[제 2 주 실습]

그래프 색칠

강 지 훈
jhkang@cnu.ac.kr

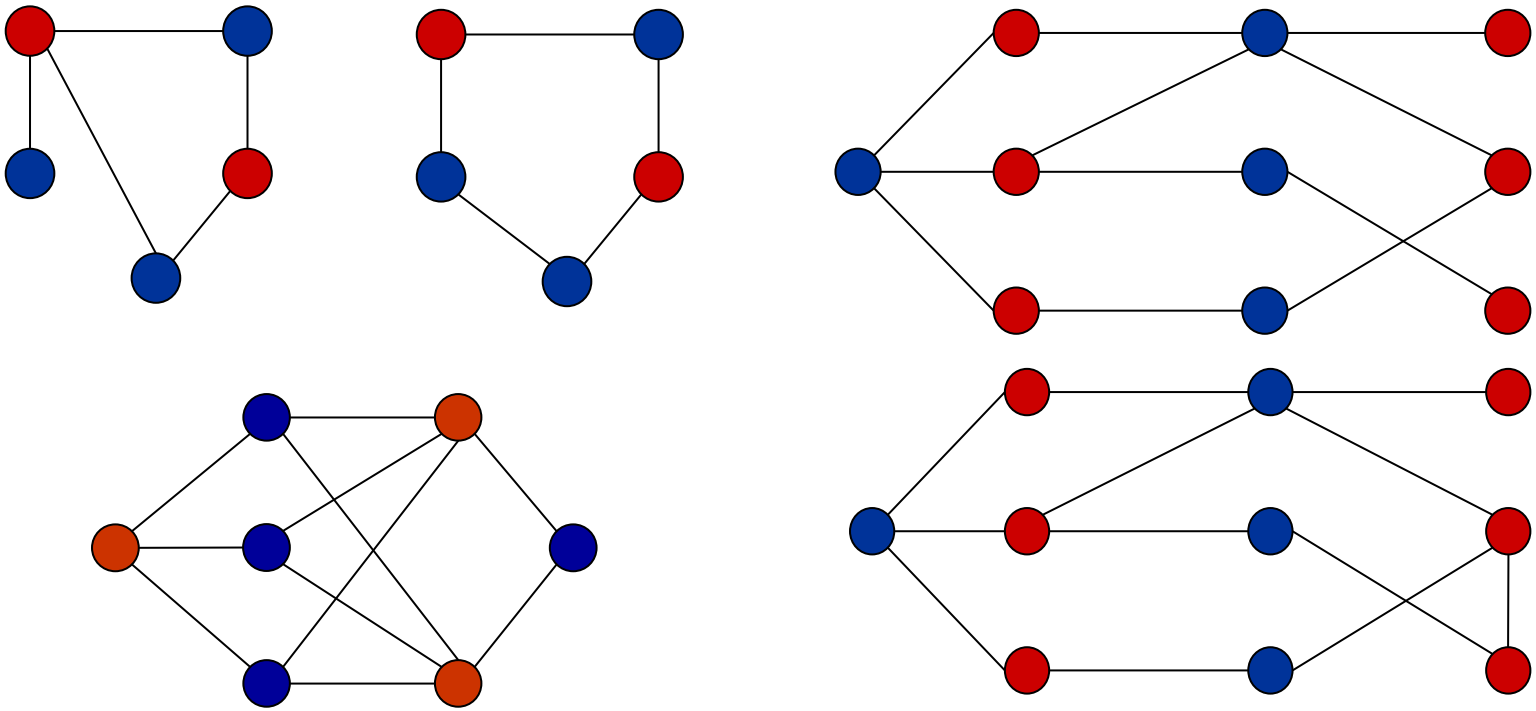
충남대학교 컴퓨터공학과

© *J.-H.Kang, CNU*

[문제 2]
그래프 색칠

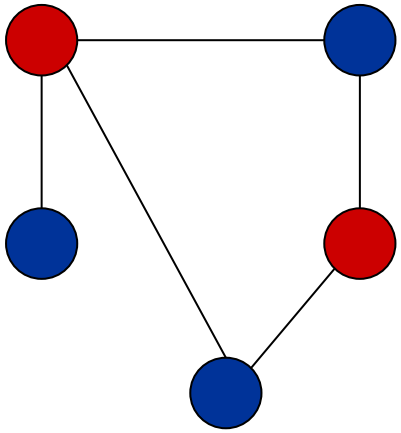
문제 개요

- Undirected graph의 각 vertex에 색칠을 한다.
  - 색은 **red**와 **blue**이다.
  - 하나의 vertex가 red이면 그 이웃 vertex는 blue이다.
  - 모든 vertex에 색칠을 한다.

