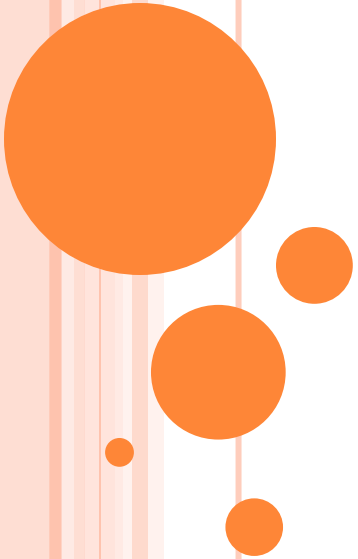


グラフの探索
Javaでの実装

二つの探索手法

- 深さ優先探索：DFS (Depth-First Search)
- 幅優先探索：BFS (Breadth-First Search)
- 共通部分
  - 元のグラフを指定して、極大木を得る

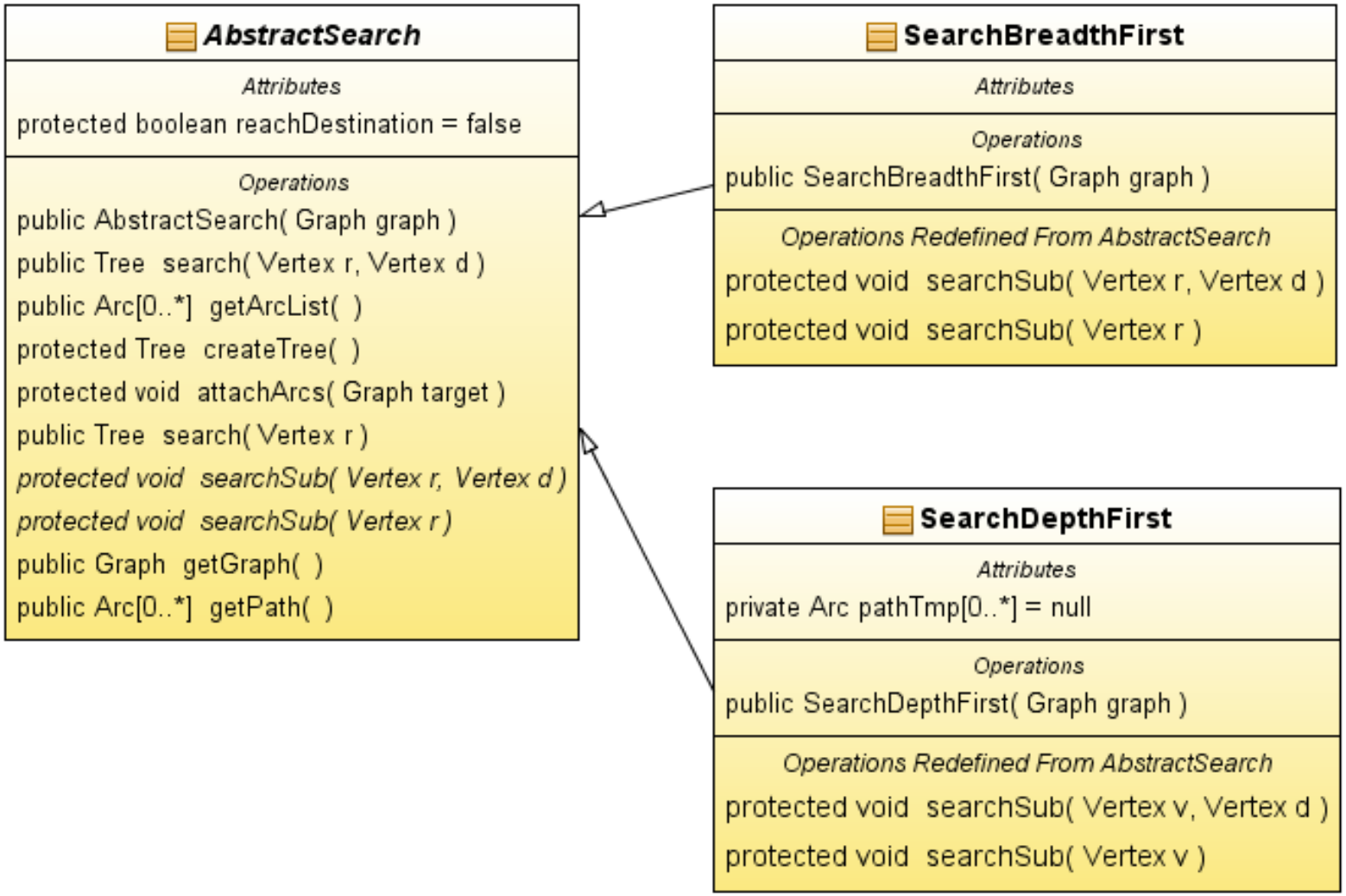
探索アルゴリズムの利用の観点から

- 利用する側からみると
  - 取り替えられる部品
  - どちらの方法が良いかはグラフに依存
  - 操作性が同じでなければ
  - 共通のクラスの派生で作ると便利
- 共通化を考える
  - 操作性の共通化
  - 共通のメソッド

Abstract classを使った共通化

- 共通のデータ構造
- 共通のメソッド
- 一部のメソッドの実装方法が異なる

- メソッド定義に修飾子abstract
  - 実装が定義されていない
  - 派生クラスで実装しなければならない
- クラス定義に修飾子abstract
  - このクラスのインスタンスが生成できないことを表す


AbstractSearch
Attributes
protected boolean reachDestination = false
Operations
public AbstractSearch(Graph graph)
public Tree search(Vertex r, Vertex d)
public Arc[0..*] getArcList( )
protected Tree createTree( )
protected void attachArcs(Graph target)
public Tree search(Vertex r)
protected void searchSub(Vertex r, Vertex d)
protected void searchSub(Vertex r)
public Graph getGraph( )
public Arc[0..*] getPath( )
SearchBreadthFirst
Attributes
Operations
public SearchBreadthFirst(Graph graph)
Operations Redefined From AbstractSearch
protected void searchSub(Vertex r, Vertex d)
protected void searchSub(Vertex r)
