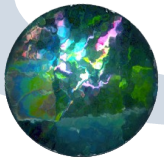
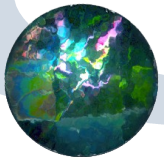
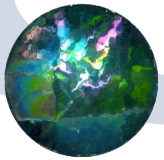


グラフの探索2



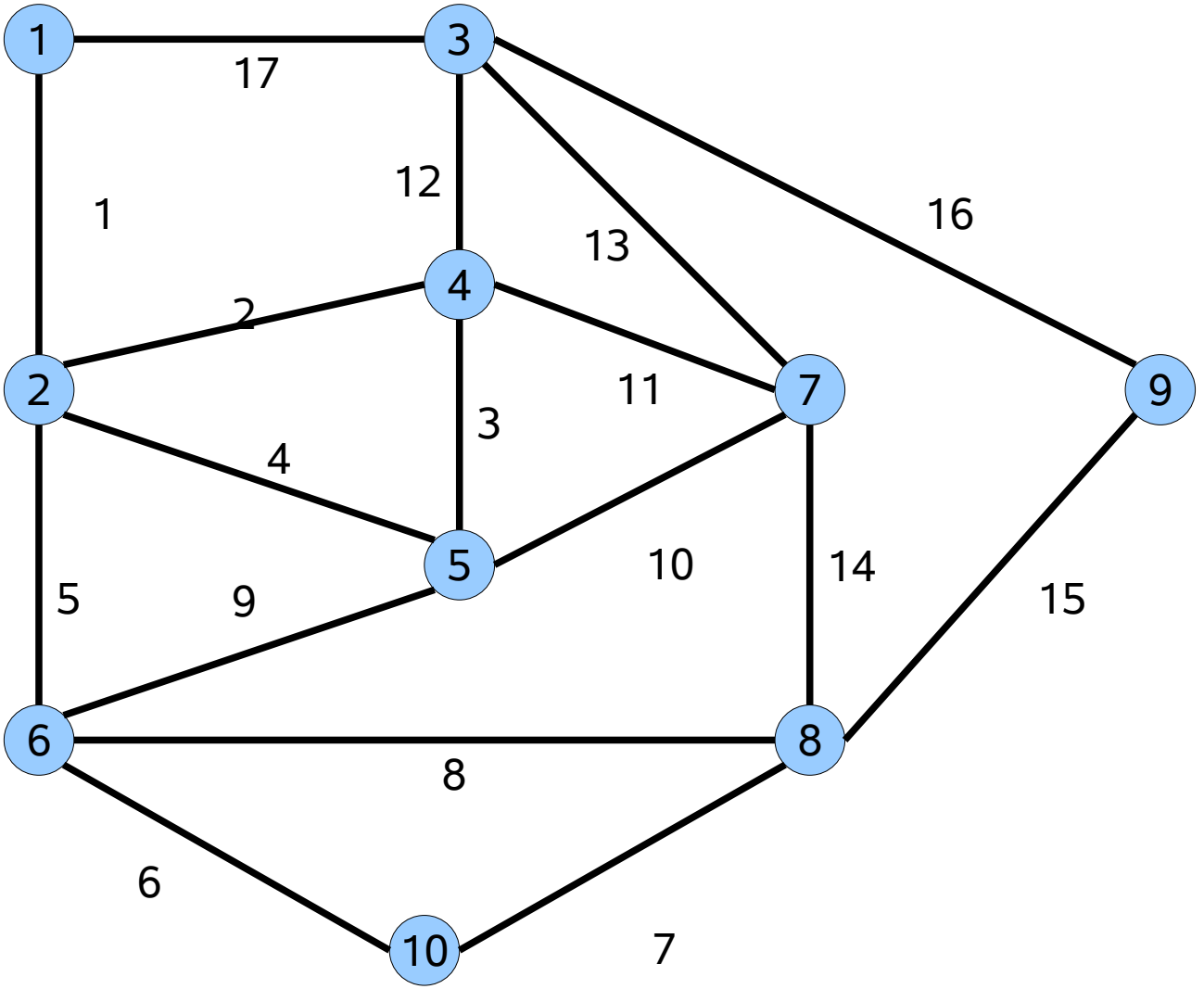
グラフ探索

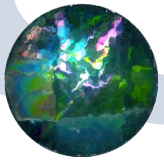
- Euler閉路の探索
  - 全ての弧を一度ずつ経由する閉路
  - 「一筆描き」
- Hamilton閉路の探索
  - 全ての頂点を一度ずつ経由する閉路
- 閉路(circle, closed path)
  - 始点と終点の一致する道



