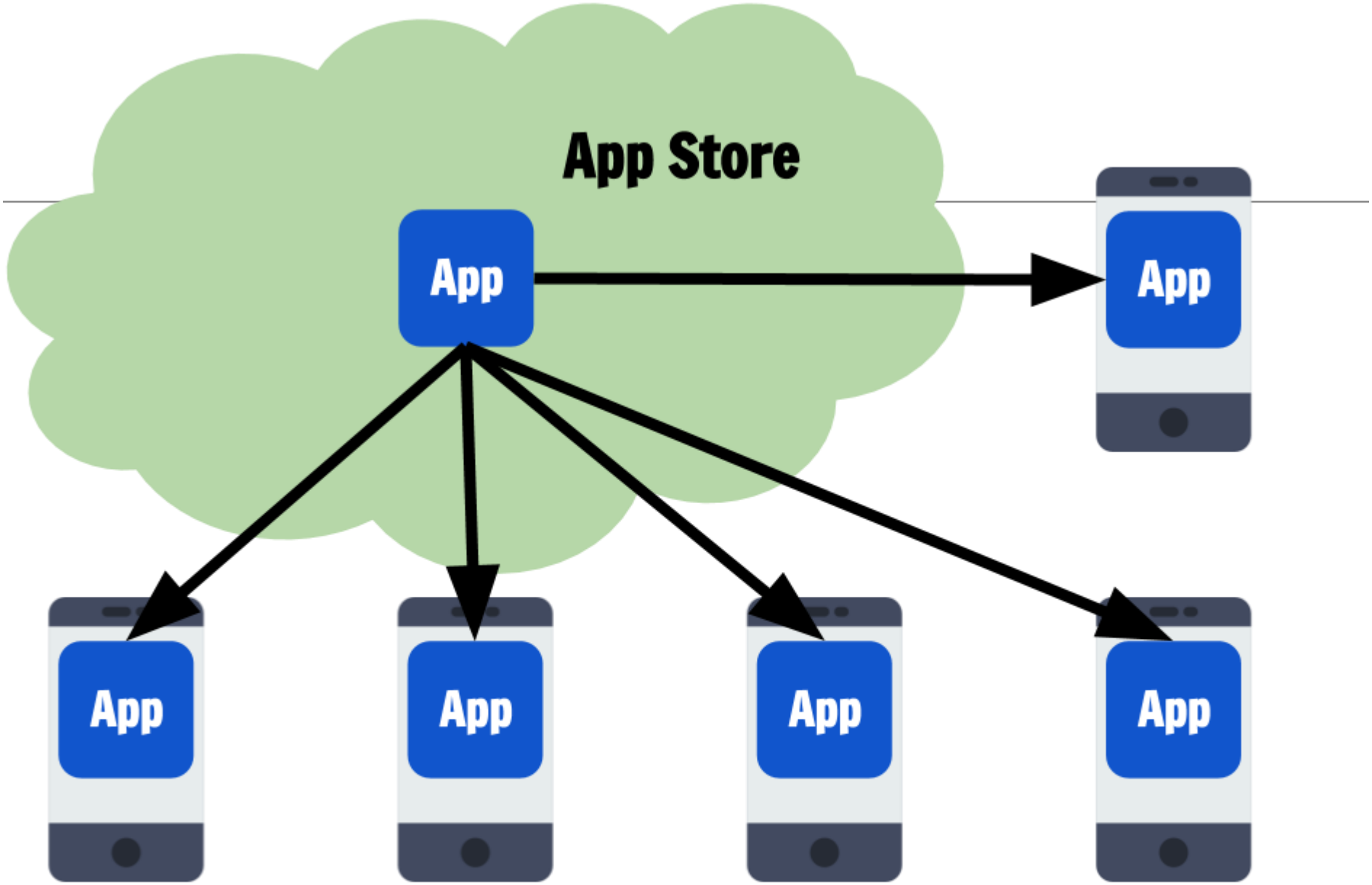
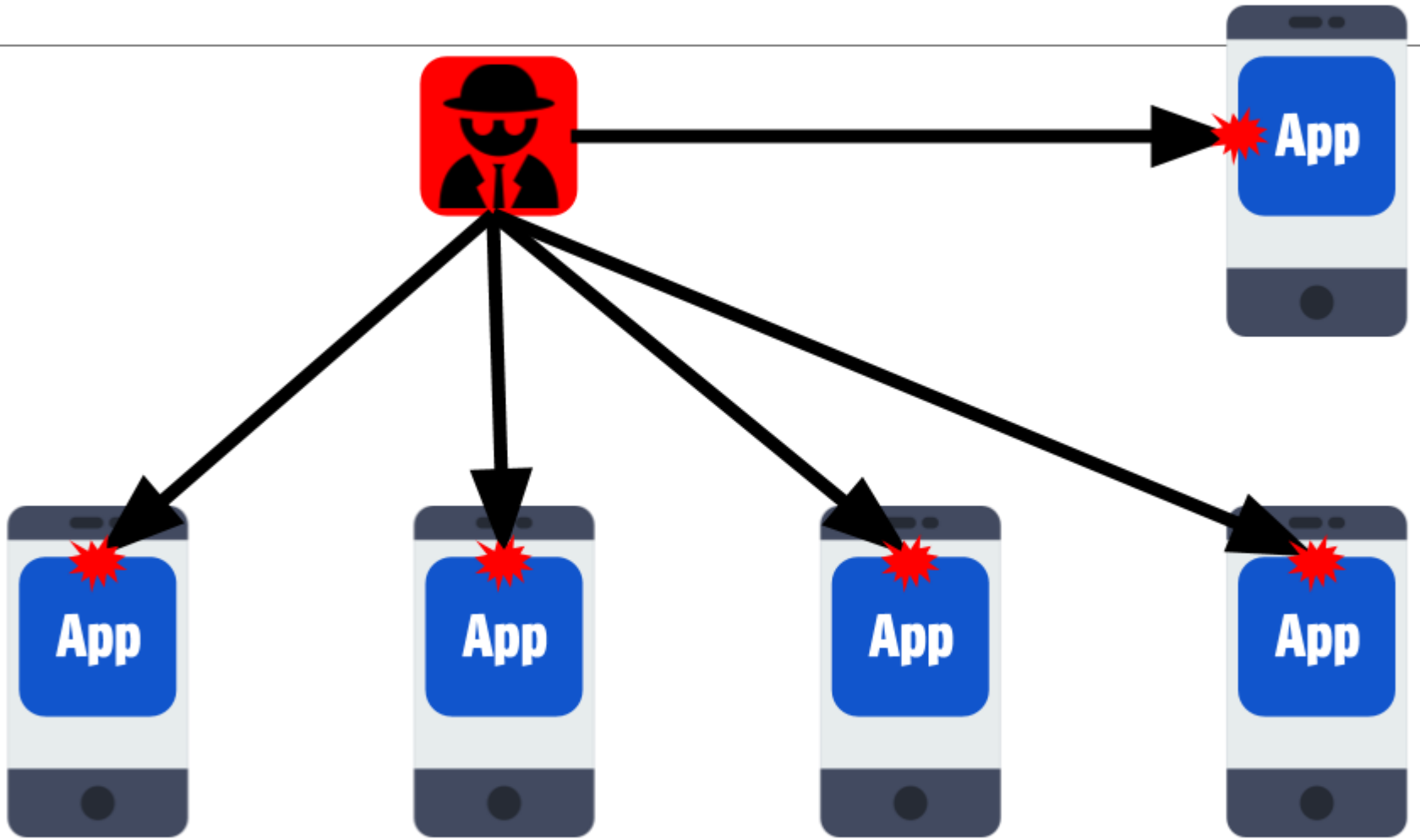
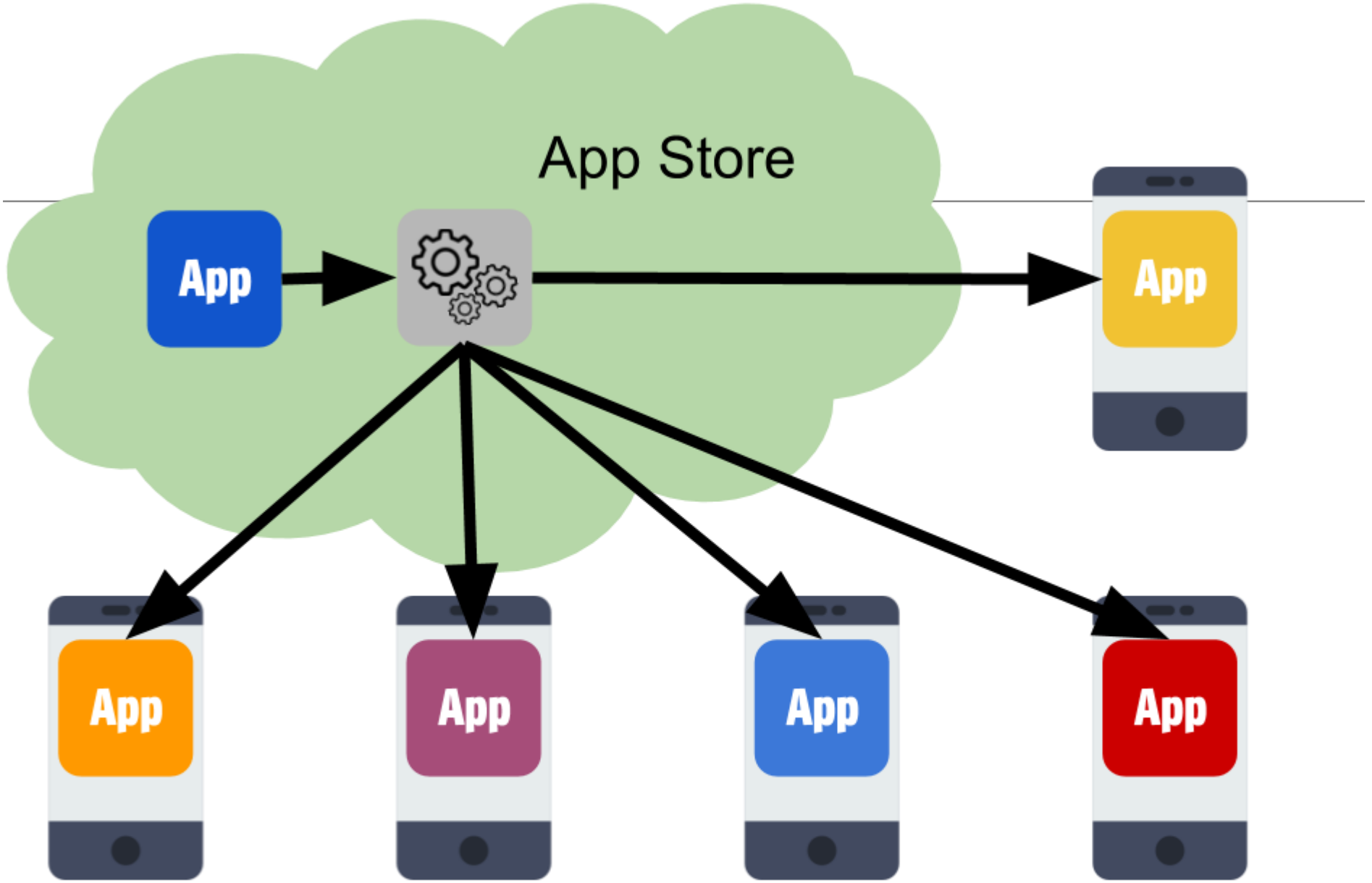
Software diversification as an obfuscation technique