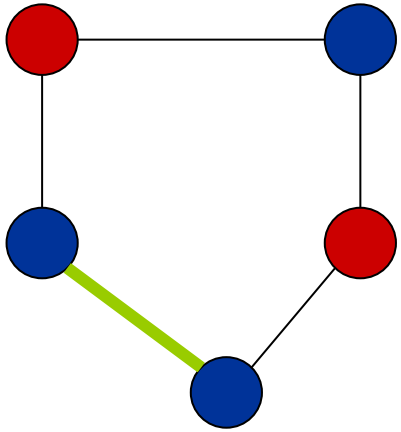


문제 개요

- 그래프의 각 edge의 양 끝 vertex가 서로 다른 색으로 칠해졌는가?
  - G1: 모든 edge가 한 끝은 red, 다른 한 끝은 blue
  - G2: 녹색 edge는 양 끝 모두 blue



Graph G1



Graph G2

입력

- Undirected graph의 입력
  - # of Vertices (numOfVertices)
    - 0보다 크거나 같은 정수
  - # of Edges (numOfEdges)
    - 0보다 크거나 같은 정수
  - A set of edges
    - numOfEdges 수만큼 반복하여 입력 받으면서 그래프를 만든다.
    - Undirected Edge는 vertex의 쌍으로 표현한다.
      - (u, v)
      - Vertex는 (0..(NumVertices-1)) 사이의 양의 정수
  - 입력 오류인 경우 재입력 받는다.

출력

그래프의 출력

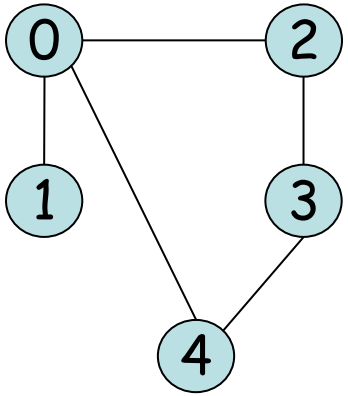
[0] -> 4 -> 2 -> 1

[1] -> 0

[2] -> 3-> 0

[3] -> 4 -> 2

[4] -> 3 -> 0

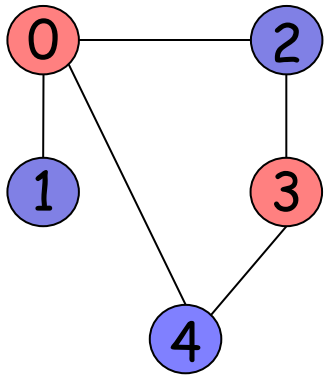


- Undirected graph를 만들 때 유의할 점:
  - Edge (u,v)가 입력되면, 양방향의 두 edge (u,v)와 (v,u)를 모두 그래프에 삽입해야 한다.

Coloring 결과 출력

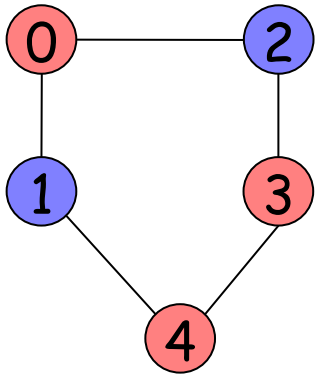
❑출력의 예 1

```
- 그래프의 vertex수와 edge수를 입력 받아야 합니다.
? 그래프의 vertex 수를 입력 하시오:5
? 그래프의 edge 수를 입력 하시오:5
- 그래프의 edge를 반복하여 5 개 입력 받아야 합니다.
  하나의 edge는 (vertex1 vertext2)의  순서로 표시됩니다.
? Edge를 입력하시오: 0 4
? Edge를 입력하시오: 0 2
? Edge를 입력하시오: 0 1
? Edge를 입력하시오: 2 3
? Edge를 입력하시오: 3 4
생성된 그래프 :
[0] -> 1 2 4
[1] -> 0
[2] -> 0 3
[3] -> 2 4
[4] -> 0 3

Vertex에 칠해진 색깔 :
0 : RED
1 : BLUE
2 : BLUE
3 : RED
4 : BLUE

끝 색이 같은 Edge :

<<Coloring을 종료합니다>>
```



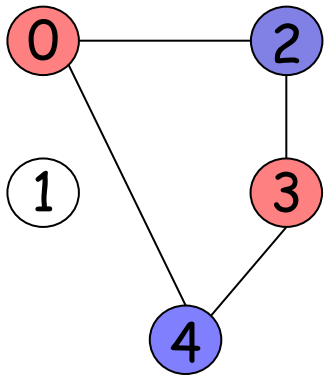
❑출력의 예 2

```
- 그래프의 vertex수와 edge수를 입력 받아야 합니다.
? 그래프의 vertex 수를 입력 하시오:5
? 그래프의 edge 수를 입력 하시오:5
- 그래프의 edge를 반복하여 5 개 입력 받아야 합니다.
  하나의 edge는 (vertex1 vertext2)의  순서로 표시됩니다.
? Edge를 입력하시오: 0 2
? Edge를 입력하시오: 0 1
? Edge를 입력하시오: 2 3
? Edge를 입력하시오: 3 4
? Edge를 입력하시오: 1 4
생성된 그래프 :
[0] -> 1 2
[1] -> 0 4
[2] -> 0 3
[3] -> 2 4
[4] -> 1 3

Vertex에 칠해진 색깔 :
0 : RED
1 : BLUE
2 : BLUE
3 : RED
4 : RED

끝 색이 같은 Edge :
3,4
4,3

<<Coloring을 종료합니다>>
```