Euler閉路(Euler Circle)

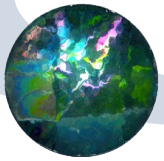
- 一筆書き
- 無向グラフが対象
- 全ての弧を一度ずつ経由して始点に戻る道
- 全ての点の次数が偶数
  - 次数が奇数だったら？


1
3
17
1
12
16
13
4
2
2
11
7
9
3
4
5
10
14
5
9
15
6
8
8
6
10
7

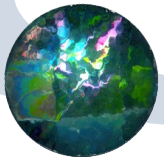


Euler閉路の列挙アルゴリズム

- A_{Euler} : すでに経由した弧の集合 ：初期 $A_{\text{Euler}}=\varnothing$
- r : 始点
- $|A|$: 弧の数



search(v){
  if (v が始点、かつ全ての辺をチェック済み)
    {save circle}
  else {
    forall(v に接続する全ての弧 a){
      if(a が A_{Euler} に含まれない){
        A_{Euler} に a を追加
        w を a の反対側の点とする
        search(w) //再帰呼び出し
        A_{Euler} から a を削除
} } } }



search(v){

if ($v = r \wedge |A_{\text{Euler}}| = |A|$) {save circle}

else {

forall($a \in \delta^{+} v$){

if($a \notin A_{\text{Euler}}$){

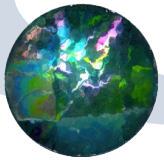
$A_{\text{Euler}} = A_{\text{Euler}} \cup \{a\}$

$w = \partial^{-} a$

search(w)

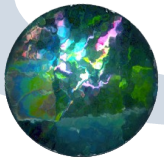
$A_{\text{Euler}} = A_{\text{Euler}} \setminus \{a\}$