Benoit Baudry, Nicolas Harrand

INRIA, France


App Store
App
App
App
App
App
App


App
App
App
App
App


App Store
App
App
App
App
App
App

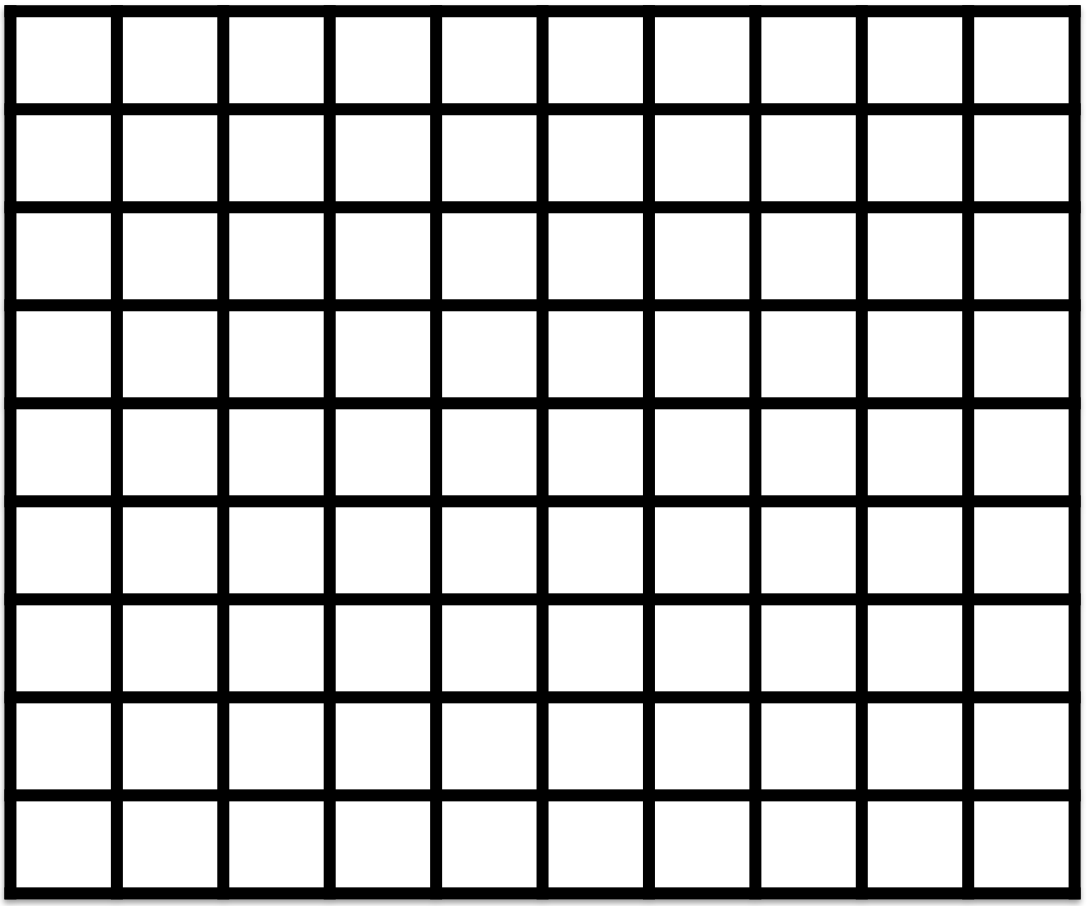
Challenges for automatic diversity

1. Foundational paradox
   - Software brittleness
2. Trade software correctness and diversity

$$\tau(P) \equiv P \wedge \delta(\tau(P), P) \neq 0$$

3. Explore the space of transformations
   - Where and how to transform
4. Quantify diversity and its impact
   - Quantity of diversity
   - Effectivness for obfuscation

