertices, 30 edges, and 20 faces)
gap> UIk:=UmbrellaDescriptorOfSurface(Ik);
[ (1,5,4,3,2), (1,6,11,7,2), (1,6,15,10,5), (2,7,12,8,3), (3,8,13,9,4),
  (4,9,14,10,5), (6,15,20,16,11), (7,12,17,16,11), (8,13,18,17,12),
  (9,14,19,18,13), (10,14,19,20,15), (16,17,18,19,20) ]
```



Up to isomorphism there is exactly one sphere resulting from applying exactly one butterfly deletion to the icosahedron. We obtain this sphere represented by it's umbrella descriptor by removing the butterfly containing the faces 1 and 2.

Example

```
gap> U1:=Zus(UIK,1,2);
[ (5,6,15,10), (3,7,12,8), (3,8,13,9,4), (4,9,14,10,5), (6,15,20,16,11),
  (7,12,17,16,11), (8,13,18,17,12), (9,14,19,18,13), (10,14,19,20,15),
  (16,17,18,19,20), (3,4,5,6,11,7) ]
gap> List(U1,Order);
[ 4, 5, 5, 4, 5, 5, 5, 5, 5, 5, 6 ]
gap> Collected(last);
[ [ 4, 2 ], [ 5, 8 ], [ 6, 1 ] ]
```

Note, applying the butterfly deletion gives rise to the butterflies [3,7] and [5,6]. But since we are only interested in butterflies which are also contained in the icosahadron, those butterflies can be ignored.

For further computations the automorphism group on the faces of the icosahedron will be helpful.

Example

```
gap> GK:=AutomorphismGroupOnFaces(Ik);
Group([ (1,2)(3,5)(6,7)(8,10)(12,15)(13,14)(17,20)(18,19), (1,2)(3,6)(4,11)
  (5,7)(8,15)(9,16)(10,12)(13,20)(14,17)(18,19), (1,5,4,3,2)(6,10,9,8,7)
  (11,15,14,13,12)(16,20,19,18,17) ])
```

Since a butterfly can be seen as the set of two neighbouring faces in the umbrella of a vertex, the set of butterflies of the icosahedron can be computed by using the cycles given by the umbrella descriptor.

Example

