

COS 426 : Precept 4

Half-Edge

Agenda

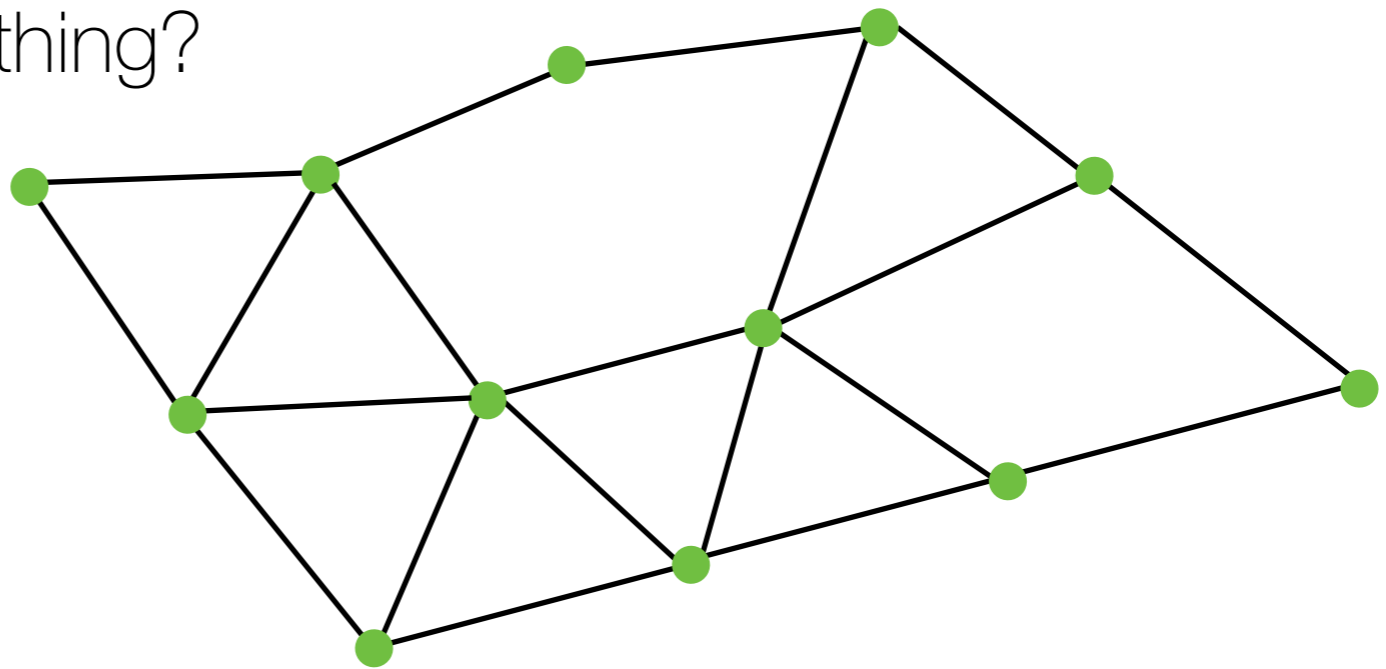
- Assignment 2 description
- Half-edge data structure
 - Traversal
 - Modification

Assignment 2

- Part 1 - Analysis
 - Implement traversal operation
 - Calculate mesh properties
 - Vertex normal, avg. edge length, etc.
- Part 2 - Filters
 - Filters and Warps similar to assignment 1
 - Topological modifiers

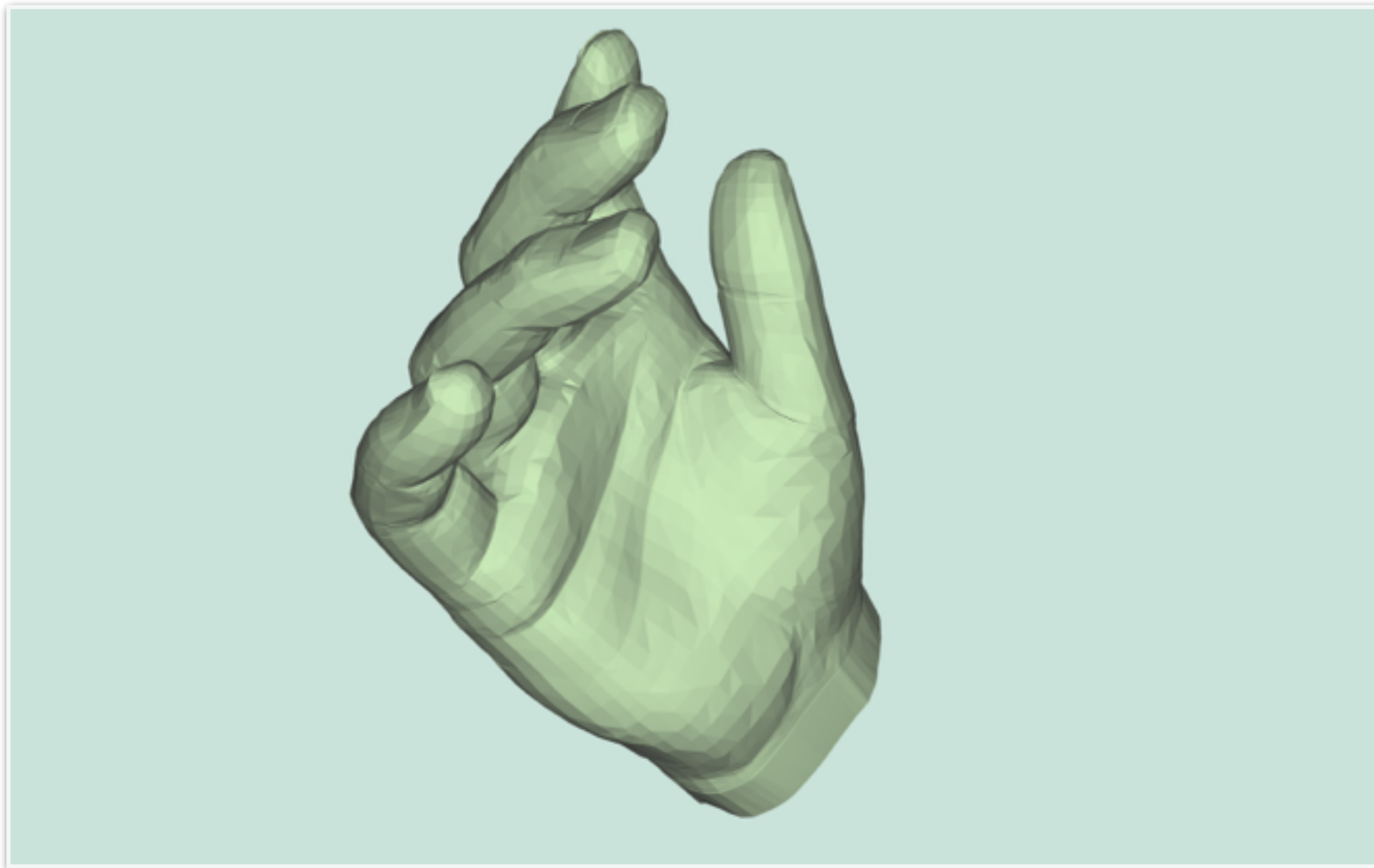
Meshes

- Images had implicit adjacency information
 - Grid around a pixel
 - Easy to express operations
- What about meshes?
 - How to apply smoothing?



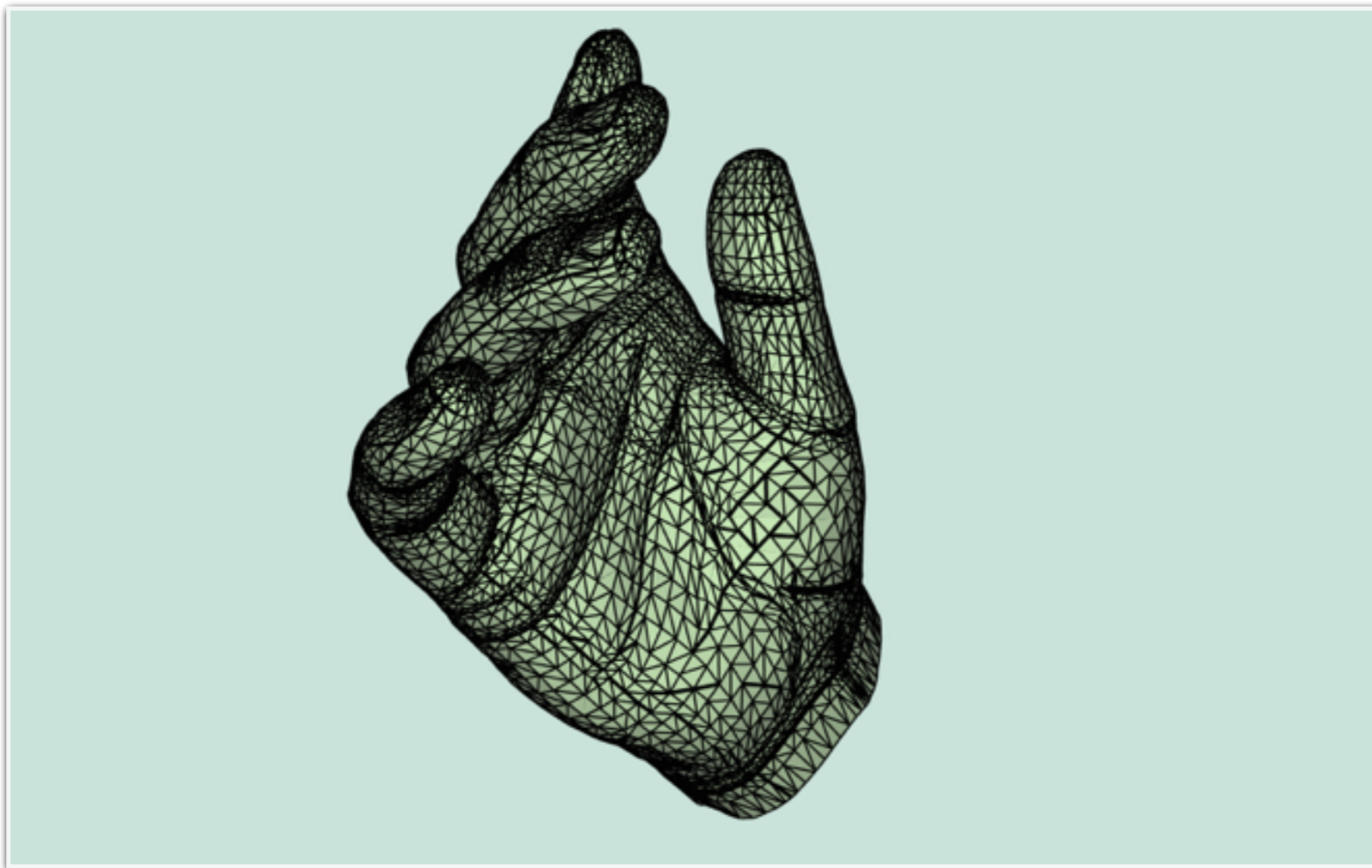
Meshes

- Meshes can be quite dense



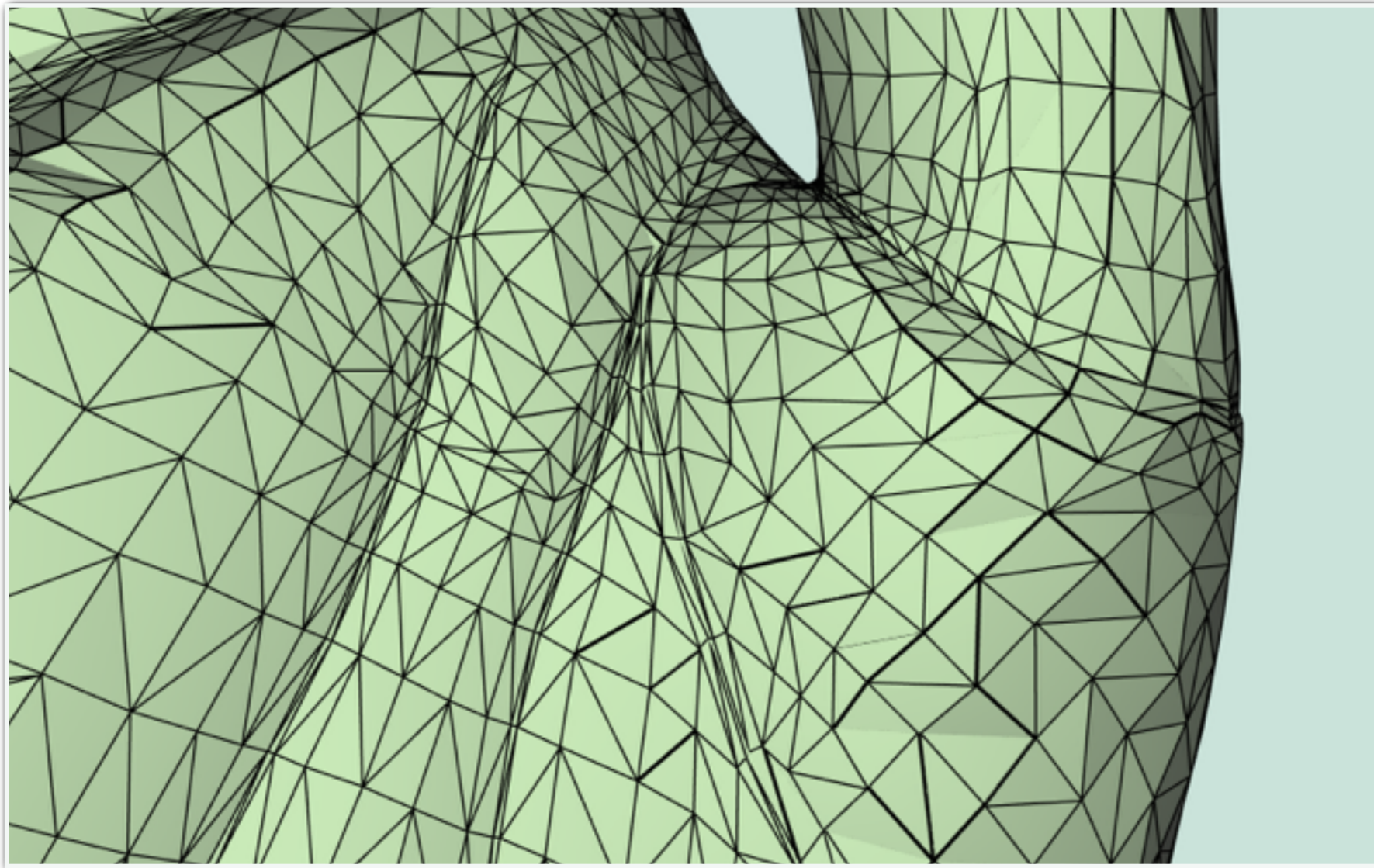
Meshes

- Meshes can be quite dense



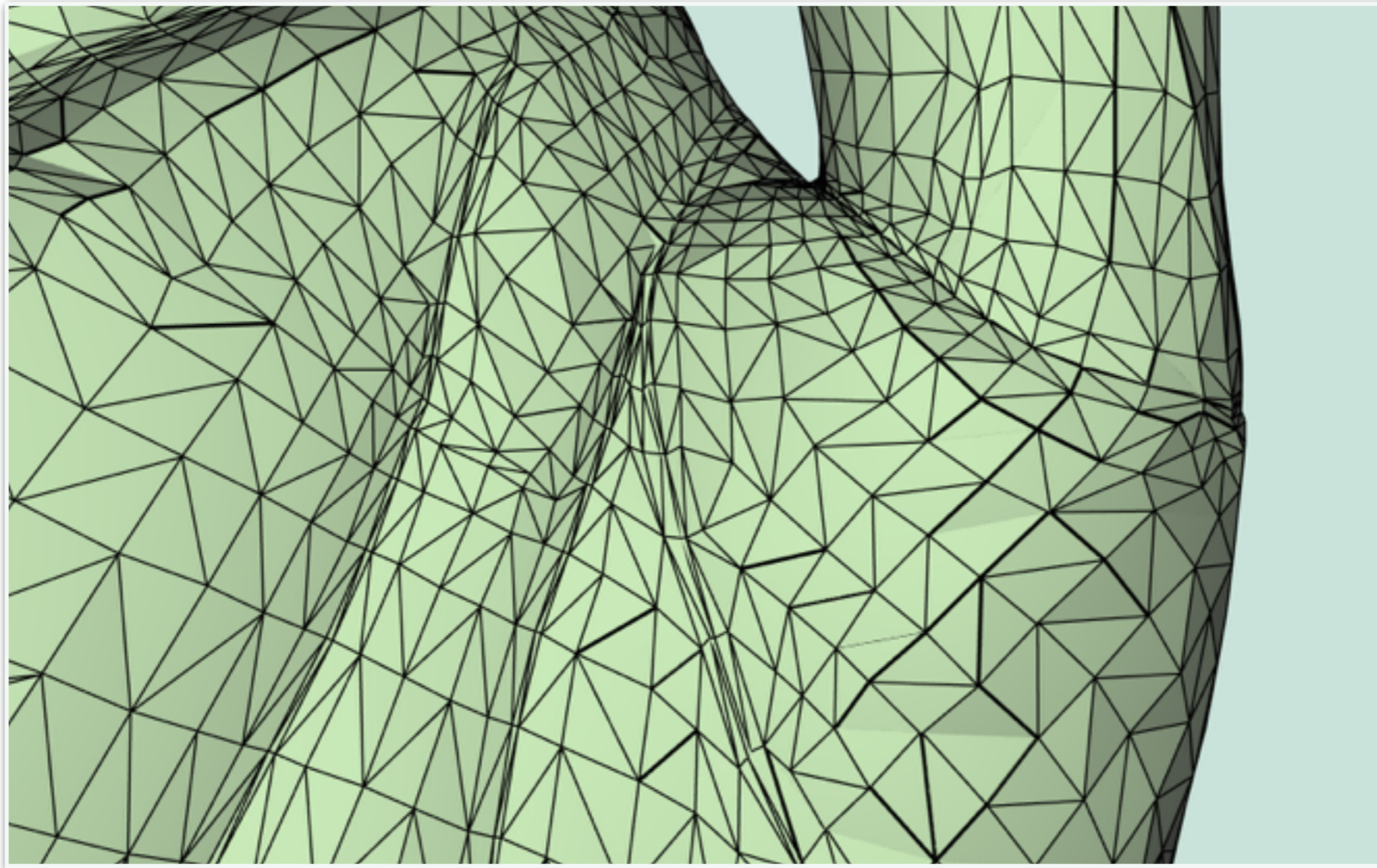
Meshes

- Meshes can be quite dense



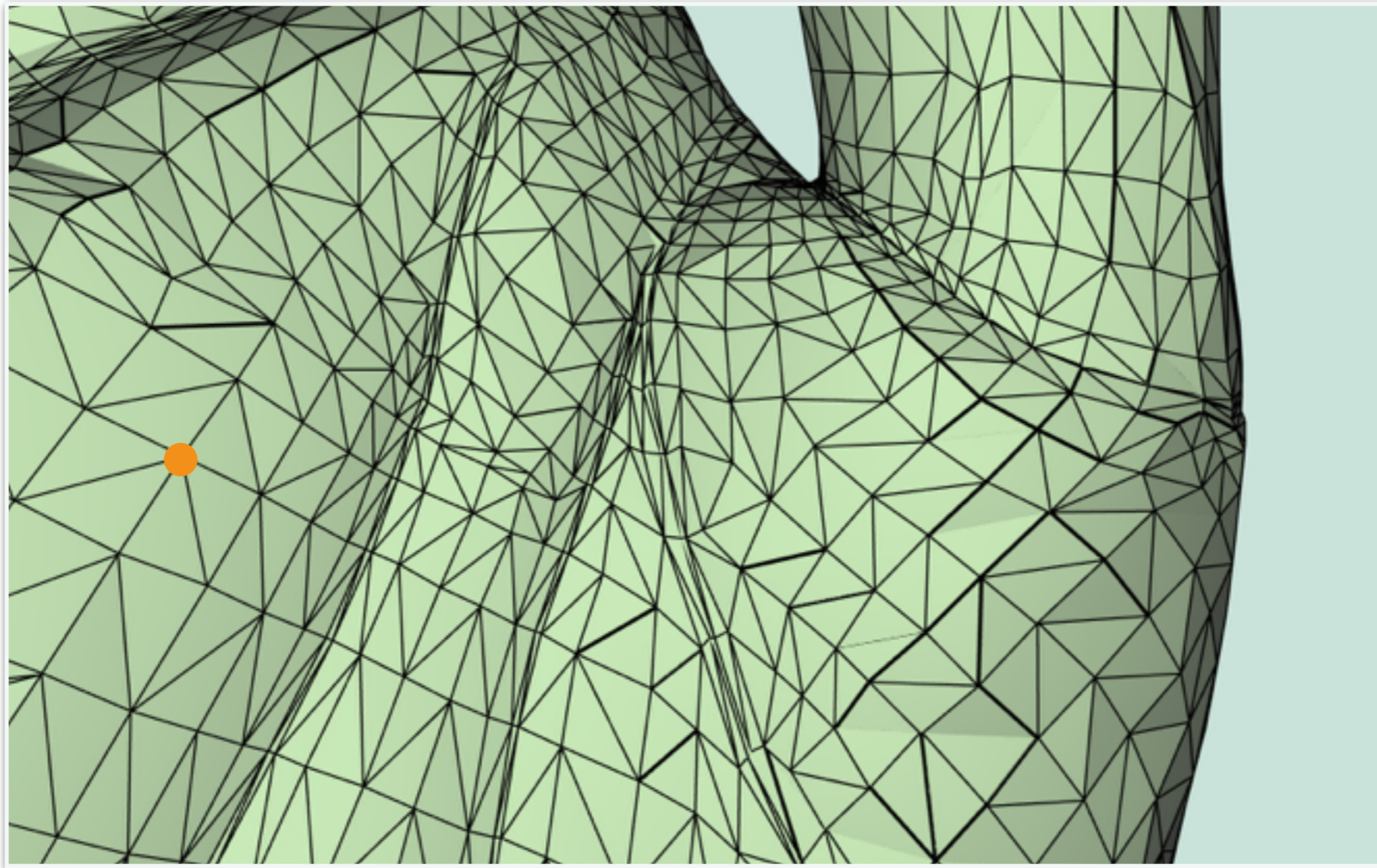
Meshes

- How to access adjacency information quickly?



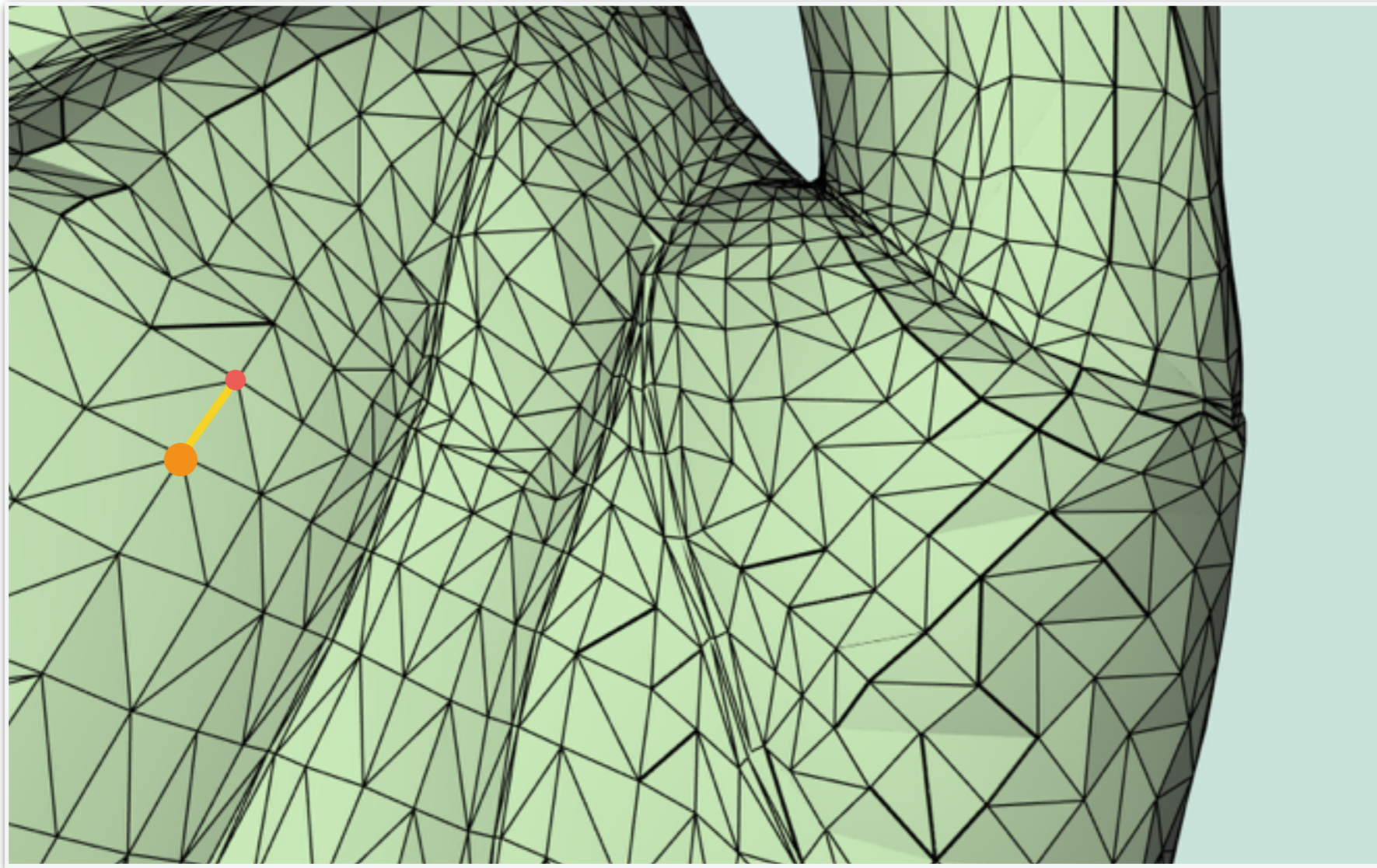
Meshes

- How to access adjacency information quickly?



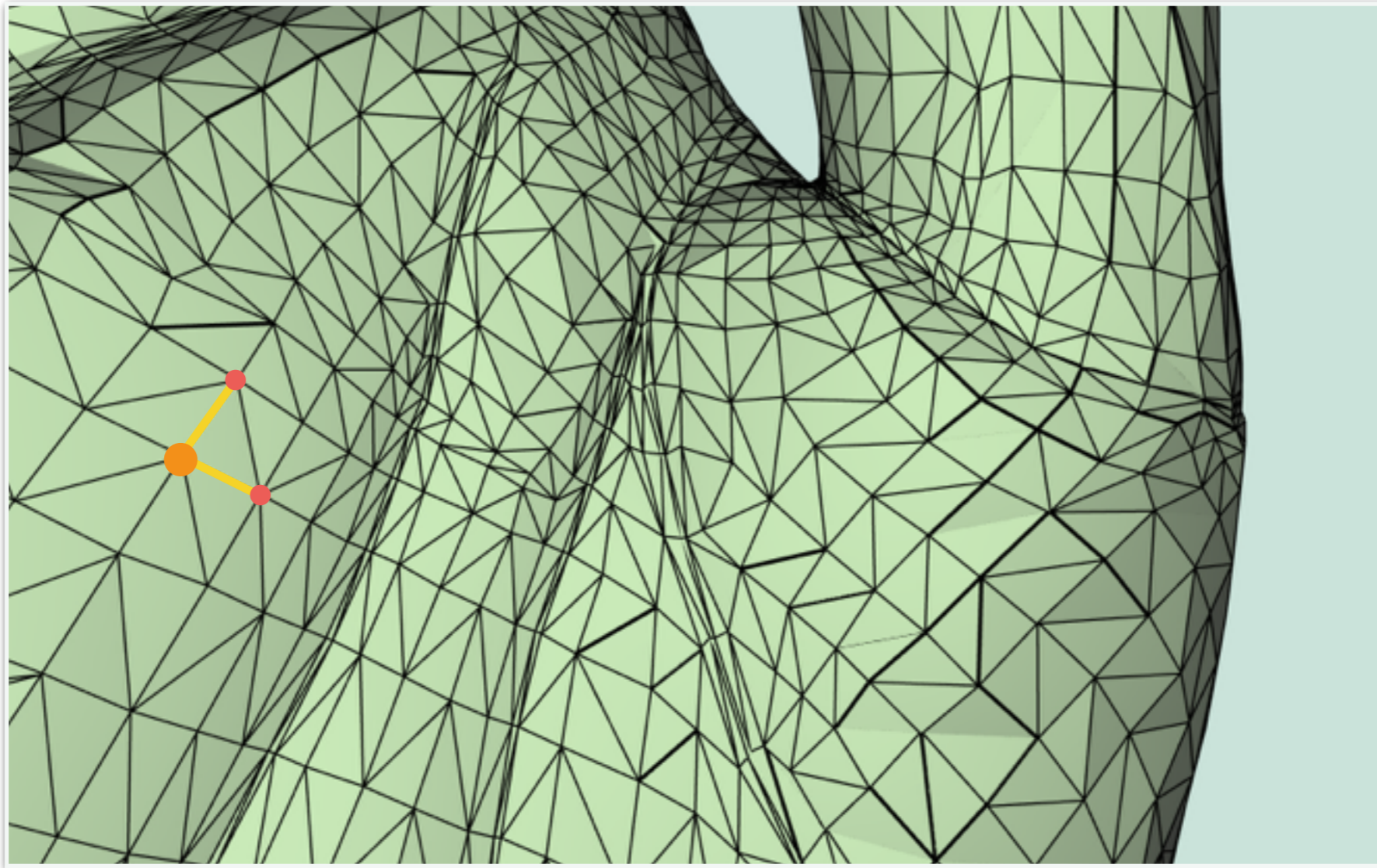
Meshes

- How to access adjacency information quickly?



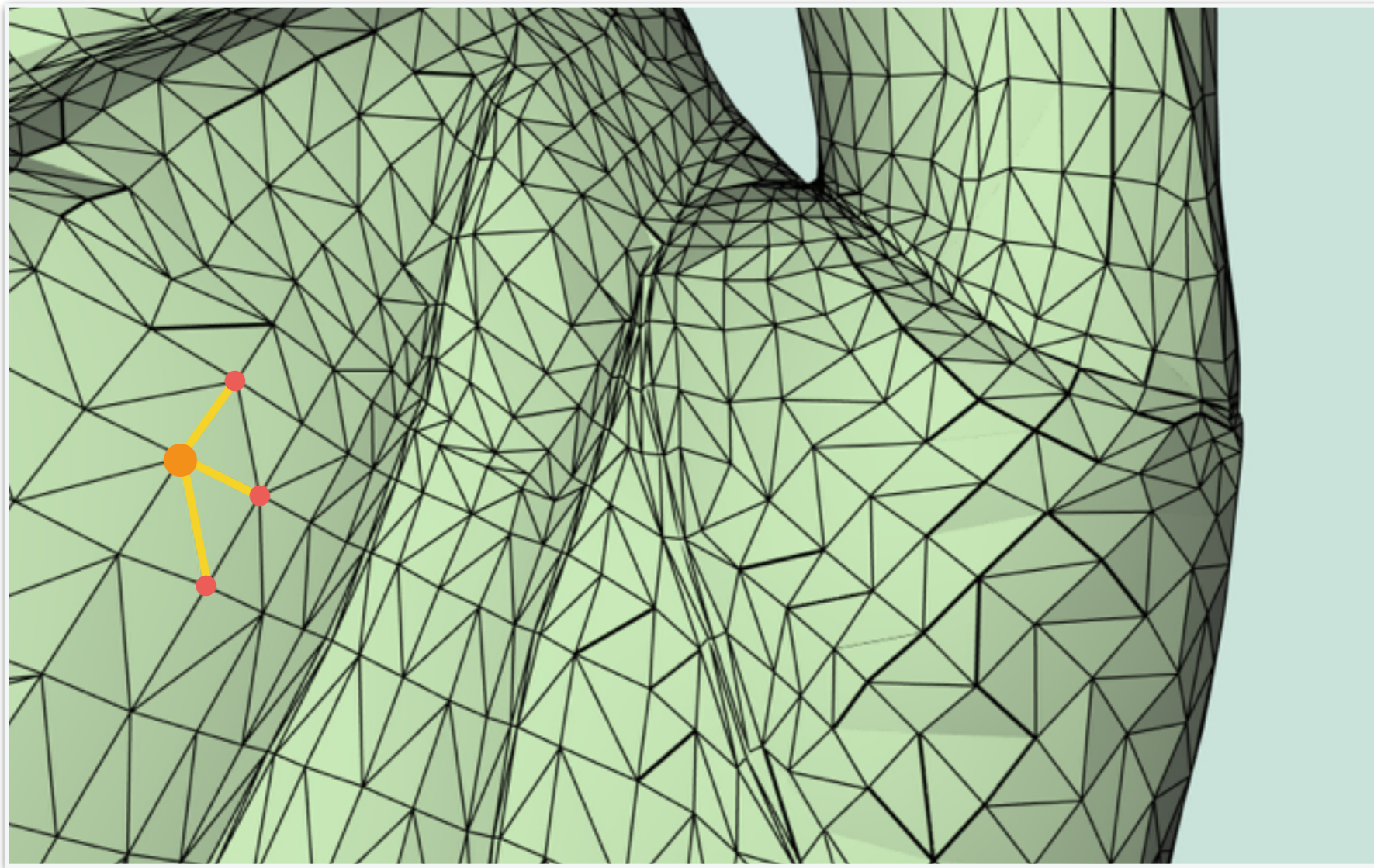
Meshes

- How to access adjacency information quickly?



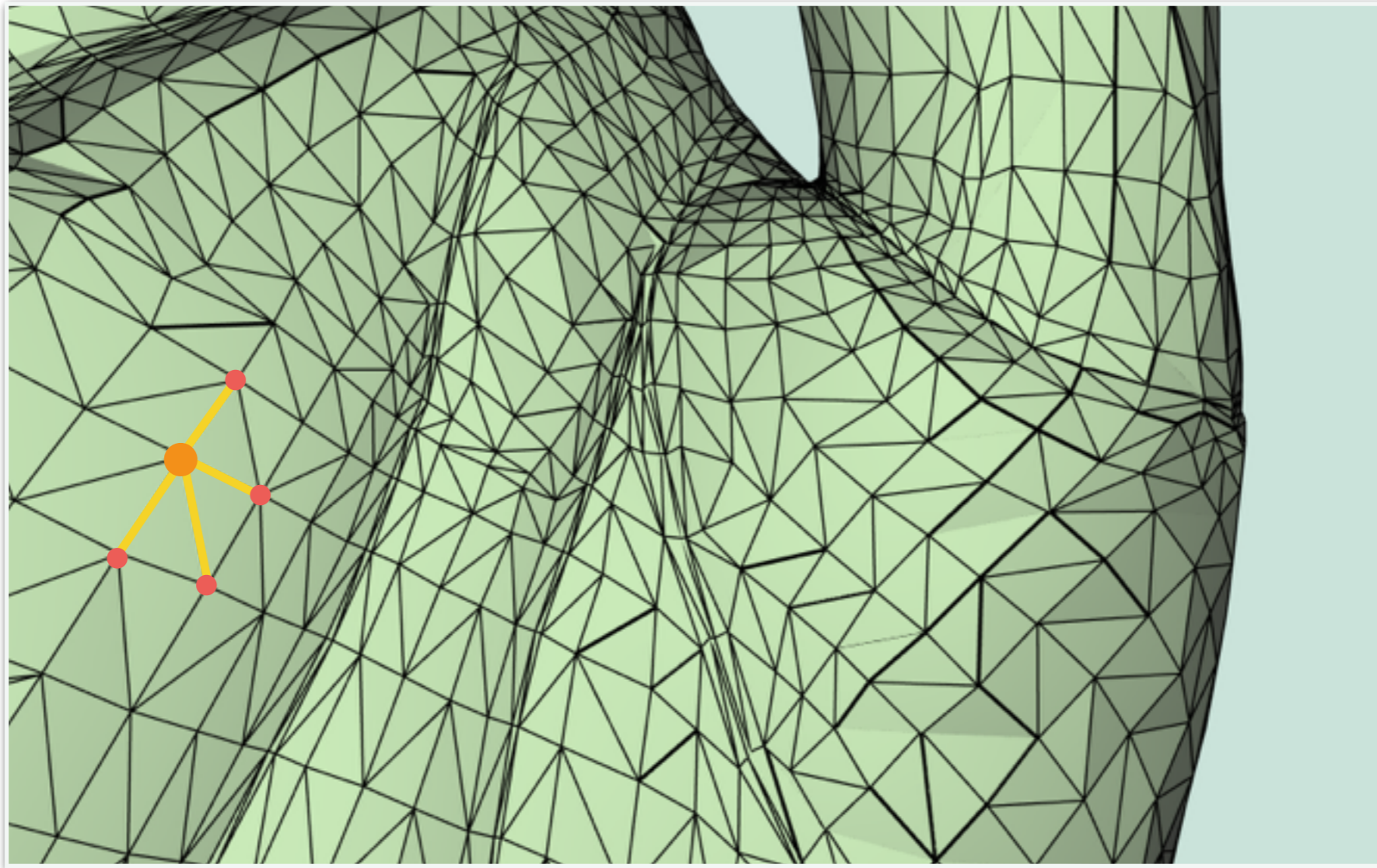
Meshes

- How to access adjacency information quickly?