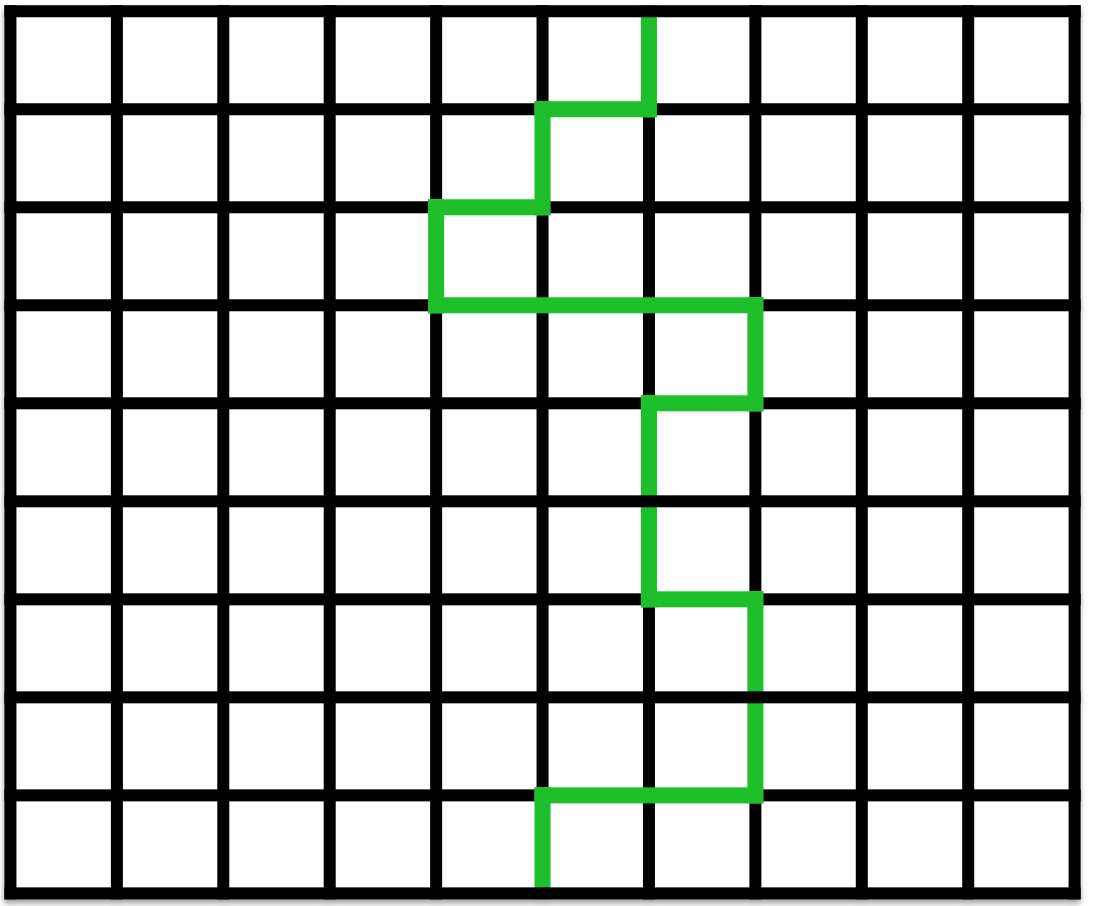
Software brittleness



SRSLSLRSRLLSSRRLRL

G. Berry. « A la chasse aux bugs, la maladie du certain » (8 juin 2011)

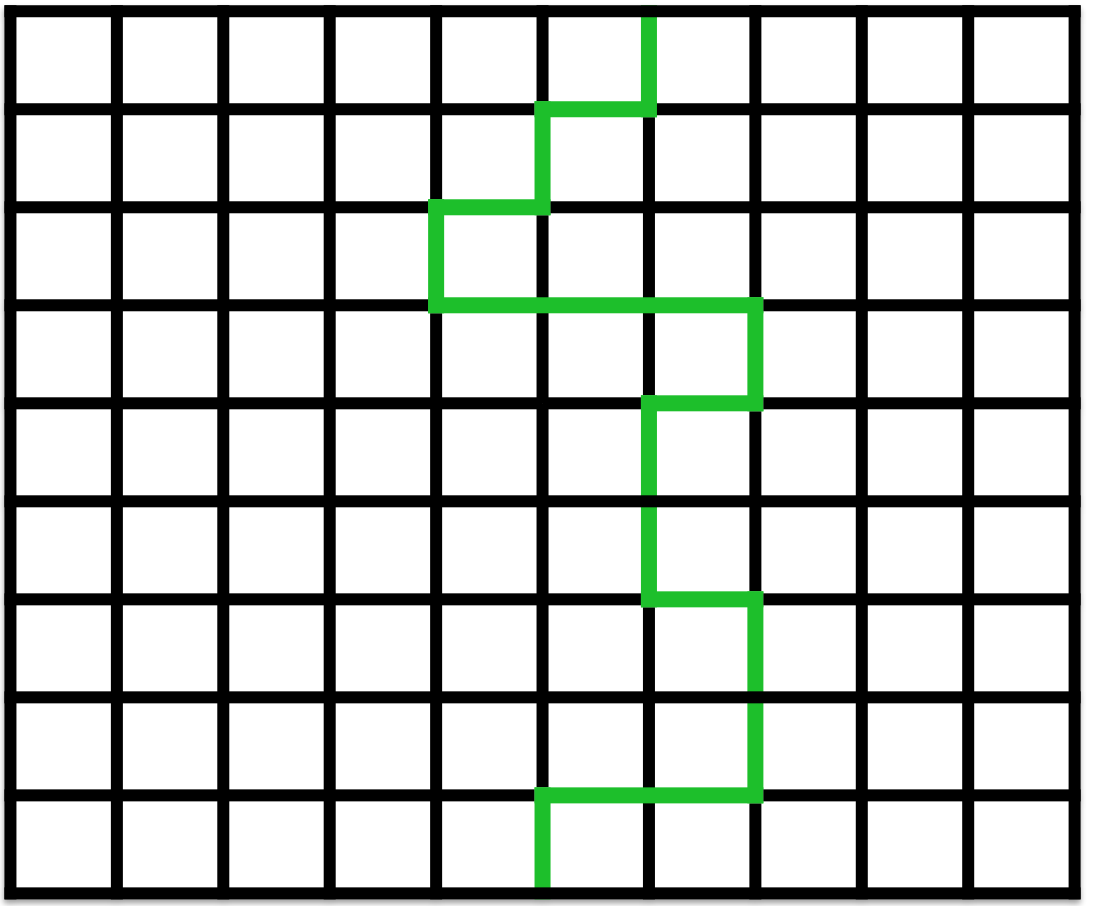
Software brittleness



SRSLSLRSRLLSSRRLRL

G. Berry. « A la chasse aux bugs, la maladie du certain » (8 juin 2011)