} } } }



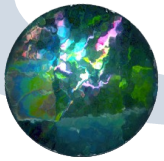
Hamilton 閉路

- 無向グラフが対象
- 全ての点を一度ずつ経由して始点に戻る道
- 巡回セールスマン問題の厳密解を得る際に必要

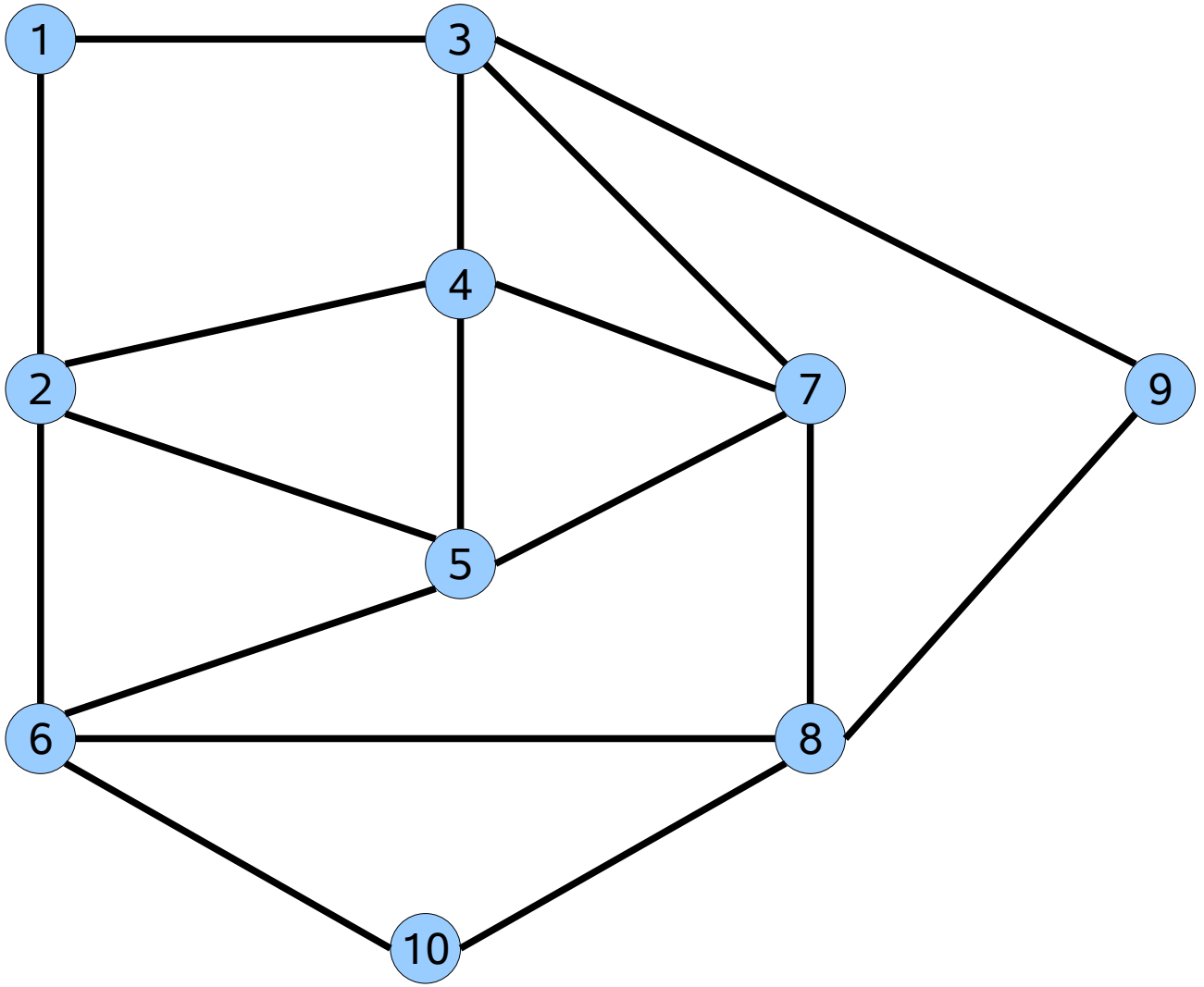


Hamilton閉路の列挙アルゴリズム