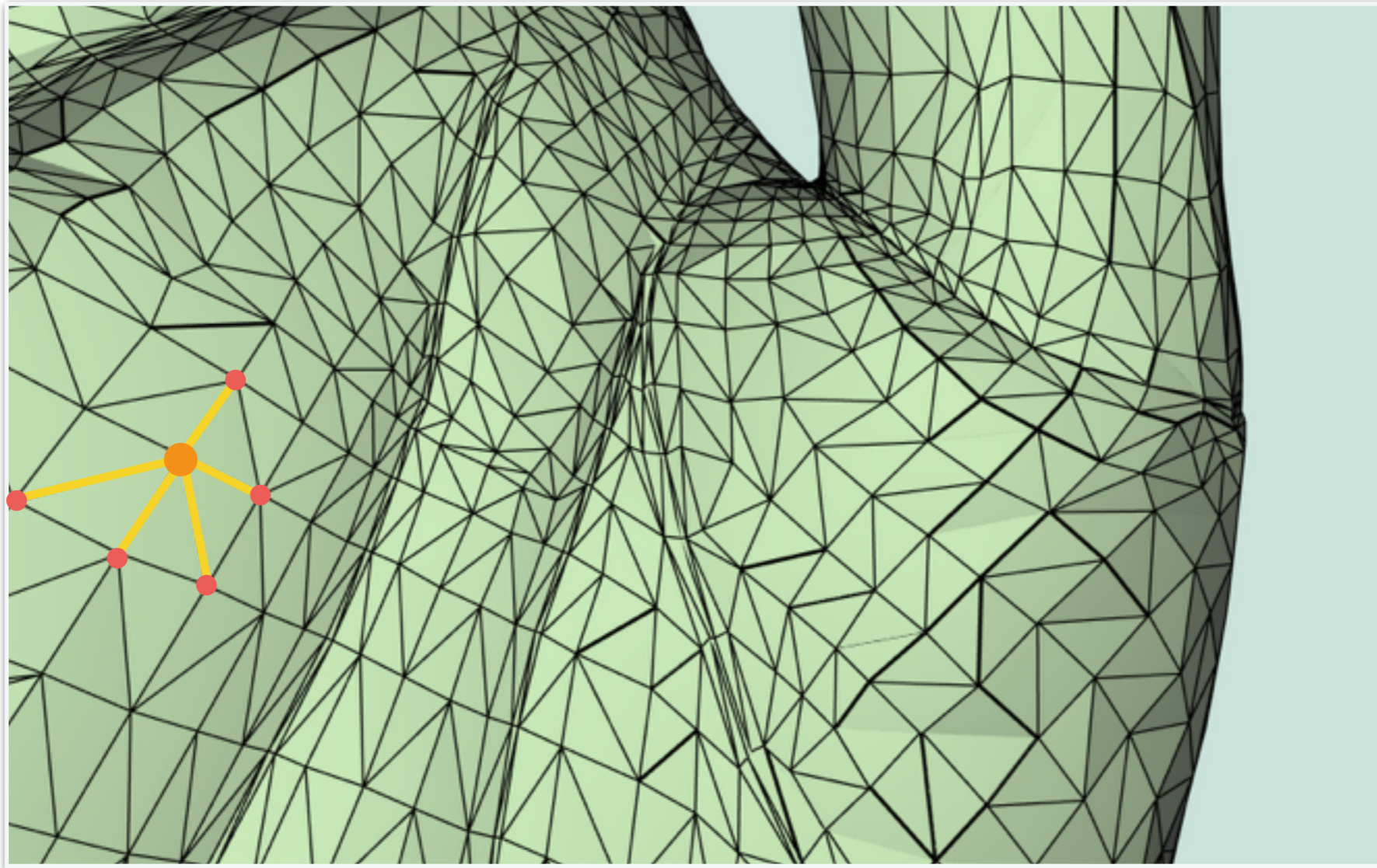
Meshes

- How to access adjacency information quickly?



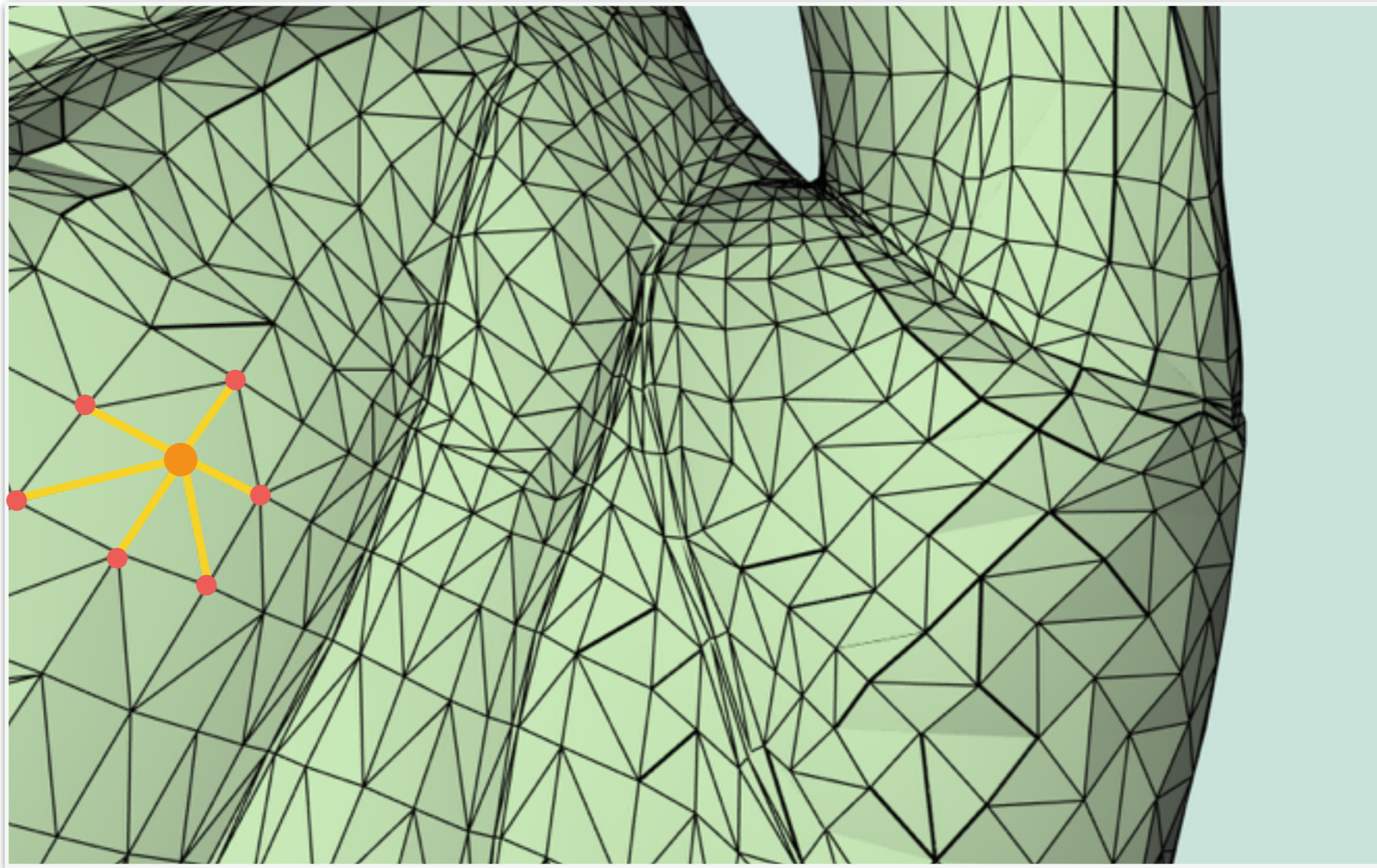
Meshes

- How to access adjacency information quickly?



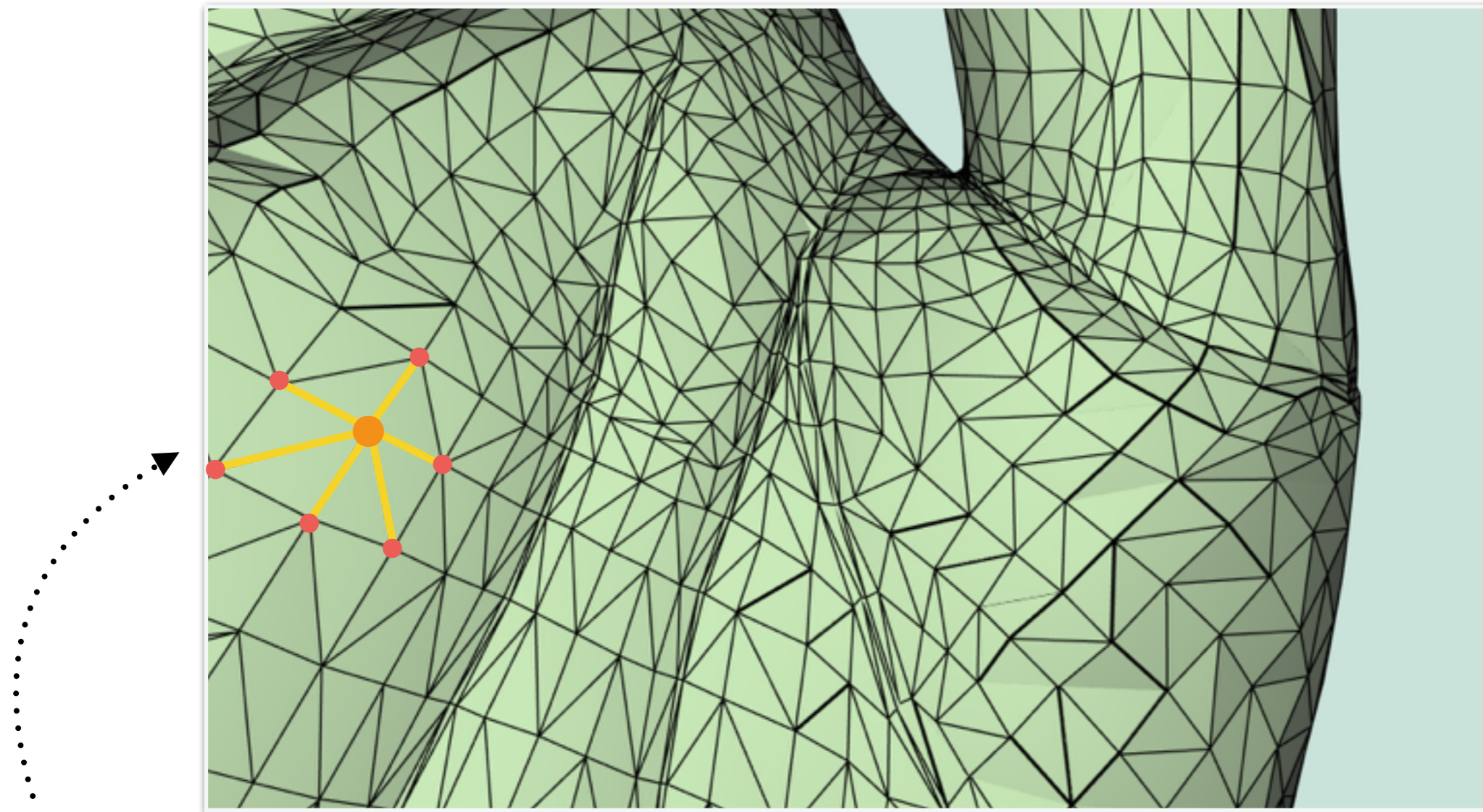
Meshes

- How to access adjacency information quickly?



Meshes

- How to access adjacency information quickly?



One - Ring Neighborhood

Half-Edge Data Structure

Half Edge

Vertex

Face

Vertex

Position

Half-Edge

Opposite
Half-Edge

Outgoing
Half-Edge

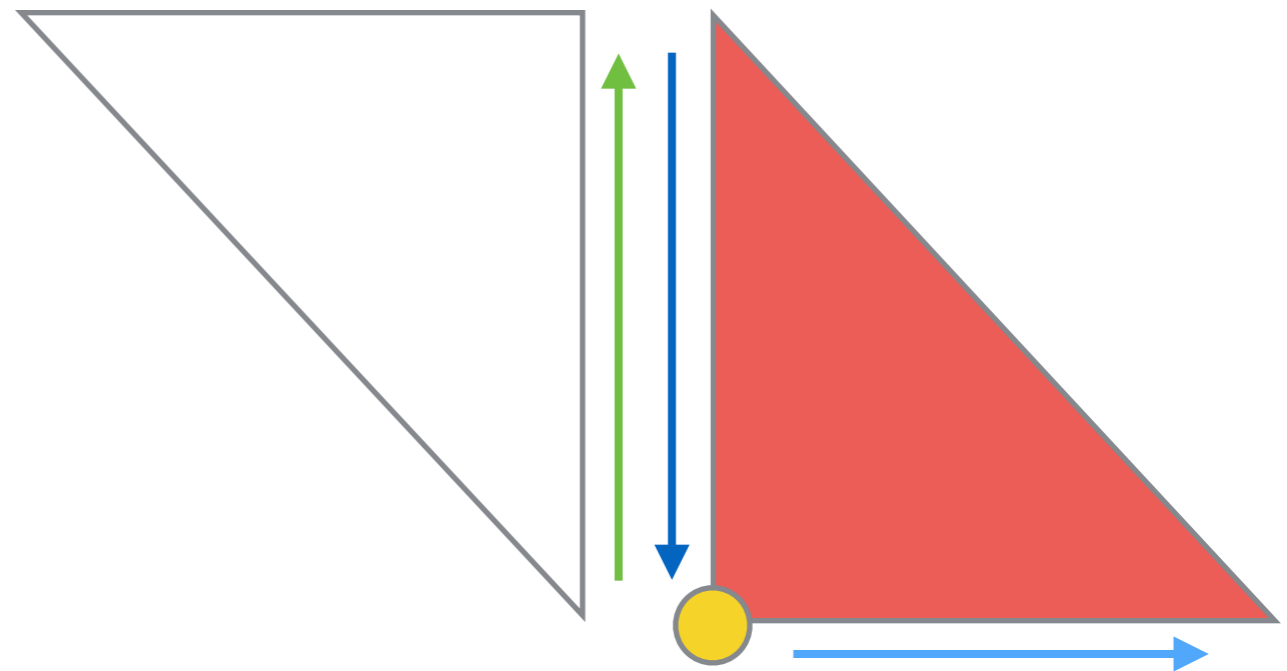
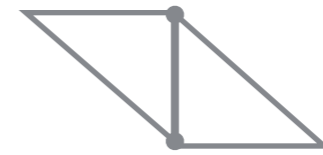
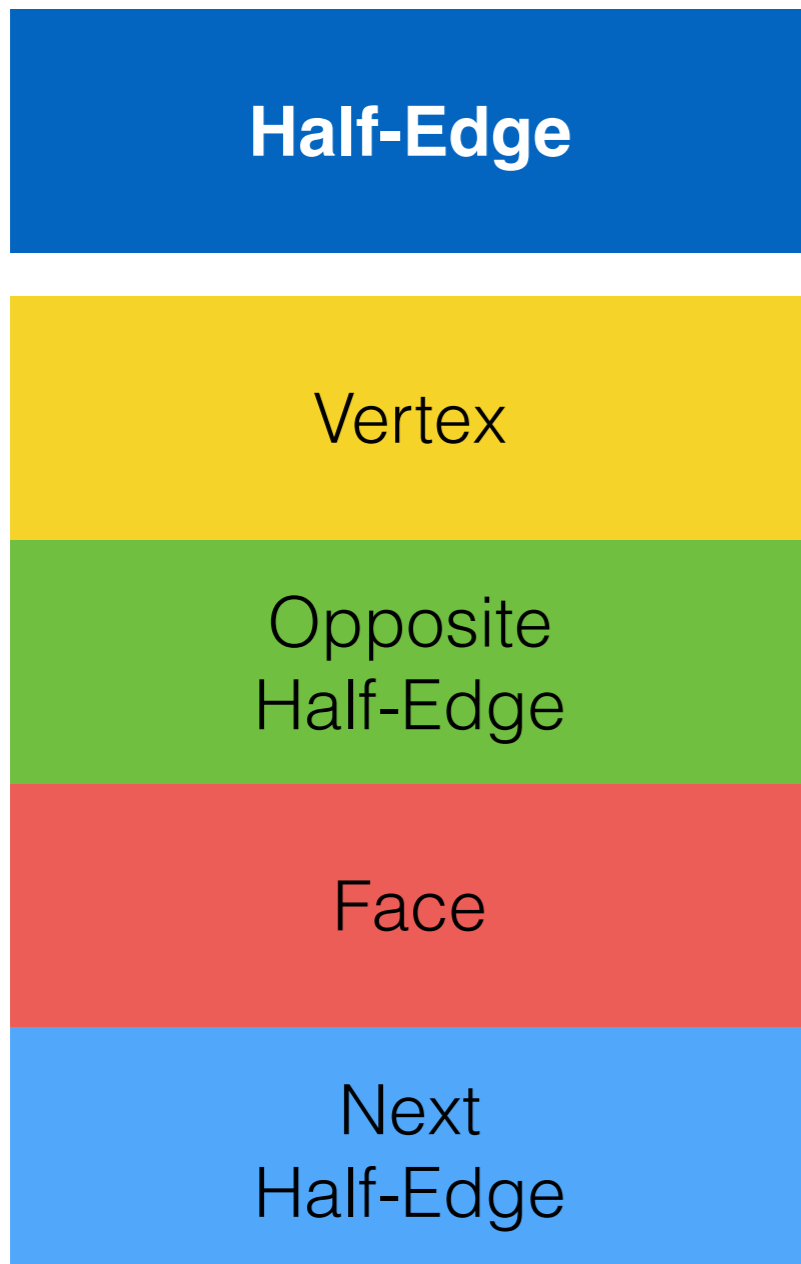
...

Face

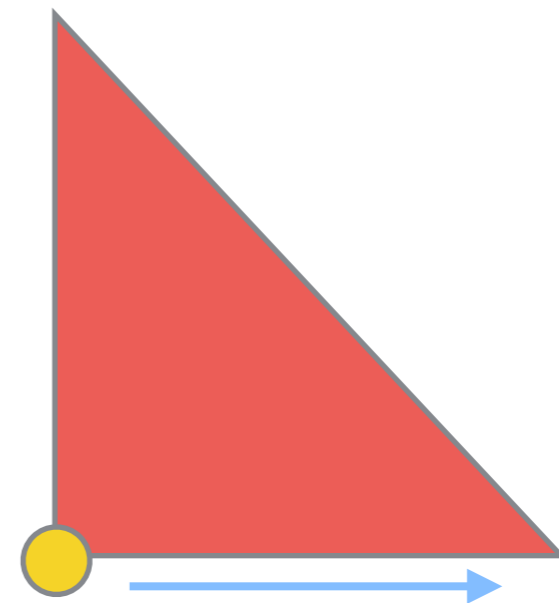
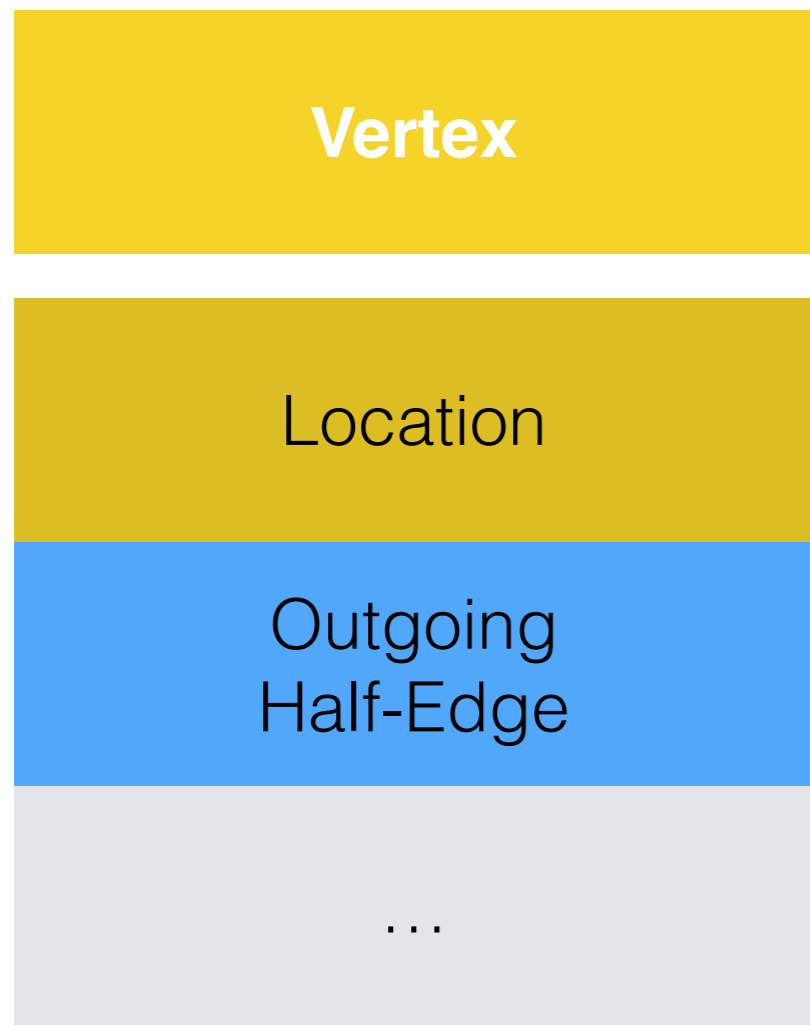
...

Next
Half-Edge

Half-Edge Data Structure

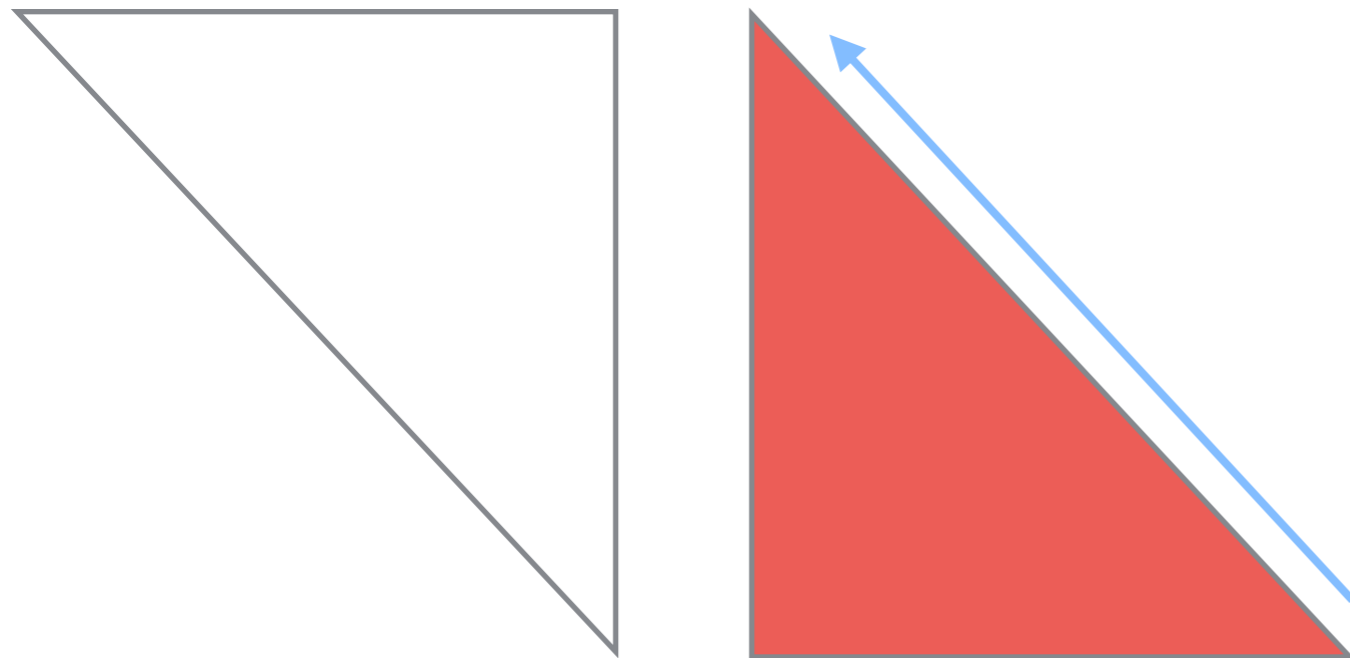


Half-Edge Data Structure

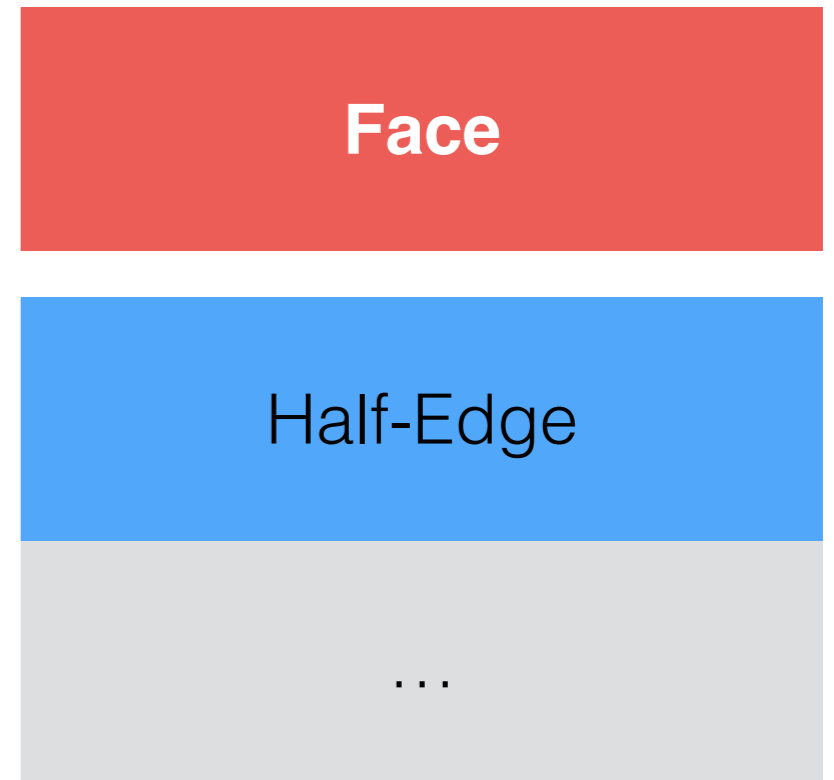


One of the two outgoing edges
will be used

Half-Edge Data Structure

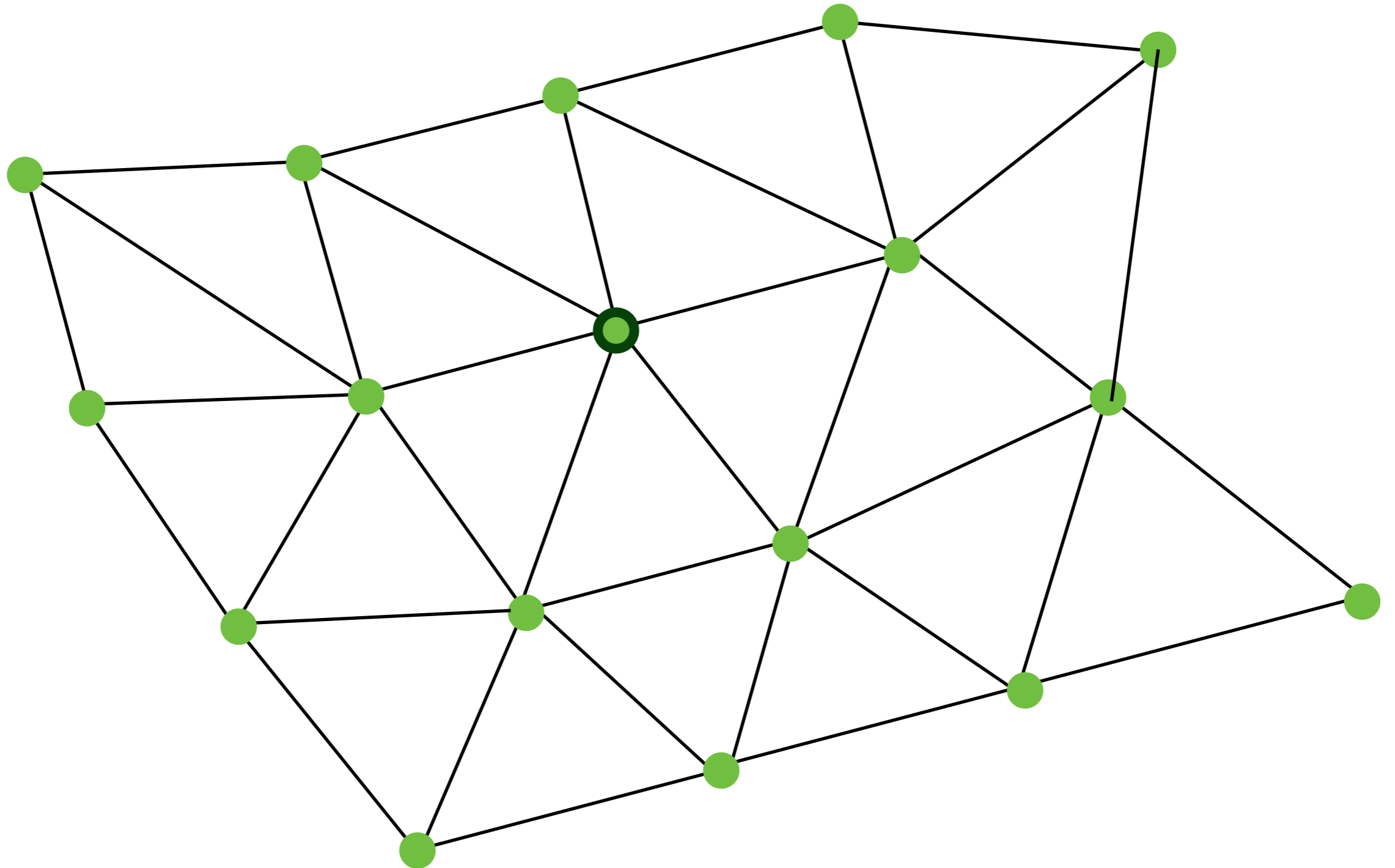


One of the three edges
will be used



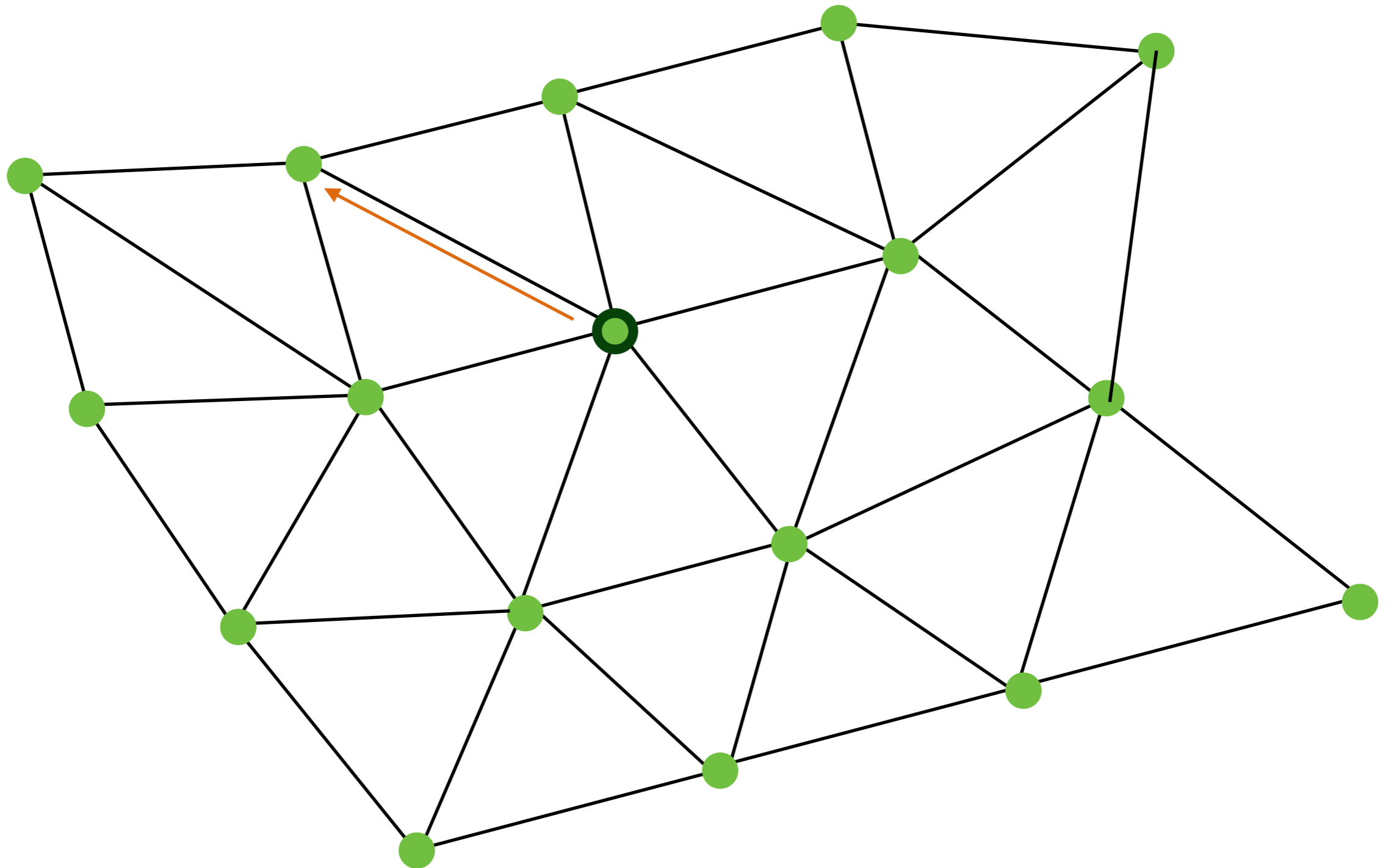
Traversal

- How to get one-ring neighbors?



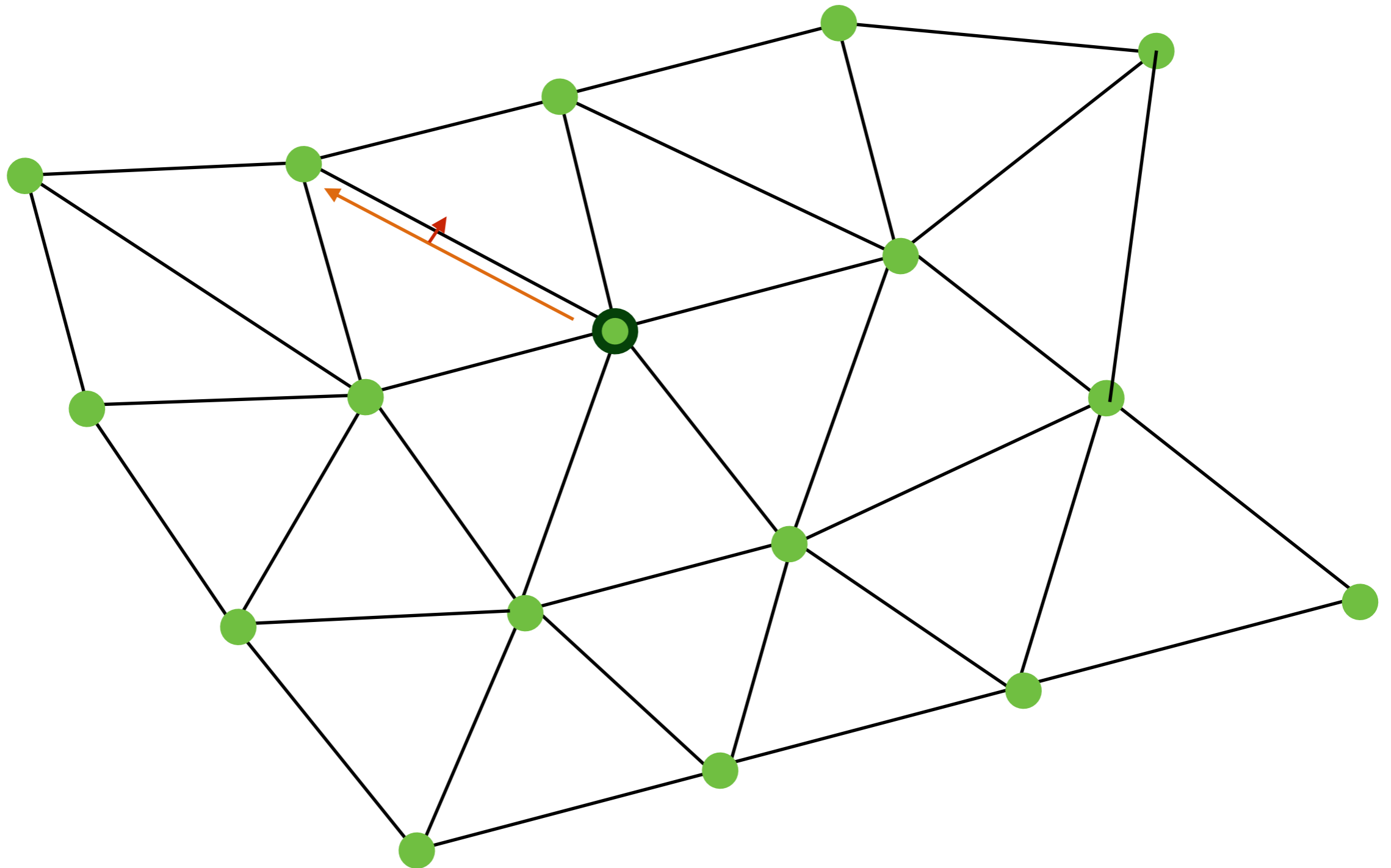
Traversal

- How to get one-ring neighbors?