Software brittleness

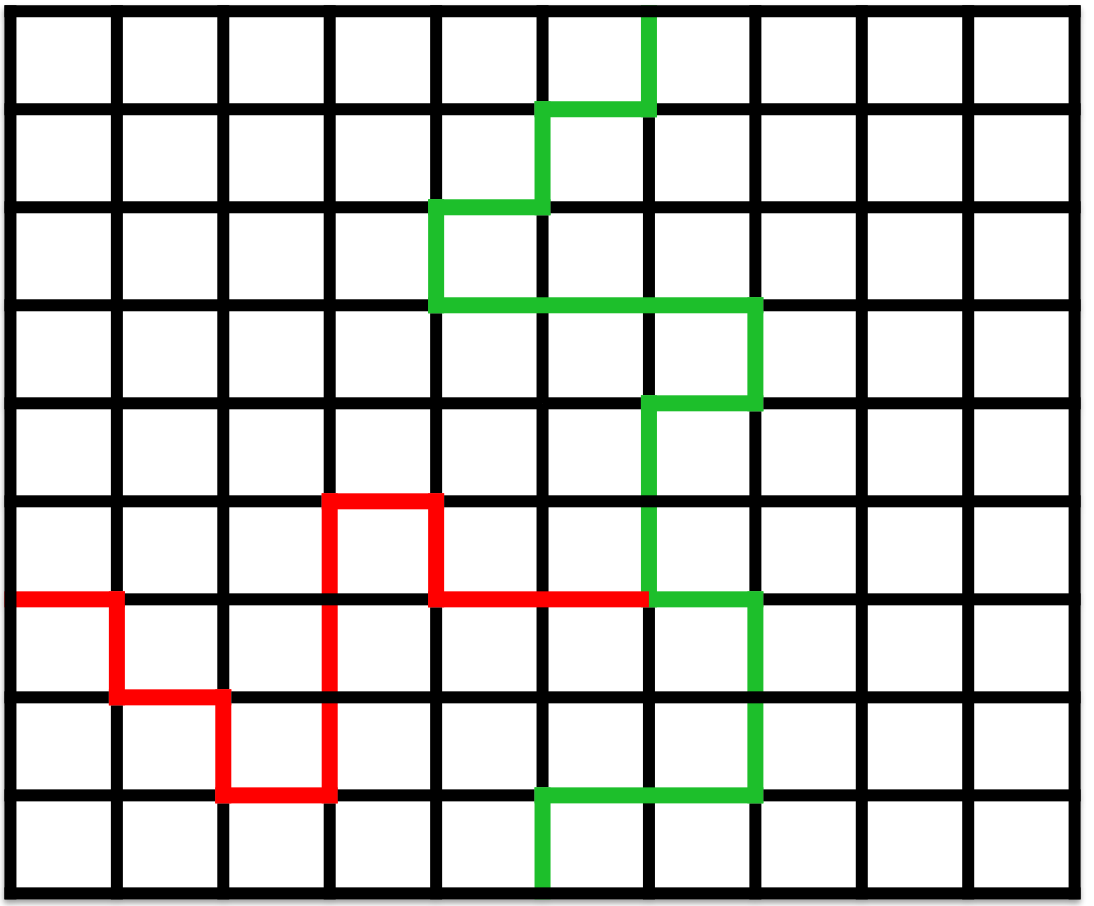


SRSLSLRSRLLSSRRLRL

SRSLSLSSRLLSSRRLRL

G. Berry. « A la chasse aux bugs, la maladie du certain » (8 juin 2011)

Software brittleness

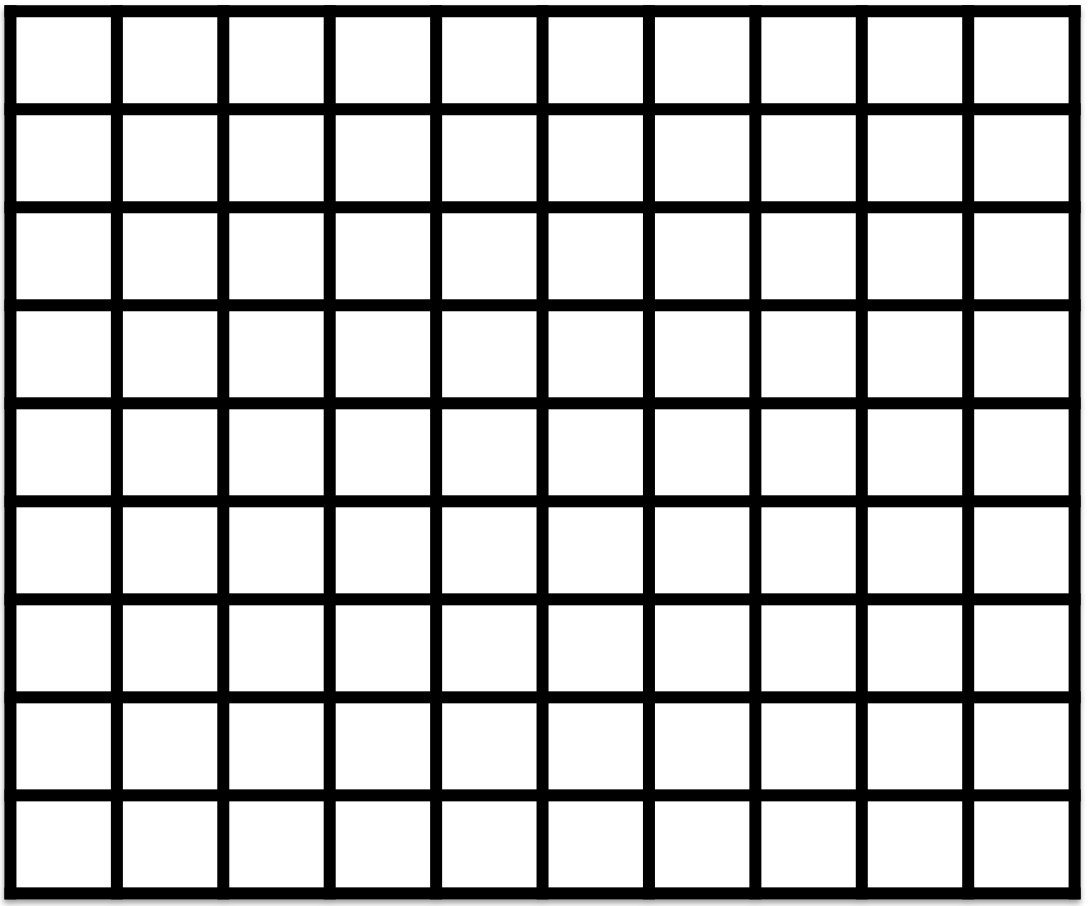


SRSLSLRSRLLSSRRLRL

SRSLSLSSRLLSSRRLRL