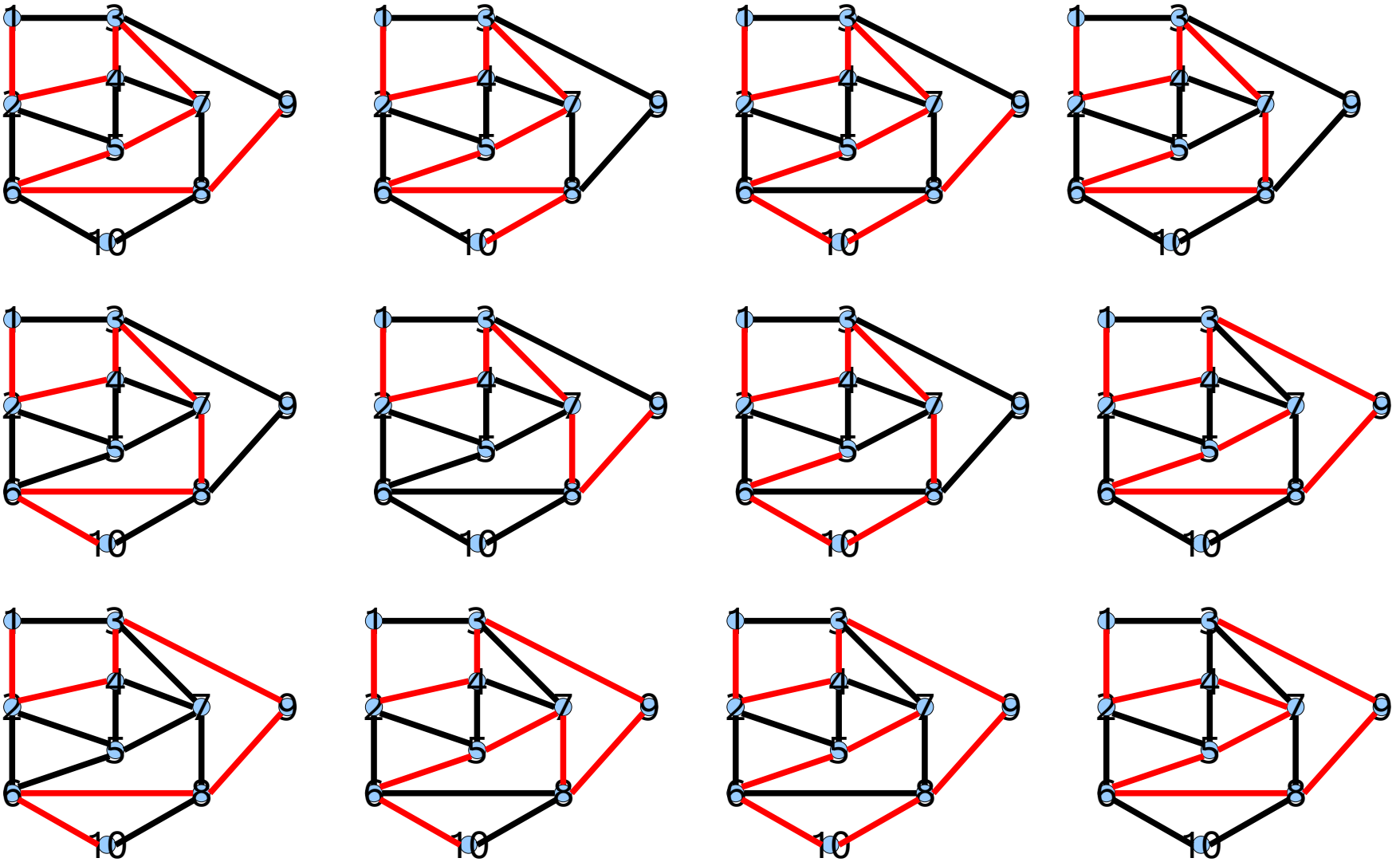
- L : すでに経由した点の集合 ：初期 L={r}
- r : 始点

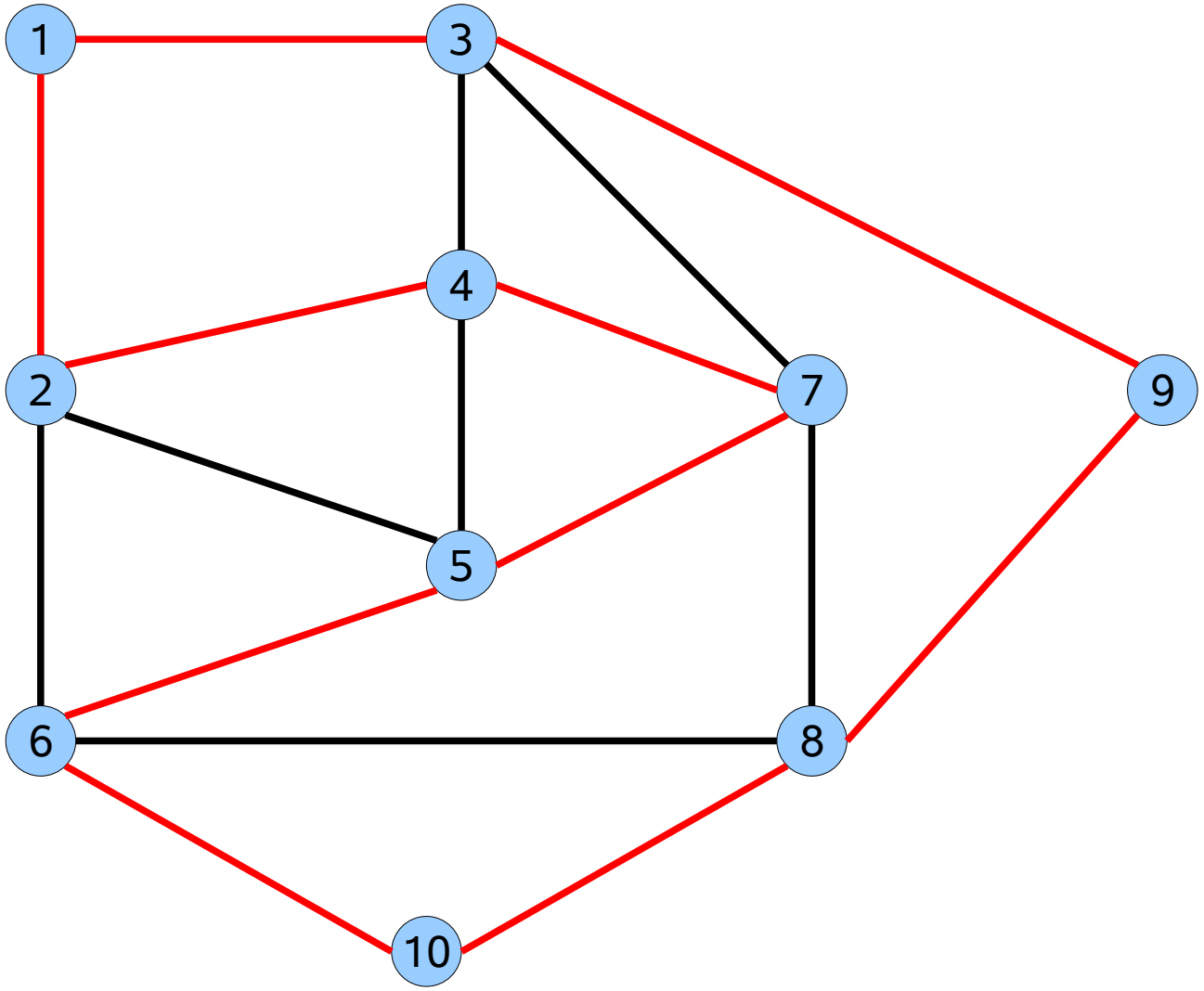


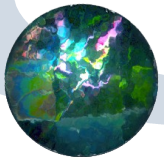
search(v){
forall($a \in \delta^{+} v$){
$w = \partial^{-} a$
if ($w = r \wedge |L| = |V|$) {save circle}
else {
if($w \notin L$){
$L = L \cup \{w\}$
search(w)
$L = L \setminus \{w\}$
}
}
} }