```
Bu:=List([1..20],i->List(UIk,g->Set([i,i^g]));
gap> Bu:=List([1..20],i->List(UIk,g->Set([i,i^g])));
[ [ [ 1, 5 ], [ 1, 6 ], [ 1, 6 ], [ 1 ], [ 1 ], [ 1 ], [ 1 ], [ 1 ], [ 1 ],
      [ 1 ], [ 1 ], [ 1 ] ],
  [ [ 1, 2 ], [ 1, 2 ], [ 2 ], [ 2, 7 ], [ 2 ], [ 2 ], [ 2 ], [ 2 ], [ 2 ],
      [ 2 ], [ 2 ], [ 2 ] ],
  [ [ 2, 3 ], [ 3 ], [ 3 ], [ 2, 3 ], [ 3, 8 ], [ 3 ], [ 3 ], [ 3 ], [ 3 ],
      [ 3 ], [ 3 ], [ 3 ] ],
  [ [ 3, 4 ], [ 4 ], [ 4 ], [ 4 ], [ 3, 4 ], [ 4, 9 ], [ 4 ], [ 4 ], [ 4 ],
      [ 4 ], [ 4 ], [ 4 ] ],
  [ [ 4, 5 ], [ 5 ], [ 1, 5 ], [ 5 ], [ 5 ], [ 4, 5 ], [ 5 ], [ 5 ], [ 5 ],
      [ 5 ], [ 5 ], [ 5 ] ],
  [ [ 6 ], [ 6, 11 ], [ 6, 15 ], [ 6 ], [ 6 ], [ 6 ], [ 6, 15 ], [ 6 ],
      [ 6 ], [ 6 ], [ 6 ], [ 6 ] ],
  [ [ 7 ], [ 2, 7 ], [ 7 ], [ 7, 12 ], [ 7 ], [ 7 ], [ 7 ], [ 7, 12 ],
      [ 7 ], [ 7 ], [ 7 ], [ 7 ] ],
  [ [ 8 ], [ 8 ], [ 8 ], [ 3, 8 ], [ 8, 13 ], [ 8 ], [ 8 ], [ 8 ],
      [ 8, 13 ], [ 8 ], [ 8 ], [ 8 ] ],
  [ [ 9 ], [ 9 ], [ 9 ], [ 9 ], [ 4, 9 ], [ 9, 14 ], [ 9 ], [ 9 ], [ 9 ],
      [ 9, 14 ], [ 9 ], [ 9 ] ],
  [ [ 10 ], [ 10 ], [ 5, 10 ], [ 10 ], [ 10 ], [ 5, 10 ], [ 10 ], [ 10 ],
      [ 10 ], [ 10 ], [ 10, 14 ], [ 10 ] ],
  [ [ 11 ], [ 7, 11 ], [ 11 ], [ 11 ], [ 11 ], [ 11 ], [ 6, 11 ], [ 7, 11 ],
      [ 11 ], [ 11 ], [ 11 ], [ 11 ] ],
  [ [ 12 ], [ 12 ], [ 12 ], [ 8, 12 ], [ 12 ], [ 12 ], [ 12 ], [ 12, 17 ],
      [ 8, 12 ], [ 12 ], [ 12 ], [ 12 ] ],
  [ [ 13 ], [ 13 ], [ 13 ], [ 13 ], [ 9, 13 ], [ 13 ], [ 13 ], [ 13 ],
      [ 13, 18 ], [ 9, 13 ], [ 13 ], [ 13 ] ],
  [ [ 14 ], [ 14 ], [ 14 ], [ 14 ], [ 14 ], [ 10, 14 ], [ 14 ], [ 14 ],
      [ 14 ], [ 14, 19 ], [ 14, 19 ], [ 14 ] ],
  [ [ 15 ], [ 15 ], [ 10, 15 ], [ 15 ], [ 15 ], [ 15 ], [ 15, 20 ], [ 15 ],
      [ 15 ], [ 15 ], [ 10, 15 ], [ 15 ] ],
  [ [ 16 ], [ 16 ], [ 16 ], [ 16 ], [ 16 ], [ 16 ], [ 11, 16 ], [ 11, 16 ],
      [ 16 ], [ 16 ], [ 16 ], [ 16, 17 ] ],
  [ [ 17 ], [ 17 ], [ 17 ], [ 17 ], [ 17 ], [ 17 ], [ 17 ], [ 16, 17 ],
      [ 12, 17 ], [ 17 ], [ 17 ], [ 17, 18 ] ],
  [ [ 18 ], [ 18 ], [ 18 ], [ 18 ], [ 18 ], [ 18 ], [ 18 ], [ 18 ],
      [ 17, 18 ], [ 13, 18 ], [ 18 ], [ 18, 19 ] ],
  [ [ 19 ], [ 19 ], [ 19 ], [ 19 ], [ 19 ], [ 19 ], [ 19 ], [ 19 ], [ 19 ],
      [ 18, 19 ], [ 19, 20 ], [ 19, 20 ] ],
  [ [ 20 ], [ 20 ], [ 20 ], [ 20 ], [ 20 ], [ 20 ], [ 16, 20 ], [ 20 ],
      [ 20 ], [ 20 ], [ 15, 20 ], [ 16, 20 ] ] ]
```

```
gap> Bu:=Union(Bu);
[ [ 1 ], [ 1, 2 ], [ 1, 5 ], [ 1, 6 ], [ 2 ], [ 2, 3 ], [ 2, 7 ], [ 3 ],
  [ 3, 4 ], [ 3, 8 ], [ 4 ], [ 4, 5 ], [ 4, 9 ], [ 5 ], [ 5, 10 ], [ 6 ],
  [ 6, 11 ], [ 6, 15 ], [ 7 ], [ 7, 11 ], [ 7, 12 ], [ 8 ], [ 8, 12 ],
  [ 8, 13 ], [ 9 ], [ 9, 13 ], [ 9, 14 ], [ 10 ], [ 10, 14 ], [ 10, 15 ],
  [ 11 ], [ 11, 16 ], [ 12 ], [ 12, 17 ], [ 13 ], [ 13, 18 ], [ 14 ],
  [ 14, 19 ], [ 15 ], [ 15, 20 ], [ 16 ], [ 16, 17 ], [ 16, 20 ], [ 17 ],
  [ 17, 18 ], [ 18 ], [ 18, 19 ], [ 19 ], [ 19, 20 ], [ 20 ] ]
gap> Bu:=Filtered(Bu, r->Size(r)=2);
[ [ 1, 2 ], [ 1, 5 ], [ 1, 6 ], [ 2, 3 ], [ 2, 7 ], [ 3, 4 ], [ 3, 8 ],
  [ 4, 5 ], [ 4, 9 ], [ 5, 10 ], [ 6, 11 ], [ 6, 15 ], [ 7, 11 ], [ 7, 12 ],
  [ 8, 12 ], [ 8, 13 ], [ 9, 13 ], [ 9, 14 ], [ 10, 14 ], [ 10, 15 ],
  [ 11, 16 ], [ 12, 17 ], [ 13, 18 ], [ 14, 19 ], [ 15, 20 ], [ 16, 17 ],
  [ 16, 20 ], [ 17, 18 ], [ 18, 19 ], [ 19, 20 ] ]
gap> Size(Bu);
30
```

Note, up to isomorphism there exists exactly one butterfly in the icosahedron.

Assuming the butterfly containing the faces 1 and 2 has already been omitted, then the stabilizer of the butterfly [1,2] will be of greater interest. The stabilizer will identify which pairs of butterflies will construct isomorphic spheres by checking which pairs are contained in the same orbit.

Example

