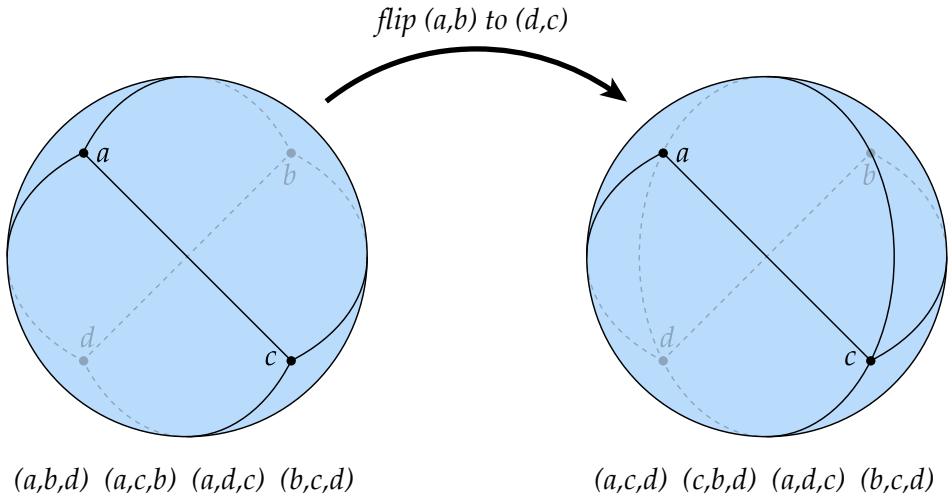
Edge flips on a tetrahedron: oriented vs. nonoriented meshes.



Recall that a polygon mesh without boundary is *manifold* if (i) each edge is contained in two faces and (ii) each vertex is contained in a single cycle of faces. On the left we visualize the connectivity of a tetrahedron on the sphere: each edge between two vertices is drawn as a circular arc (solid edges are in front of the sphere; dashed edges are behind.) This picture makes it quite clear that the tetrahedron has manifold topology, since the four triangles partition the sphere into four simple regions—and the sphere itself is manifold. On the right, we visualize the connectivity of the mesh after flipping the edge (a,b) to get the edge (d,c). When interpreting this figure, one should be careful to think about which edges are on the front and back of the sphere, as indicated by the line style (dashed or solid). Otherwise, one might mistakenly believe that, e.g., vertices *b* and *c* are connected by the solid arc touching the north pole (they are not).

Is the new mesh still manifold? Well, the four triangles partition the sphere into four simple regions—and the sphere is of course still manifold! We can also check properties (i) and (ii) to find that (i) each edge is contained in two faces and (ii) each vertex is contained in a single cycle of faces. However, there are some things about this new figure that are potentially confusing. For one thing, two of the vertices (*a* & *b*) now have degree two. There is nothing fundamentally "wrong" with having degree-two vertices in a mesh, but it may lead to confusion when drawing triangles using straight lines rather than spherical arcs. The other thing that is potentially confusing as that if we ignore the *order* of the vertices, it seems like we only have two triangles. For instance, *(a,c,d)* and *(a,d,c)* have the same vertices, but in a different order; importantly, they are related by an odd permutation, which means they have different *orientation*. In other words, these two triangles are different if we describe the surface as an oriented triangle mesh, and different if we describe it as an unoriented triangle mesh. Neither notion is "right" or "wrong"; they are simply two different types of objects that we could talk about and work with, each with different consequences.

What about our halfedge representation? Does it naturally represent oriented or unoriented mesh? And can we represent the figure on the right? The answer is that a halfedge represents an *oriented* mesh, since the halfedges clearly define an ordering of vertices with each polygon (e.g., a then b then d, as determined by "next" pointers). Therefore, we can easily represent the configuration on the right as a manifold surface. That being said, we may choose not to allow this type of edge flip for other reasons. For instance, we may not like meshes with degree-two vertices, if for no other reason that they can be confusing! (They also tend to lead to poorly-shaped geometry when we draw our triangles with straight edges, as usual).

For this assignment, you simply need to pick conventions that are self-consistent—so that your code does not crash! For instance, if you want to avoid edge flips that produce degree-two vertices, that is perfectly acceptable. If you want to handle corner cases like the one depicted above, that is also fine—as long as it continues to produce a consistent mesh (and doesn't crash!) upon subsequent edge operations!