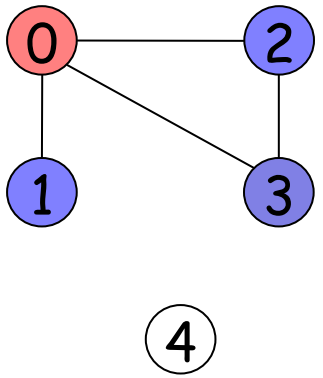
❑출력의 예 3

```
- 그래프의 vertex수와 edge수를 입력 받아야 합니다.
? 그래프의 vertex 수를 입력 하시오:5
? 그래프의 edge 수를 입력 하시오:4
- 그래프의 edge를 반복하여 4 개 입력 받아야 합니다.
  하나의 edge는 (vertex1 vertext2)의  순서로 표시됩니다.
? Edge를 입력하시오: 0 2
? Edge를 입력하시오: 0 4
? Edge를 입력하시오: 2 3
? Edge를 입력하시오: 4 3
생성된 그래프 :
[0] -> 2 4
[1] ->
[2] -> 0 3
[3] -> 2 4
[4] -> 0 3

Vertex에 칠해진 색깔 :
0 : RED
1 : NONE
2 : BLUE
3 : RED
4 : BLUE

끝 색이 같은 Edge :

<<Coloring을 종료합니다>>
```



❑출력의 예 4

```
- 그래프의 vertex수와 edge수를 입력 받아야 합니다.
? 그래프의 vertex 수를 입력 하시오:5
? 그래프의 edge 수를 입력 하시오:4
- 그래프의 edge를 반복하여 4 개 입력 받아야 합니다.
  하나의 edge는 (vertex1 vertext2)의  순서로 표시됩니다.
? Edge를 입력하시오: 0 2
? Edge를 입력하시오: 0 3
? Edge를 입력하시오: 0 1
? Edge를 입력하시오: 2 3
생성된 그래프 :
[0] -> 1 2 3
[1] -> 0
[2] -> 0 3
[3] -> 0 2
[4] ->

Vertex에 칠해진 색깔 :
0 : RED
1 : BLUE
2 : BLUE
3 : BLUE
4 : NONE

끝 색이 같은 Edge :
2,3
3,2

<<Coloring을 종료합니다>>
```



사용자에게 필요한 객체는?

- AdjacencyMatrixGraph
- Edge
- Coloring
- LinkedList
- CircularLinkedQueue
- Node
- Application

AdjacencyMatrixGraph의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - [문제1]의 것을 복사해서 수정
  - public AdjacencyMatrixGraph(int givenNumOfVertices)
  - public boolean doesVertexExist (int aVertex)
  - public boolean doesEdgeExist (Edge anEdge)
  - public int numOfVertices ()
  - public int numOfEdges ()
  - public boolean addEdge(Edge anEdge)
  - public Iterator iterator(int givenVertex)
  - public class Iterator

Edge의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - [문제1]의 것을 복사해서 사용 [수정 하지 않음]
  - public Edge(int givenTailVertex, int givenHeadVertex)
  - public void setTailVertex(int aFromVertex)
  - public int tailVertex()
  - public void setHeadVertex(int aHeadVertex)
  - public int headVertex()

Coloring의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public Coloring(AdjacencyMatrixGraph givenGraph)
  - public void runColoring()
  - public LinkedList<Edge> sameColorEdges()
  - public Iterator iterator()
  - public class Iterator

LinkedList〈T〉의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public LinkedList()
  - public LinkedList(int givenMaxSize)
  - public void clear()
  - public boolean isFull()
  - public boolean isEmpty()
  - public int size()
  - public boolean add(T anElement)
  - public Iterator iterator()
  - public class Iterator

CircularLinkedQueue〈T〉의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public CircularLinkedQueue ()
  - public CircularLinkedQueue (int initialCapacity)

  - public int maxSize()
  - public boolean isEmpty ()
  - public boolean isFull ()
  - public int size ()

  - public T frontElement ()
  - public boolean enQueue (T anElement)
  - public T deQueue ()

Node〈T〉 의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public Node()
  - public Node(T givenElement)
  - public Node(T givenElement, Node givenNode)

  - public T element()
  - public Node next()
  - public void setElement(T anElement)
  - public void setNext(Node anNode)

Application의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public void run()

DS2_02_학번_이름 Class의 구조

```
/* 항상 사용하는 main Class 구조
 * 과제 제출시 반드시 포함 되어 있어야 함*/
public class DS2_02_학번_이름 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Application application = new Application();
        application.run ();
    }
}
```

