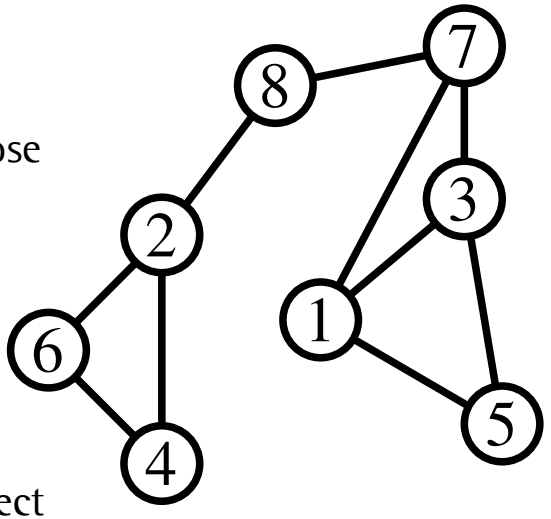
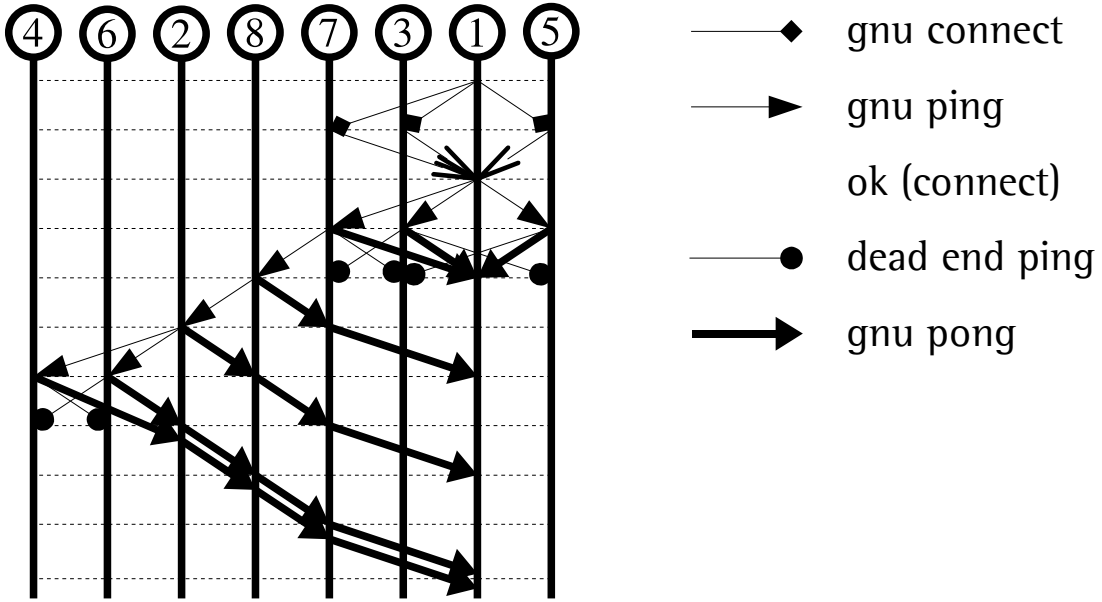
Peer-to-peer networks – (due till April 29, 2009)

Exercise 7.1: Repetition of the Gnutella protocol

Node 1 joins the network shown on the right hand side. Its cache contains its direct neighbors 3, 5 and 7. With those nodes, it establishes connection immediately. Then, the network is flooded.



Find all messages which traverse the given network according to the Gnutella protocol.

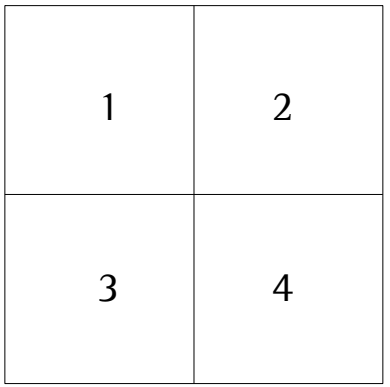


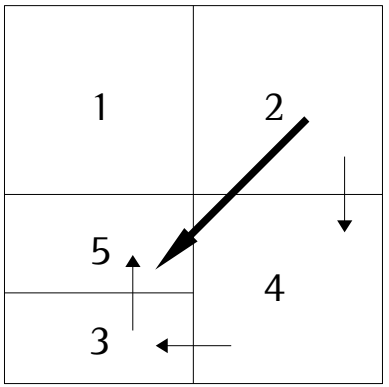
Peer-to-peer networks

Exercise 7.2: CAN

a) The idea of CAN is to navigate a query through an n-dimensional hyper cube. Given an entry-point within a cube and a target coordinate: As we observe the mid-points of the cuboids visited along the journey of a query, are the coordinates either always descending or ascending monotonously or is it possible that the direction changes with regard to one dimension?

In other words, can it happen that e.g., the x-coordinates are falling and rising within a single journey throughout the network? Find a prove that this can not happen or an example in which it actually happens.





Peer-to-peer networks

Exercise 7.2: CAN

b) Implement a member function node& findNeighbor(...) of the class CanNode in Java, C or pseudocode which chooses the next neighbor. Find an appropriate signature (input values) for the function and determine which essential fields the class CanNode needs. We assume that our knowledge about neighboring nodes is limited to routing information.

```
class Node {
   public:      // DATA
      final int DIMENSIONS = 5;
      int       min[DIMENSIONS], max[DIMENSIONS];
      node&     left_neighbor[DIMENSIONS];
      node&     right_neighbor[DIMENSIONS];

   public:      // FUNCTIONS
      node&     findNeighbor(long[] hash);

}; // class Node

node& Node::findNeighbor(long[] hash)
{
   for(int i = 0; i < DIMENSIONS; i++) {

      if(hash[i] < min[i])
         return left_neighbor[i].findNeighbor(hash);

      if(hash[i] > max[i])
         return right_neighbor[i].findNeighbor(hash);
   } // for

   return this; // we are the best!
} // findNeighbor
```

Peer-to-peer networks

Exercise 7.4: Pastry

The ID of a specific node is 35075. Our digits range from 0-7 and each hash value is 5 digits long. Fill in the following table with arbitrary values which have to meet the pastry protocol. You can assume that there are enough neighbors to fill in the entire table.

```
0: 02345 12345 22345 42345 52345 62346 72356
1: 30345 31345 32345 33345 34345 36346 37356
2: 35145 35245 35345 35445 35545 35656 35756
3: 35005 35014 35025 35035 35045 35056 35066
4: 35070 35071 35072 35073 35074 35076 35077
```