G. Berry. « A la chasse aux bugs, la maladie du certain » (8 juin 2011)

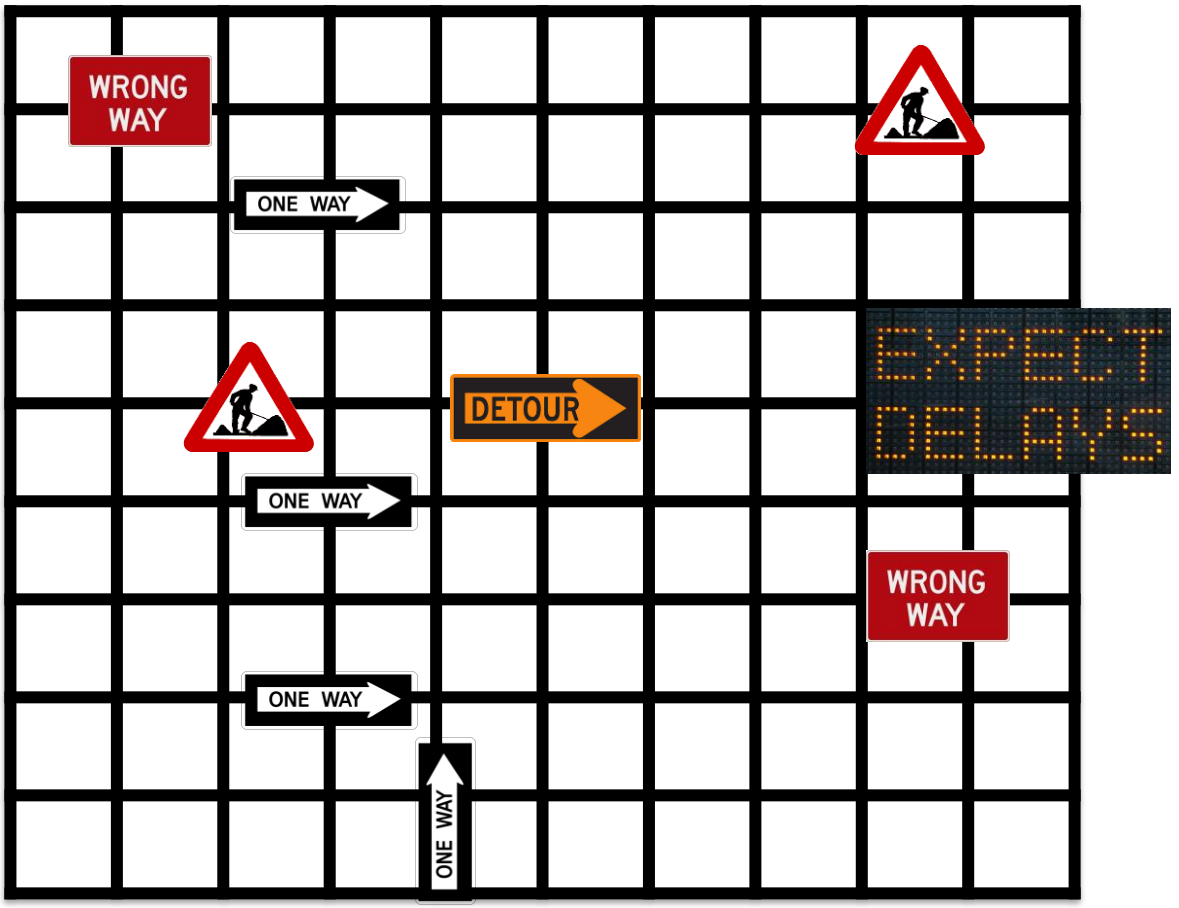
Software brittleness



SRSLSLRSRLLSSRRLRL

SRSLSLSSRLLSSRRLRL

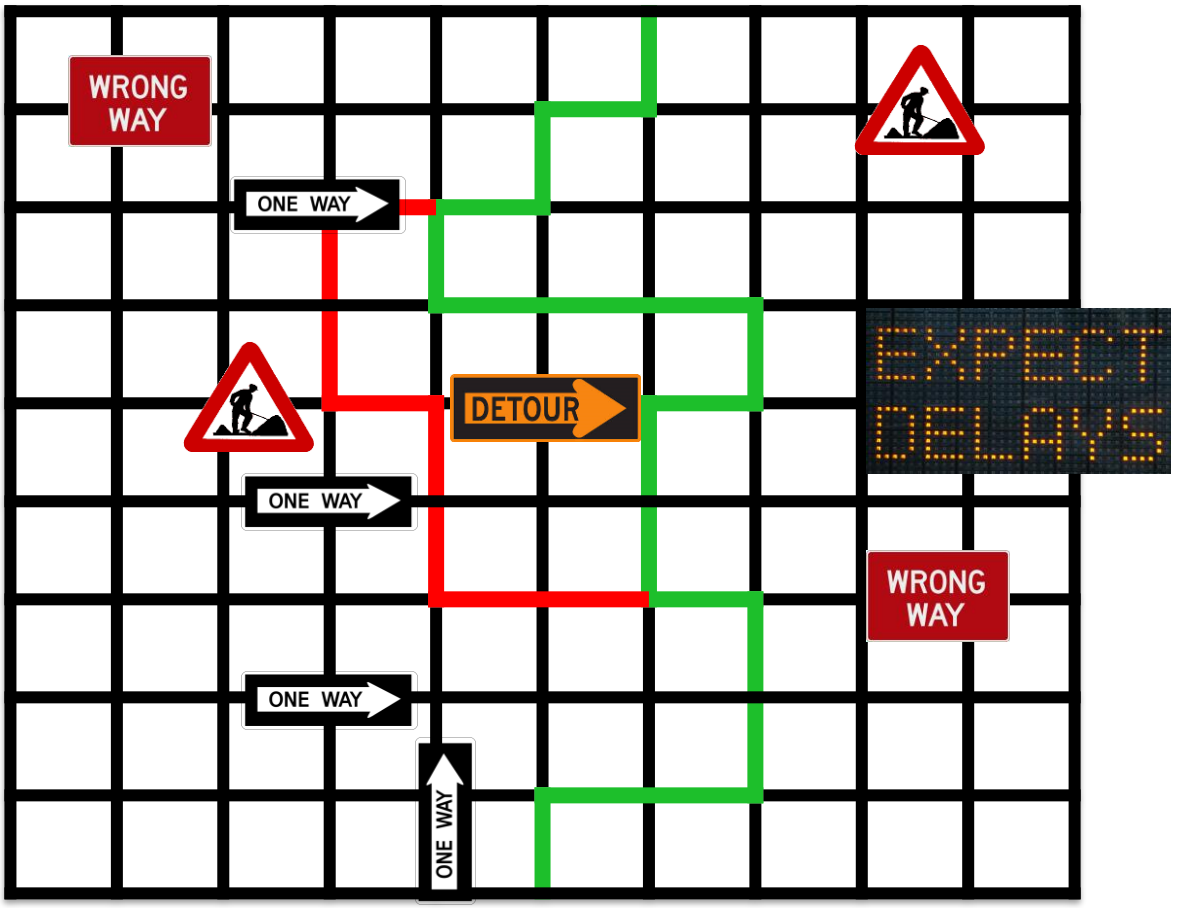
Software plasticity?



SRSLSLRSRLLSSRRLRL

SRSLSLSSRLLSSRRLRL

Software plasticity?