Application Class

Application – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class Application {
    private AdjacencyMatrixGraph _graph;
    private Coloring _coloring;
    private Scanner _scanner;
```

Application의 Public Method

- Application의 Public Member function의 사용법
  - public void run()
    - 프로그램의 실행
    - 그래프를 만든 뒤 Coloring 작업을 한다.

Application의 Private Method

- Application의 Private Member function의 사용법
  - private void inputAndMakeGraph()
    - 키보드로부터 그래프 정보를 입력 받는다.
    - 마지막에 입력된 그래프를 출력하여 보여준다.
  - private void coloring()
    - Coloring를 실행한다.
    - 마지막에 실행된 결과를 보여준다.
  - private void showColoring()
    - Coloring 결과를 보여준다.
  - private void showGraph()
    - 생성된 Graph를 보여준다.

Member Functions의 구현

- Application의 Public Member function의 구현
  - public void run ()
    - inputAndMakeGraph를 호출
    - coloring을 호출

Member Functions의 구현

- Application의 Private Member function의 구현
  - private void showGraph()
    - Graph에 Iterator 적용

```
private void    showGraph()
{
    Edge    e ;
    int tailVertex, headVertex;
    int         numOfVertices = this._graph.numOfVertices();

    System.out.println("생성된 그래프 : ");
    for ( tailVertex = 0 ; tailVertex < numOfVertices ; tailVertex++ ) {
        AdjacencyMatrixGraph.Iterator it = this._graph.iterator(tailVertex);
        System.out.print("[" + tailVertex + "] -> ");
        while ( it.hasNext() ) {
            headVertex = it.next();
            System.out.print(headVertex + " ");
        }
        System.out.println();
    }
    System.out.println();
}
```

Member Functions의 구현

- Application의 Private Member function의 구현
  - private void inputAndMakeGraph()
    - [문제1]에서 사용한 것을 사용
  - private void coloring()
    - this._coloring을 this._graph를 이용하여 생성
    - this._coloring의 runColoring을 실행
    - this.showColoring으로 Coloring 결과를 출력

Member Functions의 구현

- Application의 Private Member function의 구현
  - private void showColoring()
    - < Vertex에 칠해진 색깔을 출력 >
    - Coloring.Iterator형인 변수 it를 this._coloring.iterator()를 통해 얻어옴
    - 현재 vertex를 저장할 변수 vertex를 생성 하여 0으로 초기화
    - It에 hasNext가 있을 동안
      - Vertex와 it의 next값을 출력
    - < 끝 색이 같은 Edge 출력>
    - LinkedList<Edge>.Iterator형인 변수 listIter를 this._coloring.sameColorEdges().iterator()를 통해 얻어옴
    - listIter에 hasNext가 있을 동안
      - Edge 의 headVertex와 tailVertex를 출력
    - showGraph()를 참고하여 작성

AdjacencyMatrixGraph Class의 Iterator inner Class 구현

AdjacencyMatrixGraph Iterator

- 기존의 AdjacencyMatrixGraph에 inner Iterator를 추가

```
public Iterator iterator(int givenVertex)
{
    return new Iterator(givenVertex);
}
public class Iterator
{
    private int _nextPosition;
    private int _vertex;

    private Iterator(int givenVertex)
    {
        this._nextPosition = 0;
        this._vertex = givenVertex;
    }
    public boolean hasNext()
    {
        while(this._nextPosition != numOfVertices() &&
                _adjacency[this._vertex][this._nextPosition] != 1)
            this._nextPosition++;
        return (this._nextPosition < numOfVertices());
    }
    public int next()
    {
        int ret;
        ret = this._nextPosition;
        this._nextPosition++;
        return ret;
    }
}
```

보고서에

Iterator를 분석한 내용을 넣을 것

중간 점검

- 여기까지 구현 후 그래프의 입력 및 출력이 정상적으로 되는지 확인하세요.

Node Class

Node〈T〉 – 비공개 인스턴스 변수

```
public class Node<T> {
    private T _element;
    private Node _next;
```

Node〈T〉의 Public Method

- Node<T> 의 Public Member function의 사용법
  - public Node()
  - public Node(T givenElement)
  - public Node(T givenElement, Node givenNode)
    - 생성자
  - public T element()
    - Element 값을 받음
  - public Node next()
    - Next 값을 받음
  - public void setElement(T anElement)
    - anElement 값을 저장
  - public void setNext(Node anNode)
    - anNode 값을 next로 저장

Member Functions의 구현

- Node<T> 의 Public Member function의 구현
  - public Node()
    - this._element을 null로 초기화
    - this._next를 null로 초기화
  - public Node(T givenElement)
    - givenElement를 this._element에 저장
    - this._next를 null로 초기화
  - public Node(T givenElement, Node givenNode)
    - givenElement를 this._element에 저장
    - givenNode를 this._next에 저장

Member Functions의 구현

- Node<T> 의 Public Member function의 구현
  - public T element()
    - this._element 반환
  - public Node next()
    - this._next 반환
  - public void setElement(T anElement)
    - anElement를 this._element에 저장
  - public void setNext(Node anNode)
    - anNode를 this._next에 저장

LinkedList Class

LinkedList– 비공개 인스턴스 변수