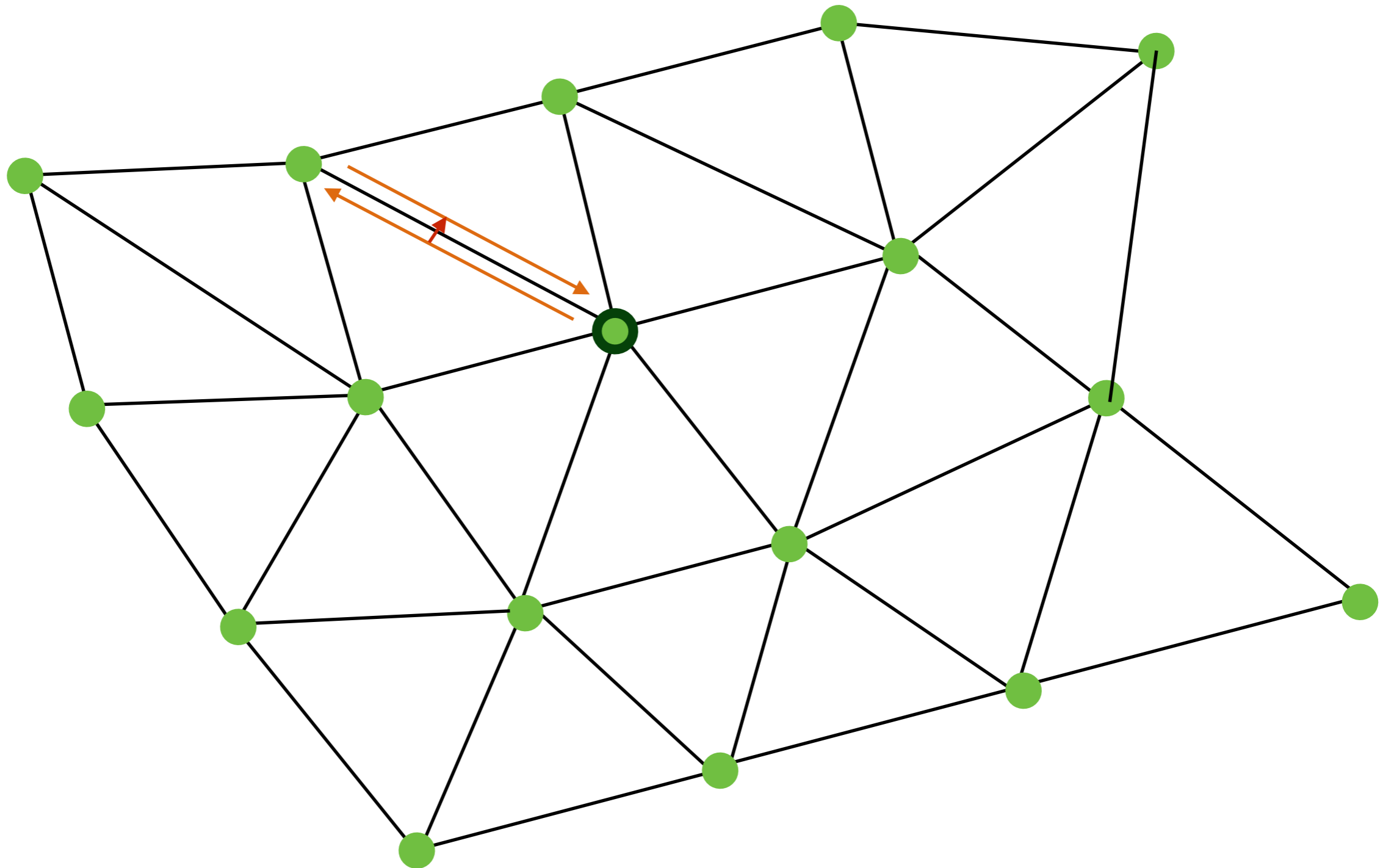
Traversal

- How to get one-ring neighbors?



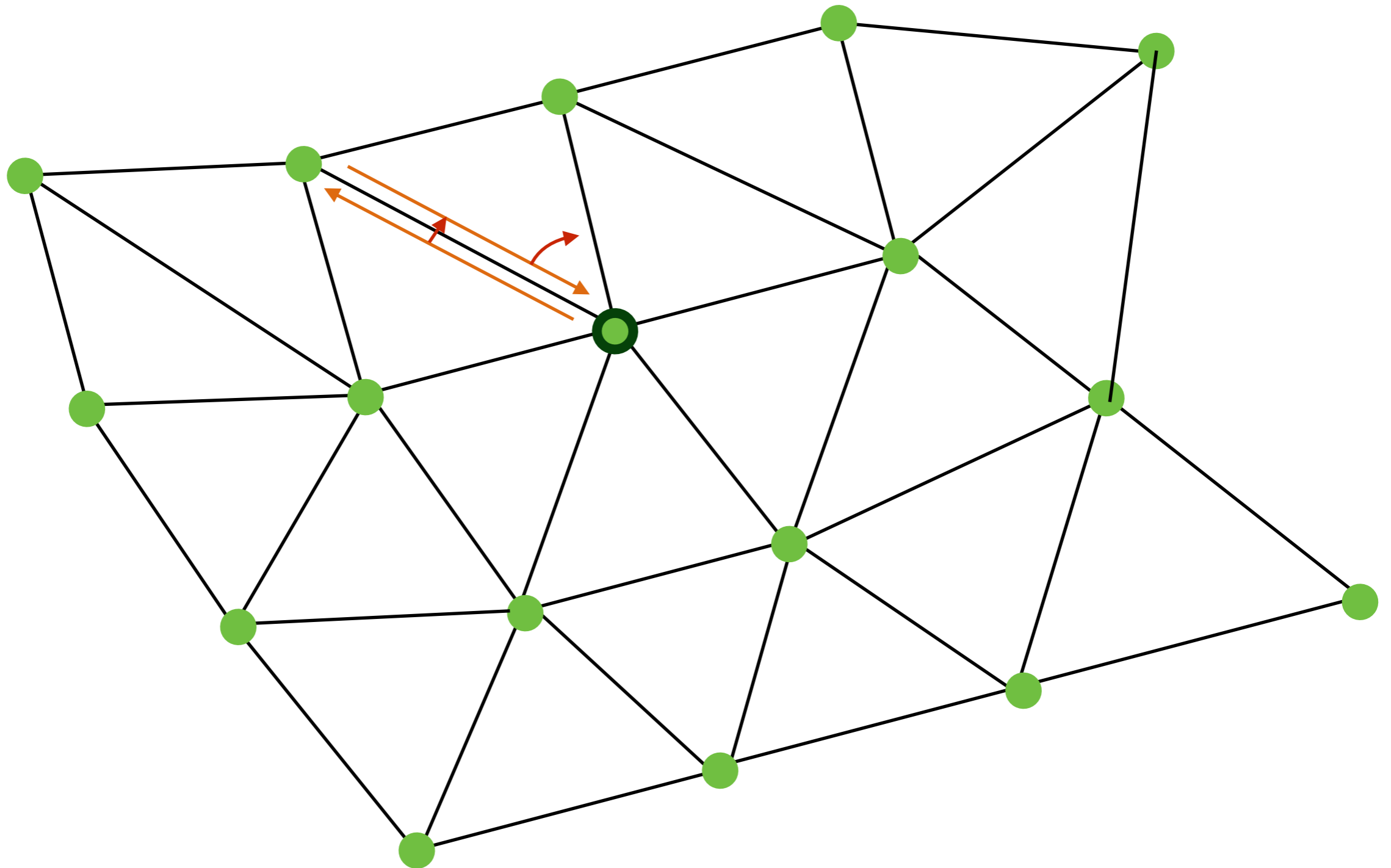
Traversal

- How to get one-ring neighbors?



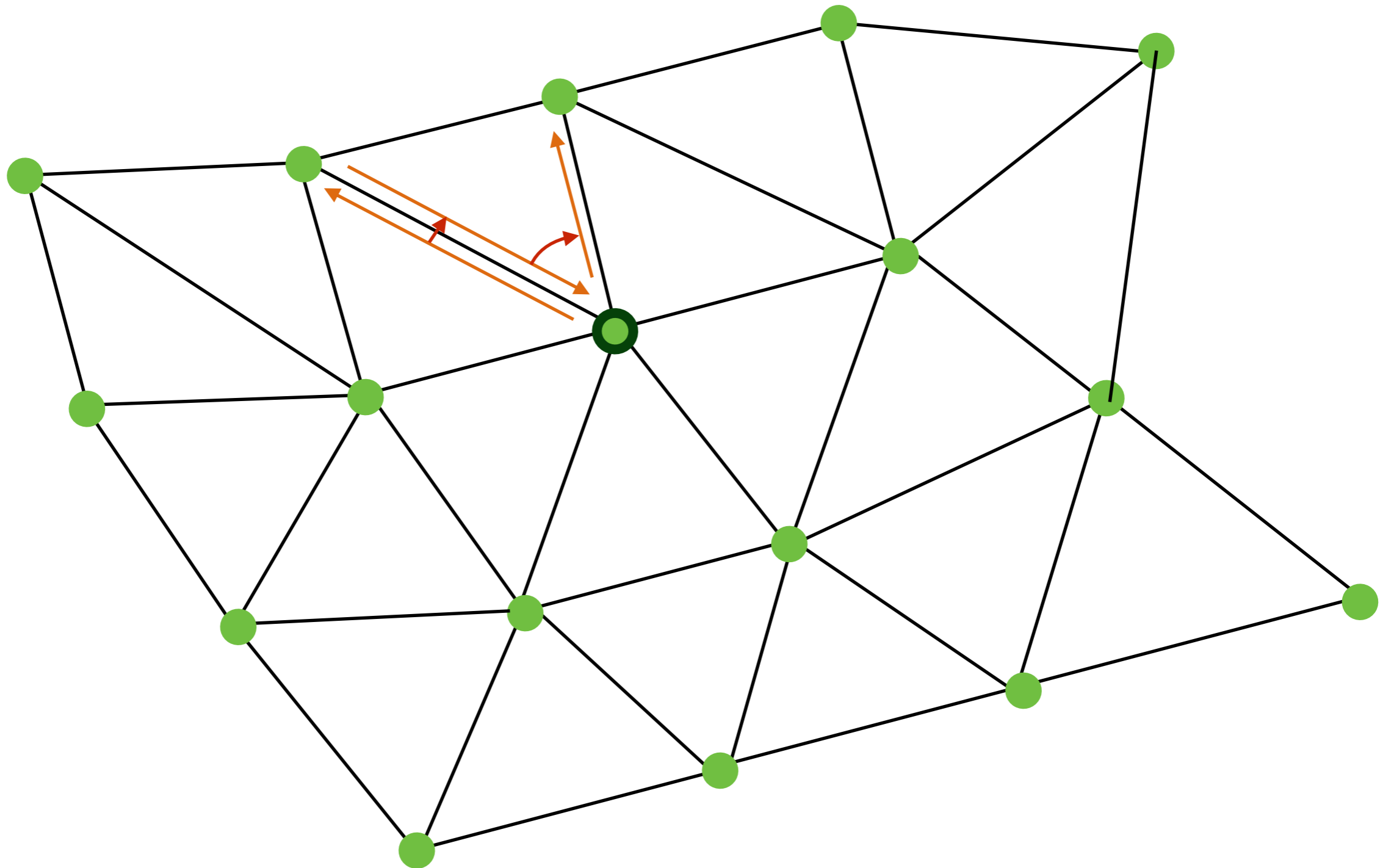
Traversal

- How to get one-ring neighbors?



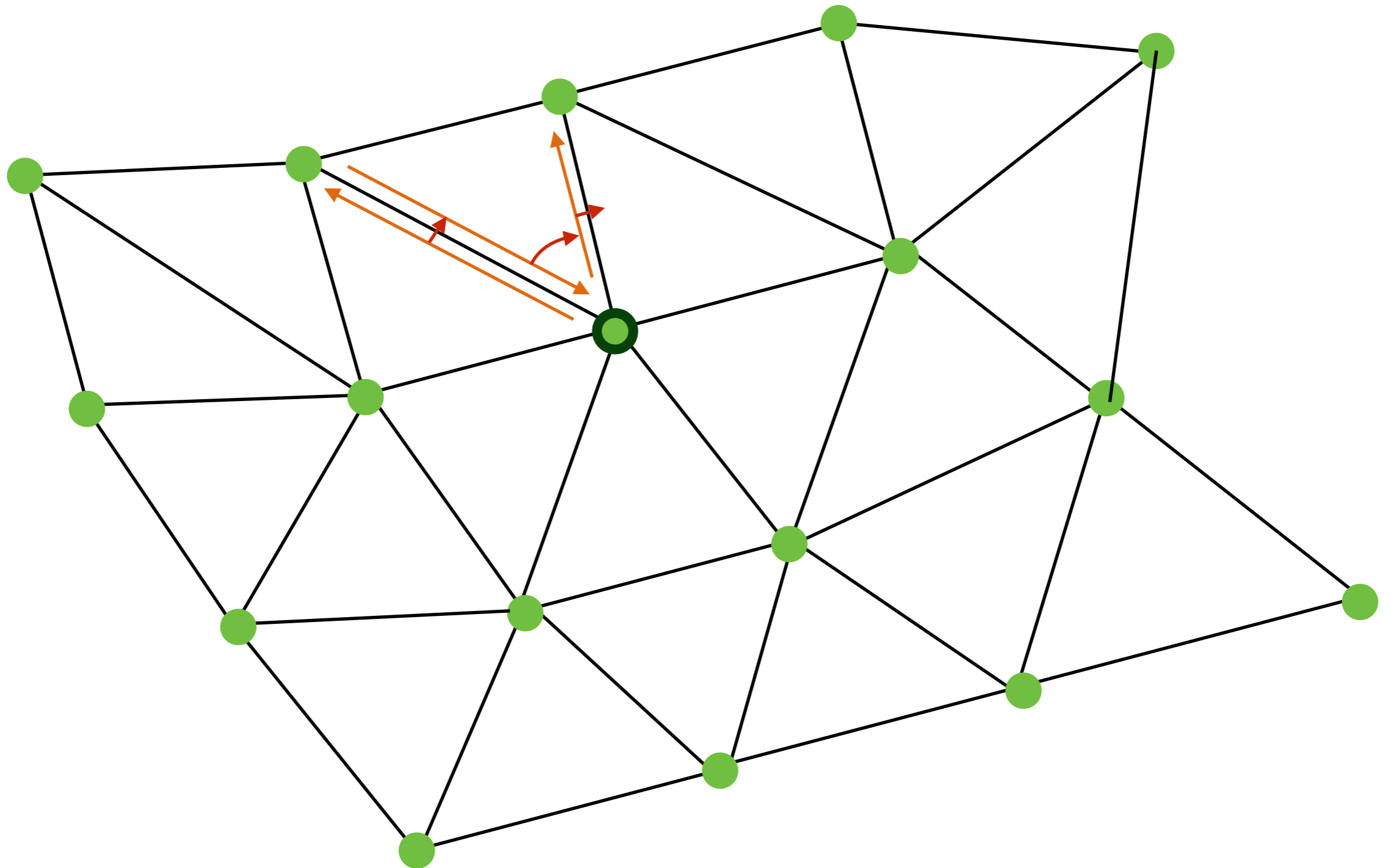
Traversal

- How to get one-ring neighbors?



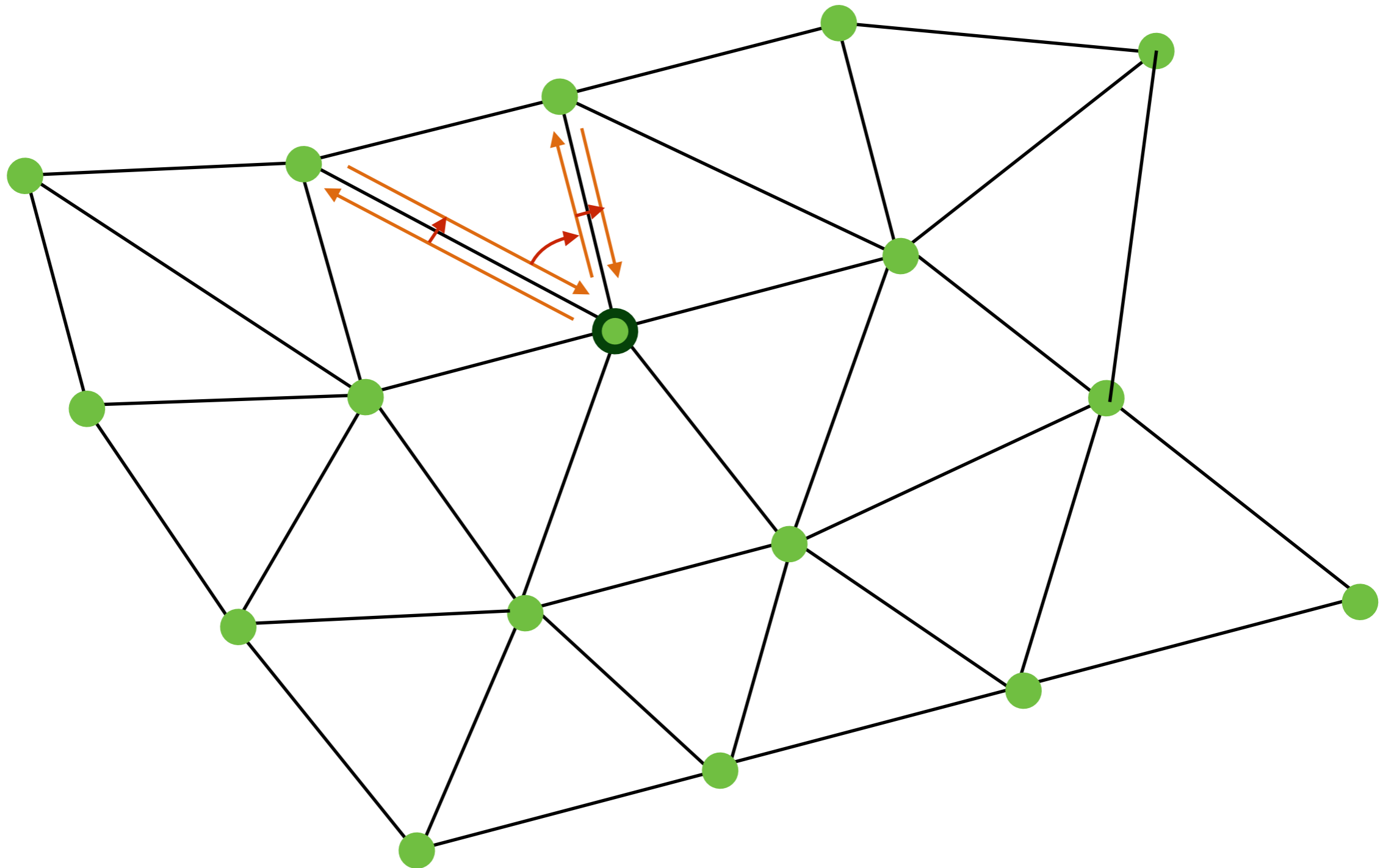
Traversal

- How to get one-ring neighbors?



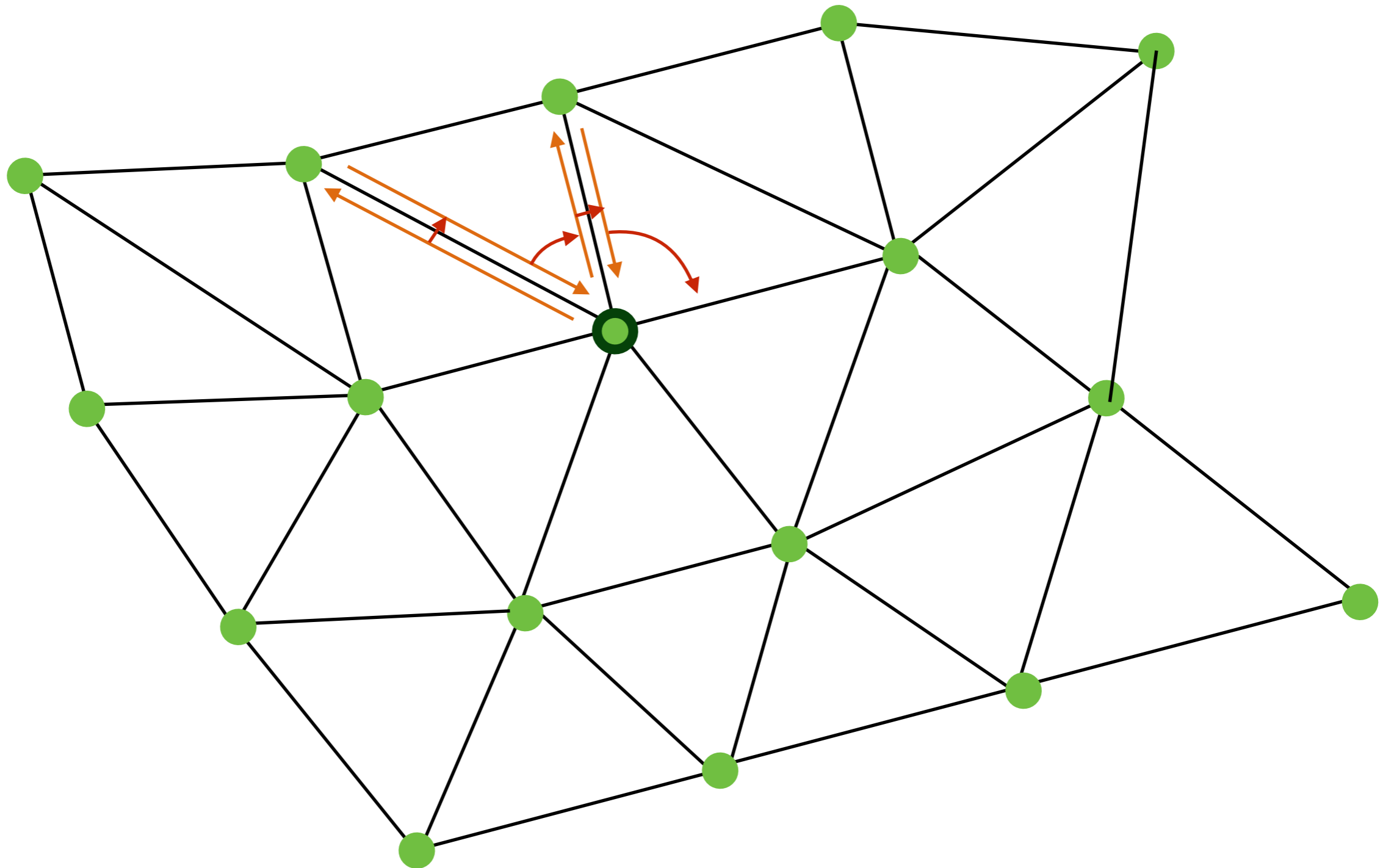
Traversal

- How to get one-ring neighbors?



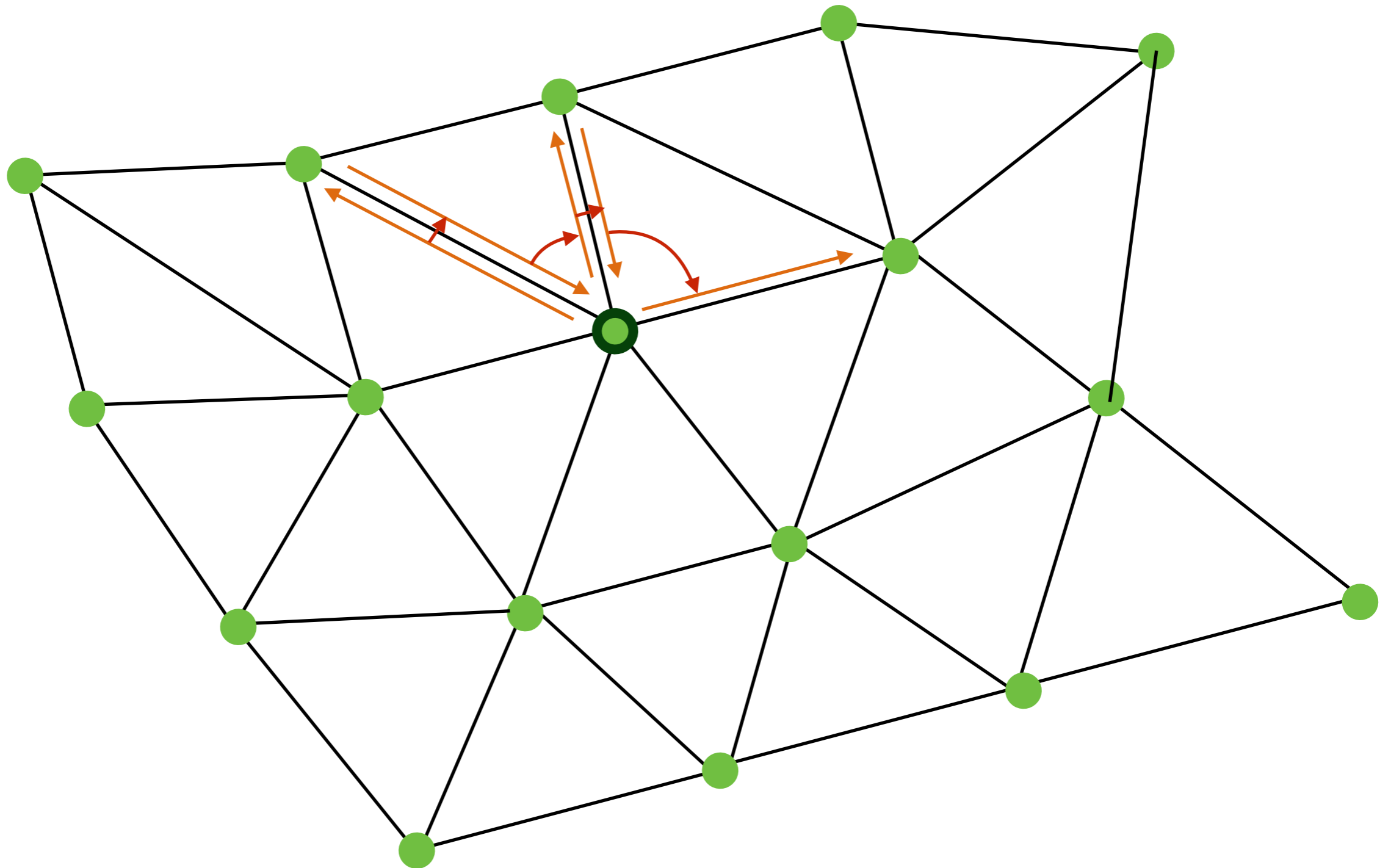
Traversal

- How to get one-ring neighbors?



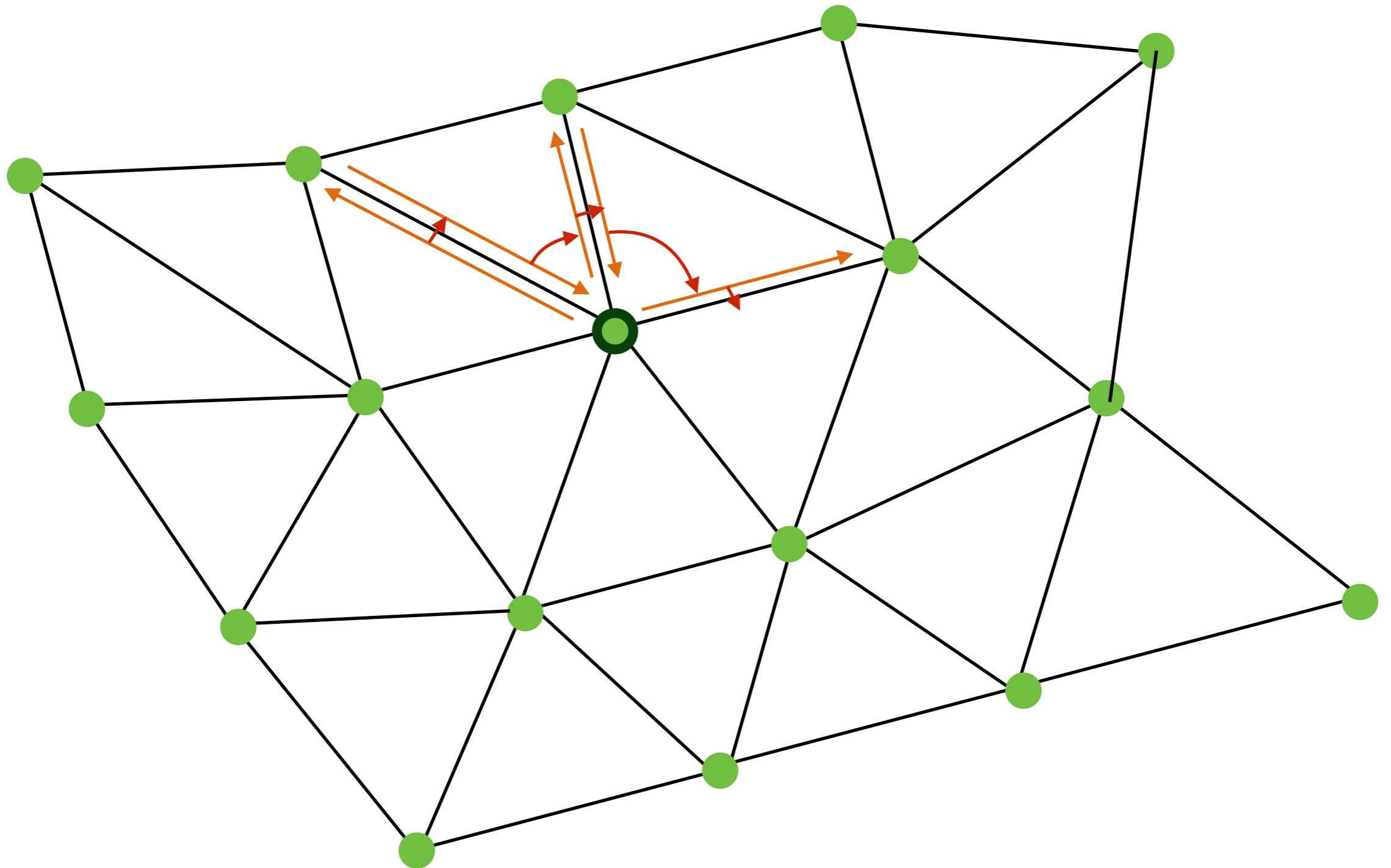
Traversal

- How to get one-ring neighbors?



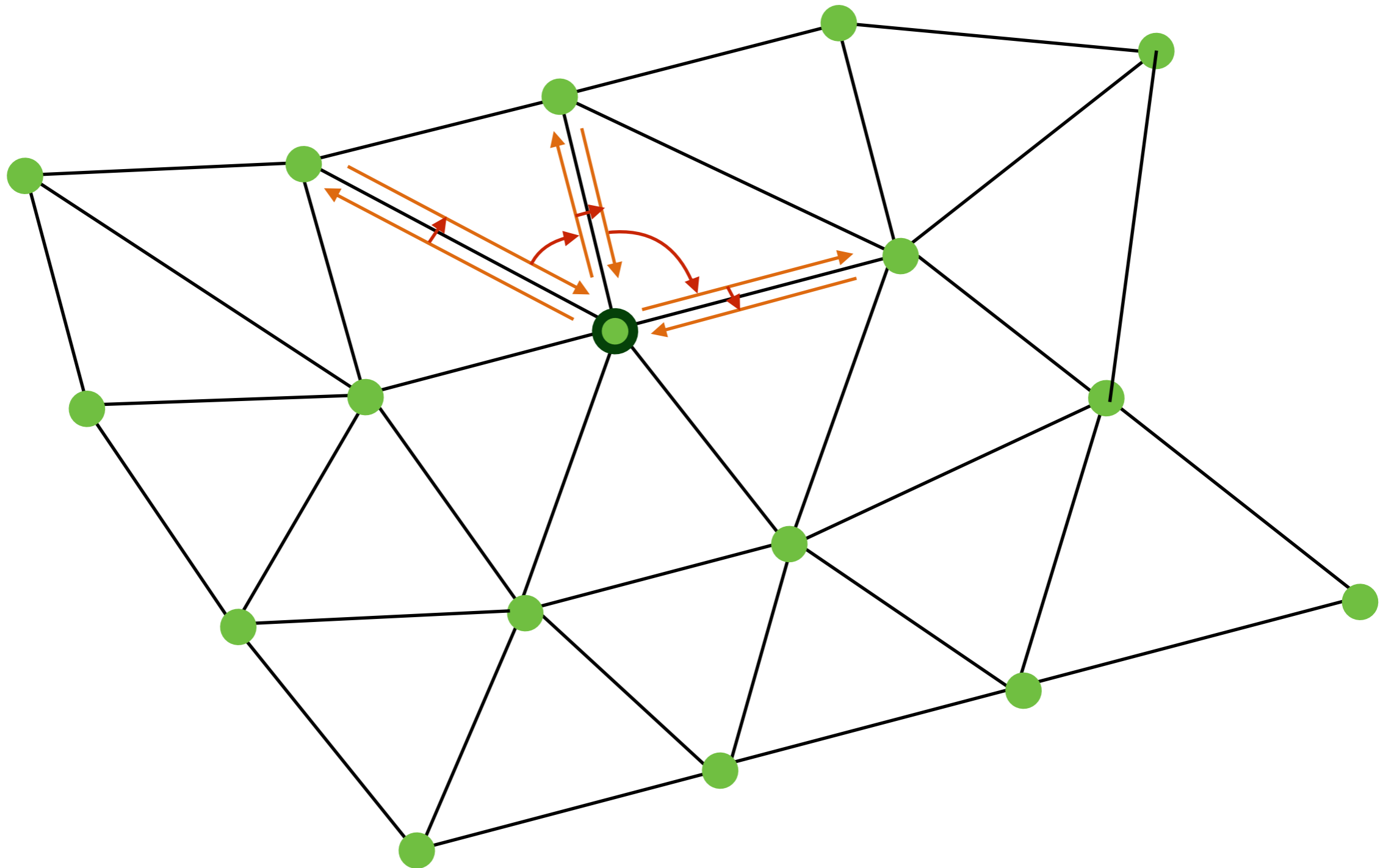
Traversal

- How to get one-ring neighbors?