SRSLSLRSRLLSSRRLRL

SRSLSLSSRLLSSRRLRL

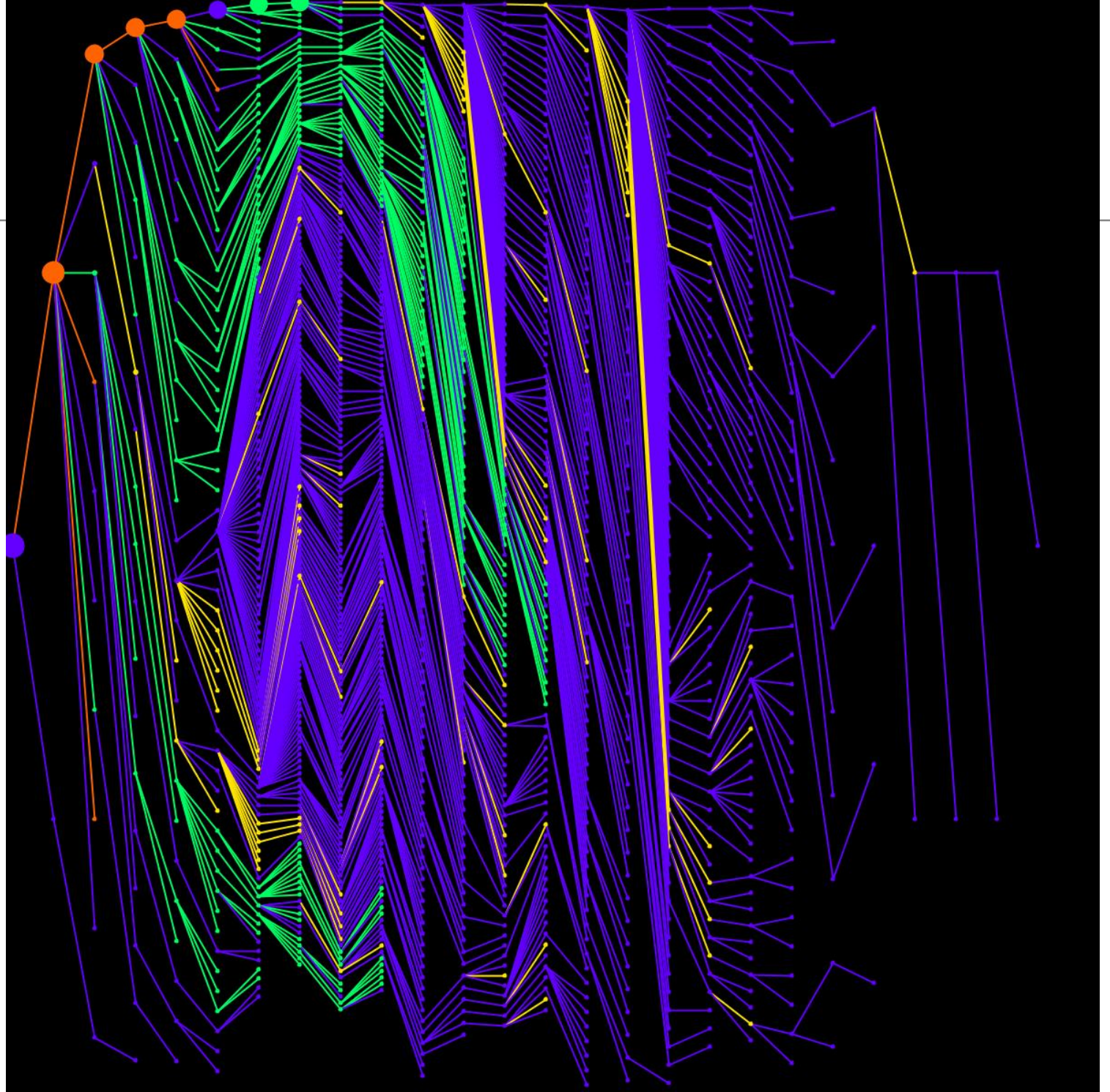
An example

```
import org.apache.commons.collections.list.TreeList;
public class QuickSort {
   TreeList sort(TreeList arr) {
      int pivot = 0;
      TreeList less = new TreeList ();
      TreeList pivotList = new TreeList ();
      TreeList more = new TreeList ();
      for(int i=1;i<arr.size();i++)  {
         if ((Integer)arr.get(i) < ((Integer)arr.get(pivot))) {
            less.add(arr.get(i));
         }
         else if ((Integer)arr.get(i) > ((Integer)arr.get(pivot)))
            more.add(arr.get(i));
            else
               pivotList.add(arr.get(i));
         }
      pivotList.add(arr.get(pivot));
      less = sort(less);
      more = sort(more);
      less.addAll(pivotList);
      less.addAll(more);
      return less;
    }
```

An example

```
import org.apache.commons.collections.list.TreeList;
public class QuickSort {
   TreeList sort(TreeList arr) {
      int pivot = 0;
      TreeList less = new TreeList ();
      TreeList pivotList = new TreeList ();
      TreeList more = new TreeList ();
      for(int i=1;i<arr.size();i++)  {
         if ((Integer)arr.get(i) < ((Integer)arr.get(pivot))) {
            less.add(arr.get(i));
         }
         else if ((Integer)arr.get(i) > ((Integer)arr.get(pivot)))
            more.add(arr.get(i));
            else
               pivotList.add(arr.get(i));
         }
      pivotList.add(arr.get(pivot));
      less = sort(less);
      more = sort(more);
      less.addAll(pivotList);
      less.addAll(more);
      return less;
    }
```

sort([1,4,2,7,9,3,7,4,0])