```
gap> SK:=Stabilizer(GK,[1,2],OnSets);
Group([ (3,7)(4,11)(5,6)(8,12)(9,16)(10,15)(13,17)(14,20), (1,2)(3,5)(6,7)
  (8,10)(12,15)(13,14)(17,20)(18,19), (1,2)(3,6)(4,11)(5,7)(8,15)(9,16)
  (10,12)(13,20)(14,17)(18,19) ])
gap> Size(SK);
4
```

TODO

How exactly can the spheres arising from exactly two butterfly deletions can be determined? Because of omitting the faces 1 and 2 from the icosahedron in the first step, the butterflies containing one of those faces can be ignored.

Example

```
gap> BuSel:=Filtered(Bu,r->Intersection([1,2],r)=[]);
[ [ 3, 4 ], [ 3, 8 ], [ 4, 5 ], [ 4, 9 ], [ 5, 10 ], [ 6, 11 ], [ 6, 15 ],
  [ 7, 11 ], [ 7, 12 ], [ 8, 12 ], [ 8, 13 ], [ 9, 13 ], [ 9, 14 ],
  [ 10, 14 ], [ 10, 15 ], [ 11, 16 ], [ 12, 17 ], [ 13, 18 ], [ 14, 19 ],
  [ 15, 20 ], [ 16, 17 ], [ 16, 20 ], [ 17, 18 ], [ 18, 19 ], [ 19, 20 ] ]
```

Among the remaining butterflies we will compute the orbits under the automorphism group to find candidates for the second butterfly for the butterfly deletion.

Example

```
gap> BeSelOr:=Orbits(SK,BuSel,OnSets);
[ [ [ 3, 4 ], [ 7, 11 ], [ 4, 5 ], [ 6, 11 ] ],
  [ [ 3, 8 ], [ 7, 12 ], [ 5, 10 ], [ 6, 15 ] ], [ [ 4, 9 ], [ 11, 16 ] ],
  [ [ 8, 12 ], [ 10, 15 ] ], [ [ 8, 13 ], [ 12, 17 ], [ 10, 14 ], [ 15, 20 ]
     ], [ [ 9, 13 ], [ 16, 17 ], [ 9, 14 ], [ 16, 20 ] ],
  [ [ 13, 18 ], [ 17, 18 ], [ 14, 19 ], [ 19, 20 ] ], [ [ 18, 19 ] ] ]
```

```
gap> List(BeSelOr,Size);
[ 4, 4, 2, 2, 4, 4, 4, 1 ]
gap> Kand:=List( BeSelOr,r->Set([Set([1,2]),r[1]]));
[ [ [ 1, 2 ], [ 3, 4 ] ], [ [ 1, 2 ], [ 3, 8 ] ], [ [ 1, 2 ], [ 4, 9 ] ],
  [ [ 1, 2 ], [ 8, 12 ] ], [ [ 1, 2 ], [ 8, 13 ] ], [ [ 1, 2 ], [ 9, 13 ] ],
  [ [ 1, 2 ], [ 13, 18 ] ], [ [ 1, 2 ], [ 18, 19 ] ] ]
```

Since different pairs of butterflies in the above set can still construct isomorphic spheres, we determine the orbits of the automorphism group on this set to check which pairs construct isomorphic spheres.

Example

```
gap> kand:=List(Kand,r->Orbit(GK,r,OnSetsSets));;
gap> Size(Set(kand));
8
gap> Size(Set(List(kand,r->Set(r))));
7
```

So two pairs of butterflies appear in the same orbit. This can also be verified by examining the corresponding stabilizers.

Example

```
gap> Filtered(Kand, r->IsSubgroup(SK,Stabilizer(GK,r,OnSetsSets)));
[ [ [ 1, 2 ], [ 4, 9 ] ], [ [ 1, 2 ], [ 8, 12 ] ] ]
```

Compute the resulting set of pairs of butterflies

Example