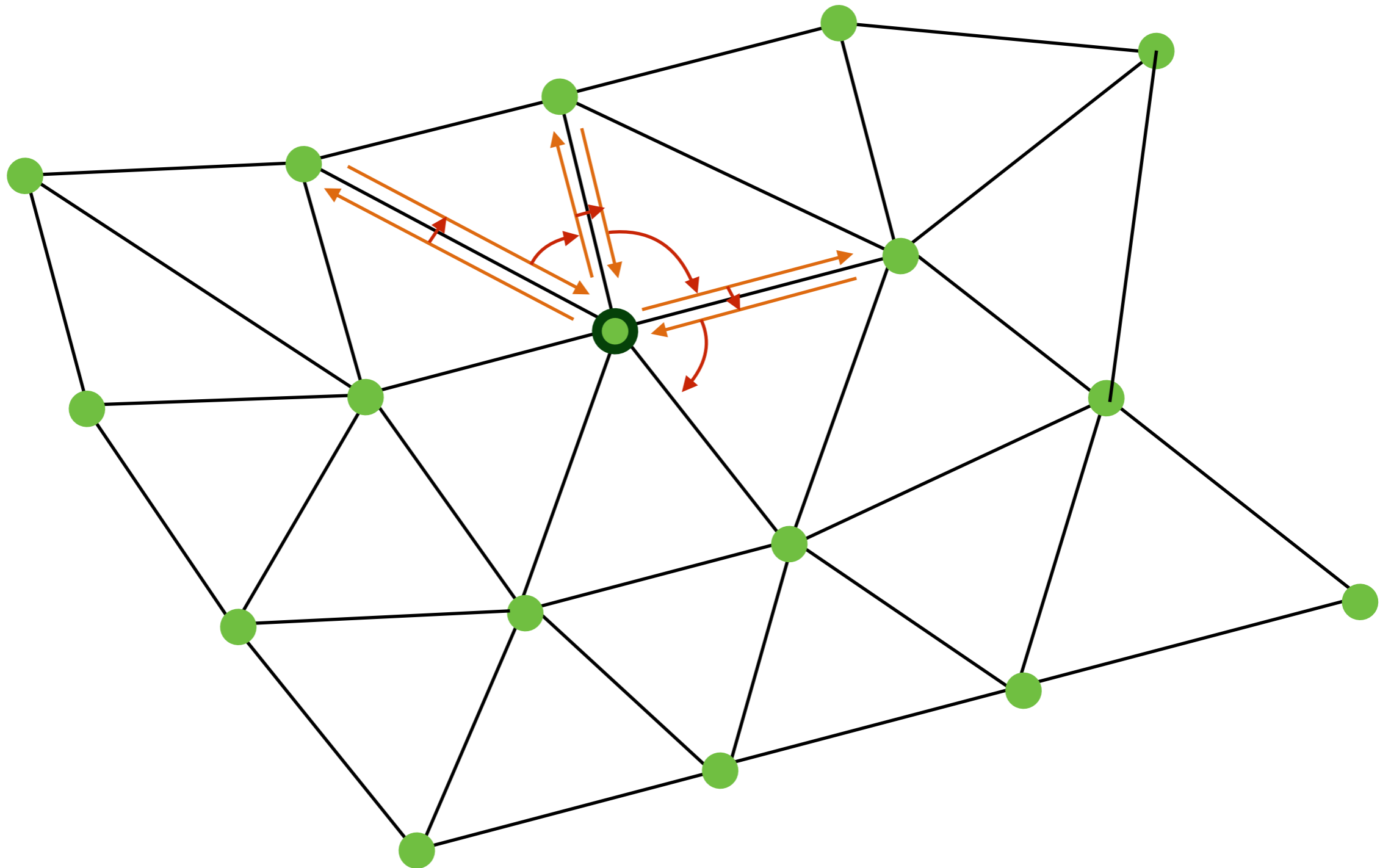
Traversal

- How to get one-ring neighbors?



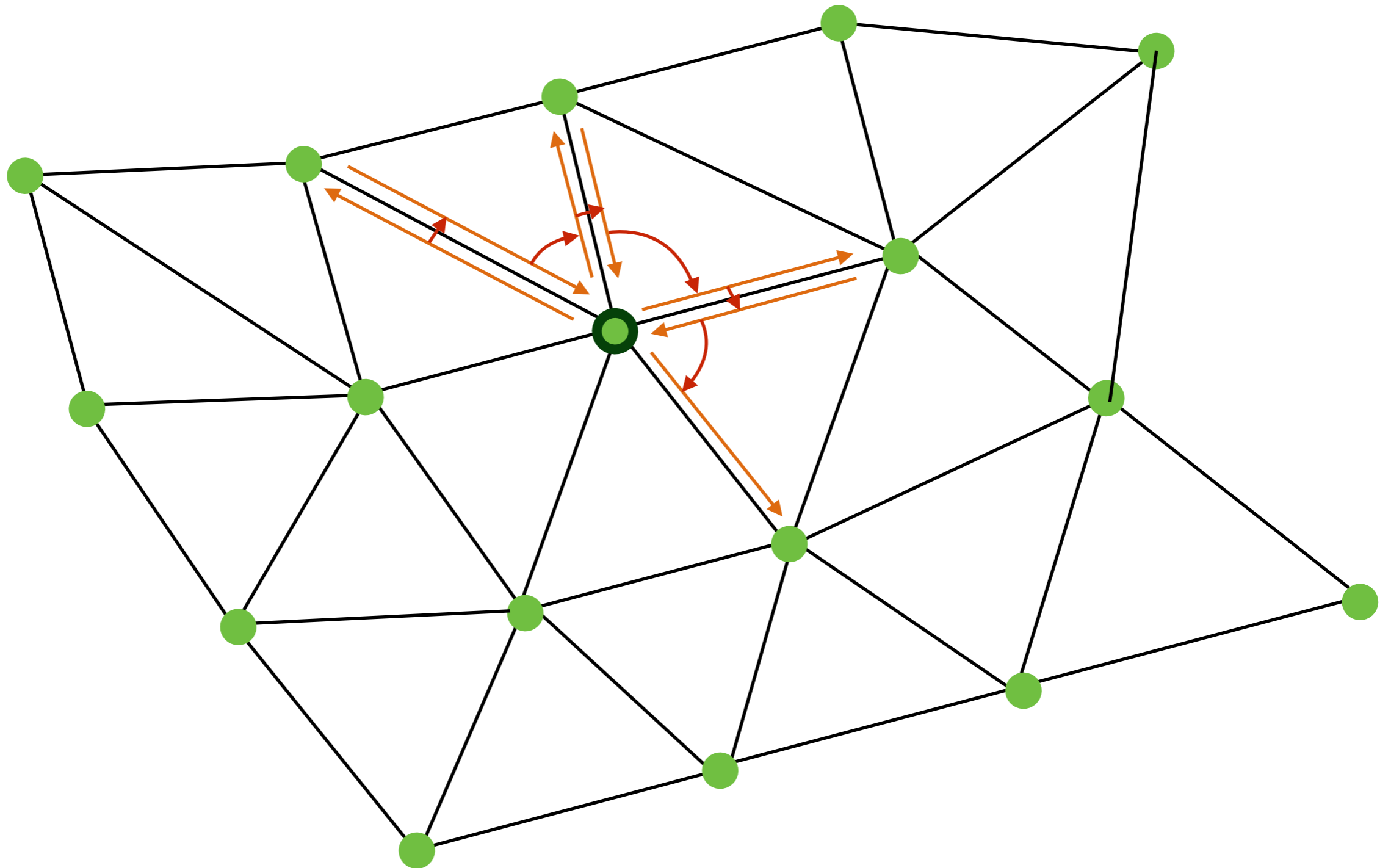
Traversal

- How to get one-ring neighbors?



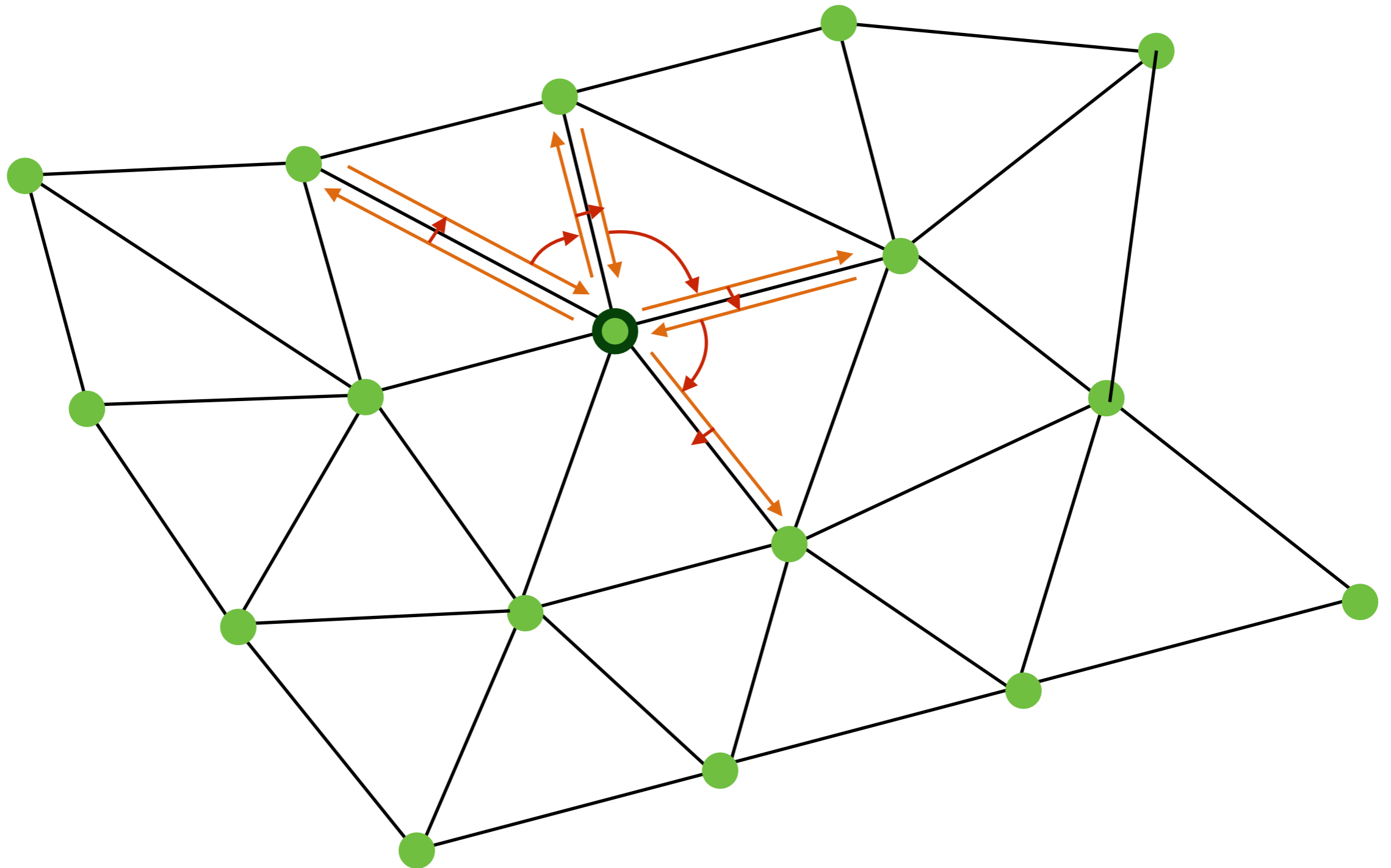
Traversal

- How to get one-ring neighbors?



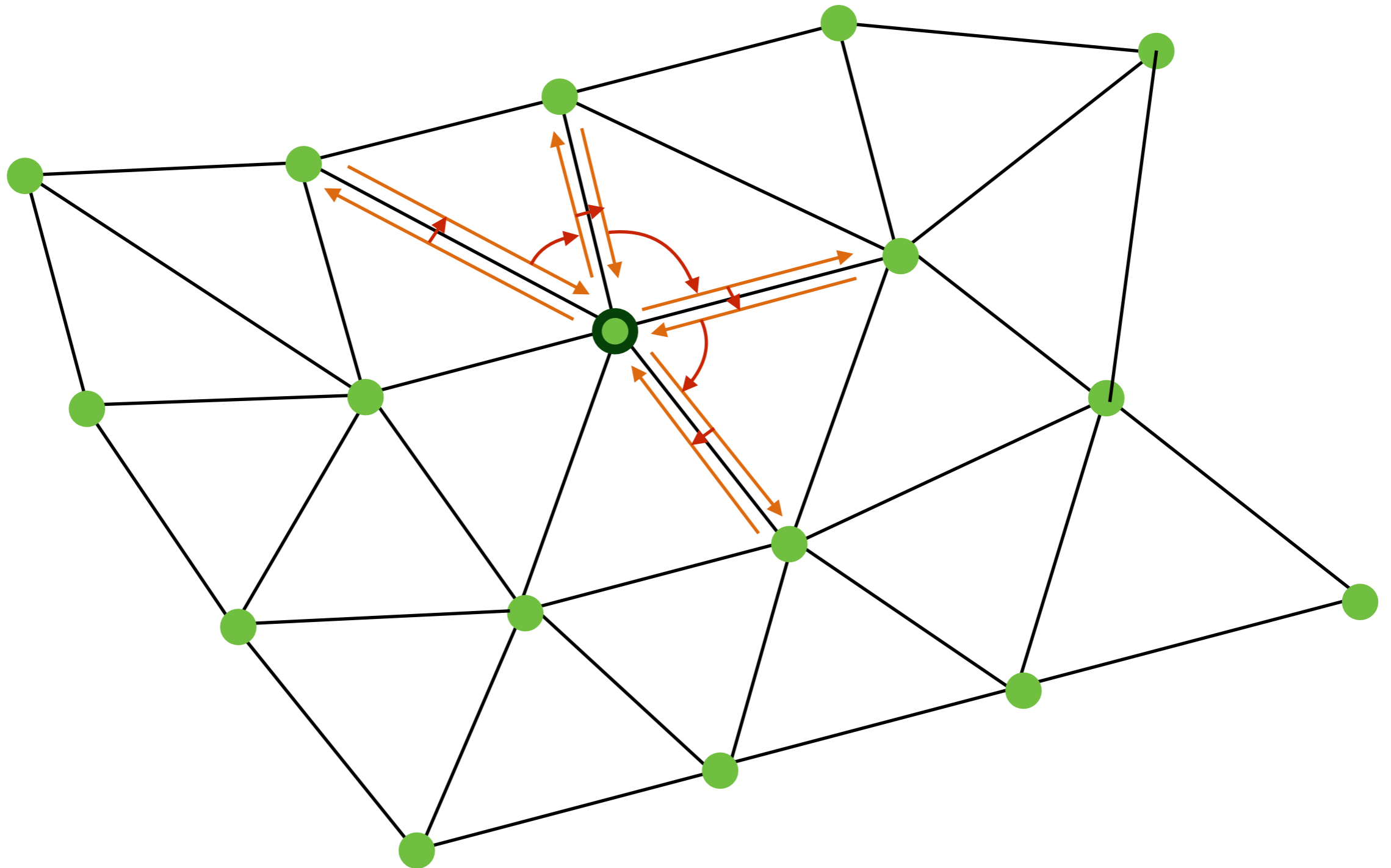
Traversal

- How to get one-ring neighbors?



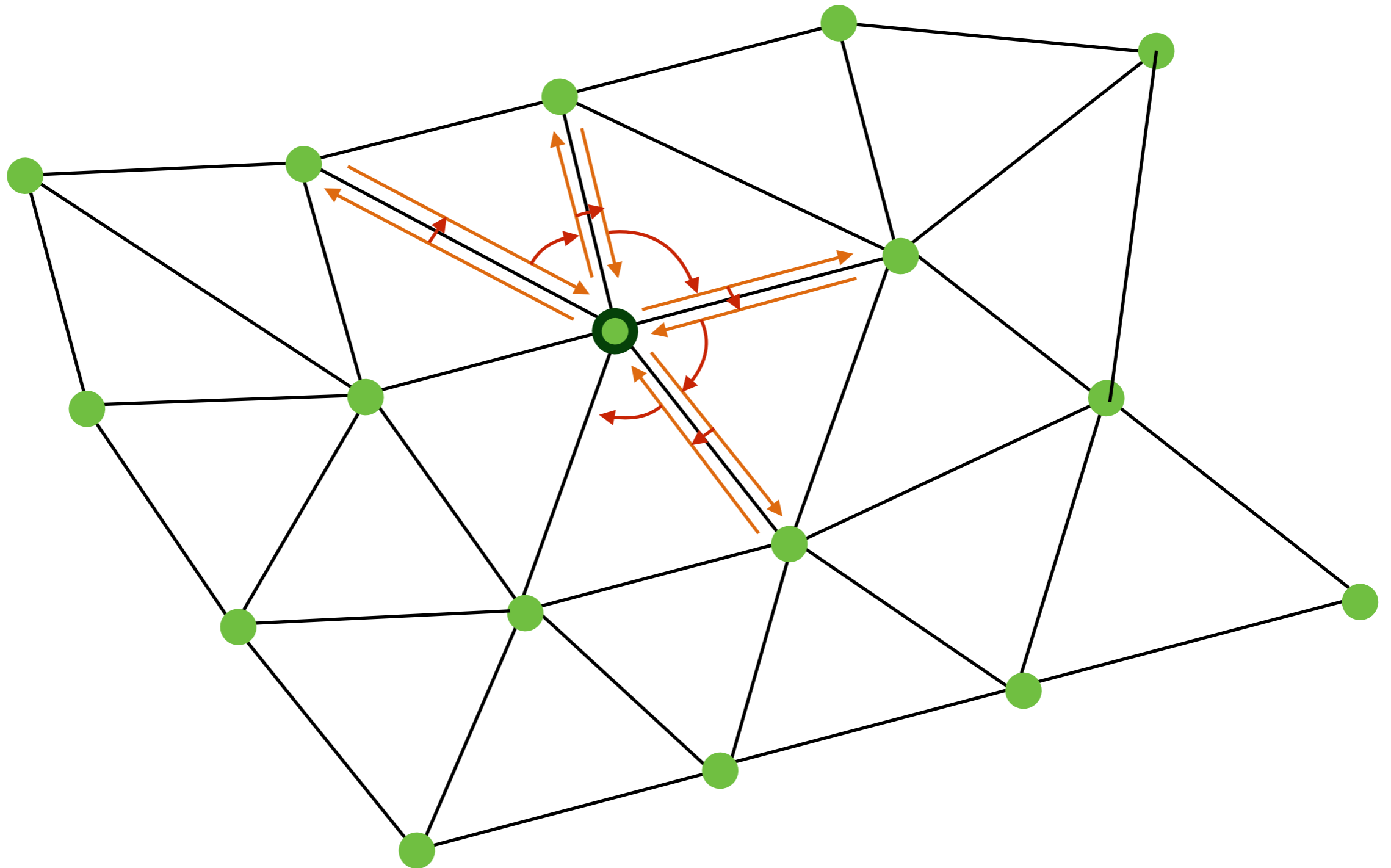
Traversal

- How to get one-ring neighbors?



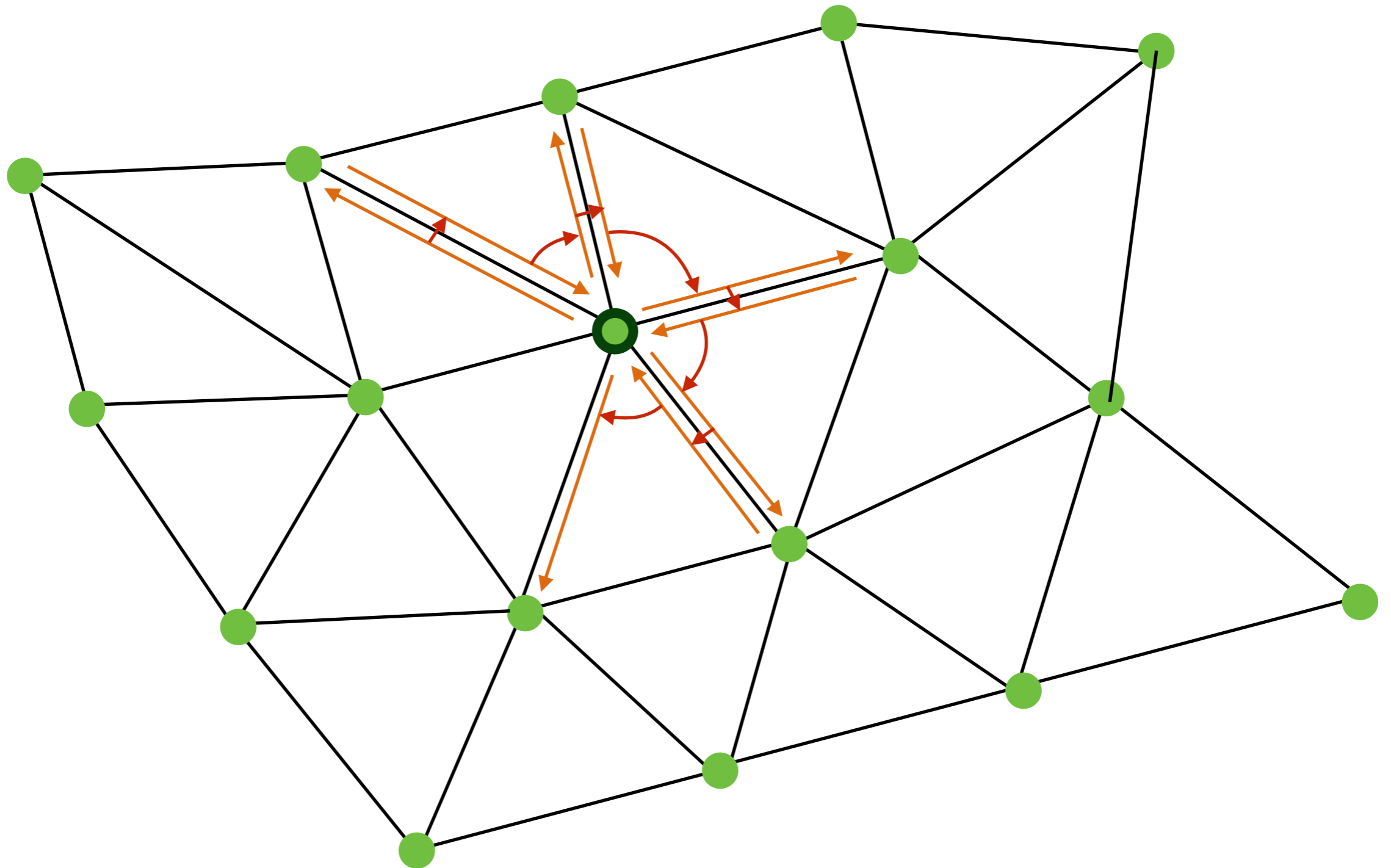
Traversal

- How to get one-ring neighbors?



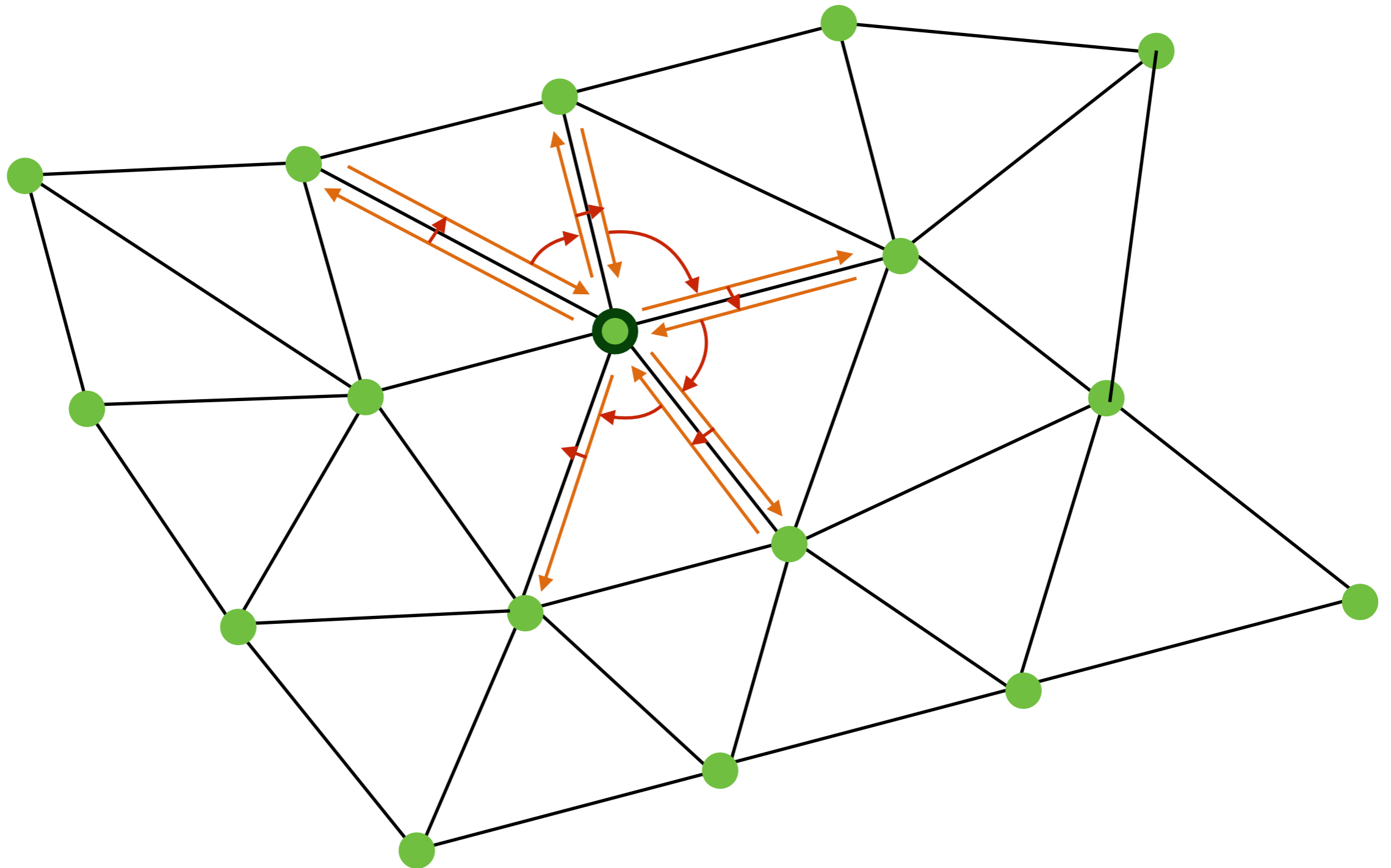
Traversal

- How to get one-ring neighbors?



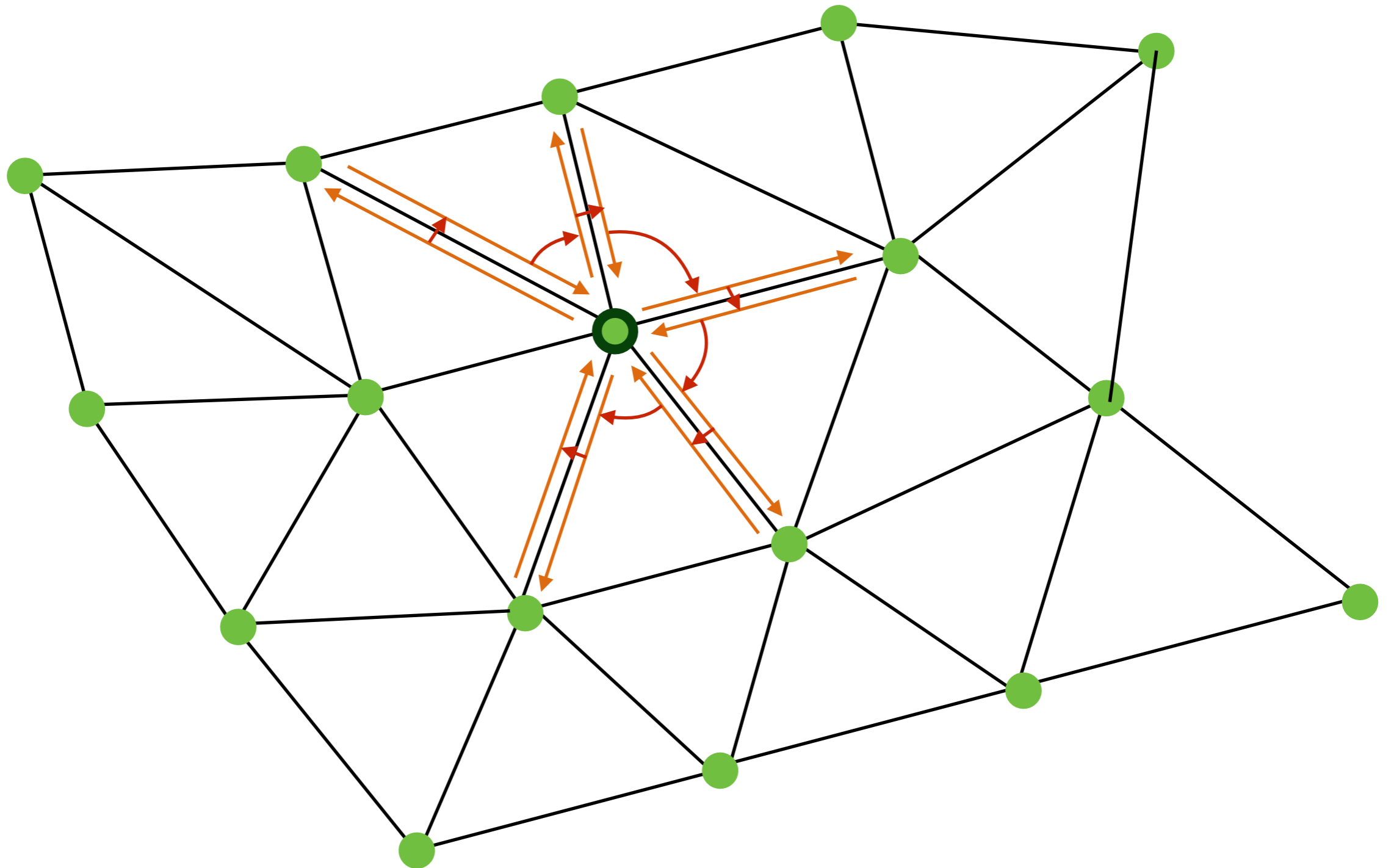
Traversal

- How to get one-ring neighbors?



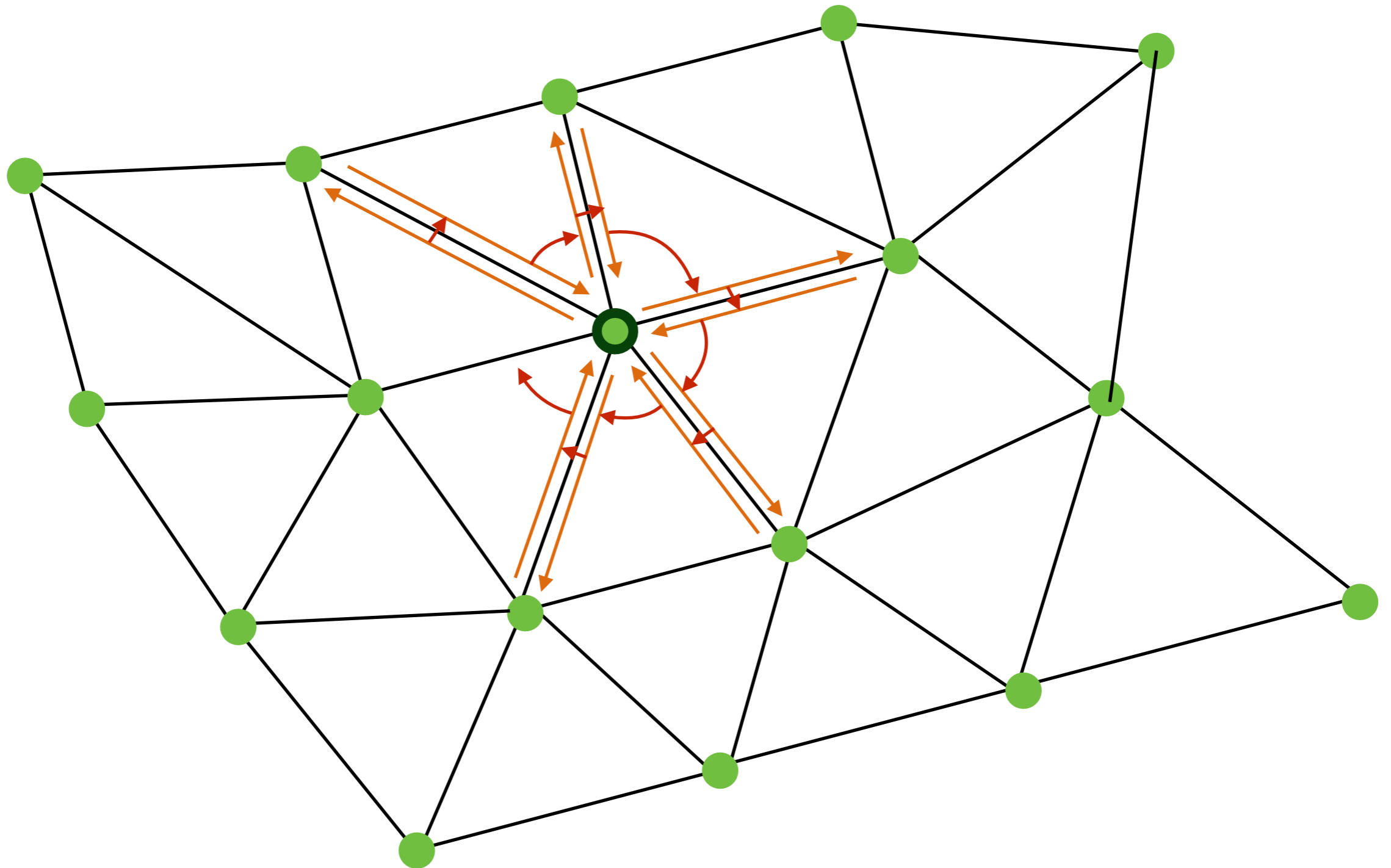
Traversal

- How to get one-ring neighbors?



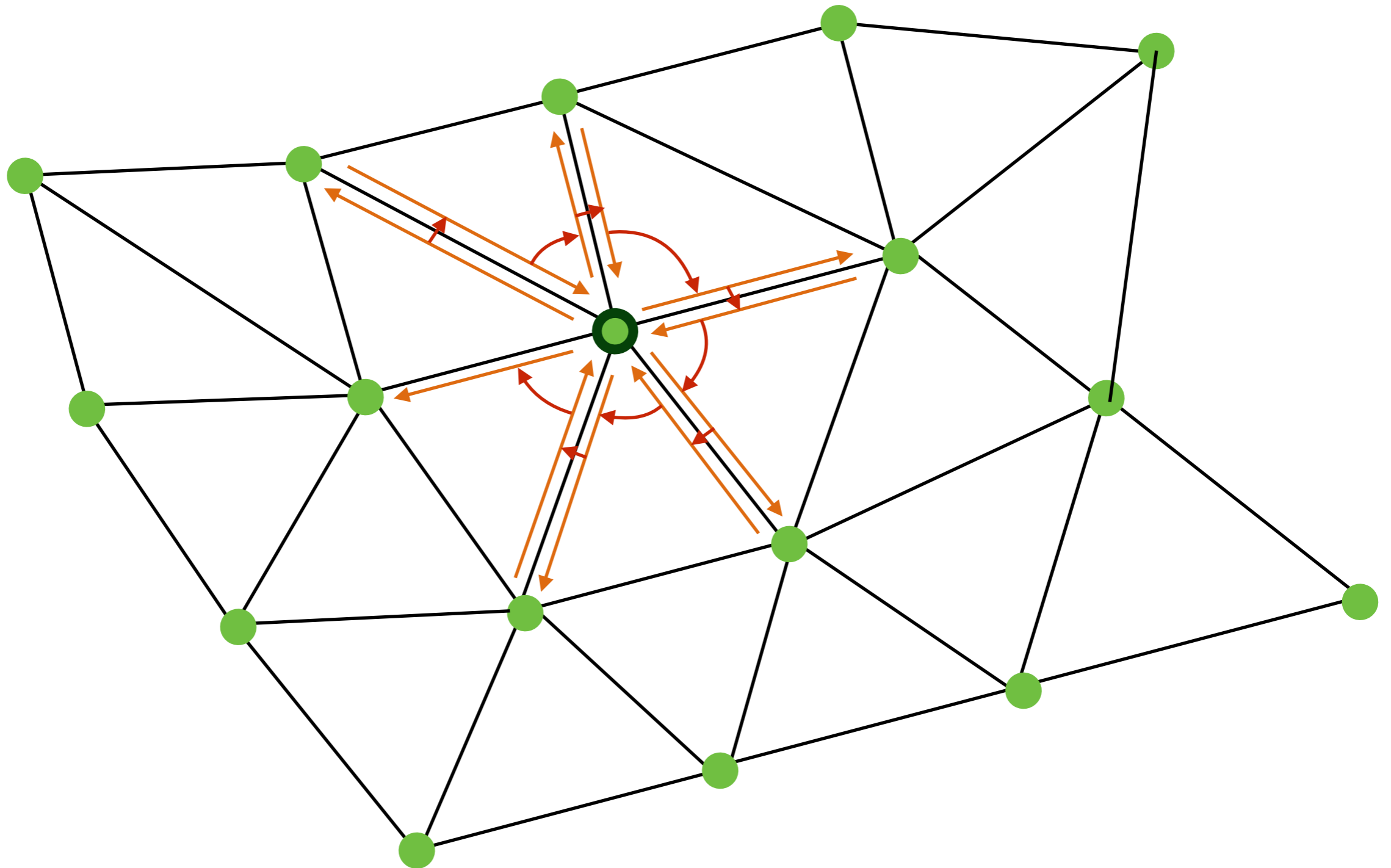
Traversal

- How to get one-ring neighbors?