1
3
4
2
7
9
5
6
8
10

Hamilton閉路の探索の様子

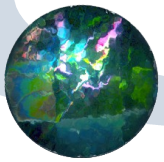


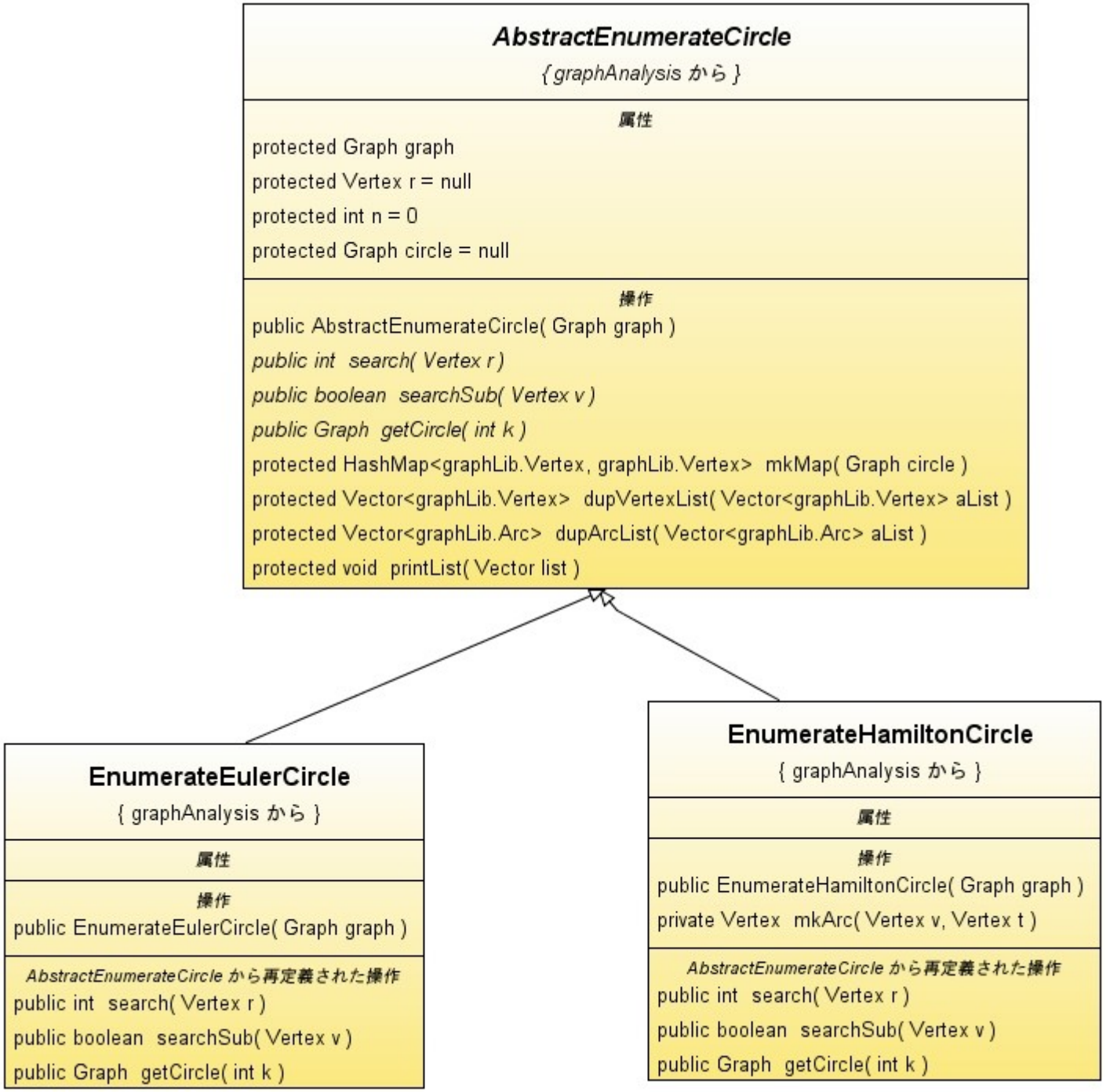

1
3
4
2
7
9
5
6
8
10

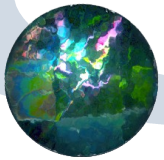


Javaプログラムへ

- アブストラクトクラス
  - AbstractEnumerateCircle
- Euler閉路の列挙
  - EnumerateEulerCircle
- Hamilton閉路の列挙
  - EnumerateHamiltonCircle



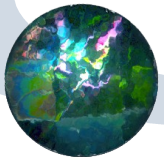

AbstractEnumerateCircle
{graphAnalysis から}
属性
protected Graph graph
protected Vertex r = null
protected int n = 0
protected Graph circle = null
操作
public AbstractEnumerateCircle(Graph graph)
public int search(Vertex r)
public boolean searchSub(Vertex v)
public Graph getCircle(int k)
protected HashMap<graphLib.Vertex, graphLib.Vertex> mkMap(Graph circle)
protected Vector<graphLib.Vertex> dupVertexList(Vector<graphLib.Vertex> aList)
protected Vector<graphLib.Arc> dupArcList(Vector<graphLib.Arc> aList)
protected void printList(Vector list)
EnumerateEulerCircle
{ graphAnalysis から }
属性
操作
public EnumerateEulerCircle(Graph graph)
AbstractEnumerateCircle から再定義された操作
public int search(Vertex r)
public boolean searchSub(Vertex v)
public Graph getCircle(int k)
EnumerateHamiltonCircle
{ graphAnalysis から }
属性
操作
public EnumerateHamiltonCircle(Graph graph)
private Vertex mkArc(Vertex v, Vertex t)
AbstractEnumerateCircle から再定義された操作
public int search(Vertex r)
public boolean searchSub(Vertex v)
public Graph getCircle(int k)



