CS749 Midsem, Spring 2016

(1) A kd-tree is a binary tree: each node has two children. A colleague proposes a “kd-three”: each node has 3 children. The children of a node correspond to *two* cuts along the chosen axis, instead of one.

(a) What are the advantages/disadvantages, if any, of a kd-three over a kd-tree?
(b) Given points P_n in node n, how would you choose the positions of the two cuts along an axis? You can assume without loss of generality that we're splitting along the x-axis.
(c) A diplomatic colleague proposes a hybrid structure, where a node can have either two children (like a kd-tree) or three children (like a kd-three). As the tree is constructed, we choose whether to split each internal node into two parts or three parts. Propose and justify a method to make this choice.

(2) You have a mesh stored with full and explicit adjacency information, using the following setup (cf slide 11 of 05_meshes.pdf):

```
class Vertex {
  Vec3 position;
  Vec3 normal;

  list<Face *> faces;
  list<Edge *> edges;
};
```

```
class Edge {
  double length;

  Vertex * endpoints[2];
      // ^^^ unordered

  list<Face *> faces;
};
```

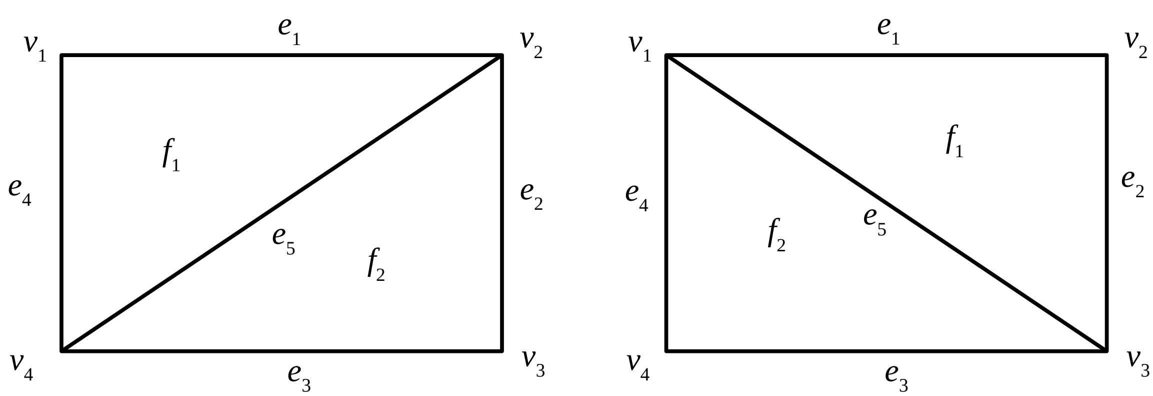
```
class Face {
  Vec3 normal;

  // Invariant:
  // vertices[i] =
  //   edges[i]->endpoint[0 or 1]
  list<Vertex *> vertices;
  list<Edge *> edges;
};
```

(Constructors, accessors and other functions omitted)

```
Mesh = [ list<Vertex>, list<Edge>, list<Face> ]
```

Now the mesh undergoes an edge flip operation:



Before   After

Write the minimum sequence of operations needed to update all the pointers in vertices v1, v2, v3, v4, faces f1, f2, and edges e1, e2, e3, e4, e5. (Don't bother about fixing the normals and lengths.) Assume the "list<T *>" class has the following functions:

```
void erase(T * t);       // removes t from list
void push_back(T * t);   // adds t to end of list
```

(3) The gradient matrix *A* we constructed for Poisson surface reconstruction has a boundary problem (slide 14 of 07_reconst.pdf). The last row has just one non-zero entry (-1), thus the derivative at the end of the range is not correctly modeled. Let's say we fix this problem by setting the first element of the last row to 1 (so the last row is "1 0 0 … 0 -1").

(a) For what sort of 1D domain is this the accurate discrete gradient?
(b) Is this matrix invertible? If yes, what is the inverse? If not, why not?

(4) Given an input mesh *M*, mesh optimization [Hoppe '92] can theoretically produce a better final simplification of *M* than a sequence of edge collapses guided by quadric error metrics [Garland '97]. Why, then, is the latter often preferred in practice? List as many reasons as you can think of.