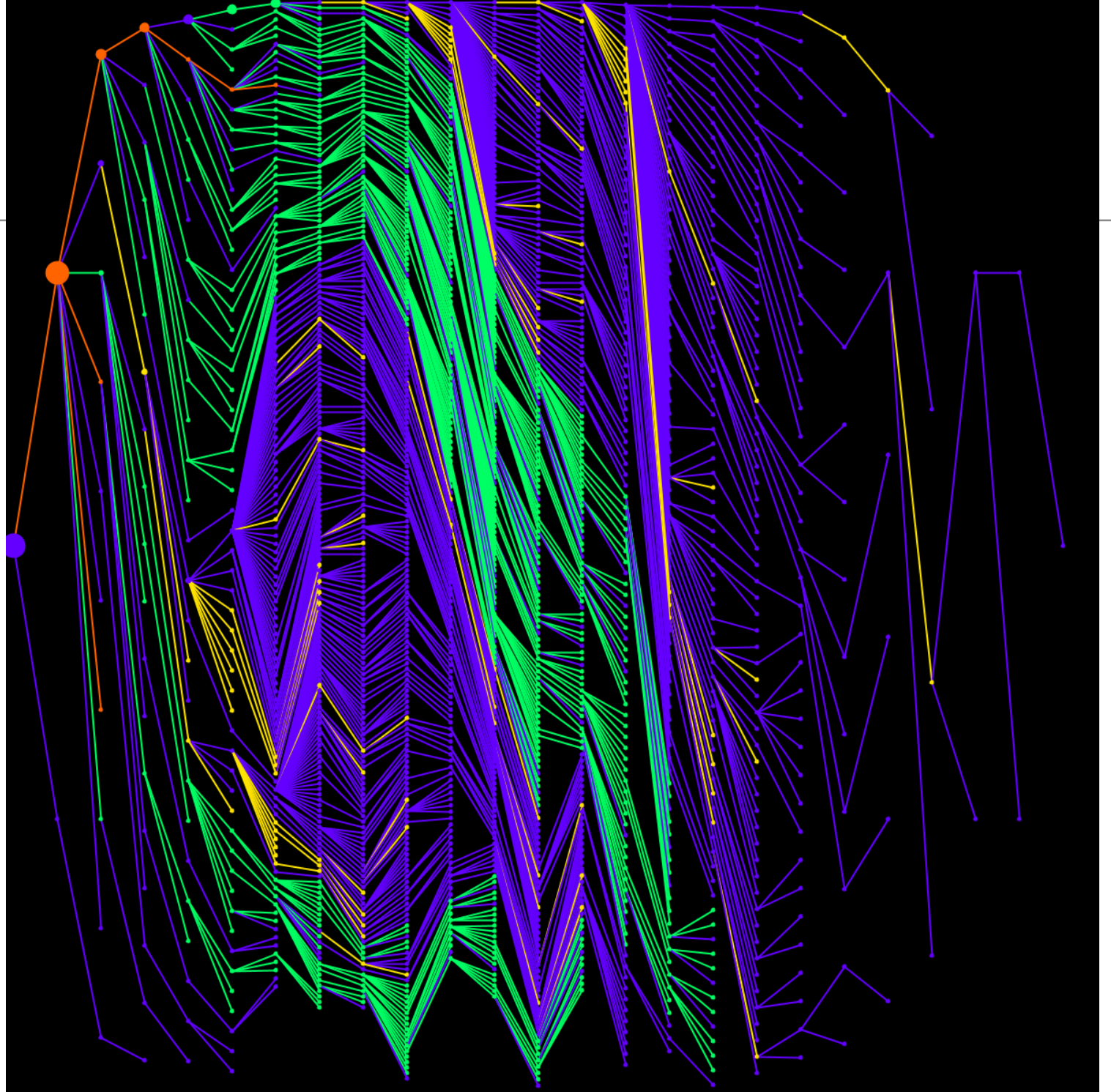


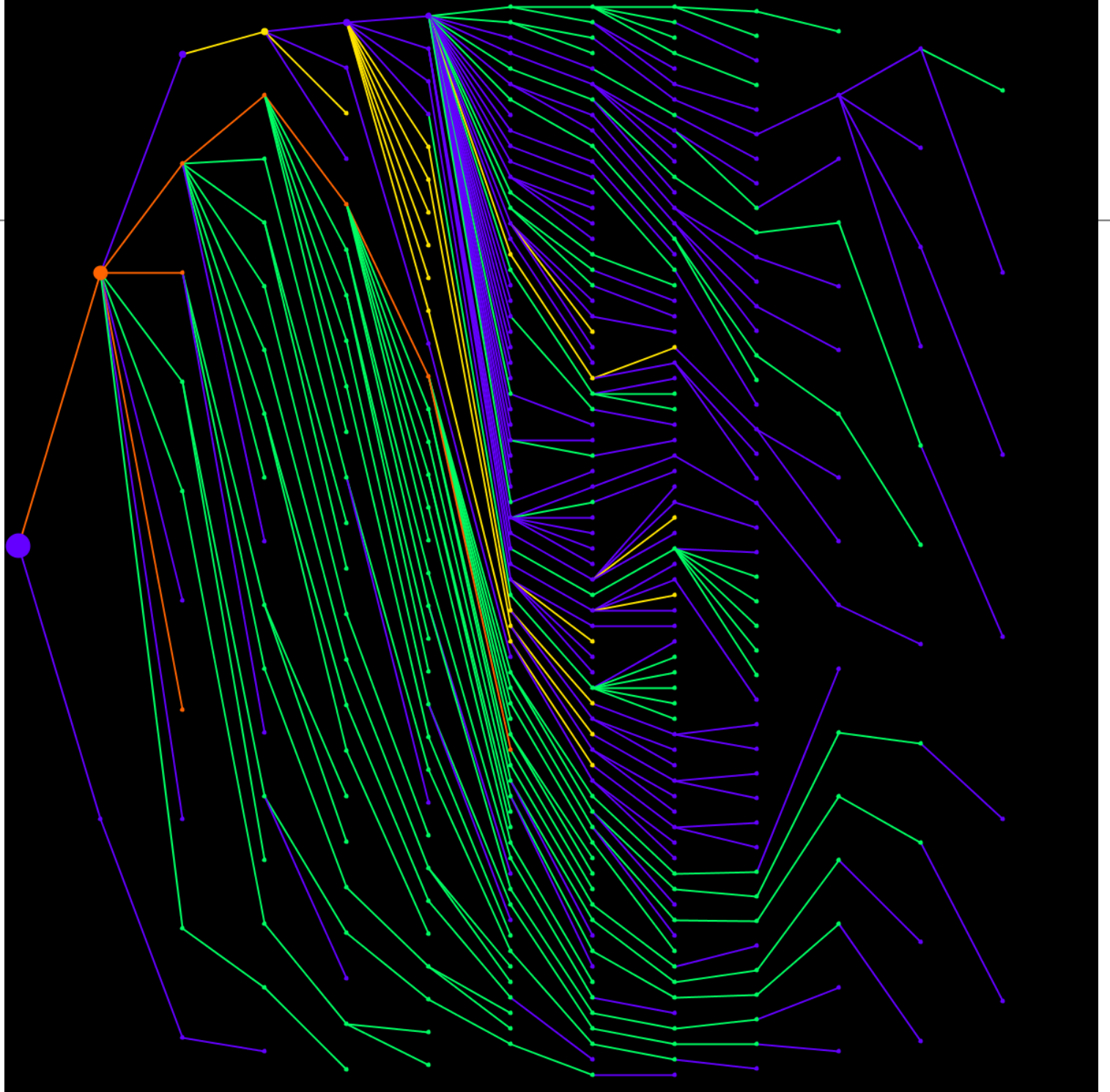
Code diversification

```
import org.apache.commons.collections.list.TreeList;
public class QuickSort {
   TreeList sort(TreeList arr) {
      int pivot = 0;
      TreeList less = new TreeList ();
      TreeList pivotList = new TreeList ();
      TreeList more = new TreeList ();
      for(int i=1;i<arr.size();i++)  {
         if ((Integer)arr.get(i) < ((Integer)arr.get(pivot))) {
            less.add(arr.get(i));
         }
         else if ((Integer)arr.get(i) > ((Integer)arr.get(pivot)))
            more.add(arr.get(i));
            else
               pivotList.add(arr.get(i));
         }
      pivotList.add(arr.get(pivot));
      less = sort(less);
      more = sort(more);
      less.addAll(pivotList);
      less.addAll(more);
      return less;
    }
```

Code diversification

```
import java.util.ArrayList;
import java.util.List;
public class QuickSort {
   List sort(List arr) {
      int pivot = 0;
      List less = new ArrayList();
      List pivotList = new ArrayList();
      List more = new ArrayList();
      for(int i=1;i<arr.size();i++)  {
         if ((Integer)arr.get(i) < ((Integer)arr.get(pivot))) {
            less.add(arr.get(i));
         }
         else if ((Integer)arr.get(i) > ((Integer)arr.get(pivot)))
            more.add(arr.get(i));
            else
               pivotList.add(arr.get(i));
         }
      pivotList.add(arr.get(pivot));
      less = sort(less);
      more = sort(more);
      less.addAll(pivotList);
      less.addAll(more);
      return less;
    }
```