SearchDepthFirst
Attributes
private Arc pathTmp[0..*] = null
Operations
public SearchDepthFirst(Graph graph)
Operations Redefined From AbstractSearch
protected void searchSub(Vertex v, Vertex d)
protected void searchSub(Vertex v)

実装時の注意

- 元のグラフを壊さない
- 結果をtreeとして得る
  - 頂点の集合をコピーする
  - 探索に使用した弧の一覧を作成する
  - 探索に使用した弧のコピーを追加する
- 終点を指定した場合
  - 始点から終点への経路のみを生成する

AbstractSearchクラスのフィールド

```
/** 探索対象となるgraphのコピー */
  protected Graph graph;
 /** 探索した頂点のリスト */
  protected List<Vertex> listOfVertex = null;
 /** 終点を指定した場合、その終点に到達したか否か */
  protected boolean reachDestination = false;
 /** 探索に使用した孤 */
  protected List<Arc> path = null;
 /** 始点 */
  protected Vertex r = null;
 /** 終点 */
  protected Vertex d = null;
```

AbstractSearchクラスの主要メソッド

public AbstractSearch(Graph graph)	コンストラクタ
public Tree search(Vertex r)	探索実行（終点指定なし）
public Tree search(Vertex r, Vertex d)	探索実行（終点指定）
protected abstract Tree createTree()	探索結果から木を得る
protected void attachArcs(Graph target)	使用した弧をtargetに追加

- 以下は、各派生クラスで実装する

protected abstract void searchSub(Vertex r)	探索の実体
protected abstract void searchSub(Vertex r, Vertex d)	