```
public class LinkedList<T> {
    private static final int DEFAULT_INITIAL_CAPACITY = 20 ;
    private int _maxSize;
    private int _size;
    private Node<T> _head;
```

LinkedList 의 Public Method

- LinkedList 의 Public Member function의 사용법
  - 실습에 사용하기 위하여 간단한 형태의 LinkedList를 구현
  - 실습에서 delete는 사용하지 않으므로 구현하지 않음
  - public LinkedList()
  - public LinkedList(int givenMaxSize)
    - 생성자
  - public void clear()
    - 현재 List 초기화
  - public boolean isFull()
    - List가 가득 차 있는지 확인
  - public boolean isEmpty()
    - List가 비어 있는지 확인
  - public int size()
    - 현재 List의 Size 확인

LinkedList 의 Public Method

- LinkedList 의 Public Member function의 사용법
  - public boolean add(T anElement)
    - anElement를 List에 삽입
  - public Iterator iterator()
  - public class Iterator

Member Functions의 구현

- LinkedList 의 Public Member function의 구현
  - public LinkedList()
    - this._maxSize를 DEFAULT_INITIAL_CAPACITY로 초기화
    - this._size를 0으로 초기화
    - this._head를 null로 초기화
  - public LinkedList(int givenMaxSize)
    - this._maxSize = givenMaxSize로 초기화
    - this._size를 0으로 초기화
    - this._head를 null로 초기화

Member Functions의 구현

- LinkedList 의 Public Member function의 구현
  - public void clear()
    - this._size를 0으로 초기화
    - this._head를 null로 초기화
  - public boolean isFull()
    - this._size가 this._maxSize와 같을 경우 true
  - public boolean isEmpty()
    - this._head가 null일 경우 true
  - public int size()
    - this._size를 반환

Member Functions의 구현

- LinkedList 의 Public Member function의 구현
  - public boolean add(T anElement)
    - List가 가득 차 있는지 확인
    - List의 가장 앞(this._head)에 anElement를 삽입
    - 자세한 구현에 대한 설명 생략 모를 경우 지난 학기 실습자료를 참고

Member Functions의 구현

LinkedList 의 Public Member function의 구현

```
public Iterator iterator()
{
    return new Iterator();
}

public class Iterator
{
    private Node _nextNode;
    private Iterator()
    {
        this._nextNode = _head;
    }
    public boolean hasNext()
    {
        return (this._nextNode != null);
    }
    public T next()
    {
        if(this._nextNode == null){
            return null;
        }
        else {
            T t = (T) this._nextNode.element();
            this._nextNode = this._nextNode.next();
            return t;
        }
    }
}
```

CircularLinkedQueue<T> Class

비공개 인스턴스 변수

```
public class CircularLinkedQueue<T> {
    // 비공개 인스턴스 변수
    private static final int DEFAULT_INITIAL_CAPACITY = 5 ;
    private int                  _maxSize ;
    private int                  _size ;
    private Node   _rear ;
```

Member Functions의 구현

- CircularLinkedQueue<T>의 Public Member function의 구현
  - 지난 학기 실습 10주차 자료를 확인하여 구현
  - http://winslab.cnu.ac.kr/lecture/2013/spring/ds/slides/prac/DS1_10.pdf
  - 보고서에 CircularLinkedQueue의 삽입을 분석한 내용을 넣을 것

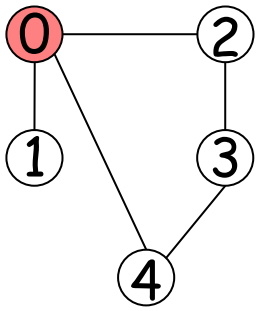
Coloring Class

bfs() 이해하기

주: Coloring을 위해 구현할 필요는 없음
구현은 runColoring()을 하면 됨

```
private void bfs()
{
    int  fromVertex, toVertex ;
    boolean [] visited = new boolean[this._graph.numOfVertices()];
    for (int v = 0 ; v < this._graph.numOfVertices() ; v++)
        visited[v] = false ;
    CircularLinkedQueue<Integer> bfsQ = new CircularLinkedQueue();

    visited[this._startingVertex] = true ;
    bfsQ.enQueue(this._startingVertex);