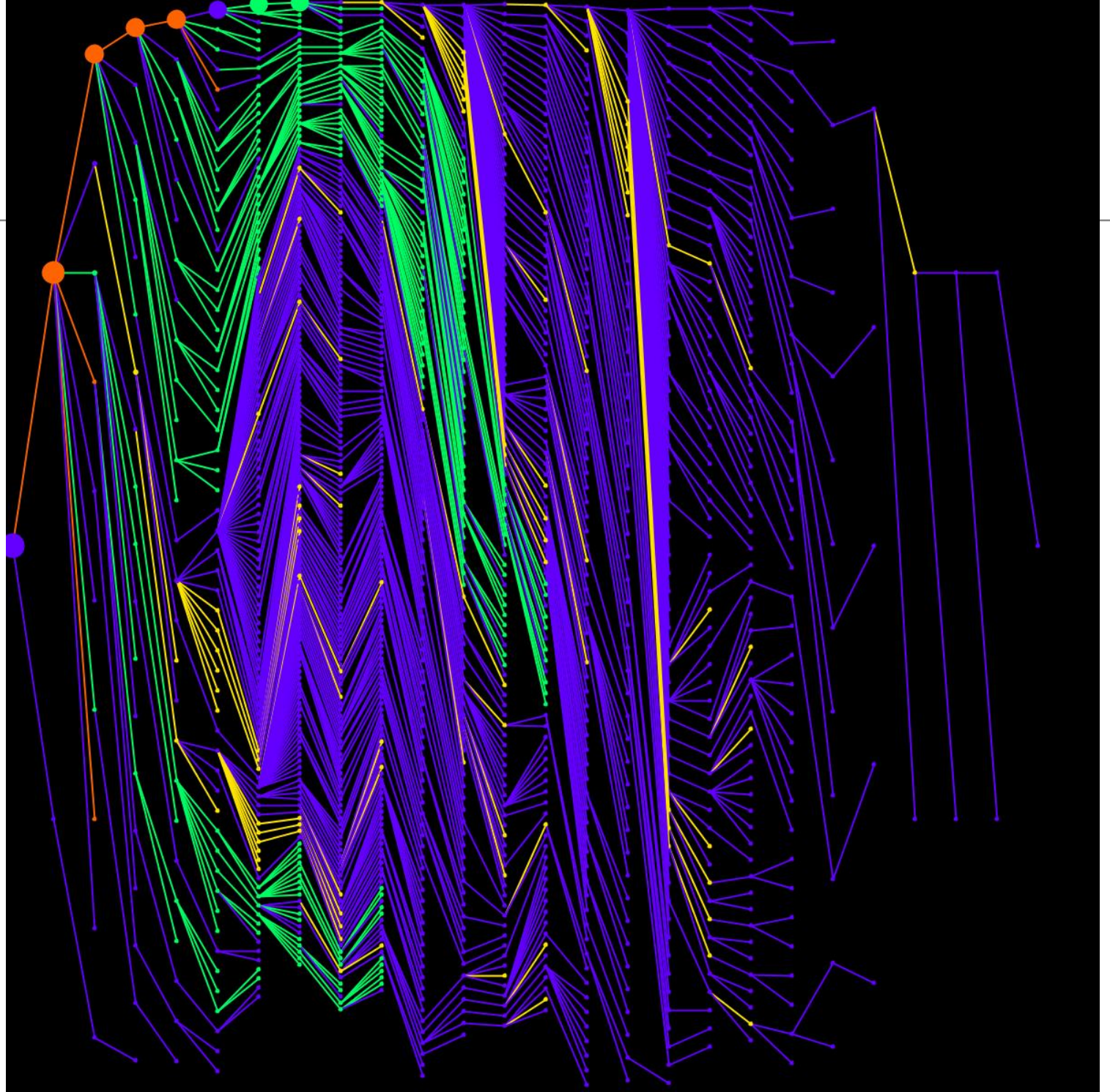


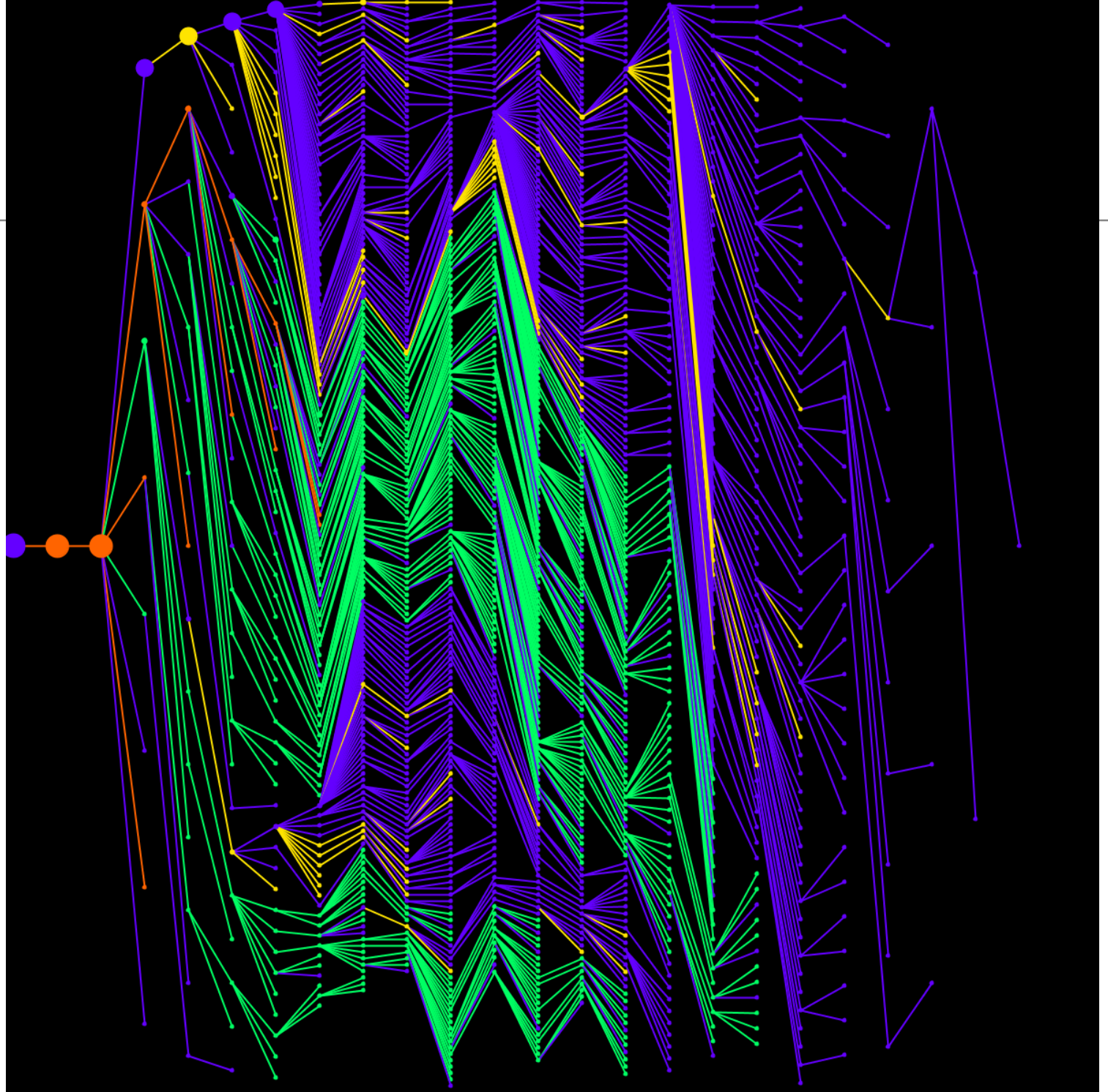
Code diversification

```
import org.apache.commons.collections.list.TreeList;
public class QuickSort {
   TreeList sort(TreeList arr) {
      int pivot = 0;
      TreeList less = new TreeList ();
      TreeList pivotList = new TreeList ();
      TreeList more = new TreeList ();
      for(int i=1;i<arr.size();i++)  {
         if ((Integer)arr.get(i) < ((Integer)arr.get(pivot))) {
            less.add(arr.get(i));
         }
         else if ((Integer)arr.get(i) > ((Integer)arr.get(pivot)))
            more.add(arr.get(i));
            else
               pivotList.add(arr.get(i));
         }
      pivotList.add(arr.get(pivot));
      less = sort(less);
      more = sort(more);
      less.addAll(pivotList);
      less.addAll(more);
      return less;
    }
```

Code diversification

```
import org.apache.commons.collections.list.TreeList;
public class QuickSort {
   TreeList sort(TreeList arr) {
      int pivot = 0;
      TreeList less = new TreeList ();
      TreeList pivotList = new TreeList ();
      TreeList more = new TreeList ();
      for(int i=1;i<arr.size();i++)  {
         if ((Integer)arr.get(i) < ((Integer)arr.get(pivot))) {
            less.add(arr.get(i));
         }
         else if ((Integer)arr.get(i) > ((Integer)arr.get(pivot)))
            more.add(arr.get(i));
            else
               pivotList.add(arr.get(i));
         }
      isSorted(less);
      pivotList.add(arr.get(pivot));
      less = sort(less);
      more = sort(more);
      less.addAll(pivotList);
      less.addAll(more);
      return less;
    }
```





Conclusion

- Software diversity
  - is based on exact and approximate transformations
  - is an obfuscation technique
- How can we measure
  - computation diversity?
  - impact on protection?

B. Baudry, M. Monperrus. « The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond ». ACM Comp. Surveys, 2015.

B. Baudry, S. Allier, M. Monperrus. « Tailored source code transformations to synthesize computationally diverse program variants ». ISSTA, 2014.