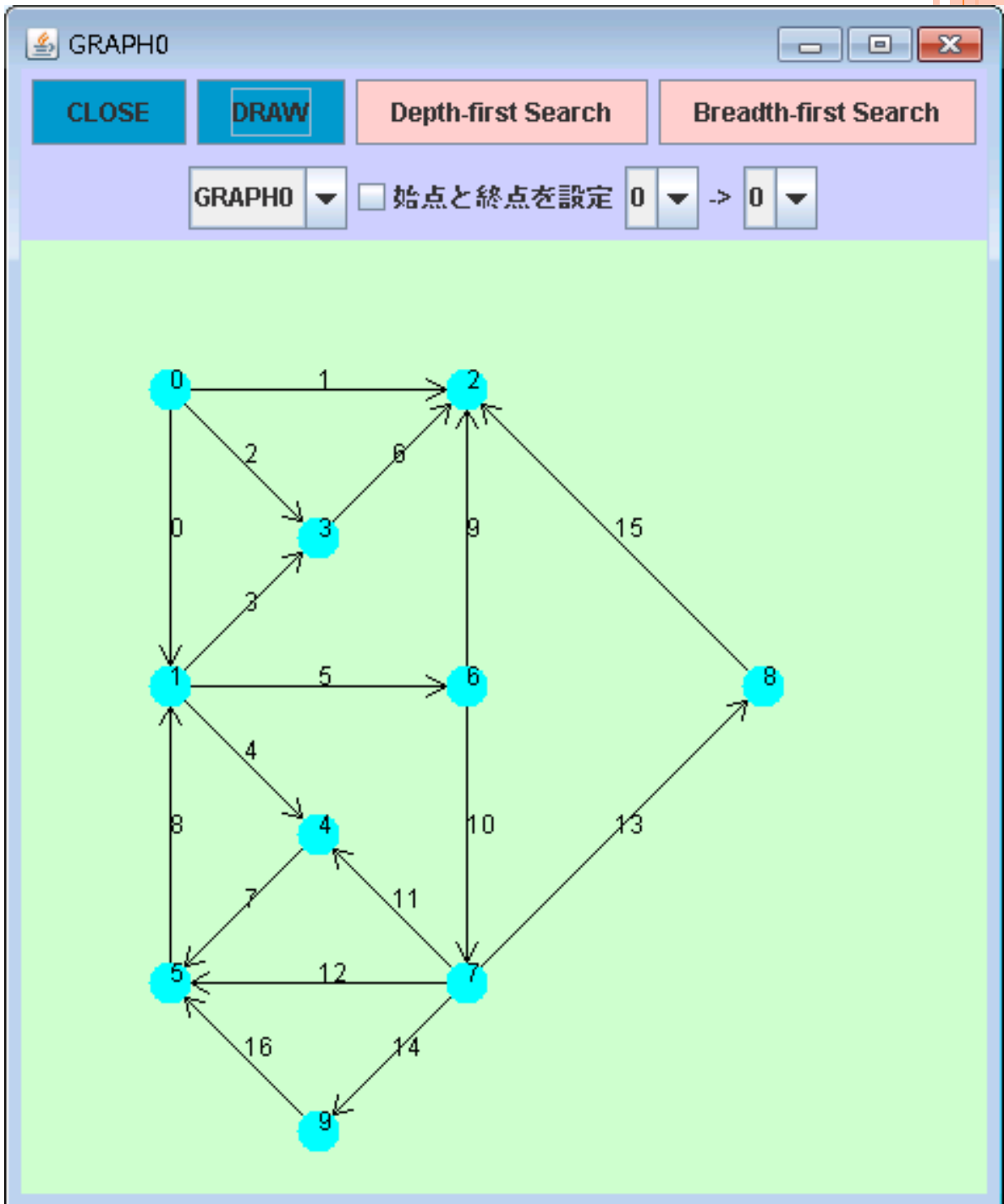
深さ優先探索

```
protected void searchSub(Vertex v) {
        for (Arc a : v.getArcs()) {//頂点v から出ている弧
            //弧の先の頂点
            Vertex to = a.terminal(v);
            if (!listOfVertex.contains(to)) {//弧の先の頂点はまだtreeに無い
                //頂点を追加
                listOfVertex.add(to);
                /** 探索に使用した弧を保存 */
                path.add(a);
                searchSub(to);
            }
        }
    }
```