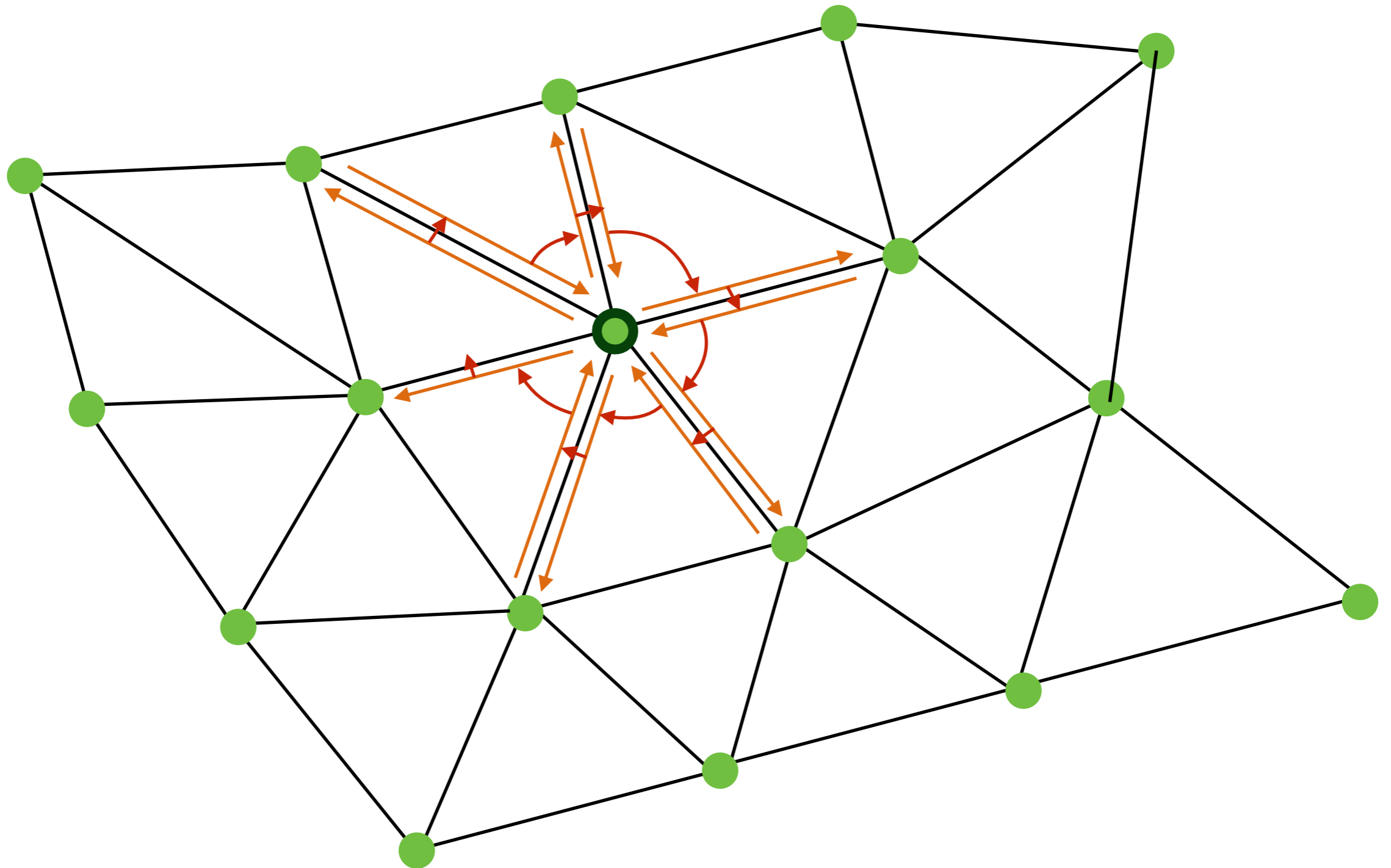
Traversal

- How to get one-ring neighbors?



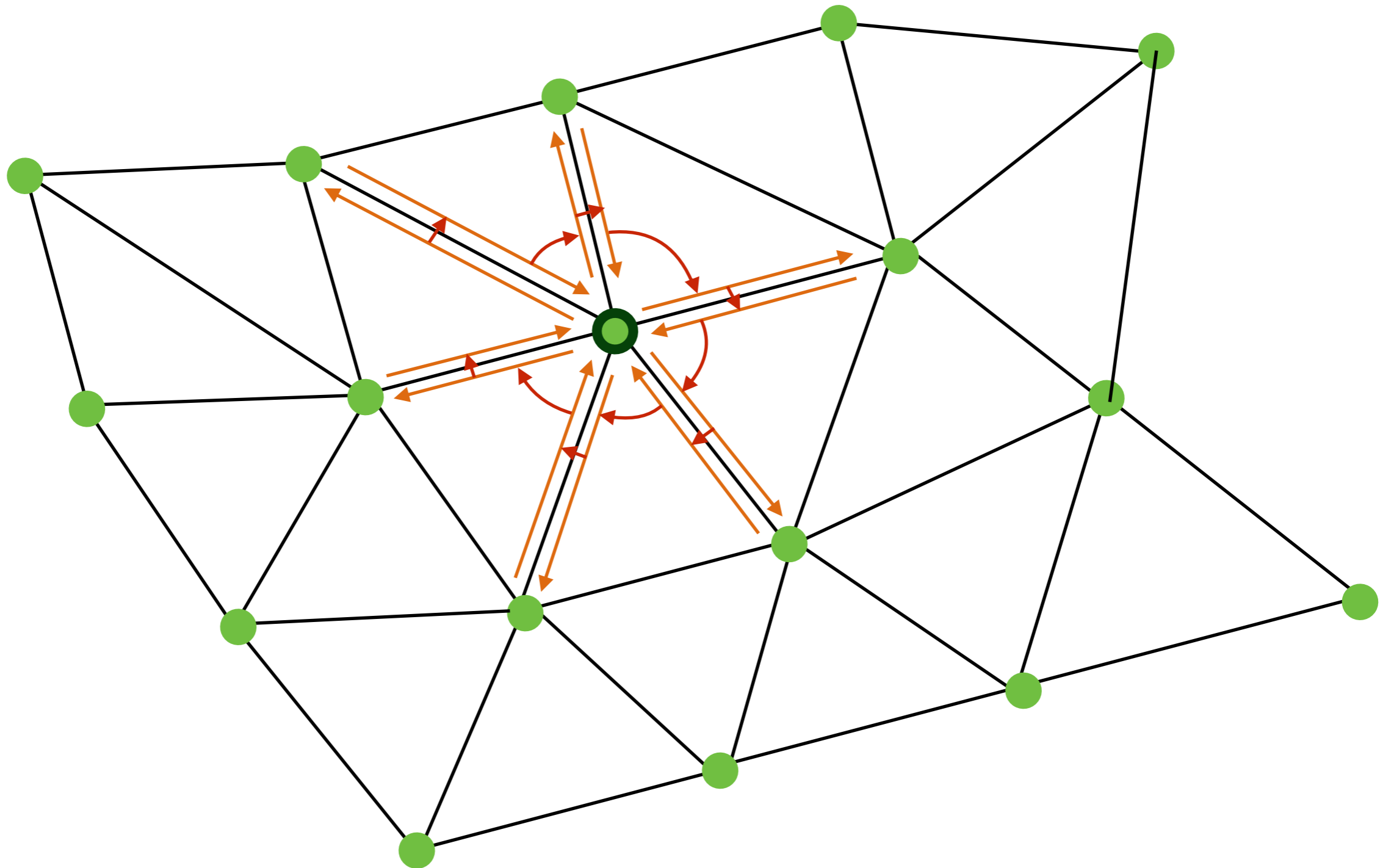
Traversal

- How to get one-ring neighbors?



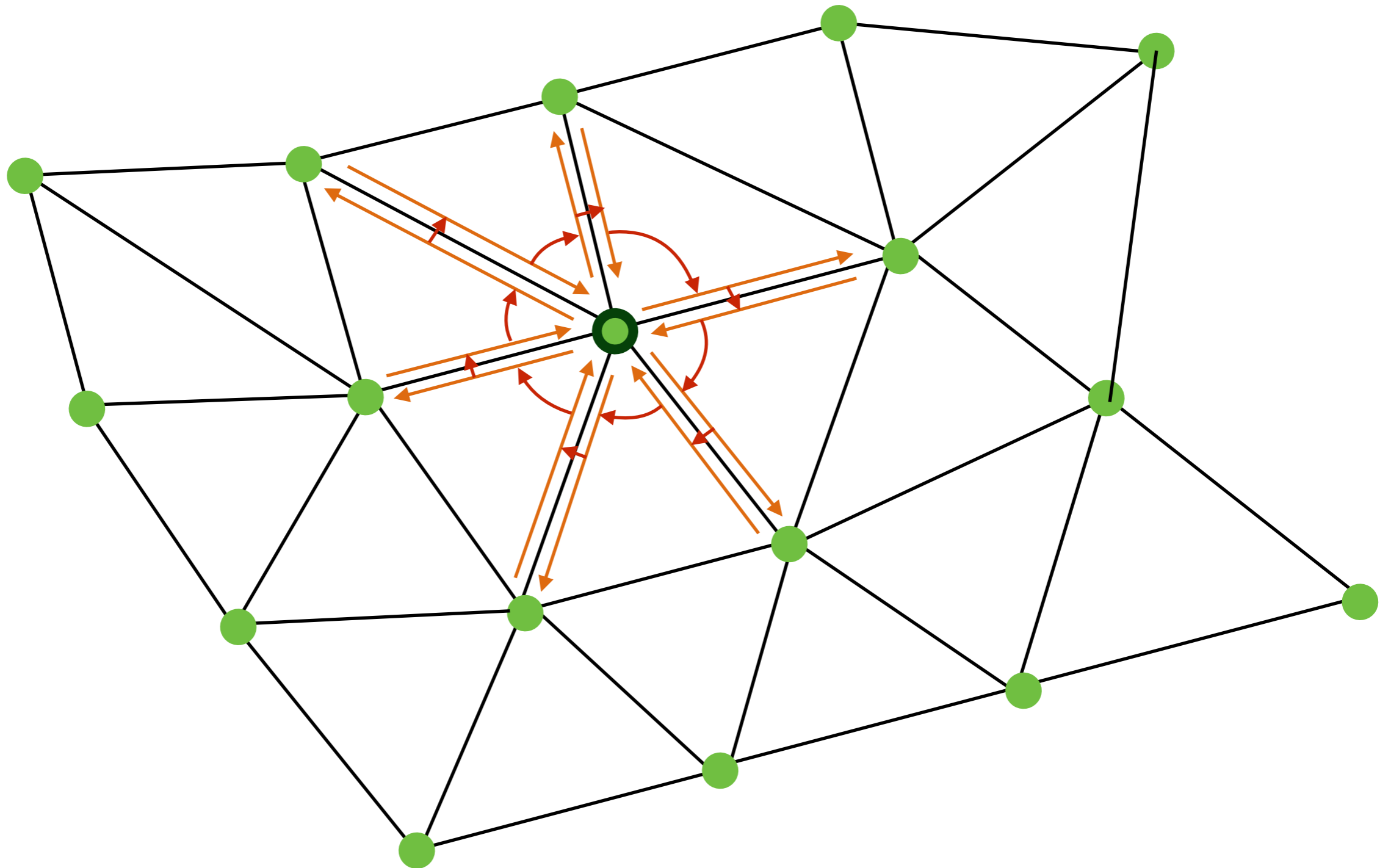
Traversal

- How to get one-ring neighbors?



Traversal

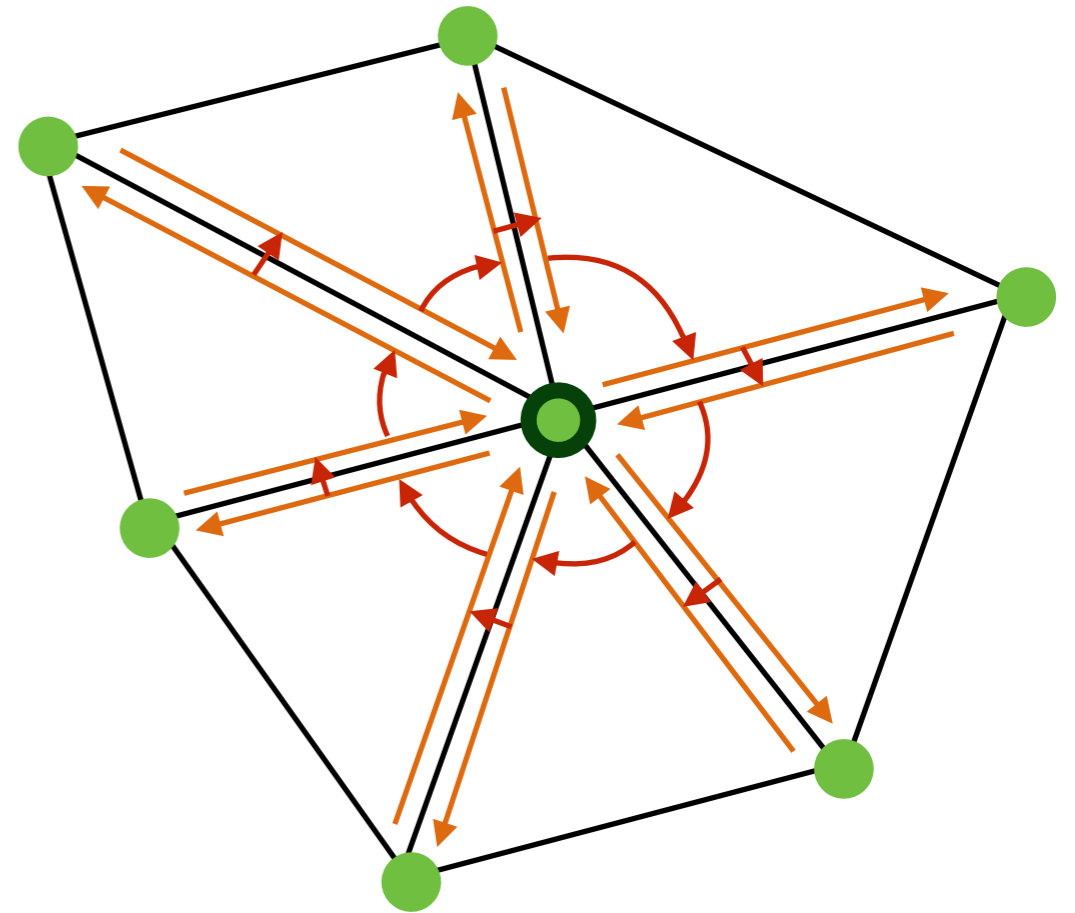
- How to get one-ring neighbors?



Traversal

- How to get one-ring neighbors?

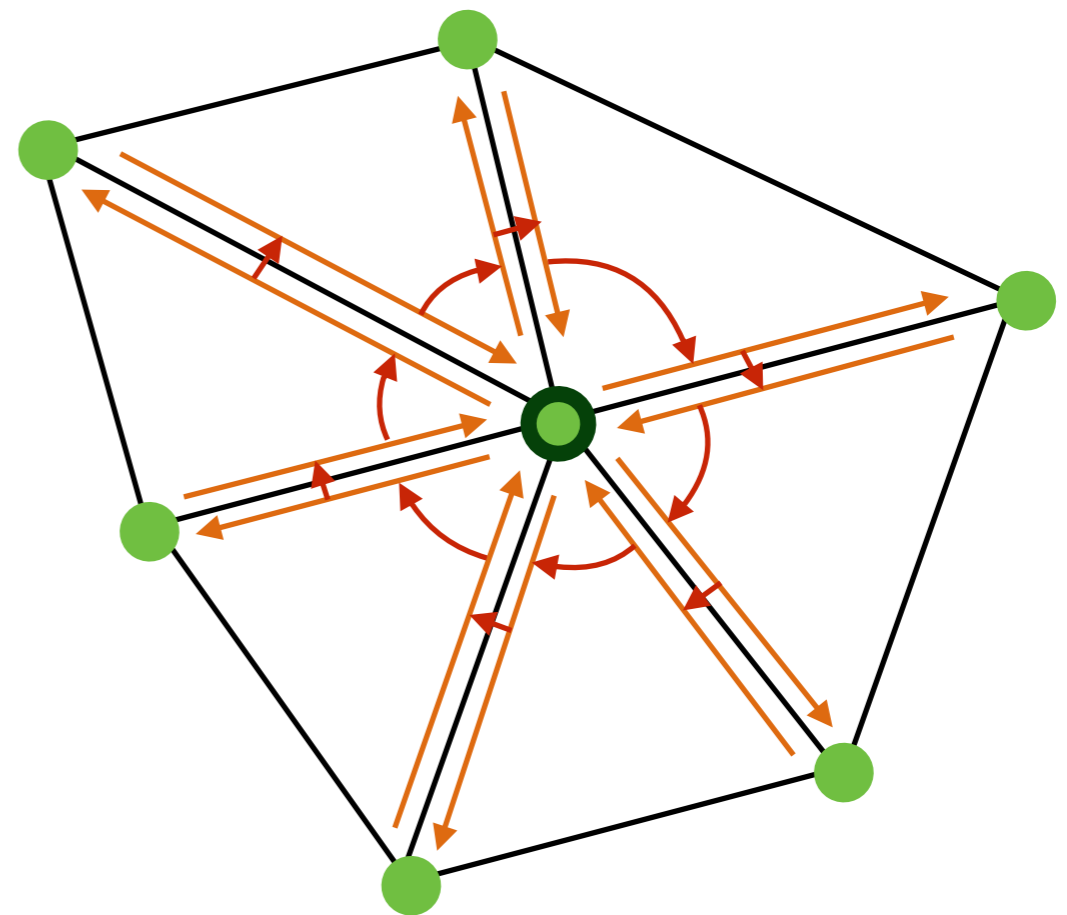
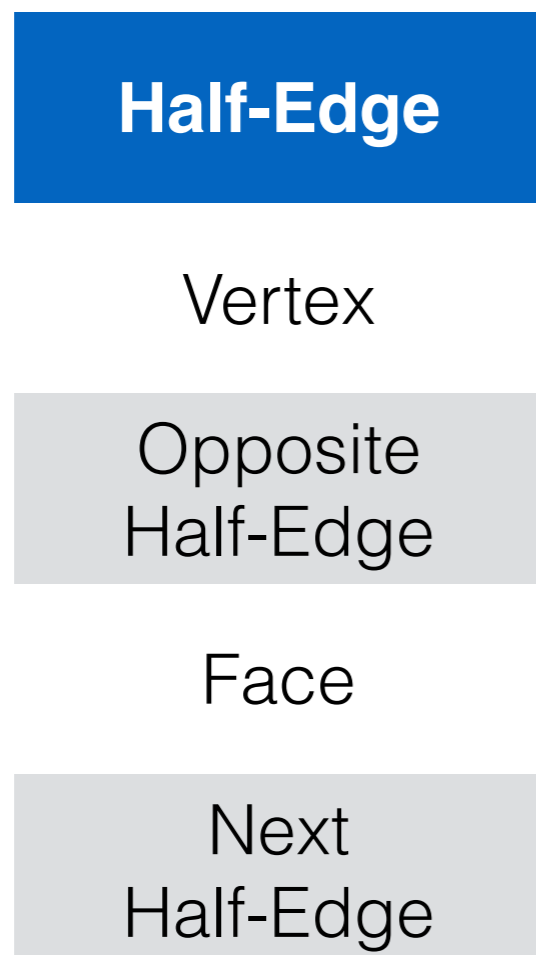
```
original_he = vertex.he;
current = original_he;
do {
    // do something with data
    current = he.opposite.next;
} while ( he != original_he)
```



- Assignment will ask you for other kind of adjacency queries
 - Vertices around Face, Faces around Vertex etc.

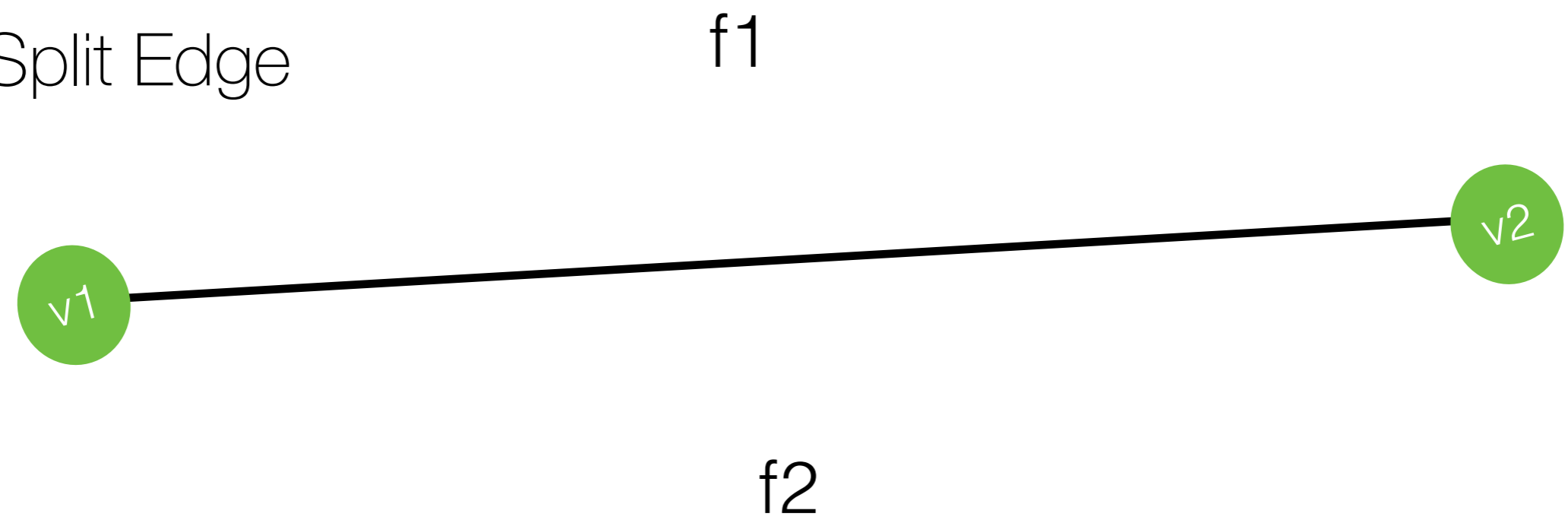
Traversal

- Vertex Normals are defined as weighted average of adjacent faces (weighted by face area)
- How would you compute vertex normals given per face normal and area?