    while (! bfsQ.isEmpty()) {
        fromVertex = bfsQ.deQueue();
        AdjacencyMatrixGraph.Iterator it = this._graph.iterator(fromVertex);
        while (! it.hasNext() ) {
            toVertex= it.next();
            if (! visited[toVertex] ) {
                this.visit(toVertex);
                visited[toVertex] = true ;
                bfsQ.enQueue(toVertex);
            }
        }
    }
}
```

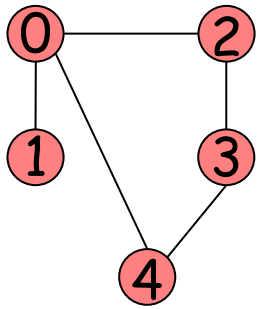
Coloring_bfs()



visited[]

[0]	[1]	[2]	[3]	[4]
T	F	F	F	F

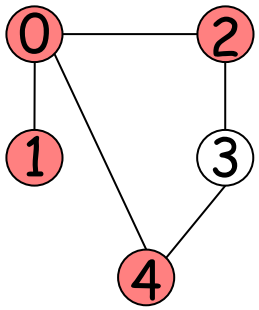
bfsQ: →0→



visited[]

[0]	[1]	[2]	[3]	[4]
T	T	T	T	T

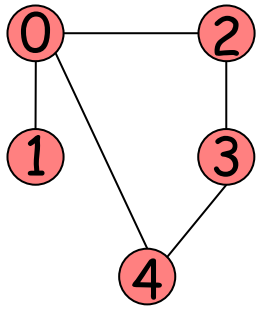
bfsQ: →3→4→



visited[]

[0]	[1]	[2]	[3]	[4]
T	T	T	F	T

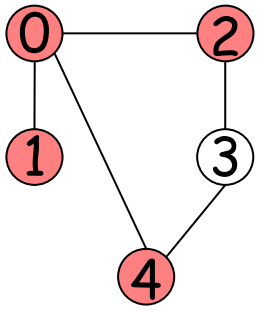
bfsQ: →4→2→1→



visited[]

[0]	[1]	[2]	[3]	[4]
T	T	T	T	T

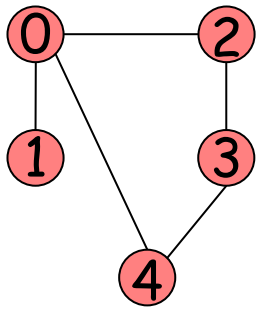
bfsQ: →3→



visited[]

[0]	[1]	[2]	[3]	[4]
T	T	T	F	T

bfsQ: →4→2→



visited[]

[0]	[1]	[2]	[3]	[4]
T	T	T	T	T

bfsQ: →→

Coloring – 비공개 인스턴스 변수

```
public class Coloring {
    private enum Color{NONE, RED, BLUE};
    private Color [] _color; // 각 vertex의 color를 저장할 배열
    private int _startingVertex;
    LinkedList<Edge> _sameColorEdges; // 끝 색이 같은 edge들의 리스트
    private AdjacencyMatrixGraph _graph;
```

Coloring의 Public Method

- Coloring 의 Public Member function의 사용법
  - public Coloring(AdjacencyMatrixGraph givenGraph)
    - 생성자
  - public LinkedList<Edge> sameColorEdges()
    - 같은 색깔을 가진 Edge를 가지고 있는 EdgeList를 반환한다
  - public void runColoring()
    - Coloring을 한다.
  - public Iterator iterator()
  - public class Iterator

Coloring 의 Private Method

- Coloring 의 Private Member function의 사용법
  - private void checkColors()
    - 그래프를 탐색하여서, 모든 edge를 검사하여 끝 색이 같은 edge들의 리스트를 만든다.

Member Functions의 구현

- Coloring 의 Public Member function의 구현
  - public Coloring(AdjacencyMatrixGraph givenGraph)
    - _startingVertex를 0으로 초기화
    - givenGraph를 this._graph에 저장
    - this._sameColorEdges를 새로 생성
    - this._color를 this._graph.numOfVertices만큼 생성
    - 생성된 this._color의 모든 값을 Color.NONE로 초기화
  - public LinkedList<Edge> sameColorEdges()
    - _sameColorEdges를 반환

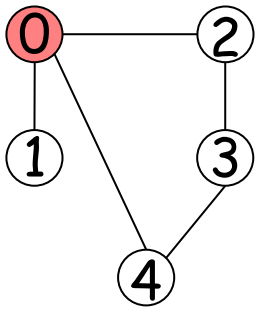
Member Functions의 구현

Coloring 의 Public Member function의 구현

bfs() 의 응용

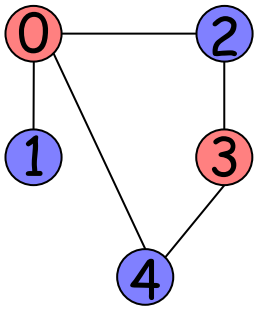
```
public void runColoring()
{
    int tailVertex, headVertex;
    Color   headVertexColor ;
    int     numOfVertices = this._graph.numOfVertices() ;

    // 출발 vertex의 color는 RED로 함
    this._color[this._startingVertex] = Color.RED;