```
gap> Filtered(Kand, r-> r<>[ [ 1, 2 ], [ 8, 12 ] ]);
[ [ [ 1, 2 ], [ 3, 4 ] ], [ [ 1, 2 ], [ 3, 8 ] ], [ [ 1, 2 ], [ 4, 9 ] ],
  [ [ 1, 2 ], [ 8, 13 ] ], [ [ 1, 2 ], [ 9, 13 ] ], [ [ 1, 2 ], [ 13, 18 ] ],
  [ [ 1, 2 ], [ 18, 19 ] ] ]
gap> Kand:=last;
[ [ [ 1, 2 ], [ 3, 4 ] ], [ [ 1, 2 ], [ 3, 8 ] ], [ [ 1, 2 ], [ 4, 9 ] ],
  [ [ 1, 2 ], [ 8, 13 ] ], [ [ 1, 2 ], [ 9, 13 ] ], [ [ 1, 2 ], [ 13, 18 ] ],
  [ [ 1, 2 ], [ 18, 19 ] ] ]
```

Now use these pairs to finally obtain the simplicial surfaces constructed from the icosahedron

Example

```
gap> Fl:=List(Kand,r->Zus(Zus(UIk,r[1,1],r[1,2]),r[2,1],r[2,2]));
[ [ (5,6,15,10), (7,12,8), (5,9,14,10), (6,15,20,16,11), (7,12,17,16,11),
      (8,13,18,17,12), (9,14,19,18,13), (10,14,19,20,15), (16,17,18,19,20),
      (5,9,13,8,7,11,6) ],
  [ (5,6,15,10), (4,9,14,10,5), (6,15,20,16,11), (7,12,17,16,11),
      (12,13,18,17), (9,14,19,18,13), (10,14,19,20,15), (16,17,18,19,20),
      (4,5,6,11,7), (4,9,13,12,7) ],
  [ (5,6,15,10), (3,7,12,8), (6,15,20,16,11), (7,12,17,16,11),
      (8,13,18,17,12), (13,14,19,18), (10,14,19,20,15), (16,17,18,19,20),
      (3,5,6,11,7), (3,5,10,14,13,8) ],
  [ (5,6,15,10), (3,7,12), (4,9,14,10,5), (6,15,20,16,11), (7,12,17,16,11),
      (9,14,19,18), (10,14,19,20,15), (16,17,18,19,20), (3,4,5,6,11,7),
      (3,12,17,18,9,4) ],
```

```
  [ (5,6,15,10), (3,7,12,8), (4,14,10,5), (6,15,20,16,11), (7,12,17,16,11),
      (8,18,17,12), (10,14,19,20,15), (16,17,18,19,20), (3,4,5,6,11,7),
      (3,4,14,19,18,8) ],
  [ (5,6,15,10), (3,7,12,8), (3,8,9,4), (4,9,14,10,5), (6,15,20,16,11),
      (7,12,17,16,11), (10,14,19,20,15), (16,17,19,20), (3,4,5,6,11,7),
      (8,9,14,19,17,12) ],
  [ (5,6,15,10), (3,7,12,8), (3,8,13,9,4), (4,9,14,10,5), (6,15,20,16,11),
      (7,12,17,16,11), (8,13,17,12), (10,14,20,15), (3,4,5,6,11,7),
      (9,13,17,16,20,14) ] ]
gap> List(Fl,r->Collected(List(r,Order)));
[ [ [ 3, 1 ], [ 4, 2 ], [ 5, 6 ], [ 7, 1 ] ], [ [ 4, 2 ], [ 5, 8 ] ],
  [ [ 4, 3 ], [ 5, 6 ], [ 6, 1 ] ],
  [ [ 3, 1 ], [ 4, 2 ], [ 5, 5 ], [ 6, 2 ] ],
  [ [ 4, 4 ], [ 5, 4 ], [ 6, 2 ] ], [ [ 4, 4 ], [ 5, 4 ], [ 6, 2 ] ],
  [ [ 4, 4 ], [ 5, 4 ], [ 6, 2 ] ] ]
gap> VerCou:=last;;
```

Those pairs can also be identified by computing the distances of the faces contained in a pair of butterlfies.

Example

```
gap> List(Kand, r->List(r[1],i->List(r[2],j->DistanceOfFaces(Ik,i,j))));
[ [ [ 2, 2 ], [ 1, 2 ] ], [ [ 2, 3 ], [ 1, 2 ] ], [ [ 2, 3 ], [ 2, 3 ] ],
  [ [ 3, 4 ], [ 2, 3 ] ], [ [ 3, 4 ], [ 3, 3 ] ], [ [ 4, 5 ], [ 3, 4 ] ],
  [ [ 5, 4 ], [ 4, 5 ] ] ]
```

This identification can be simplified by computing the distances of the faces of the butterflies to the face 1.

Example

```
gap> Collected(List([1..20],i->DistanceOfFaces(Ik,1,i)));
[ [ 0, 1 ], [ 1, 3 ], [ 2, 6 ], [ 3, 6 ], [ 4, 3 ], [ 5, 1 ] ]
```