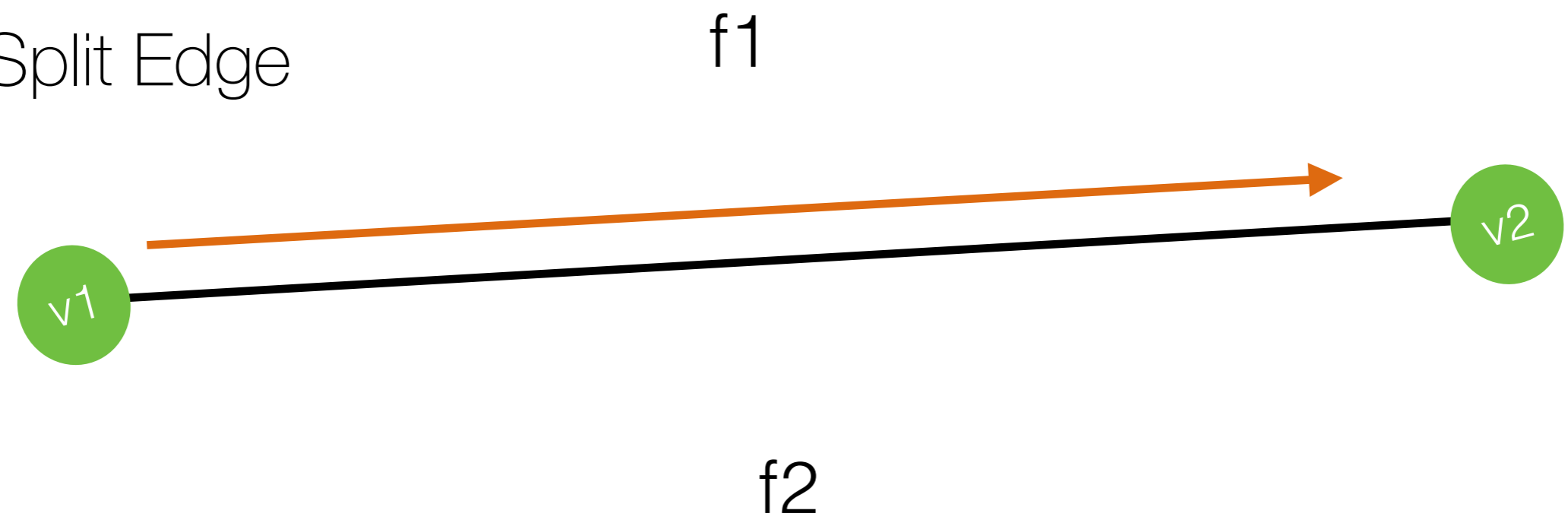
Data Structure Modification

- Split Edge



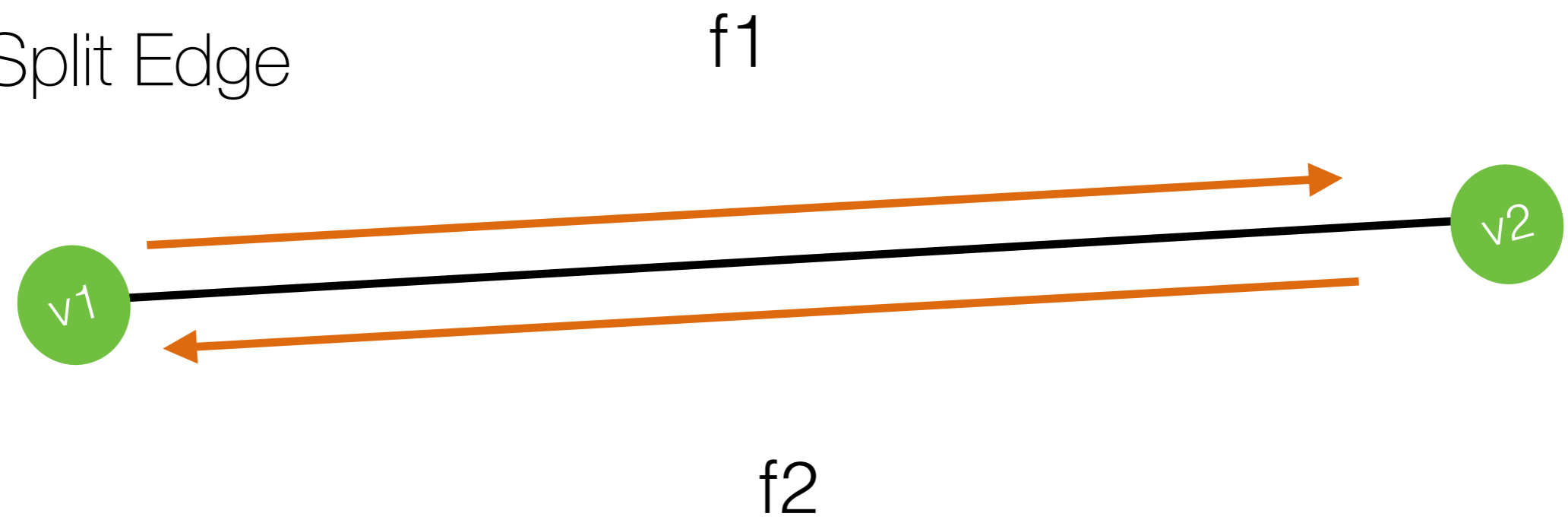
Data Structure Modification

- Split Edge



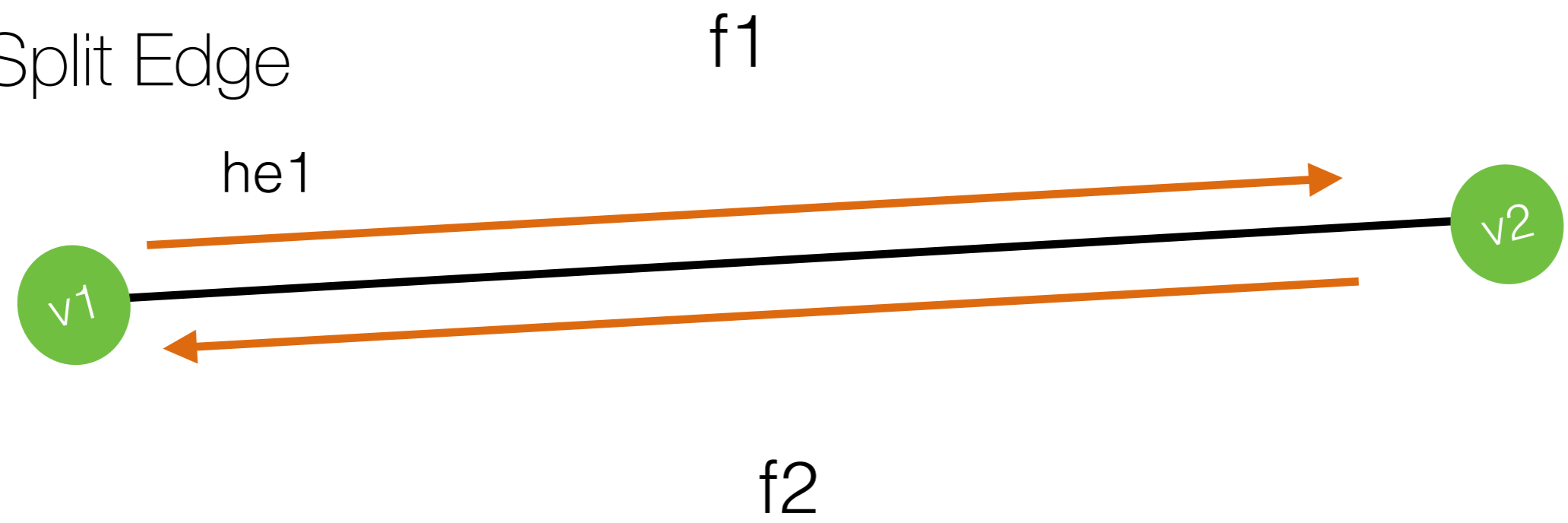
Data Structure Modification

- Split Edge



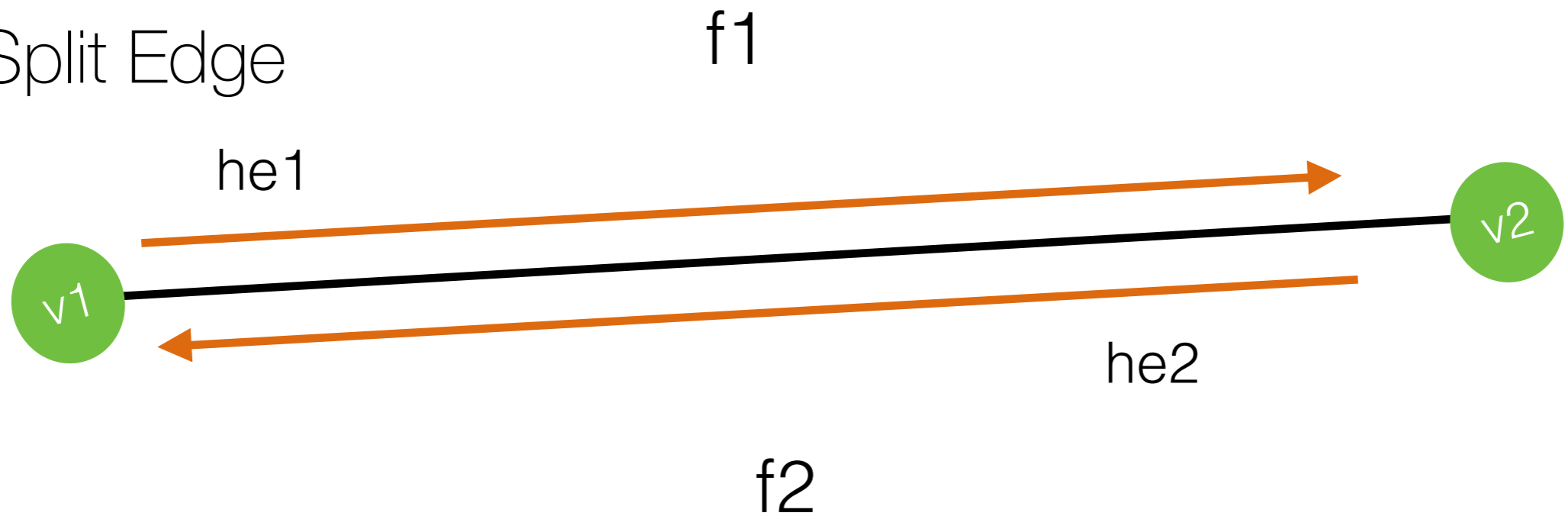
Data Structure Modification

- Split Edge



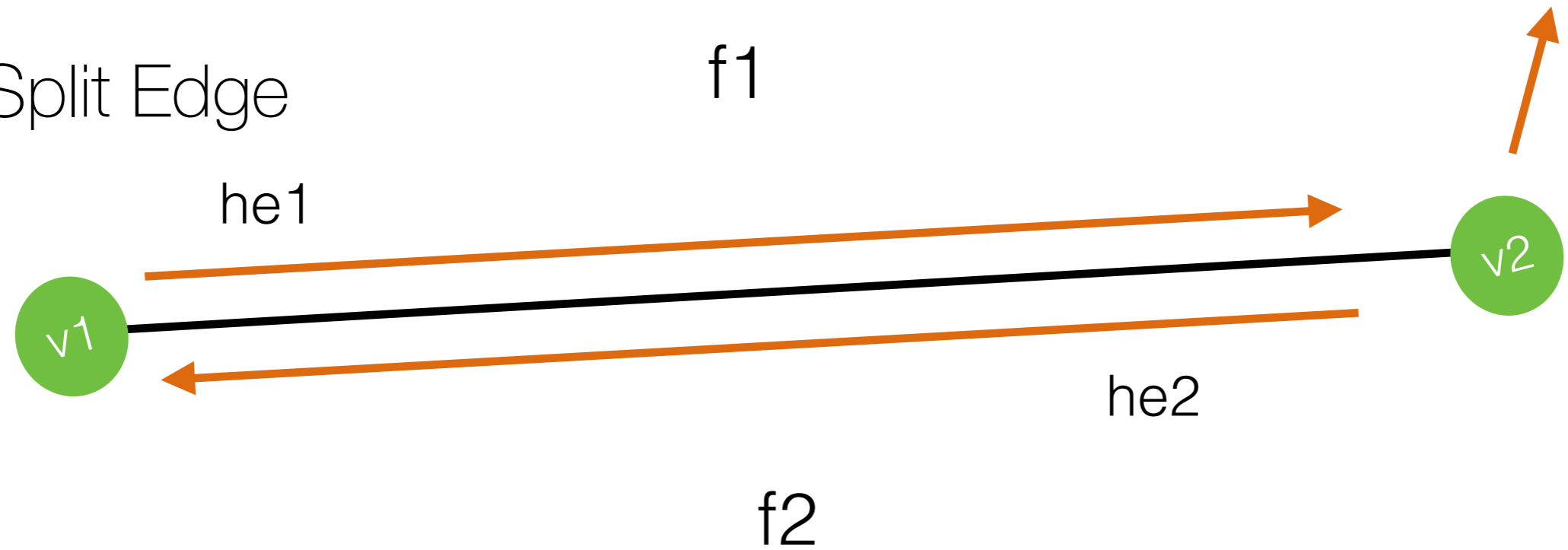
Data Structure Modification

- Split Edge



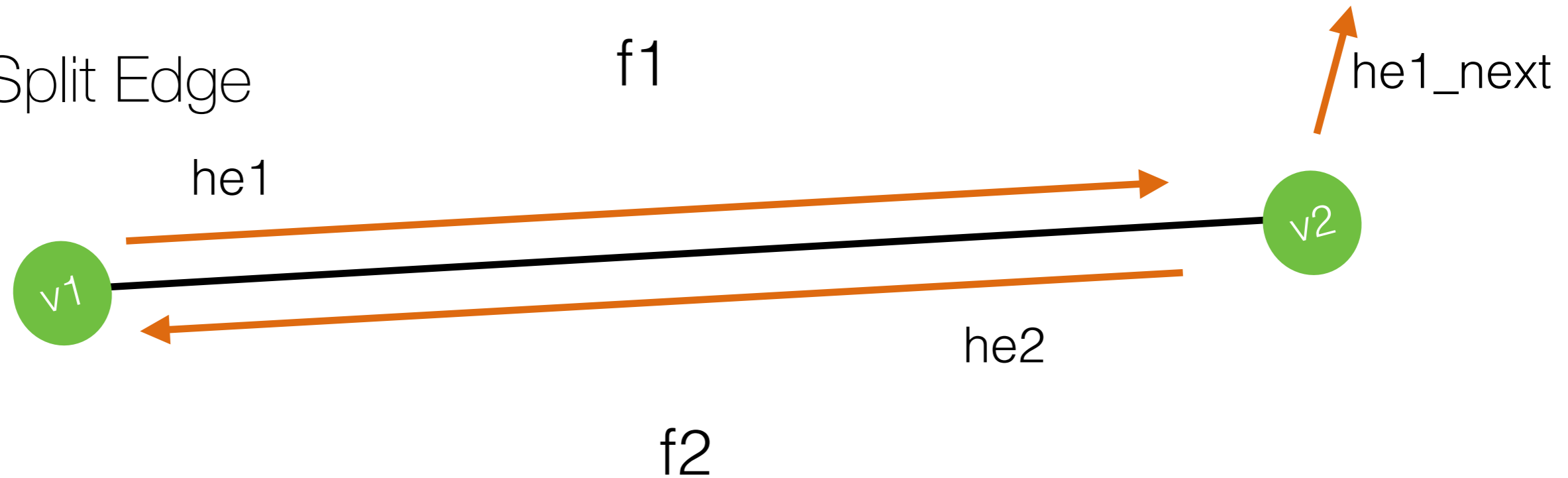
Data Structure Modification

- Split Edge



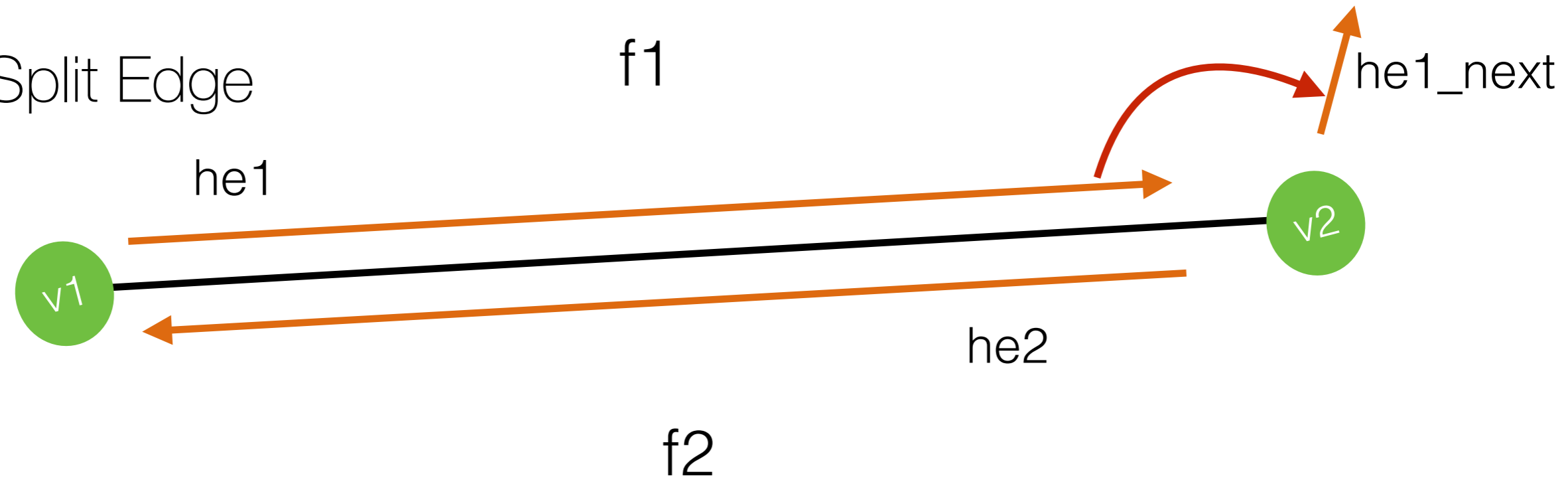
Data Structure Modification

- Split Edge



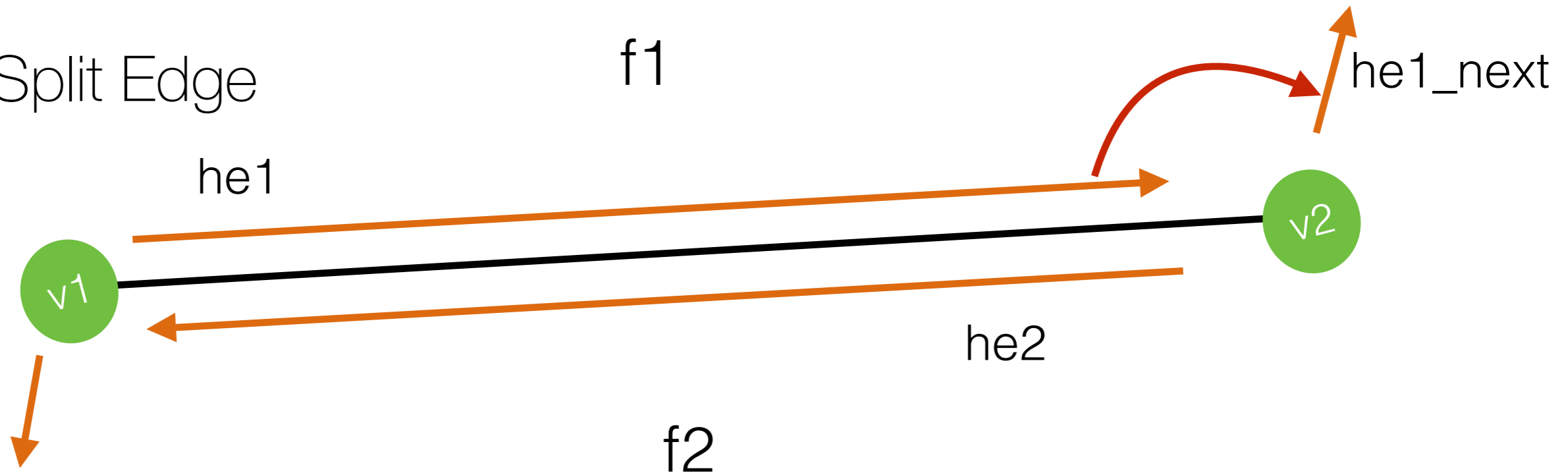
Data Structure Modification

- Split Edge



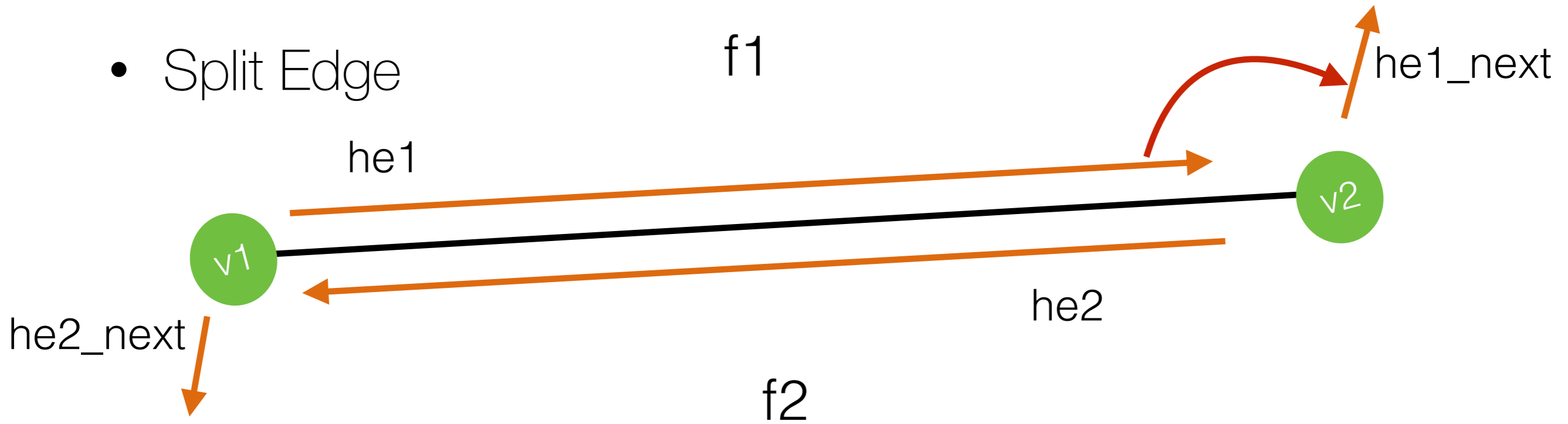
Data Structure Modification

- Split Edge



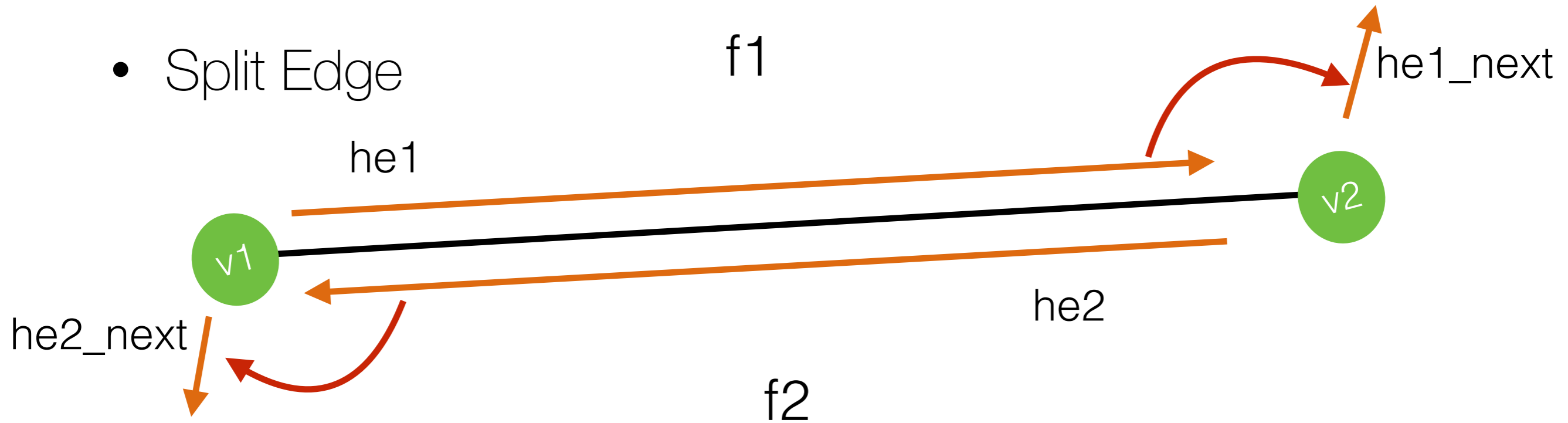
Data Structure Modification

- Split Edge



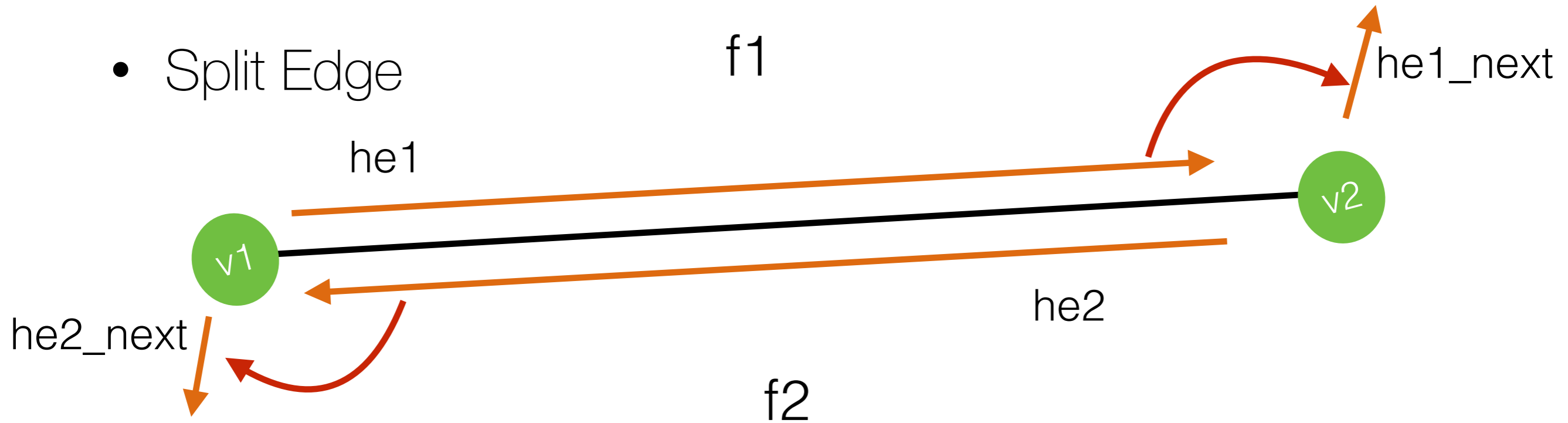
Data Structure Modification

- Split Edge



Data Structure Modification

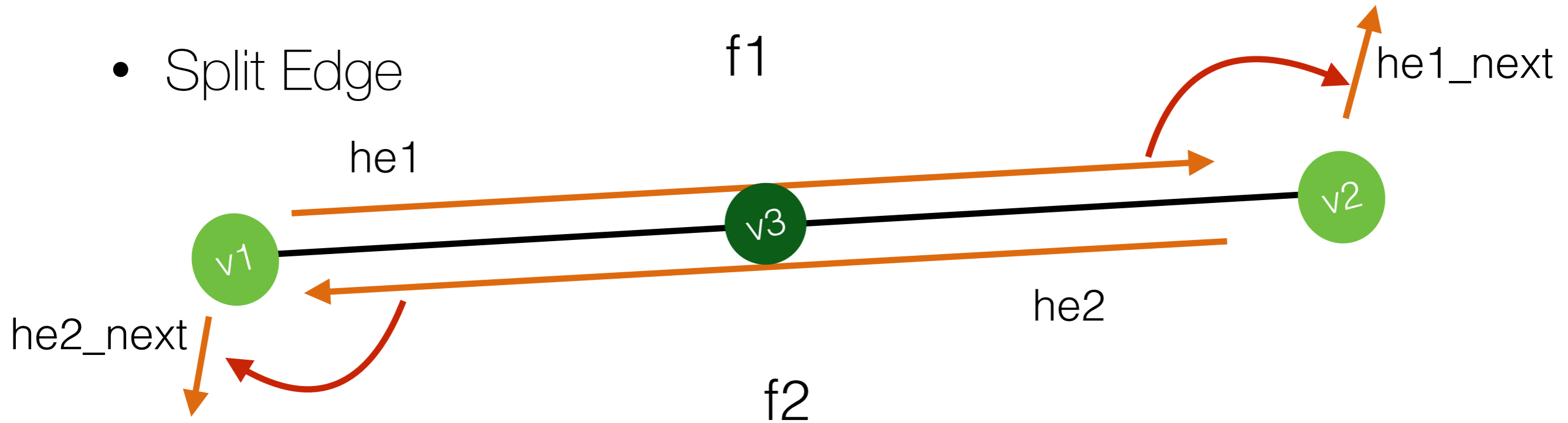
- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

Data Structure Modification

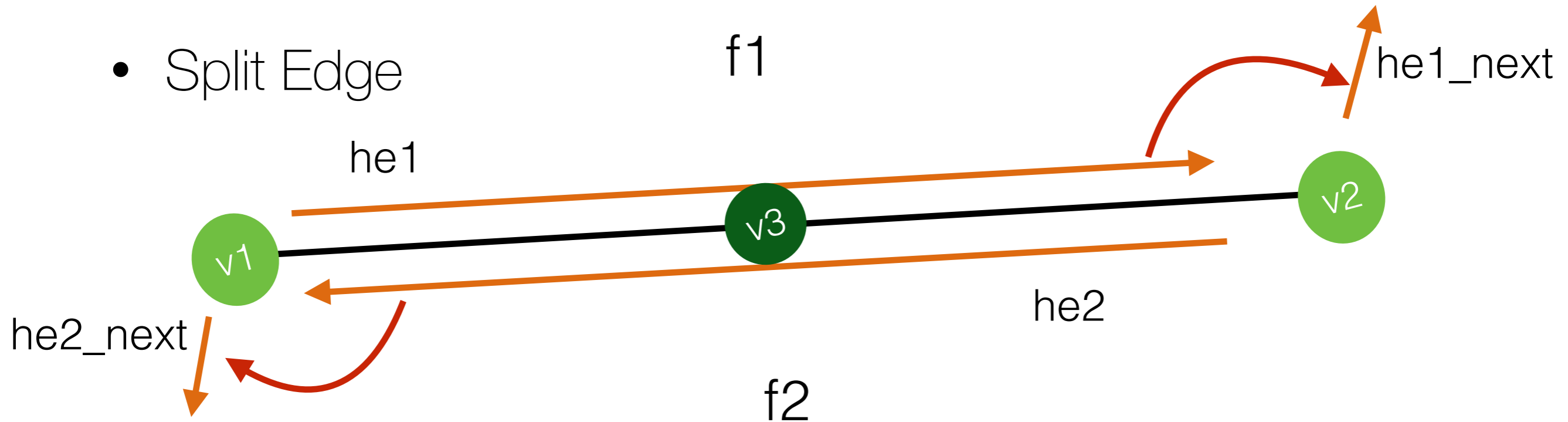
- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

Data Structure Modification

- Split Edge



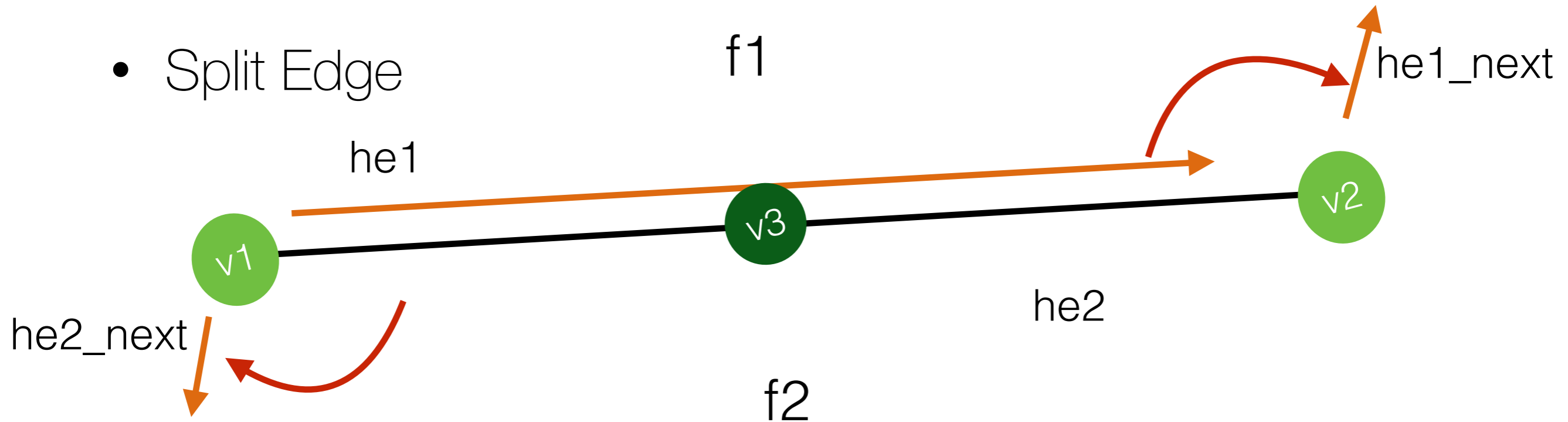
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

Data Structure Modification

- Split Edge



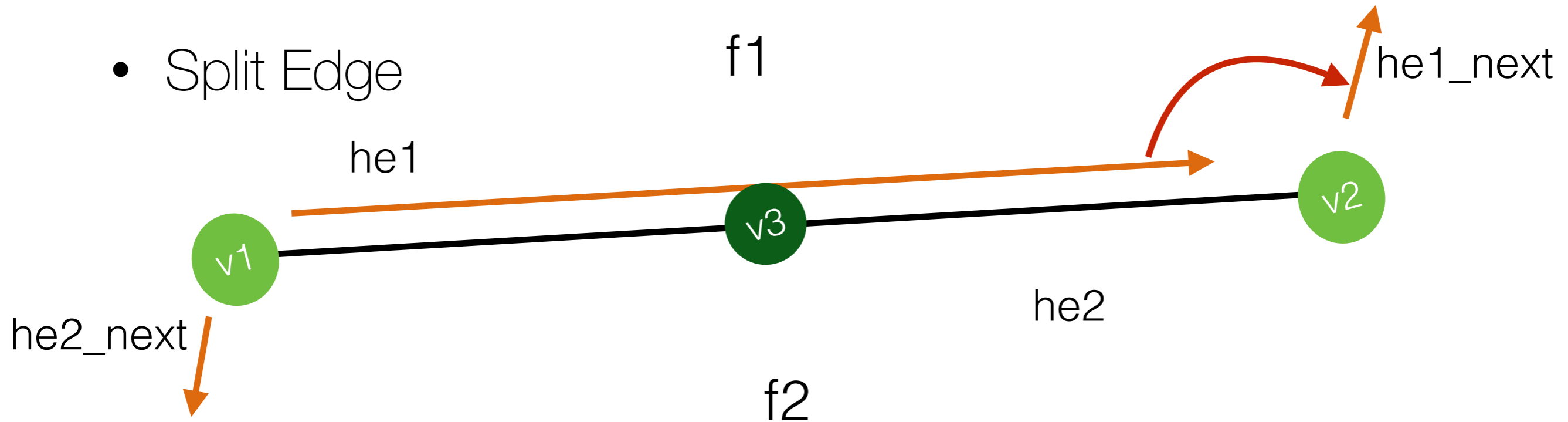
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

Data Structure Modification

- Split Edge



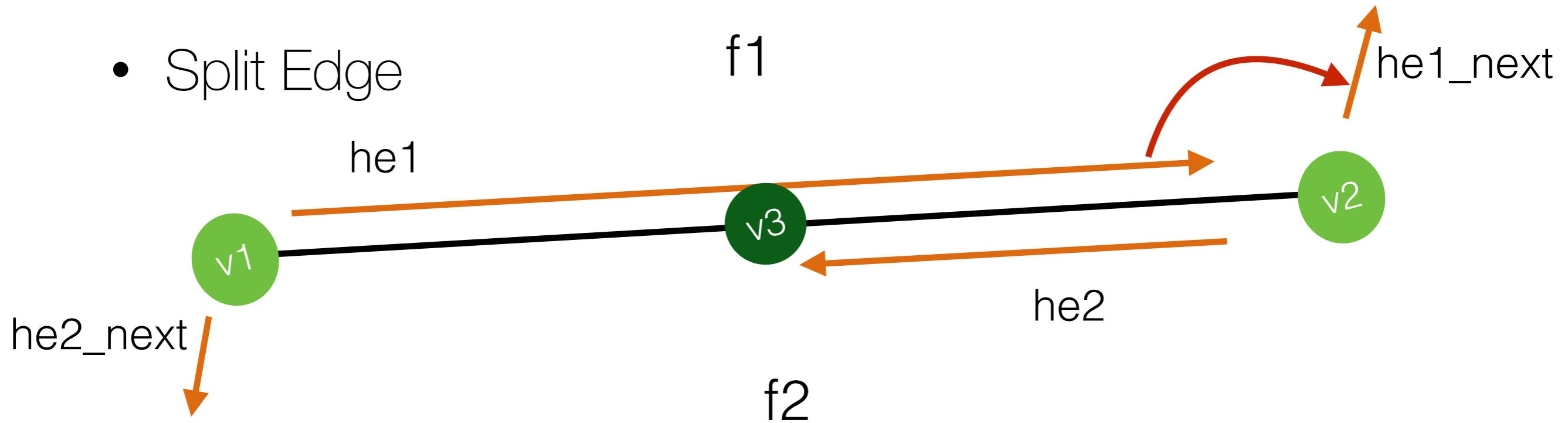
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

Data Structure Modification

- Split Edge



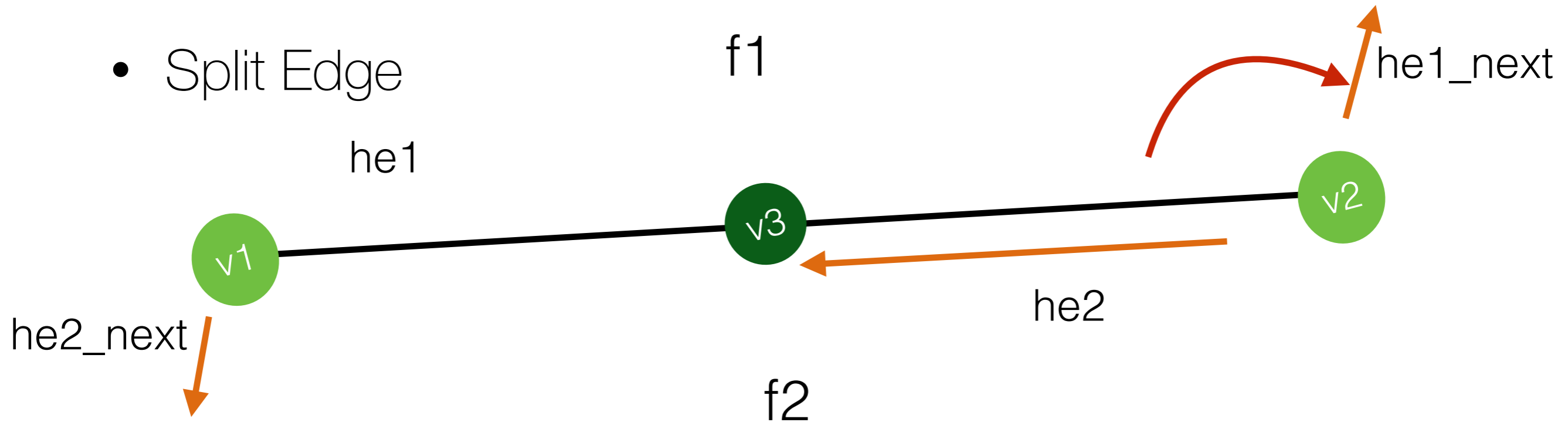
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```


Data Structure Modification

- Split Edge



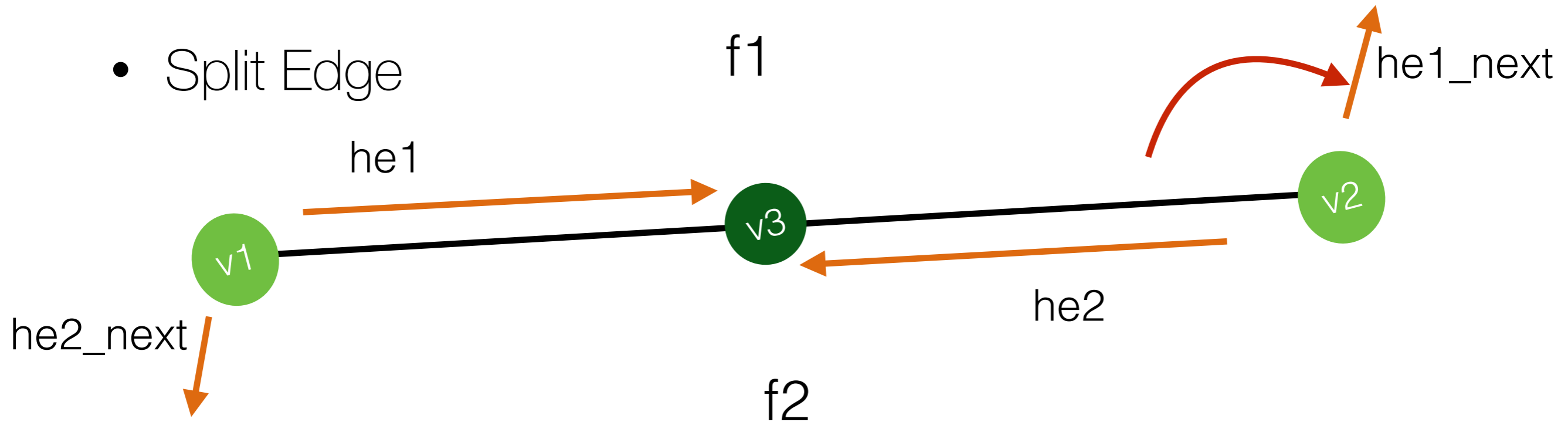
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

Data Structure Modification

- Split Edge



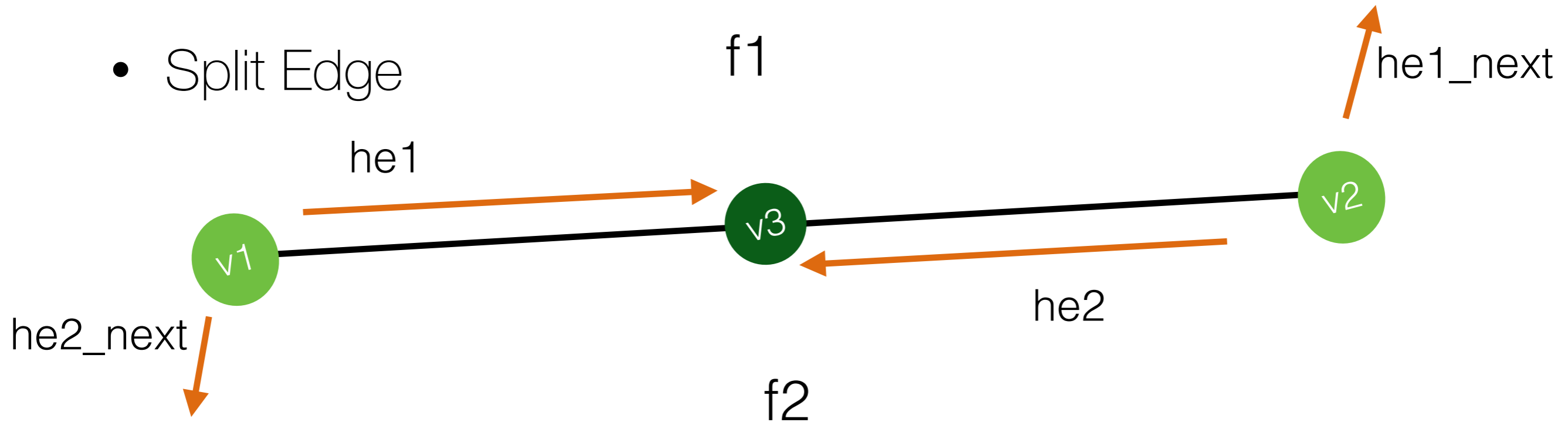
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

Data Structure Modification

- Split Edge



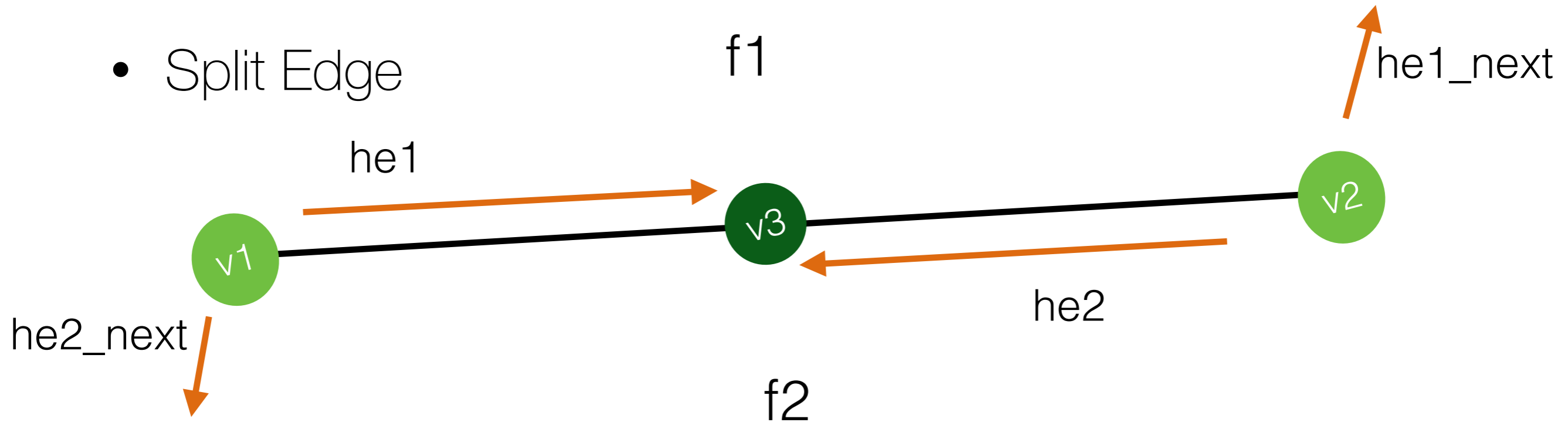
```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

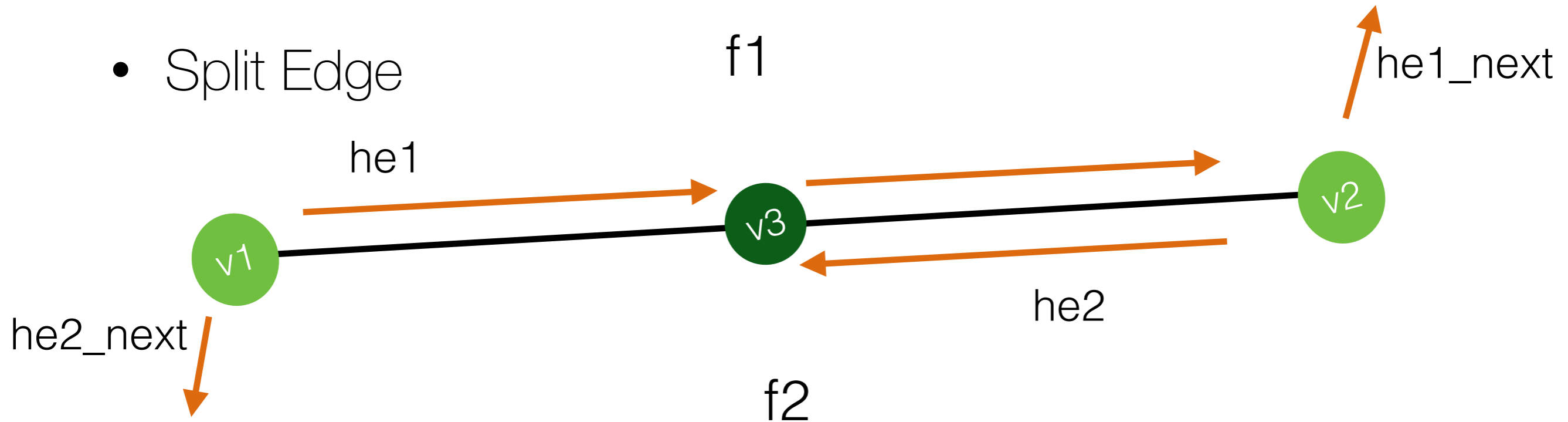
```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

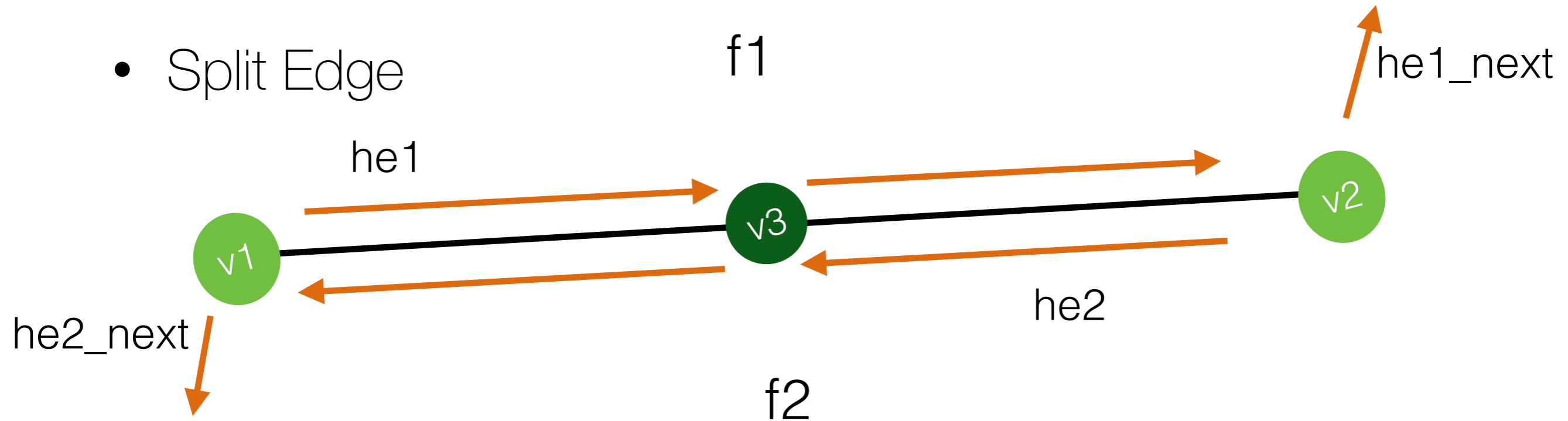
```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

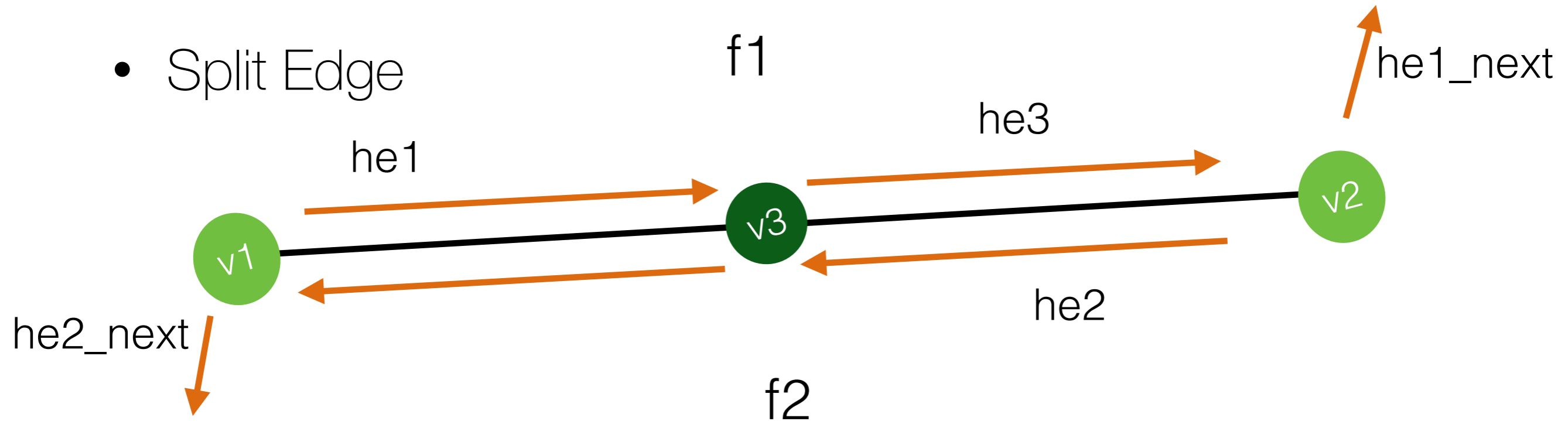
```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

