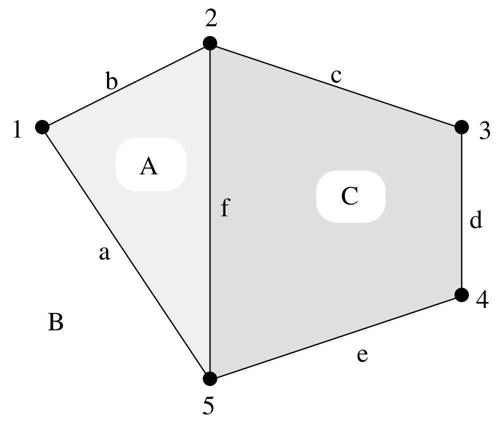

Figure 4: Example 2 without dangling edges.

Type-Definition 1 :

```
typedef vector<Dcel_edge_type<I> >   Edges_container;
typedef vector<Dcel_face_type<I> >   Faces_container;
typedef vector<Dcel_vertex_type<I> > Vertices_container;
```

Also the most important geometric types are predefined in I:

Type-Definition 2 :

```
typedef typename I::Point   Point;
typedef typename I::Segment Segment;
typedef typename I::Polygon Polygon;
```

2.2 DCEL Traverse

A DCEL can be traversed in several ways:

- Iterating over all edges, vertices, or faces.
- Iterating over all edges incident to a certain vertex or face.
- Walking around a given face and collecting all vertices in a certain order, i.e. clockwise or counter-clockwise.

Since we are iterating over a set of elements or around a vertex or face, we are using the so called *iterators*. The *iterator* concept can be seen as a generalization of pointer concept and was invented in the STL-library ([SL95],[MS96]).

For iterating on the element containers of the DCEL we use the following iterator types:

Type-Definition 3 :