    CircularLinkedQueue<Integer> bfsQ = new CircularLinkedQueue();
    bfsQ.enQueue(this._startingVertex);

    while (! bfsQ.isEmpty() ) {
        tailVertex = (Integer)bfsQ.deQueue();
        headVertexColor = ((this._color[tailVertex])== Color.RED)
                ? Color.BLUE : Color.RED ;
        AdjacencyMatrixGraph.Iterator it = this._graph.iterator(tailVertex);
        while ( it.hasNext() ) {
            headVertex= it.next();
            if (this._color[headVertex] == Color.NONE) {
                this._color[headVertex] = headVertexColor ;
                bfsQ.enQueue(headVertex) ;
            }
        }
    }
    this.checkColors();
}
```

runColoring()



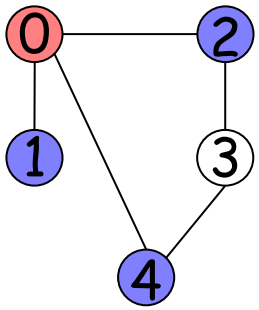
[0]	[1]	[2]	[3]	[4]
R	N	N	N	N

bfsQ: →0→



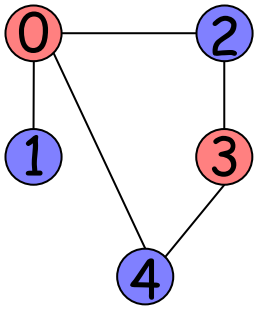
[0]	[1]	[2]	[3]	[4]
R	B	B	R	B

bfsQ: →3→4→



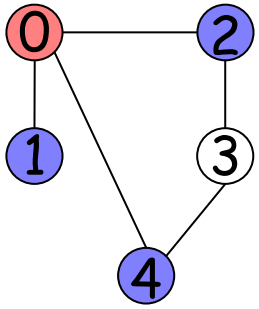
[0]	[1]	[2]	[3]	[4]
R	B	B	N	B

bfsQ: →4→2→1→



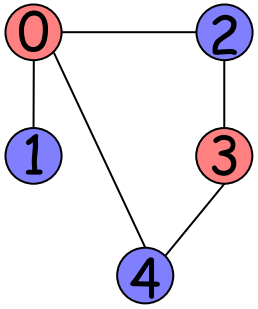
[0]	[1]	[2]	[3]	[4]
R	B	B	R	B

bfsQ: →3→



[0]	[1]	[2]	[3]	[4]
R	B	B	N	B

bfsQ: →4→2→



[0]	[1]	[2]	[3]	[4]
R	B	B	R	B

bfsQ: →→

Member Functions의 구현

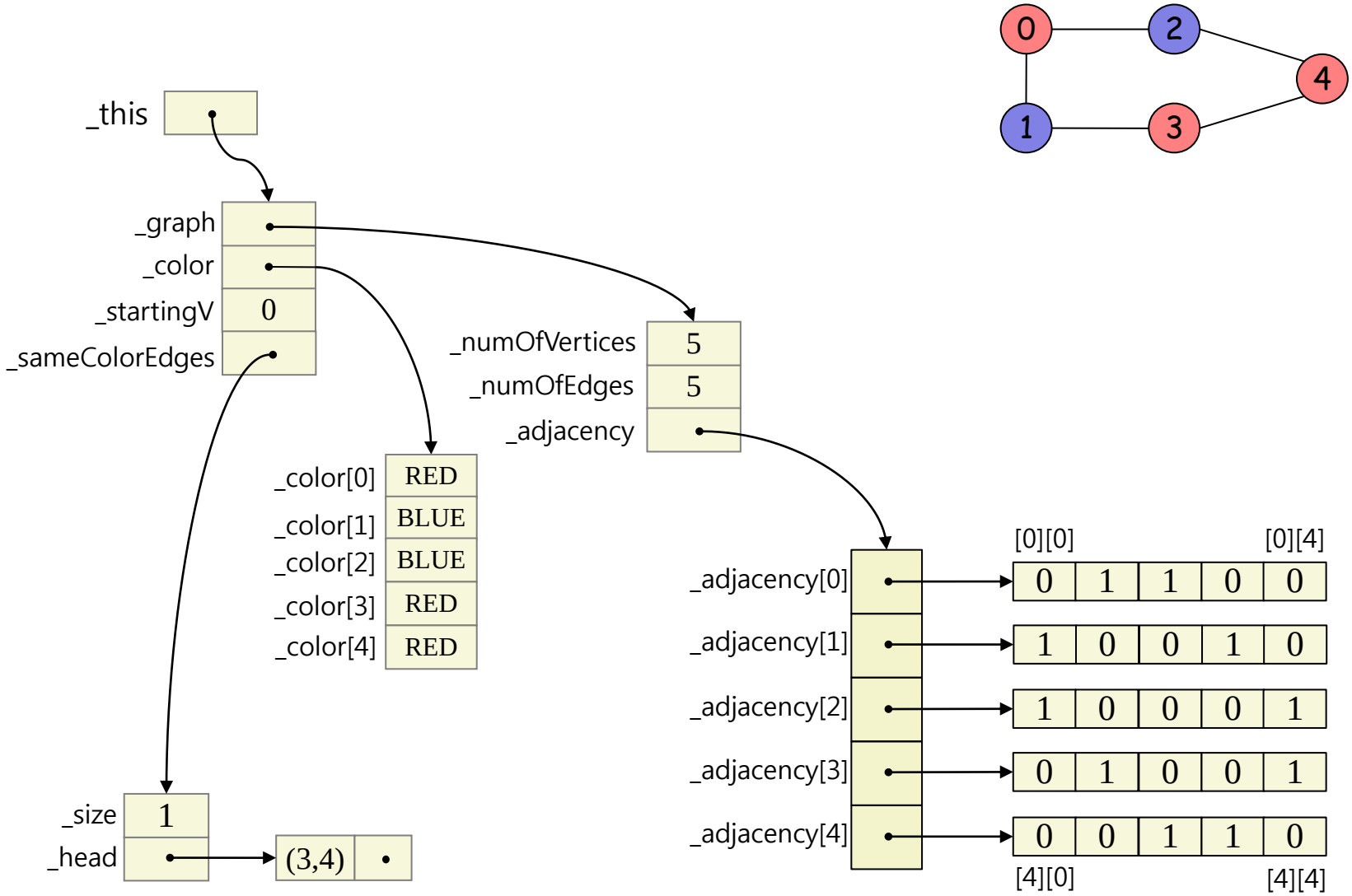
- Coloring 의 Public Member function의 구현
  - public Iterator iterator()
    - 새로운 Iterator Class를 생성하여 반환
  - public class Iterator
    - AdjacencyMatrixGraph에서 구현한 Iterator를 참고하여 구현
    - private Iterator()
    - public boolean hasNext()
    - public Color next()

Member Functions의 구현

Coloring 의 Private Member function의 구현