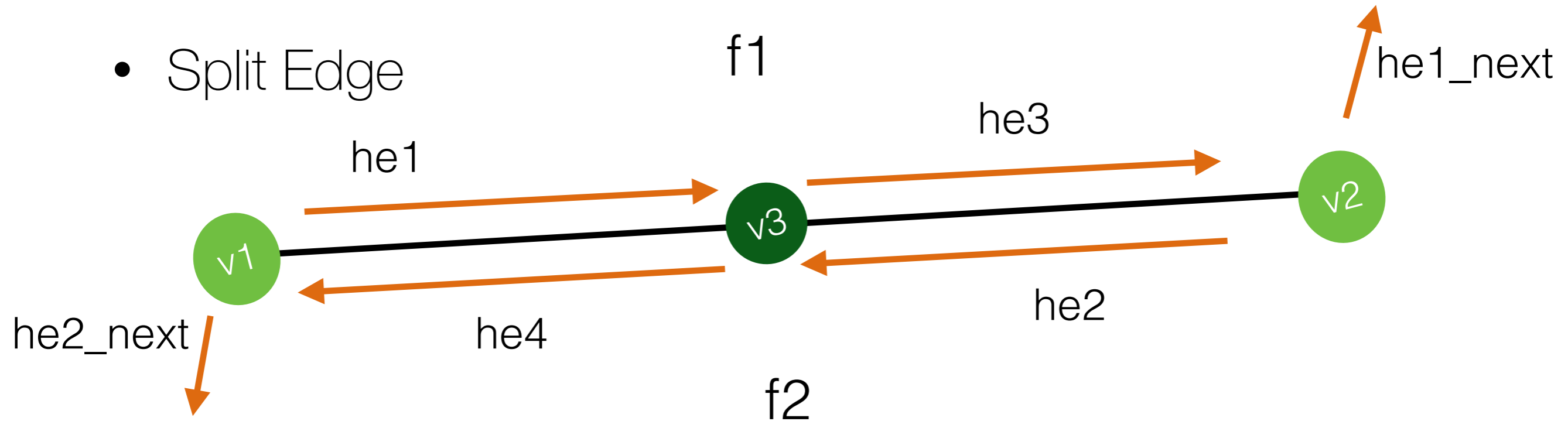
```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```


Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

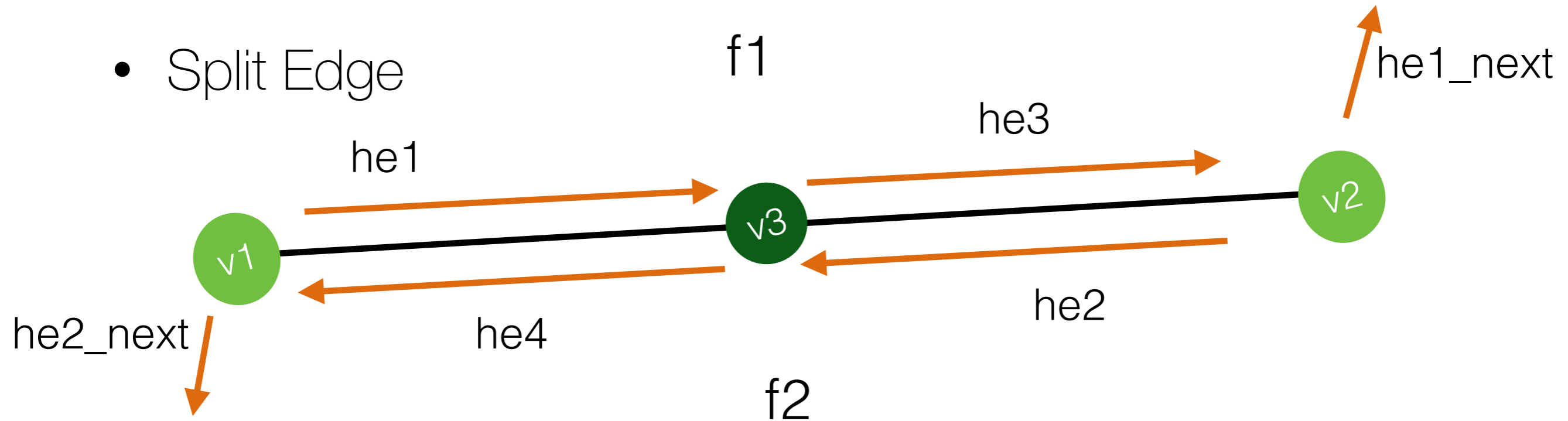
```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

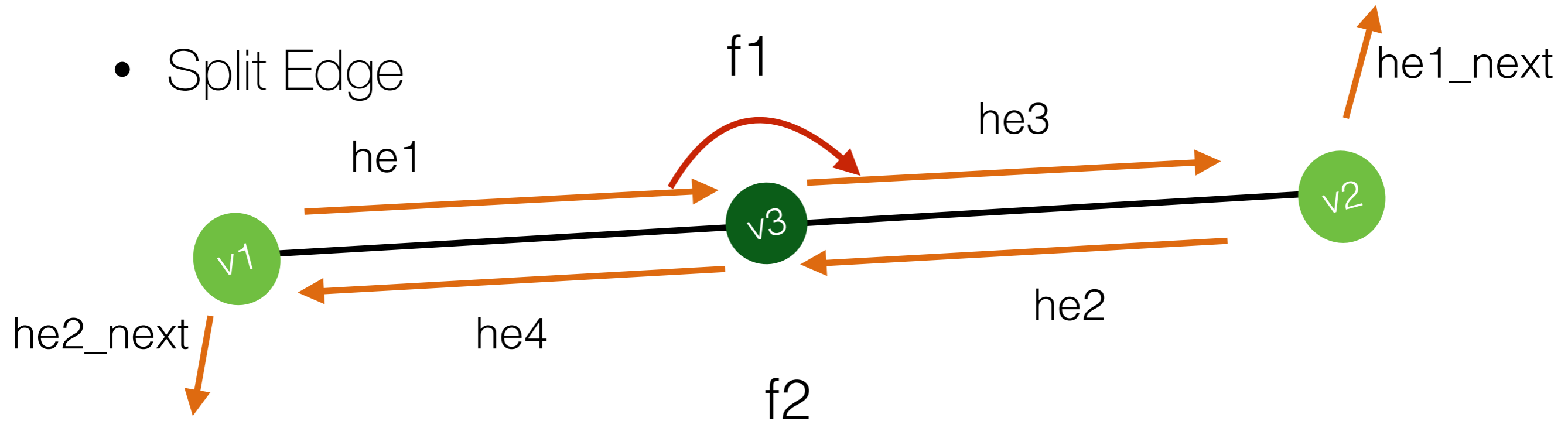
```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

```
he2.next = he4;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

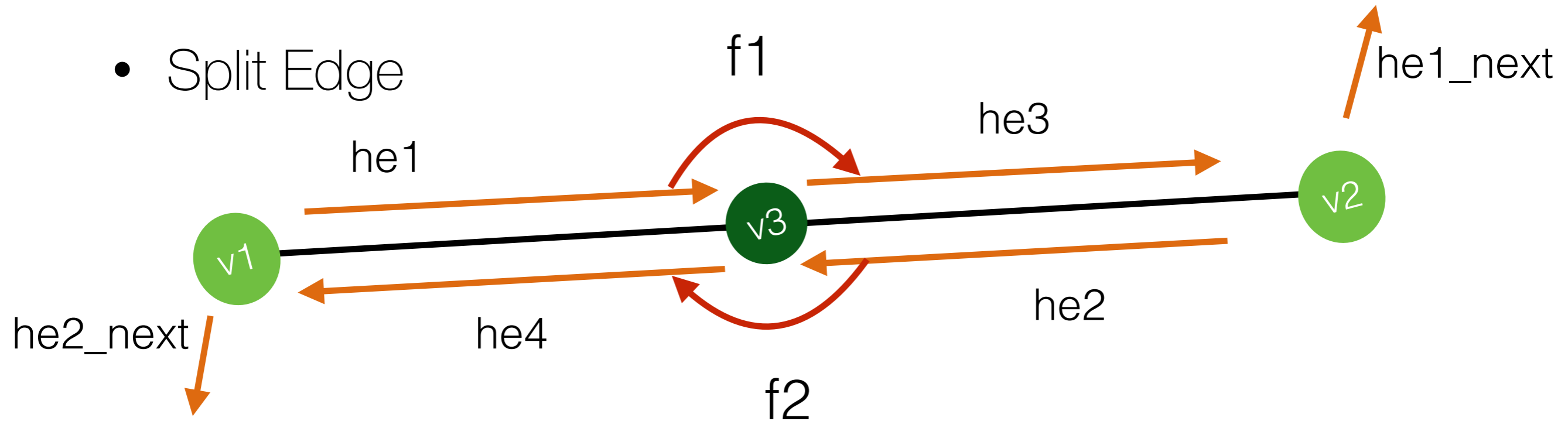
```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

```
he2.next = he4;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

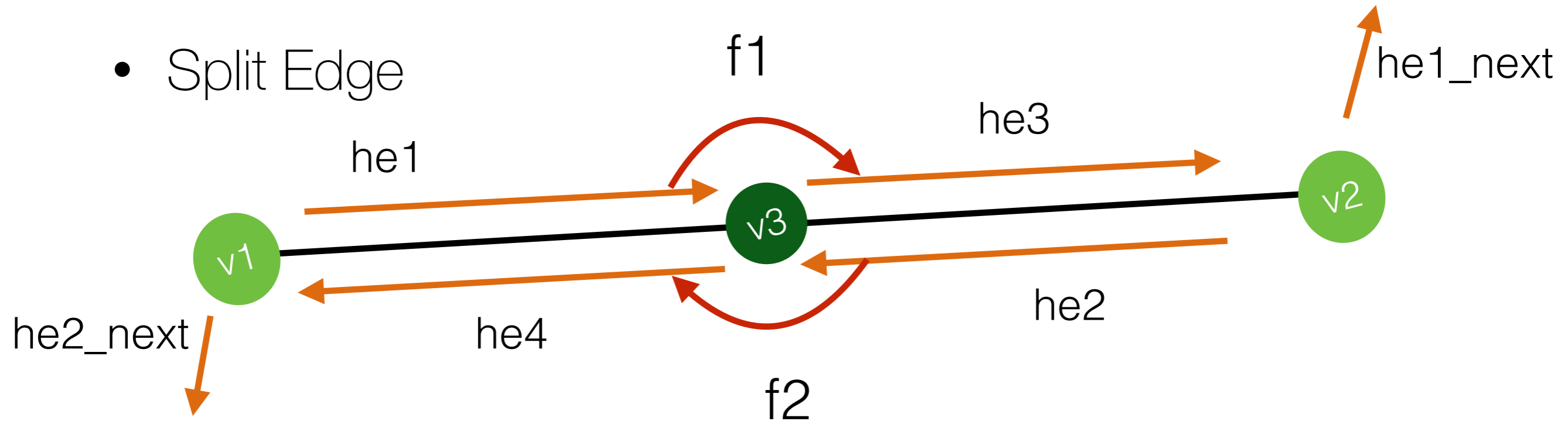
```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

```
he2.next = he4;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

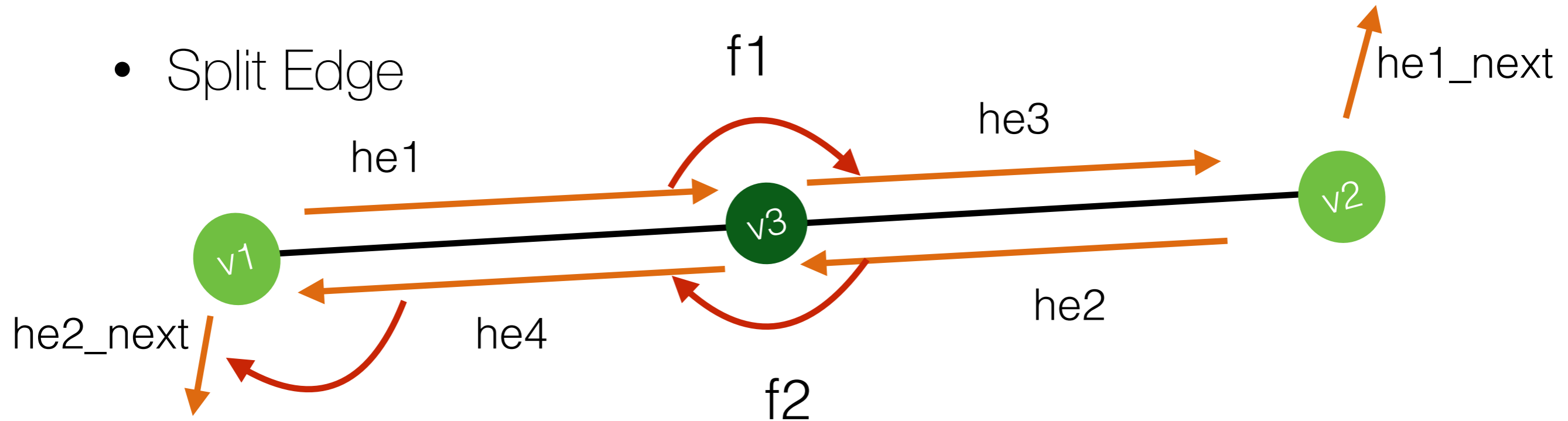
```
he2.next = he4;
```

```
he3.next = he1_next;
```

```
he4.next = he2_next;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

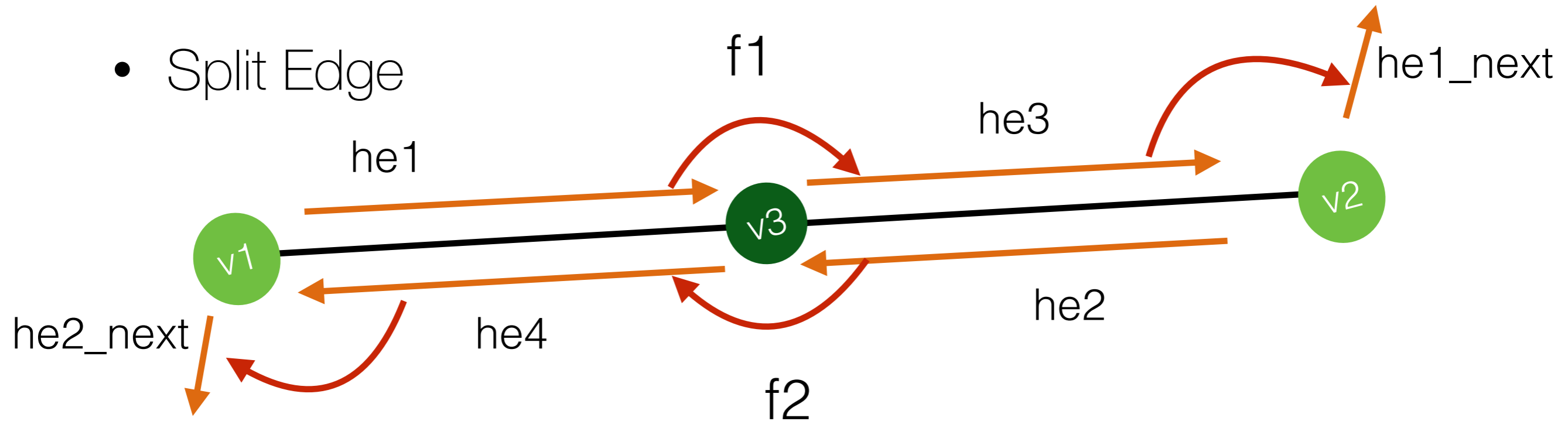
```
he2.next = he4;
```

```
he3.next = he1_next;
```

```
he4.next = he2_next;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

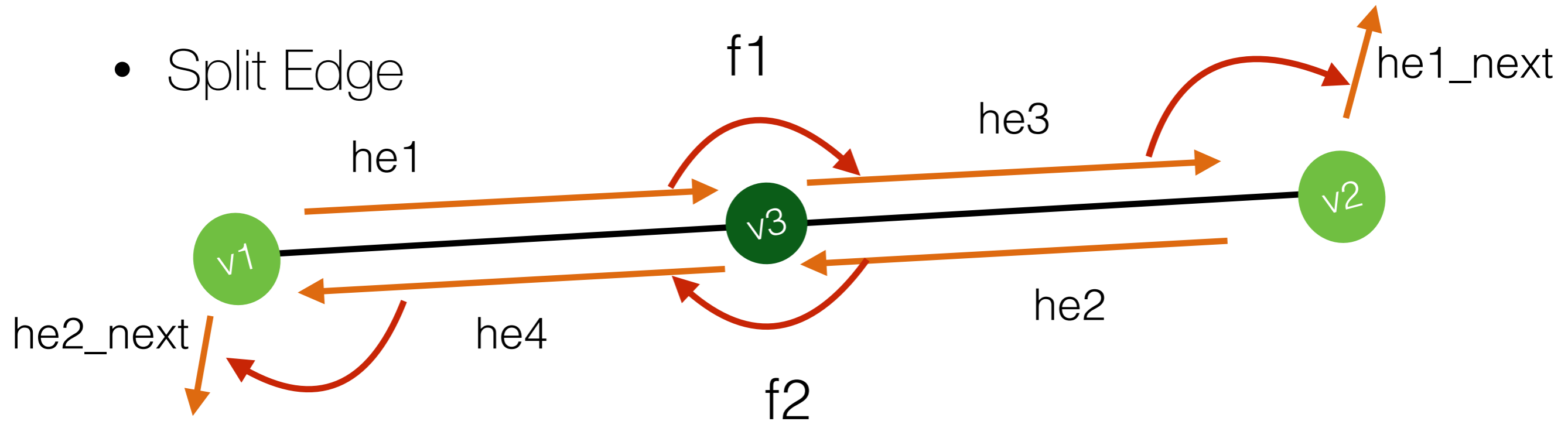
```
he2.next = he4;
```

```
he3.next = he1_next;
```

```
he4.next = he2_next;
```


Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

```
he2.next = he4;
```

```
he3.next = he1_next;
```

```
he4.next = he2_next;
```

```
he1.opposite = he4;
```

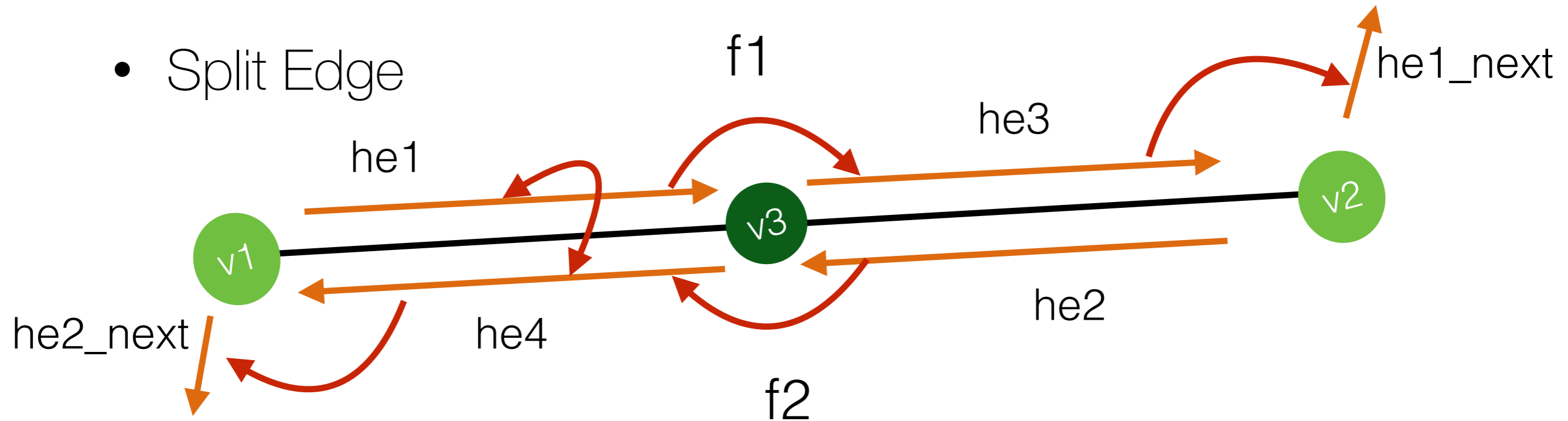
```
he4.opposite = he1;
```

```
he2.opposite = he3;
```

```
he3.opposite = he2;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

```
he2.next = he4;
```

```
he3.next = he1_next;
```

```
he4.next = he2_next;
```

```
he1.opposite = he4;
```

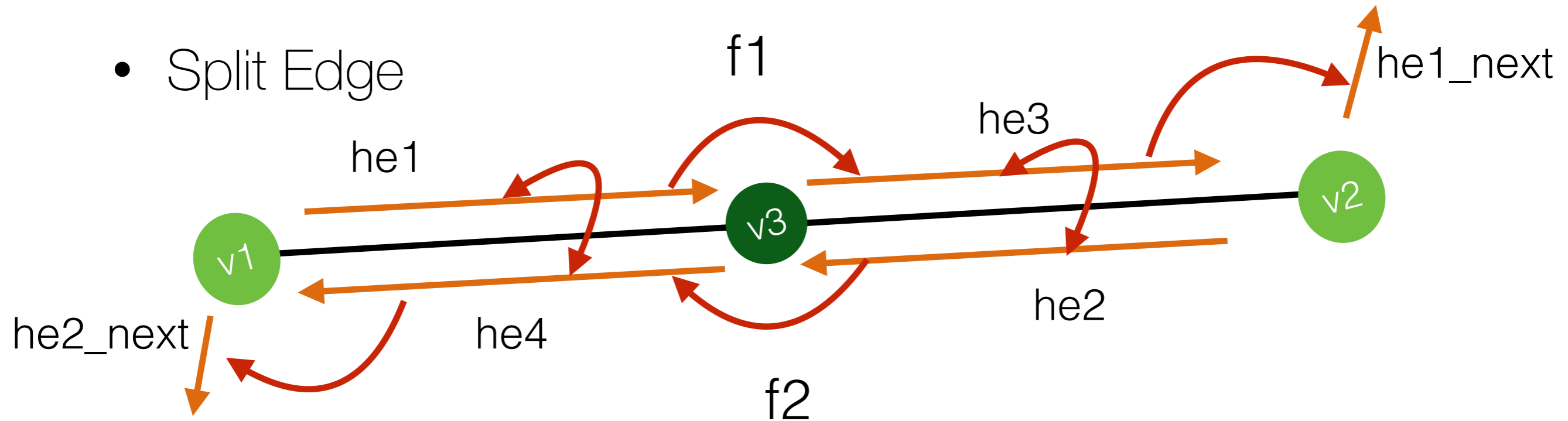
```
he4.opposite = he1;
```

```
he2.opposite = he3;
```

```
he3.opposite = he2;
```

Data Structure Modification

- Split Edge



```
v3 = addVertex( weightedAvgPos(v1, v2, factor) );
```

```
he1.vertex = v3;
```

```
he2.vertex = v3;
```

```
he3 = addHalfEdge( v3, v2, f1 );
```

```
he4 = addHalfEdge( v3, v1, f2 );
```

```
he1.next = he3;
```

```
he2.next = he4;
```

```
he3.next = he1_next;
```

```
he4.next = he2_next;
```

```
he1.opposite = he4;
```

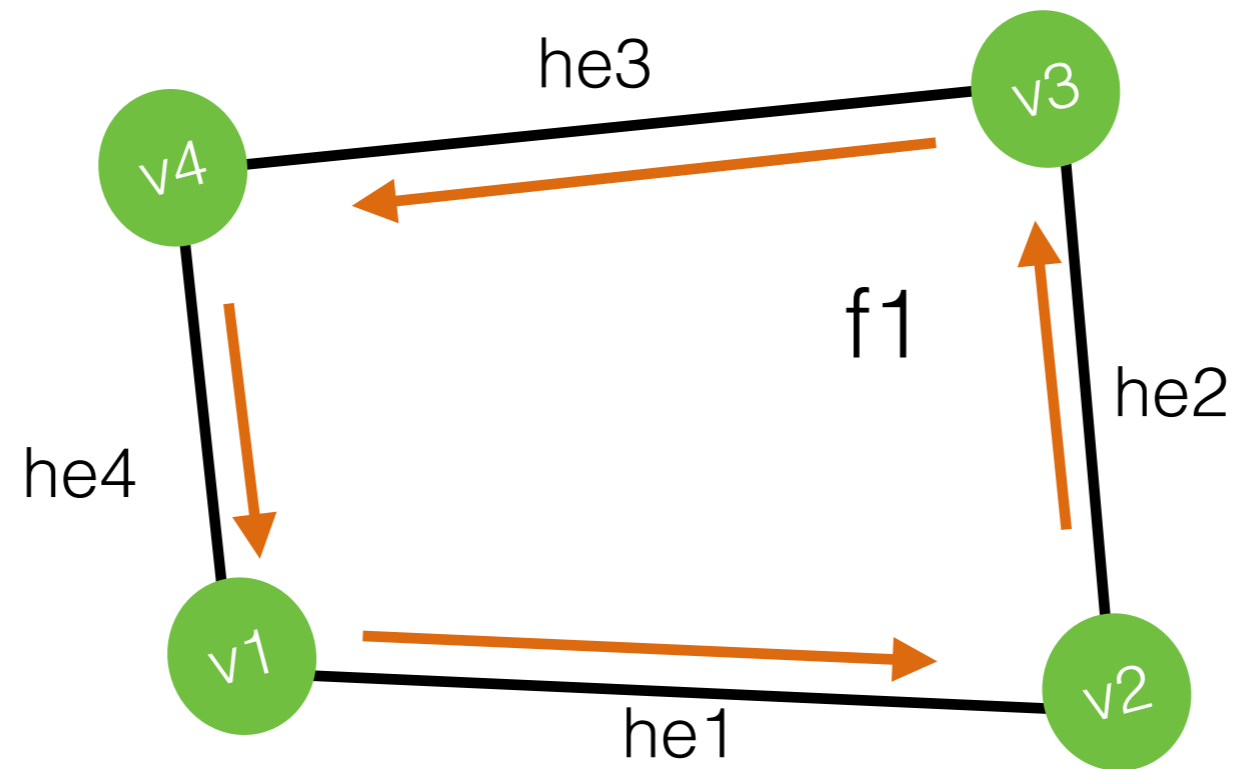
```
he4.opposite = he1;
```

```
he2.opposite = he3;
```

```
he3.opposite = he2;
```

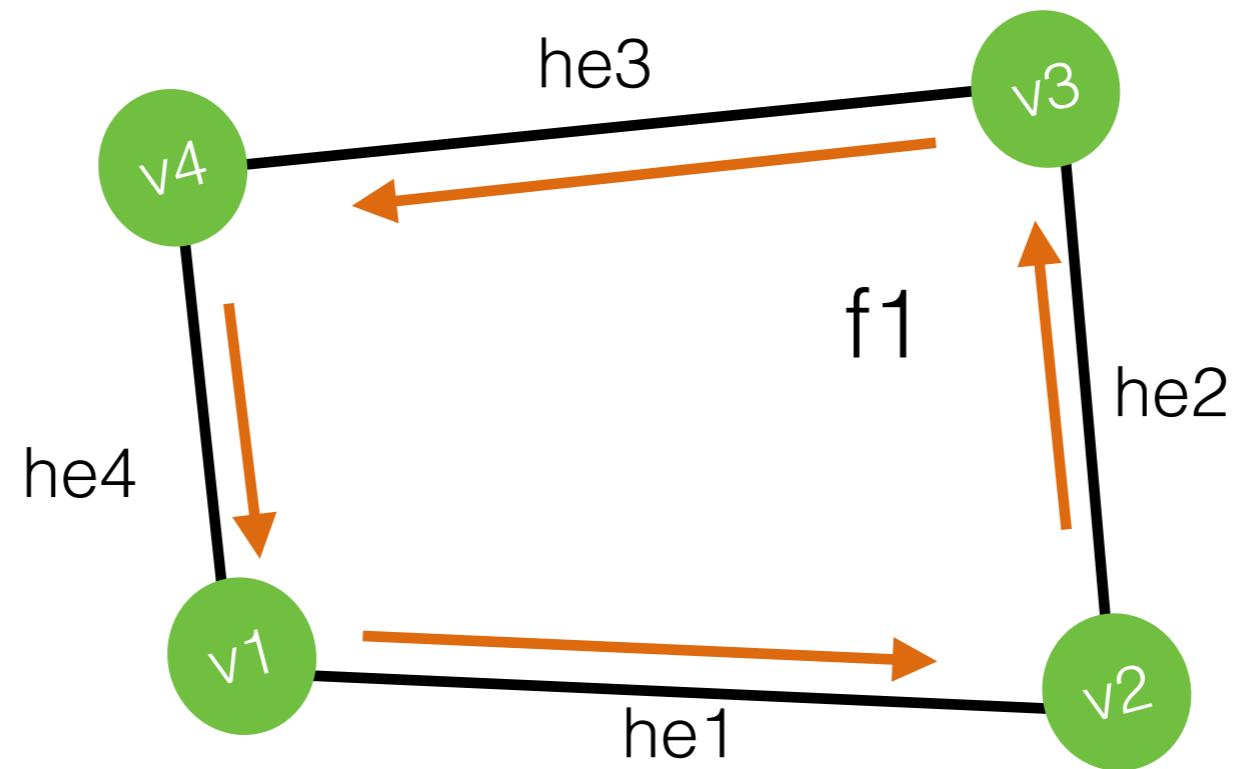
Data Structure Modification

- Split Face



Data Structure Modification

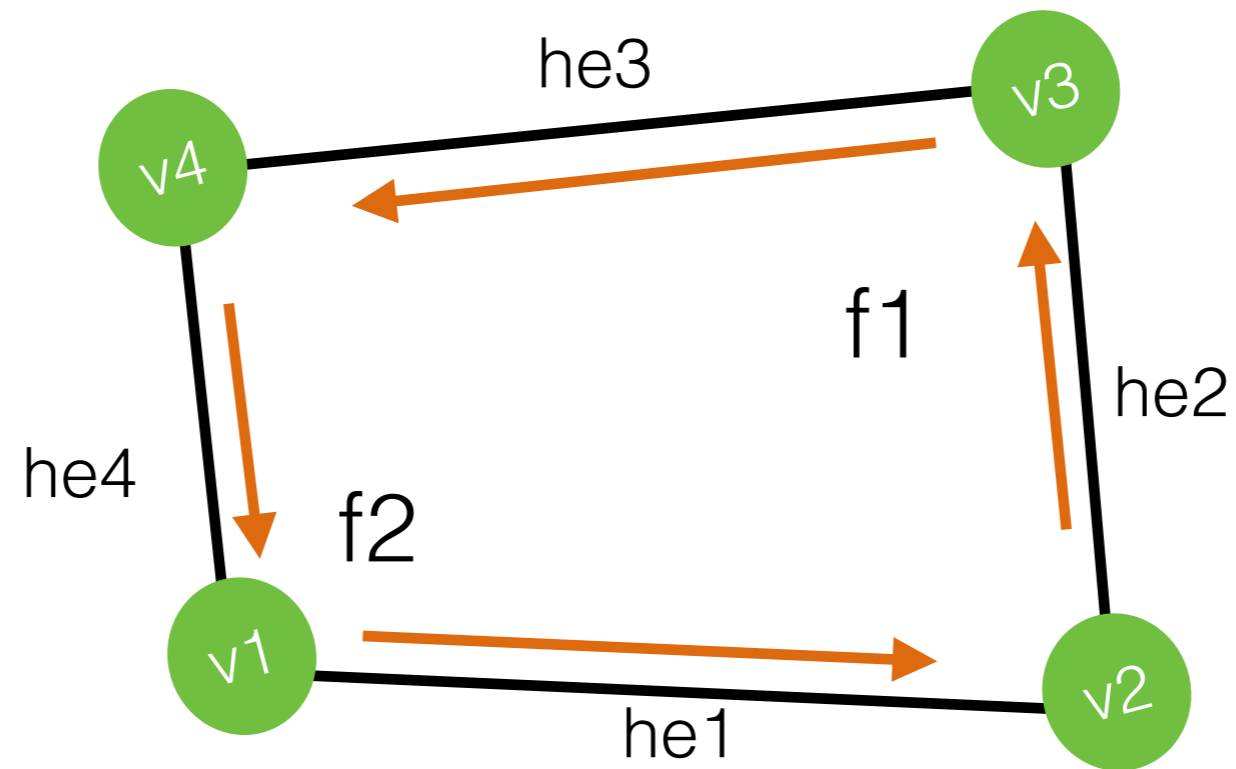
- Split Face



`f2 = addFace();`

Data Structure Modification

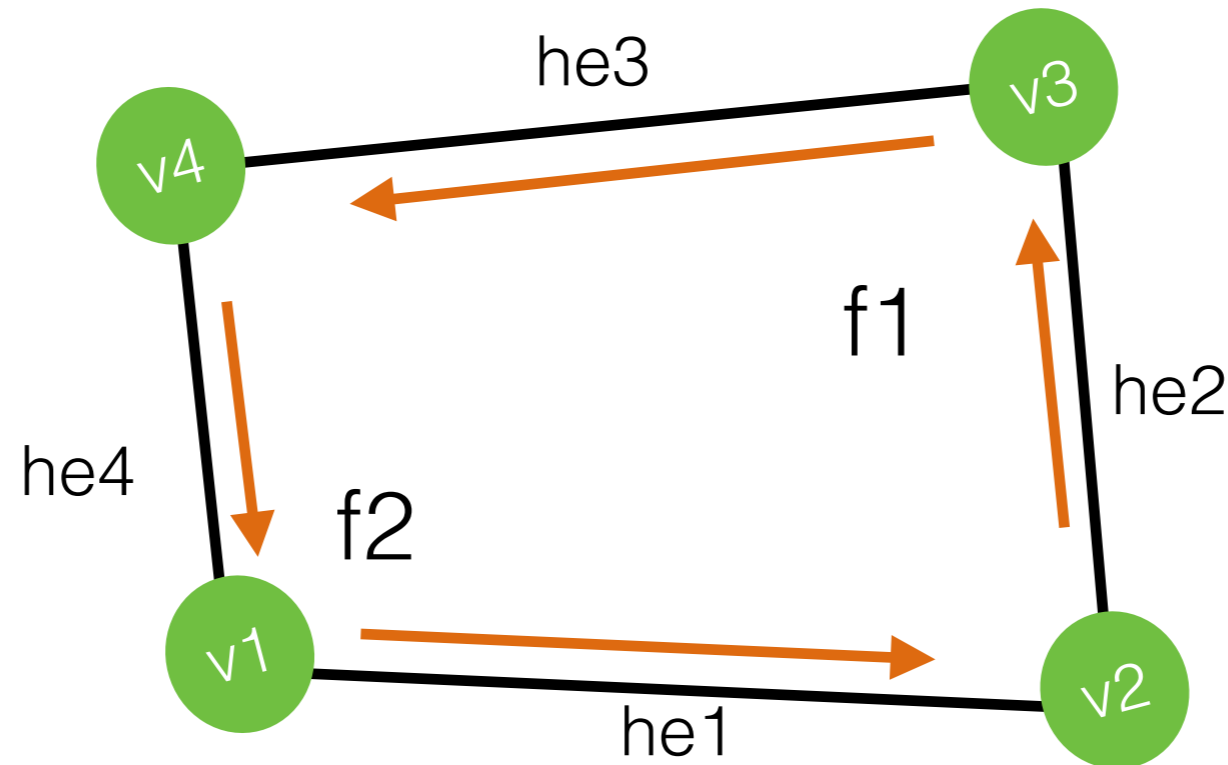
- Split Face



f2 = addFace();

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

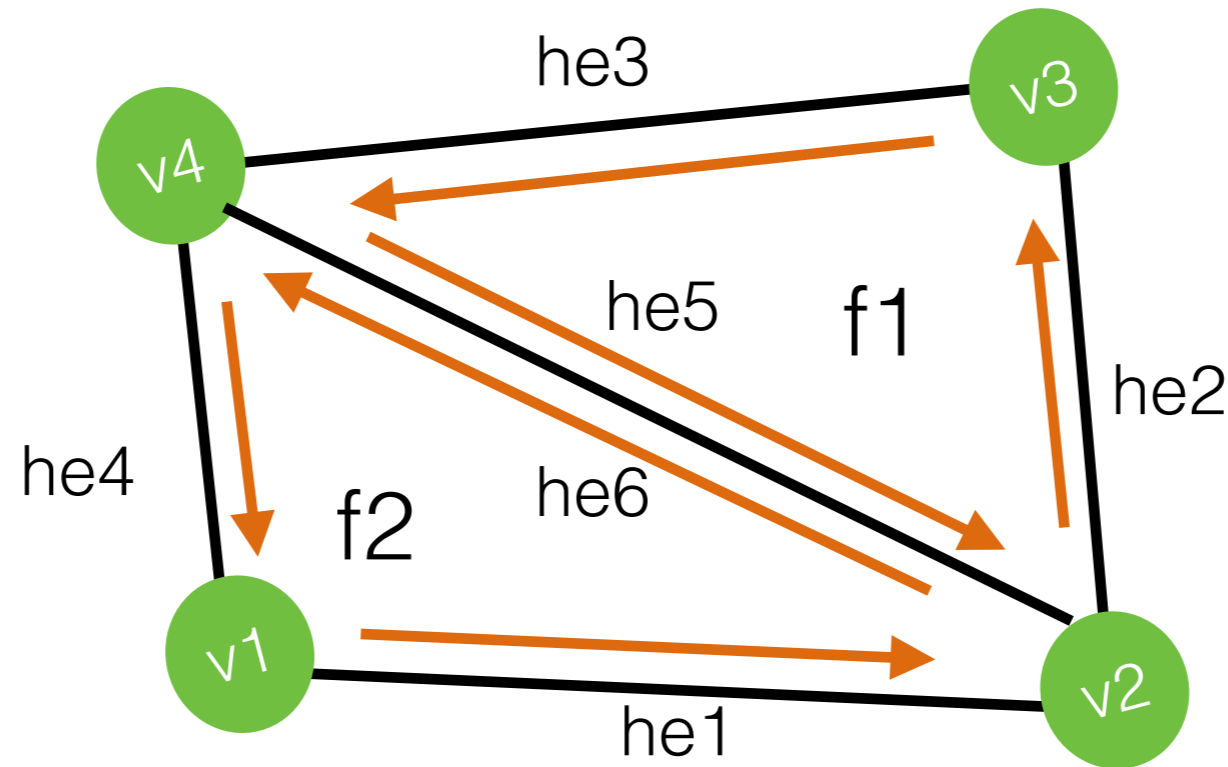
```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```


Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

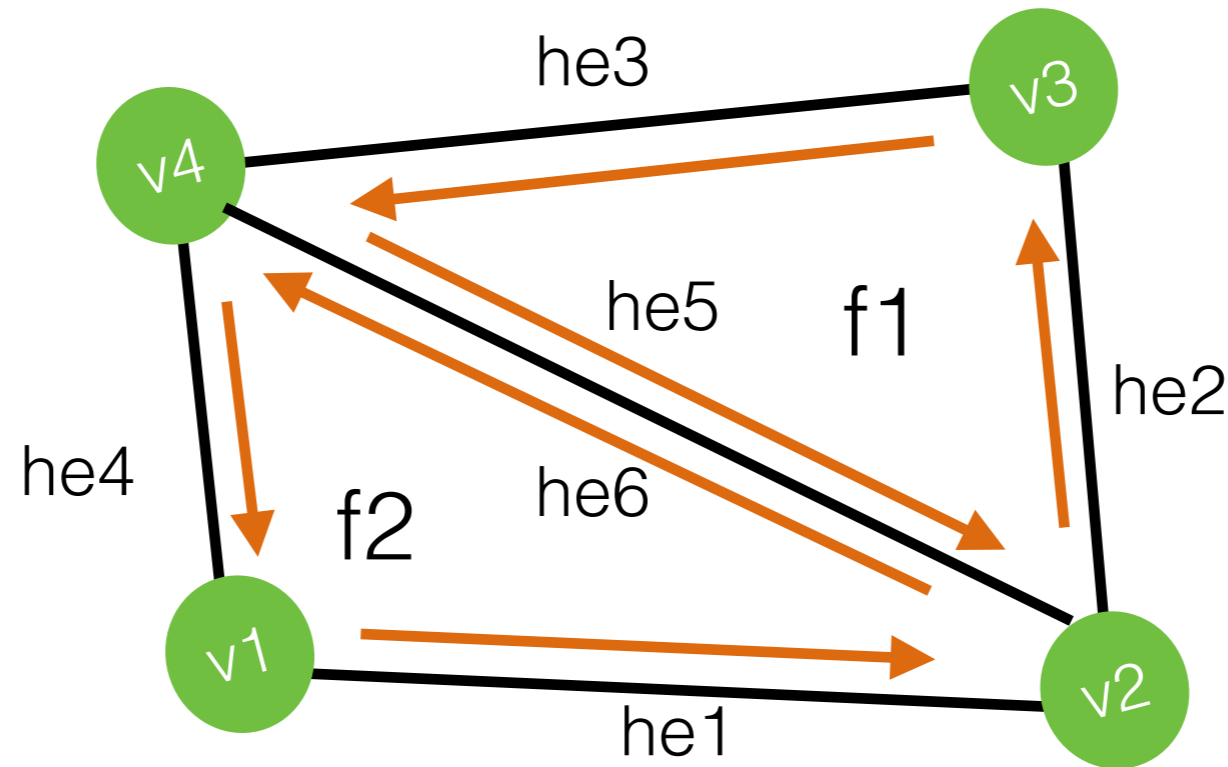
```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

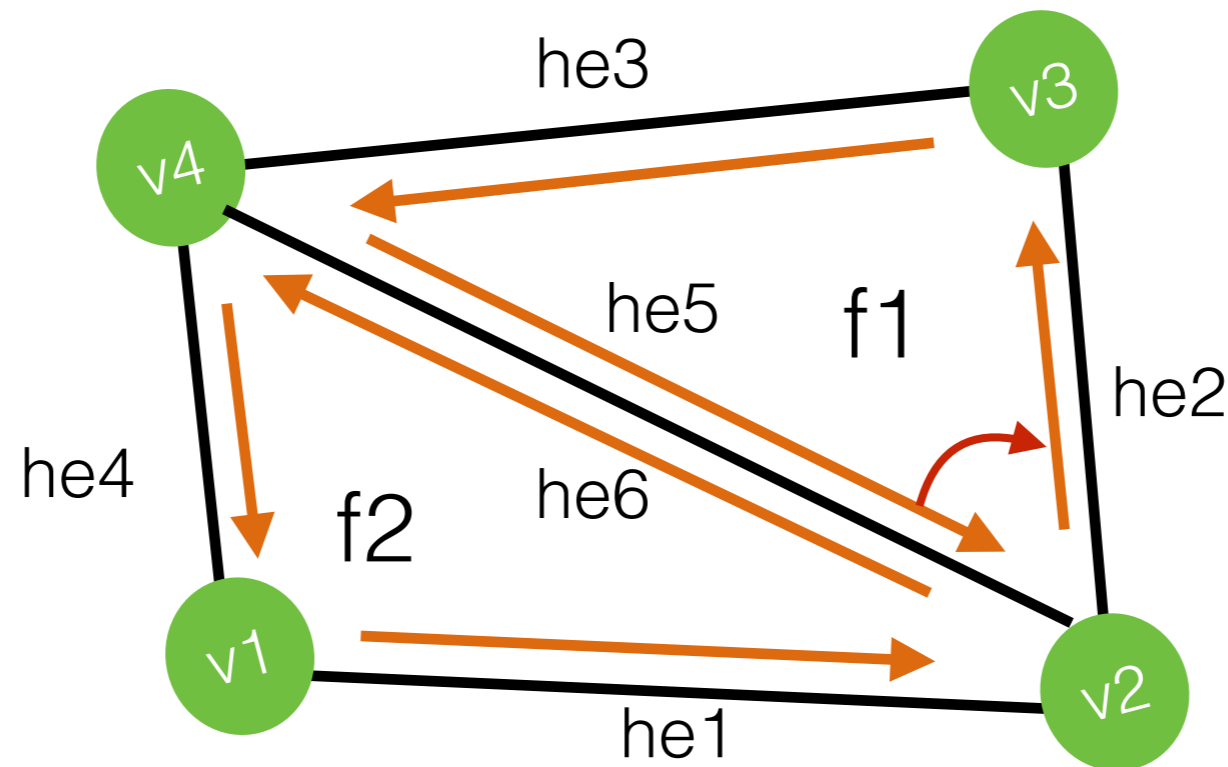
```
he3.next = he5;
```

```
he1.next = he6;
```

```
he6.next = he4;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

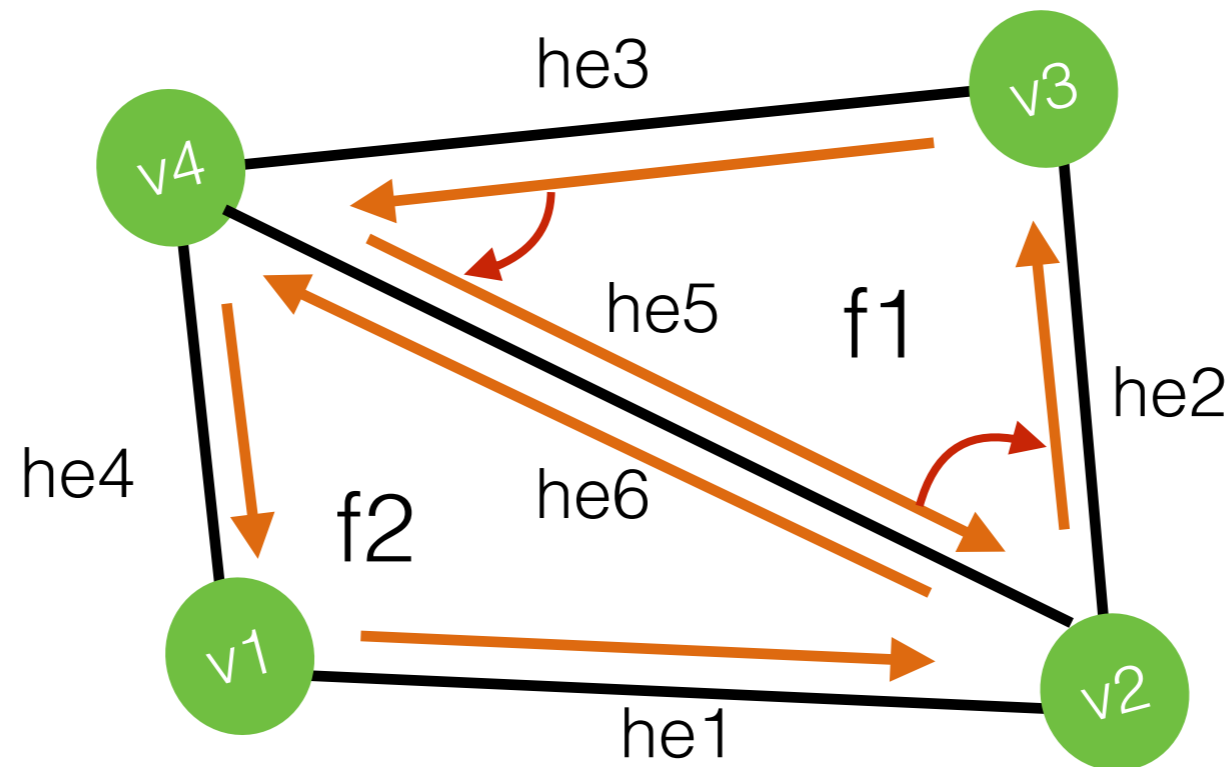
```
he3.next = he5;
```

```
he1.next = he6;
```

```
he6.next = he4;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

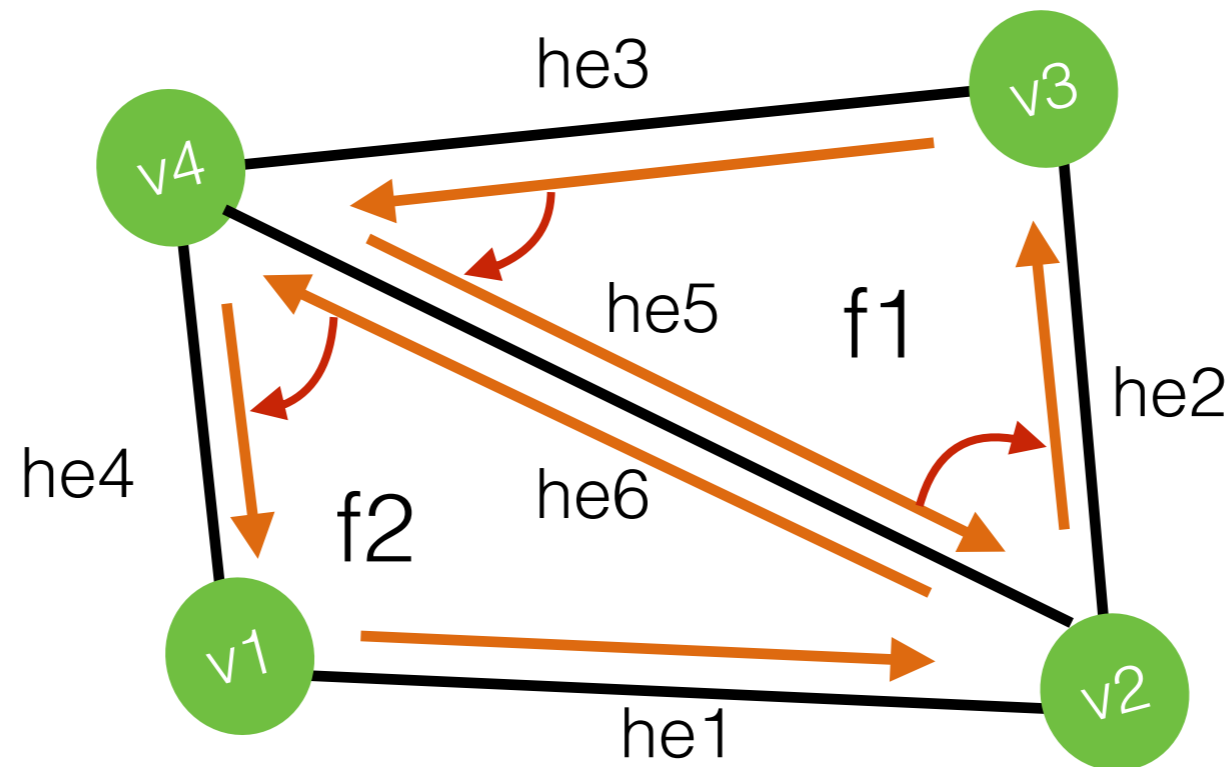
```
he3.next = he5;
```

```
he1.next = he6;
```

```
he6.next = he4;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

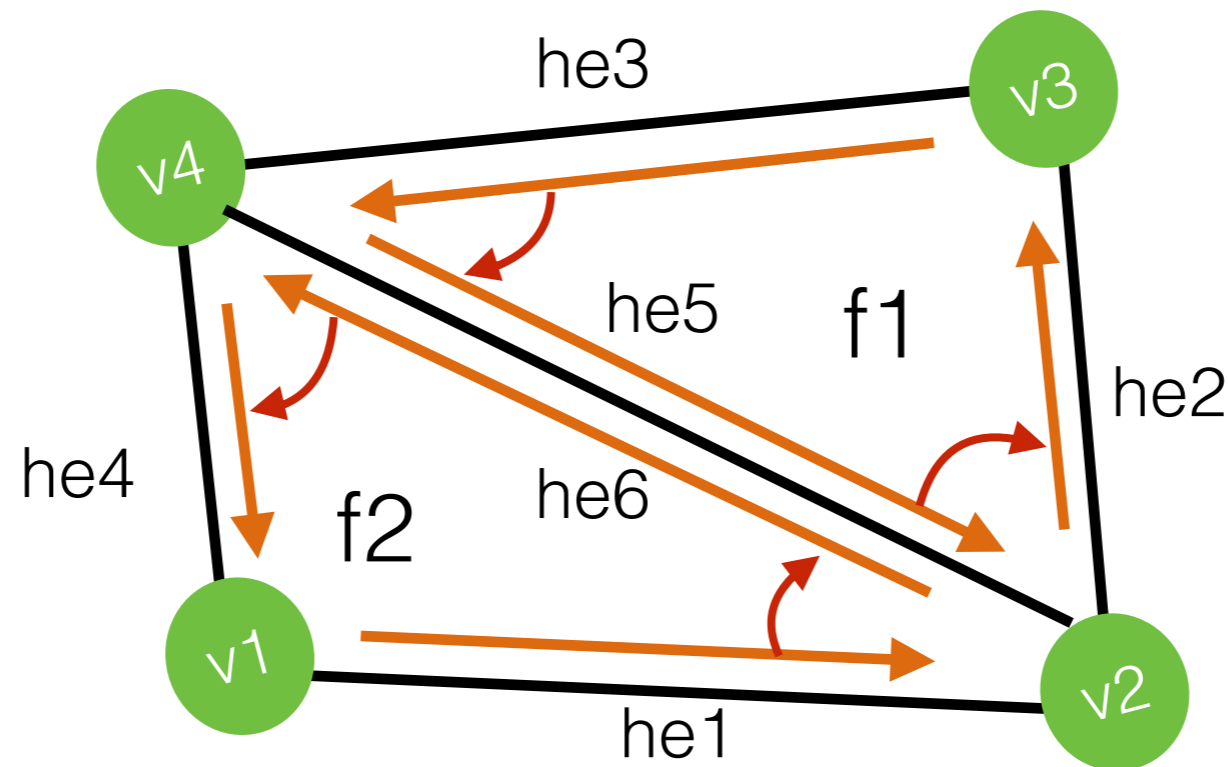
```
he3.next = he5;
```

```
he1.next = he6;
```

```
he6.next = he4;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

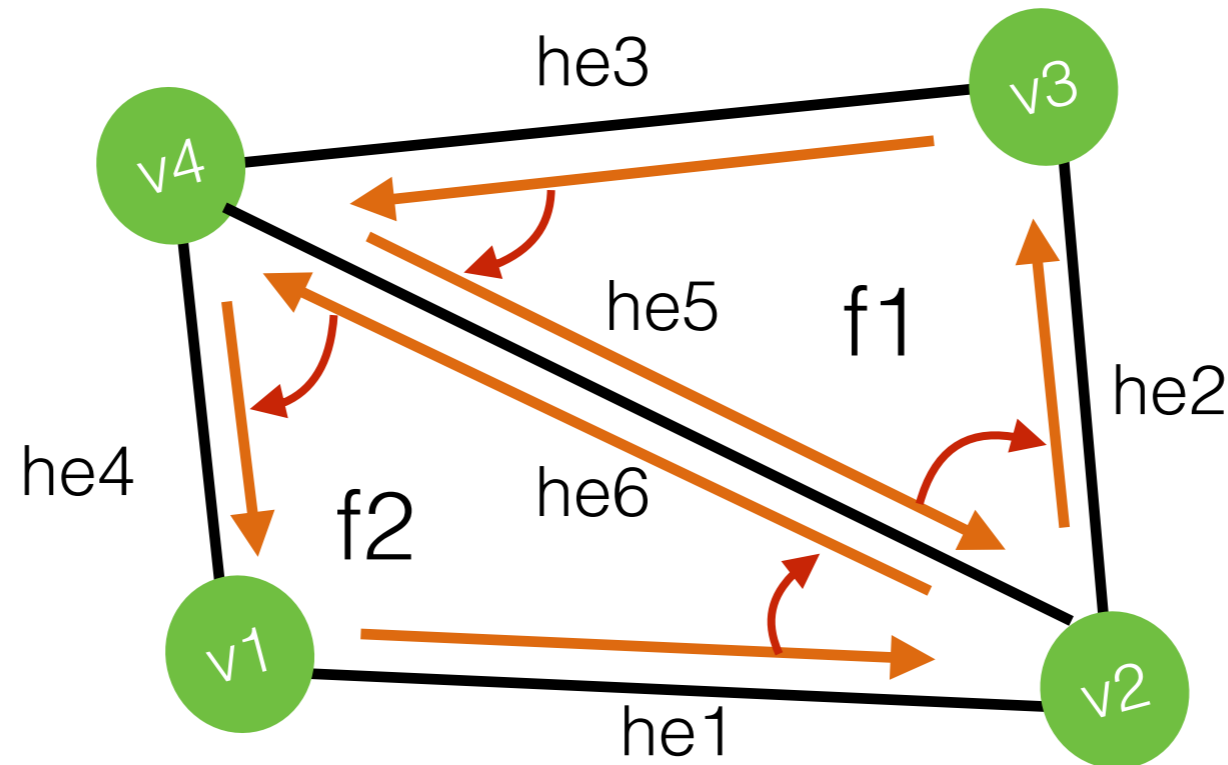
```
he3.next = he5;
```

```
he1.next = he6;
```

```
he6.next = he4;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

```
he3.next = he5;
```

```
he1.next = he6;
```

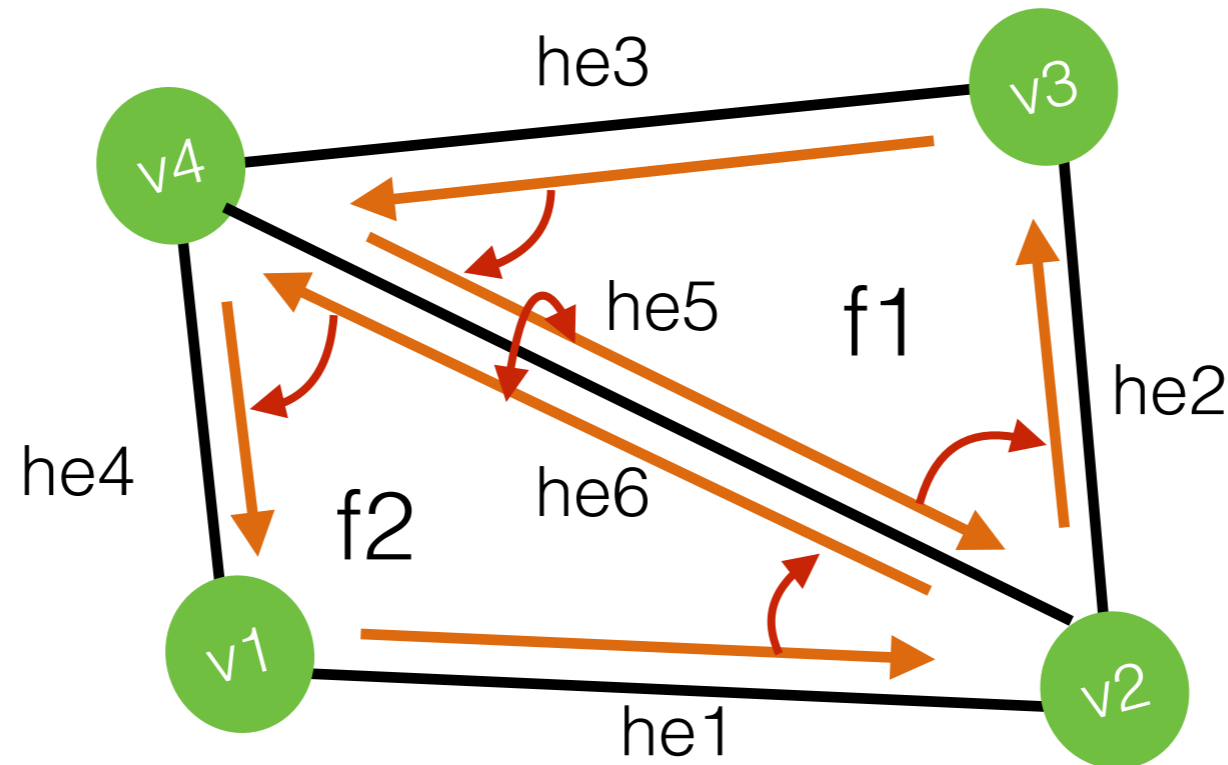
```
he6.next = he4;
```

```
f1.halfedge = he5;
```

```
f2.halfedge = he6;
```

Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

```
he3.next = he5;
```

```
he1.next = he6;
```

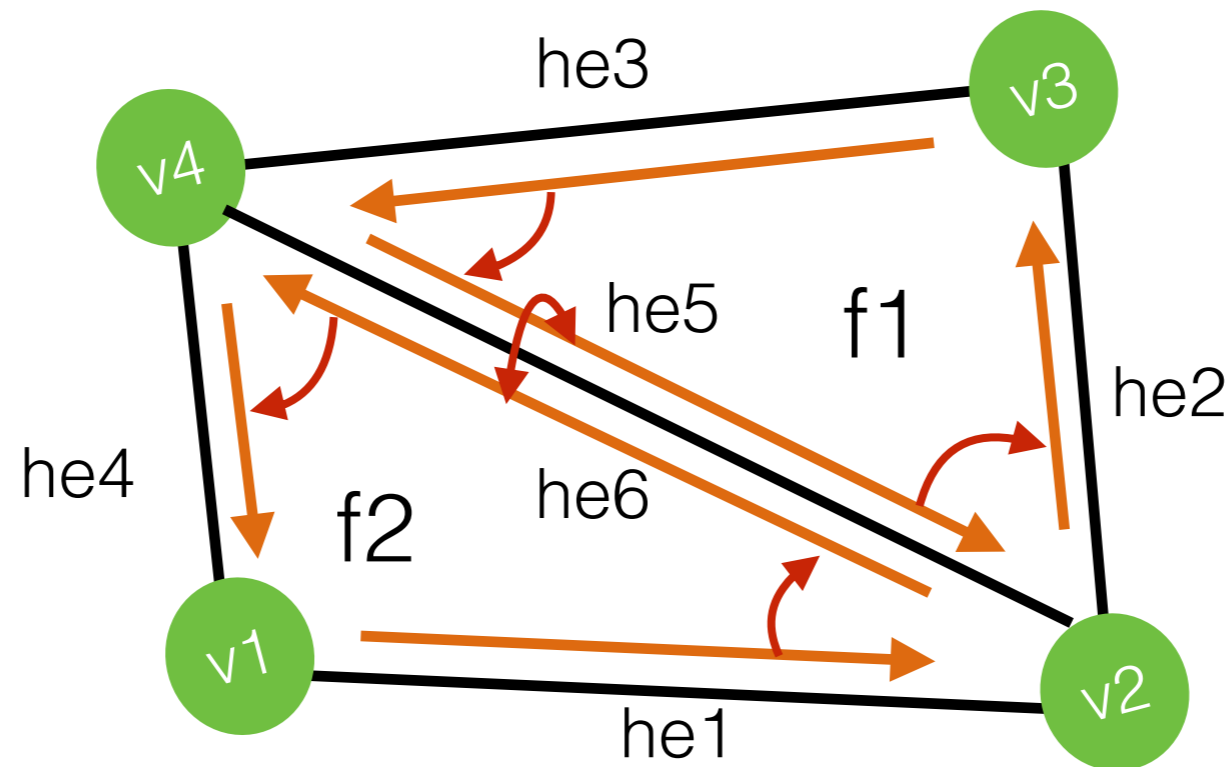
```
he6.next = he4;
```

```
f1.halfedge = he5;
```

```
f2.halfedge = he6;
```


Data Structure Modification

- Split Face



```
f2 = addFace();
```

```
he5 = addHalfEdge( v4, v2, f1 );
```

```
he6 = addHalfEdge( v2, v4, f2 );
```

```
he5.opposite = he6;
```

```
he6.opposite = he5;
```

```
he5.next = he2;
```

```
he3.next = he5;
```

```
he1.next = he6;
```

```
he6.next = he4;
```

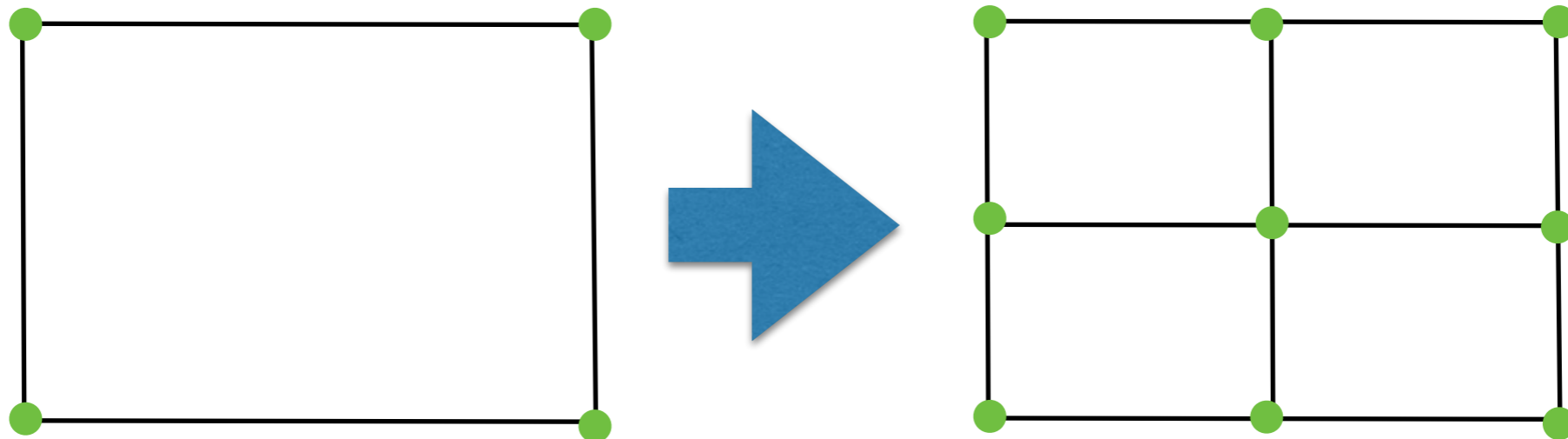
```
f1.halfedge = he5;
```

```
f2.halfedge = he6;
```

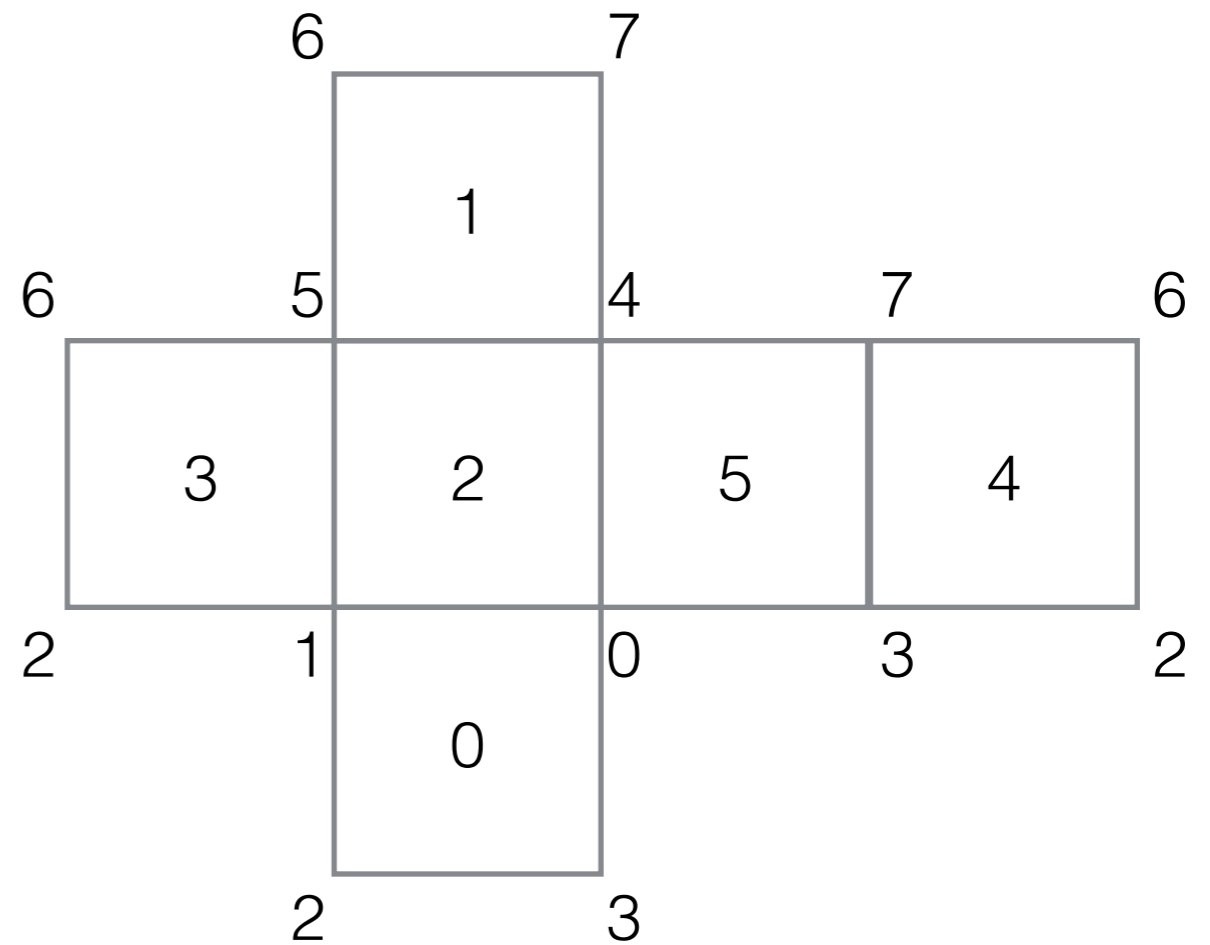
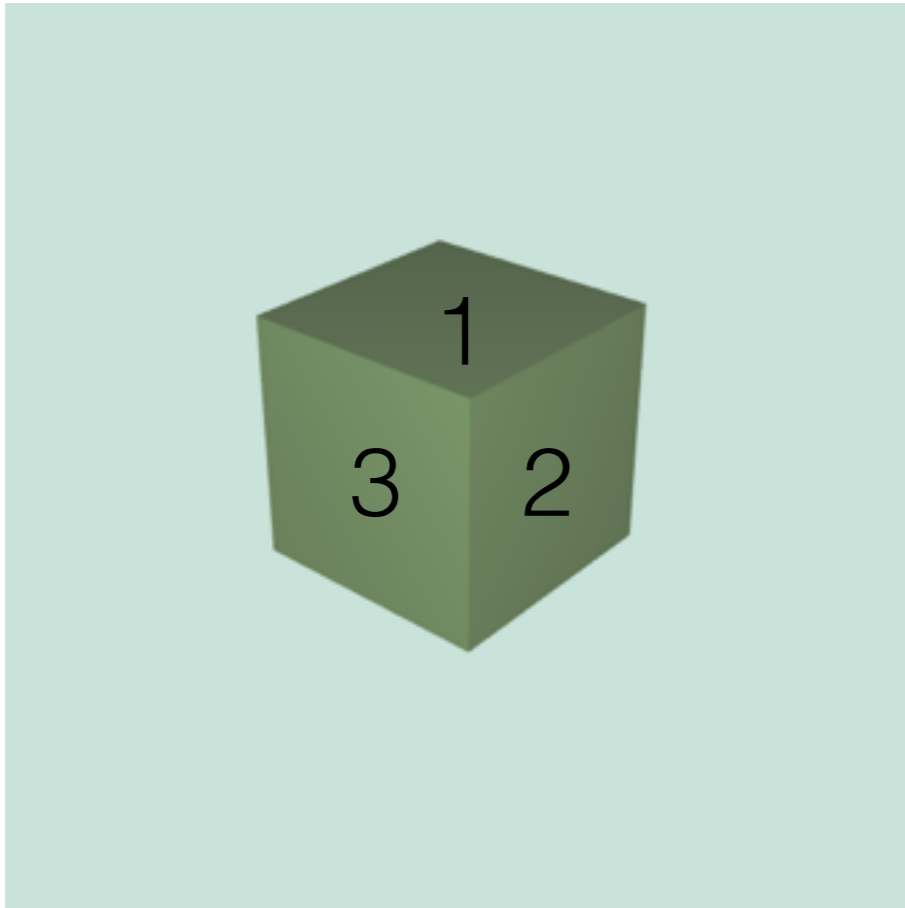
Remember to re-link he4 and he1 to point to f2

Data Structure Modification

- How would you go about subdividing a quad face?
 - You're given split edge and split face
 - Just use those - guaranteed validity of dataset after use!
- Part of the assignment
 - Think about it during tomorrows class!



Cube Cheat Sheet



Vertex neighbors:

- $0 \rightarrow \{1, 3, 4\}$
- $1 \rightarrow \{0, 2, 5\}$
- $2 \rightarrow \{1, 6, 3\}$

- $3 \rightarrow \{0, 2, 7\}$
- $4 \rightarrow \{7, 5, 0\}$
- $6 \rightarrow \{5, 2, 7\}$