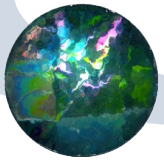
AbstractEnumerateCircle.java

```
public abstract int search(graphLib.Vertex r);
public abstract boolean searchSub(graphLib.Vertex v);
public abstract graphLib.Graph getCircle(int k);

protected   java.util.HashMap<graphLib.Vertex,graphLib.Vertex>
        mkMap(graphLib.Graph circle);
protected java.util.Vector<graphLib.Vertex> dupVertexList(
        java.util.Vector<graphLib.Vertex> aList);
protected java.util.Vector<graphLib.Arc> dupArcList(
        java.util.Vector<graphLib.Arc> aList);
protected void printList(java.util.Vector list);
```



- メソッドsearch
  - 始点を指定して、探索を準備する。
  - 探索結果(circle)を保存する準備
  - circleに対応する頂点または弧のリストを準備
  - 再帰的メソッドsearchSubを呼び出す
  - 発見したcircleの数を返す



- メソッドsearchSub
  - 再帰的メソッド
- メソッドgetCircle
  - *k*番目のcircleを返す
  - 指定されたcircleに対応する頂点または弧のリストからグラフを構成する

```java
/*
 * AbstractEnumerateCircle.java
 *
 * Created on 2007/06/08, 12:27
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package graphAnalysis;

/**
 *
 * @author tadaki
 */


public abstract class AbstractEnumerateCircle {
    /* 探索対象となるgraph */
    protected graphLib.Graph graph;
    protected java.util.Vector<graphLib.Arc> arcList=null;
    protected java.util.Vector<java.util.Vector<graphLib.Arc>> arcListList=null;
    protected java.util.Vector<graphLib.Vertex> vertexList=null;
    protected java.util.Vector<java.util.Vector<graphLib.Vertex>> vertexListList=null;
    protected graphLib.Vertex r=null;  //出発点
    protected int n=0;
    protected graphLib.Graph circle=null;
    /* graph  とcircle の頂点の対応関係 */
    java.util.HashMap<graphLib.Vertex,graphLib.Vertex> vertexMap=null;
    /** Creates a new instance of AbstractEnumerateCircle */
    public AbstractEnumerateCircle(graphLib.Graph graph) {
        this.graph=graph;
    }
    public abstract int search(graphLib.Vertex r);
    public abstract boolean searchSub(graphLib.Vertex v);
    public abstract graphLib.Graph getCircle(int k);

    protected java.util.HashMap<graphLib.Vertex,graphLib.Vertex> mkMap(graphLib.Graph circle){
        /* graph  とcircle の頂点の対応関係 */
        java.util.HashMap<graphLib.Vertex,graphLib.Vertex> vertexMap
                =new java.util.HashMap<graphLib.Vertex,graphLib.Vertex>();
        for(graphLib.Vertex v: graph.getVertexes()){
            graphLib.Vertex vt=new graphLib.Vertex(v.toString());
            java.awt.geom.Point2D.Double p=v.getPoint();
            vt.setPoint(p.x,p.y);
            vertexMap.put(v,vt);
            circle.addVertex(vertexMap.get(v));
        }
        return vertexMap;
    }

    protected java.util.Vector<graphLib.Vertex> dupVertexList(java.util.Vector<graphLib.Vertex> aList){
        java.util.Vector<graphLib.Vertex> aListNew=new java.util.Vector<graphLib.Vertex>();
        for(int i=0;i<aList.size();i++){
            aListNew.add(aList.get(i));
        }
        return aListNew;
    }

    protected java.util.Vector<graphLib.Arc> dupArcList(java.util.Vector<graphLib.Arc> aList){
        java.util.Vector<graphLib.Arc> aListNew=new java.util.Vector<graphLib.Arc>();
        for(int i=0;i<aList.size();i++){
            aListNew.add(aList.get(i));
        }
        return aListNew;
    }

    protected void printList(java.util.Vector list){
        for(int i=0;i<list.size();i++){
            System.out.print(list.get(i).toString()+" ");
        }
```

```
        System.out.println();
    }

}
```

```java
/*
 * EnumerateEulerCircle.java
 *
 * Created on 2007/03/16, 21:43
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package graphAnalysis;

/**
 *
 * @author tadaki
 */


public class EnumerateEulerCircle extends AbstractEnumerateCircle{

    /** Creates a new instance of EnumerateEulerCircle */
    public EnumerateEulerCircle(graphLib.Graph graph) {
       super(graph);
        n=graph.getNumArcs();
    }
    /**
     * 探索実行
     * @param r 探索を開始する頂点
     * @return 探索結果のcircleの数
     */