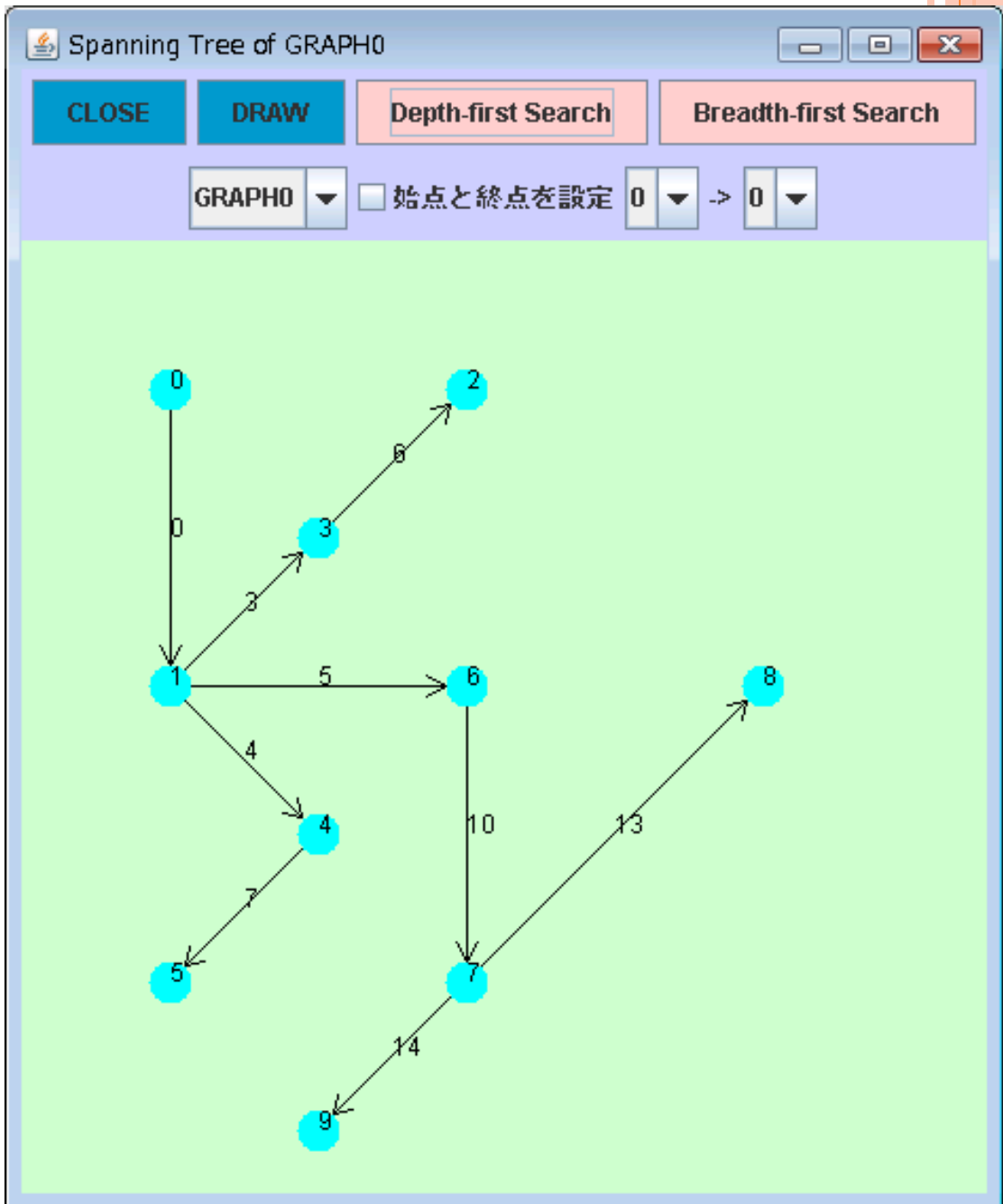
幅優先探索

```
protected void searchSub(Vertex r) {
    //待ち行列の作成
    ConcurrentLinkedQueue<Vertex> queue =
        new ConcurrentLinkedQueue<Vertex>();
    queue.add(r);
    while (!queue.isEmpty()) {
      Vertex v = queue.poll();
      for (Arc a : v.getArcs()) {
        Vertex to = a.terminal(v);//vと反対側
        if (!listOfVertex.contains(to) && !queue.contains(to)) {
          listOfVertex.add(to);
          path.add(a);
          queue.add(to);
        }
      }
    }
  }
```