```
private void checkColors()
{
    Edge    e ;
    int tailVertex, headVertex;
    int numOfVertices = this._graph.numOfVertices();

    for ( tailVertex = 0 ; tailVertex < numOfVertices ; tailVertex++ ) {
        AdjacencyMatrixGraph.Iterator it = this._graph.iterator(tailVertex);
        while ( it.hasNext() ) {
            headVertex = it.next();
            if ( this._color[tailVertex] == this._color[headVertex]){
                Edge newEdge = new Edge(tailVertex, headVertex);
                this._sameColorEdges.add(newEdge);
            }
        }
    }
}
```

Coloring 객체의 구조



과제 제출

© J.-H.Kang, CNU

과제 제출

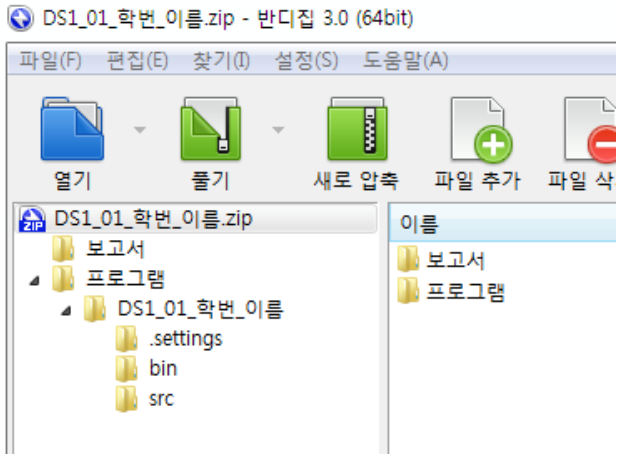
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_02_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 9월 10일(화) 23시59분까지
  - 시간 내 제출 엄수

  - 제출을 하지 않을 경우 0점 처리하고, 숙제를 50% 이상 제출하지 않으면 F 학점 처리하며, 2번 이상 제출하지 않으면 A 학점을 받을 수 없다.

과제 제출

- 파일 이름 작명 방법
  - DS2_02_학번_이름.zip
  - 폴더의 구성
    - DS2_02_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_02_학번_이름.java
      - 보고서
        - 이곳에 보고서 문서 파일을 저장한다.
        - 입력과 실행 결과는 화면 image로 문서에 포함시킨다.
        - 문서는 pdf 파일로 만들어 제출한다.



보고서 작성 방법

- 겉장
  - 제목: 자료구조 실습 보고서
  - [제xx주] 숙제명
  - 제출일
  - 학번/이름
- 내용
  1. 프로그램 설명서
     1. 주요 알고리즘 /자료구조 /기타
     2. 함수 설명서
     3. 종합 설명서 : 프로그램 사용방법 등을 기술
  2. 구현 후 느낀 점 : 요약의 내용을 포함하여 작성한다.
  3. 실행 결과 분석
     1. 입력과 출력 (화면 capture : 실습예시와 다른 예제로 할 것)
     2. 결과 분석

     ----- 표지 제외한 3번까지의 내용을 A4 세 장 내외의 분량으로 작성 할 것 -----
  4. 소스코드 : 화면 capture가 아닌 소스를 붙여넣을 것

     소스는 장수 제한이 없음.

[제 2 주 실습] 끝