    public int search(graphLib.Vertex r){
        this.r=r;
        arcListList = new java.util.Vector<java.util.Vector<graphLib.Arc>>();
        arcList = new java.util.Vector<graphLib.Arc>();
        searchSub(r);
        return arcListList.size();
    }

    public boolean searchSub(graphLib.Vertex v){
        if(v.equals(r) && arcList.size()==n){
            arcListList.add(dupArcList(arcList));
//            printList(arcList);
        } else {
            int degree=v.degree();
            for(int i=0;i<degree;i++){
                graphLib.Arc a=v.getArc(i);
                if(!arcList.contains(a)){
                    arcList.add(a);
                    searchSub(a.terminal(v));
                    arcList.remove(a);
                }
            }
        }
        return true;
    }

    public synchronized graphLib.Graph getCircle(int k){
        if(k>=arcListList.size())return null;
        circle = new graphLib.Graph(String.valueOf(k)+"-th Euler Circle");
        vertexMap=mkMap(circle);

        java.util.Vector<graphLib.Arc> aList
                =dupArcList(arcListList.get(k));
        graphLib.Vertex v=r;
        for(int i=0;i<aList.size();i++){
            graphLib.Arc a=aList.get(i);
            if(a.from().equals(v) || a.to().equals((v))){
                graphLib.Vertex t=a.terminal(v);
                circle.addArc(vertexMap.get(v),vertexMap.get(t),a.toString());
                v=t;
            } else {
                System.err.println("Error");
```

```
                }
            }
            return circle;
        }
}
```

```java
/*
 * EnumerateHamiltonCircle.java
 *
 * Created on 2007/03/24, 20:28
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package graphAnalysis;

/**
 *
 * @author tadaki
 */


public class EnumerateHamiltonCircle extends AbstractEnumerateCircle{

    /** Creates a new instance of EnumerateHamiltonCircle */
    public EnumerateHamiltonCircle(graphLib.Graph graph) {
        super(graph);
        n=graph.getSize();
    }
    /**
     * 探索実行
     * @param r 探索を開始する頂点
     * @return 探索結果のcircleの数
     */
    public int search(graphLib.Vertex r){
        this.r=r;
        vertexListList = new java.util.Vector<java.util.Vector<graphLib.Vertex>>();
        vertexList = new java.util.Vector<graphLib.Vertex>();
        vertexList.add(r);
        searchSub(r);
        return vertexListList.size();
    }

    public boolean searchSub(graphLib.Vertex v){
        int degree=v.degree();
        for(int i=0;i<degree;i++){
            graphLib.Arc a=v.getArc(i);
            graphLib.Vertex t = a.terminal(v);
            if(t.equals(r) && vertexList.size()==n){
                vertexListList.add(dupVertexList(vertexList));
            } else {
                if(!vertexList.contains(t)){
                    vertexList.add(t);
                    searchSub(t);
                    vertexList.remove(t);
                }
            }
        }
        return true;
    }

    public synchronized graphLib.Graph getCircle(int k){
        if(k>=vertexListList.size())return null;
        circle = new graphLib.Graph(String.valueOf(k)+"-th Hamilton Circle");
        vertexMap=mkMap(circle);
        java.util.Vector<graphLib.Vertex> vList
                =dupVertexList(vertexListList.get(k));
        graphLib.Vertex v=r;
        for(int i=0;i<vList.size()-1;i++){
            v=mkArc(v,vList.get(i+1));
        }
        mkArc(vList.lastElement(),r);
        return circle;
    }

    private graphLib.Vertex mkArc(graphLib.Vertex v, graphLib.Vertex t){
```

```
        boolean b=true;
        int degree=v.degree();
        for(int j=0;j<degree && b;j++){
            graphLib.Arc a=v.getArc(j);
            if(a.terminal(v).equals(t)){
                circle.addArc(vertexMap.get(v),vertexMap.get(t),a.toString());
                v=t;
                b=false;
            }
        }
        return v;
    }
}
```