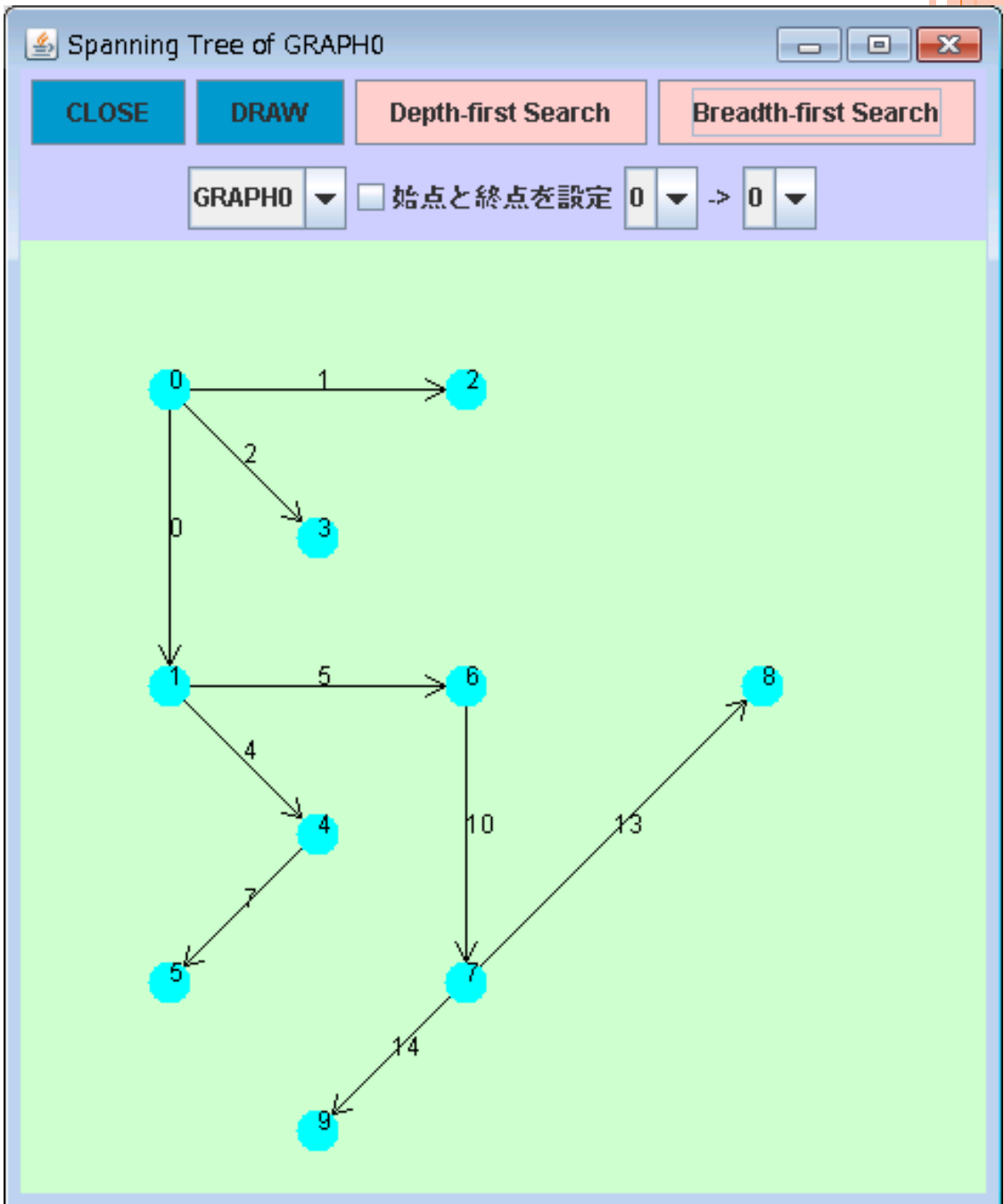
対象グラフの表示


GRAPH0
CLOSE
DRAW
Depth-first Search
Breadth-first Search
GRAPH0
始点と終点を設定
0
->
0
0
1
2
3
4
5
6
7
8
9
1
2
6
0
3
9
15
3
5
4
8
10
13
7
11
12
16
14

深さ優先探索の実施


Spanning Tree of GRAPH0
CLOSE
DRAW
Depth-first Search
Breadth-first Search
GRAPH0
始点と終点を設定
0
->
0
0
2
6
0
3
3
1
5
6
8
4
4
10
13
7
5
7
14
9

幅優先探索の実施


Spanning Tree of GRAPH0
CLOSE
DRAW
Depth-first Search
Breadth-first Search
GRAPH0
始点と終点を設定
0
->
0
0
1
2
3
4
5
6
7
8
9
1
2
0
5
4
7
10
13
14

終点を指定した深さ優先探索の実施