[제 5 주 실습]

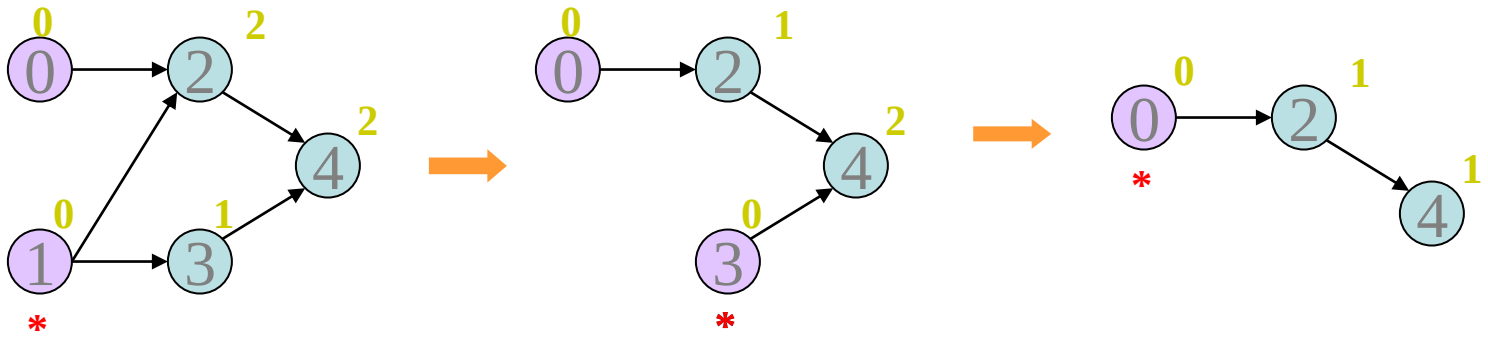
위상 정렬

강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

Topological Sort

Example: Topological Sort

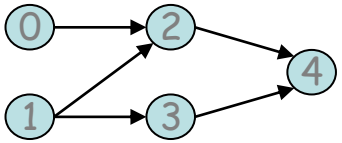


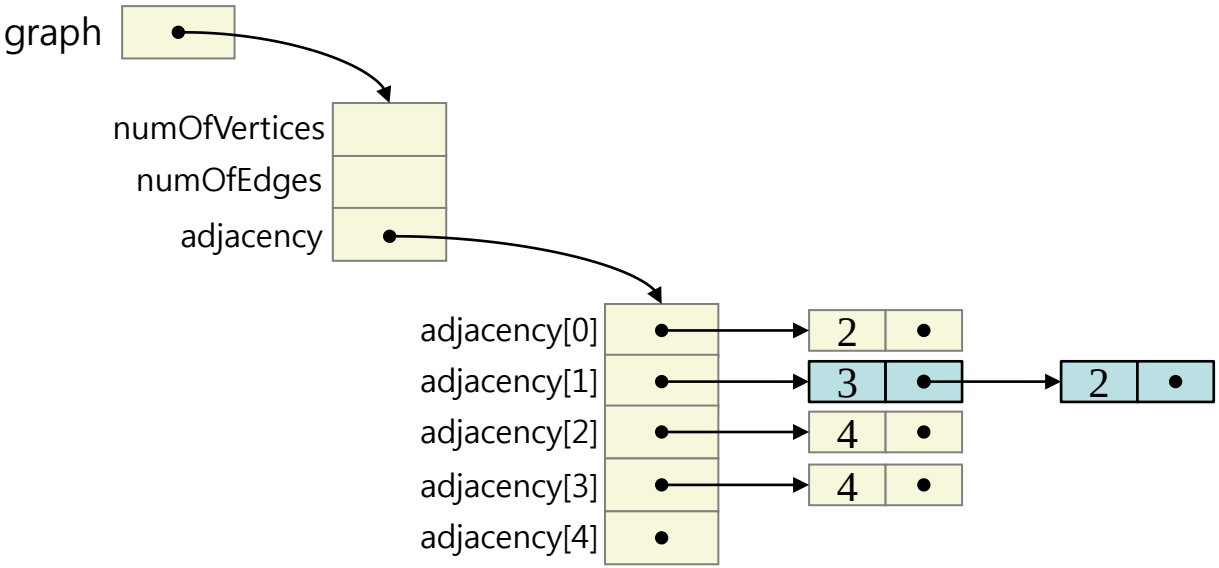
Algorithm

```
1.  Input the AOV network.
    (Let N be # of vertices)
2.  i = 0 ;
3.  while ( (i < N) && (there is a vertex with no predecessors) ) {
4.          pick a vertex v which has no predecessors ;
5.          output v ;
6.          delete v and all edges adjacent from v ;
7.          i++ ;
8.  }
9.  if ( i < N) {
10.         ..... // We cannot find topological order since the graph has a cycle.
11. }
12. else {
13.         ..... // We have got a topological order
14. }
```

Data Representation

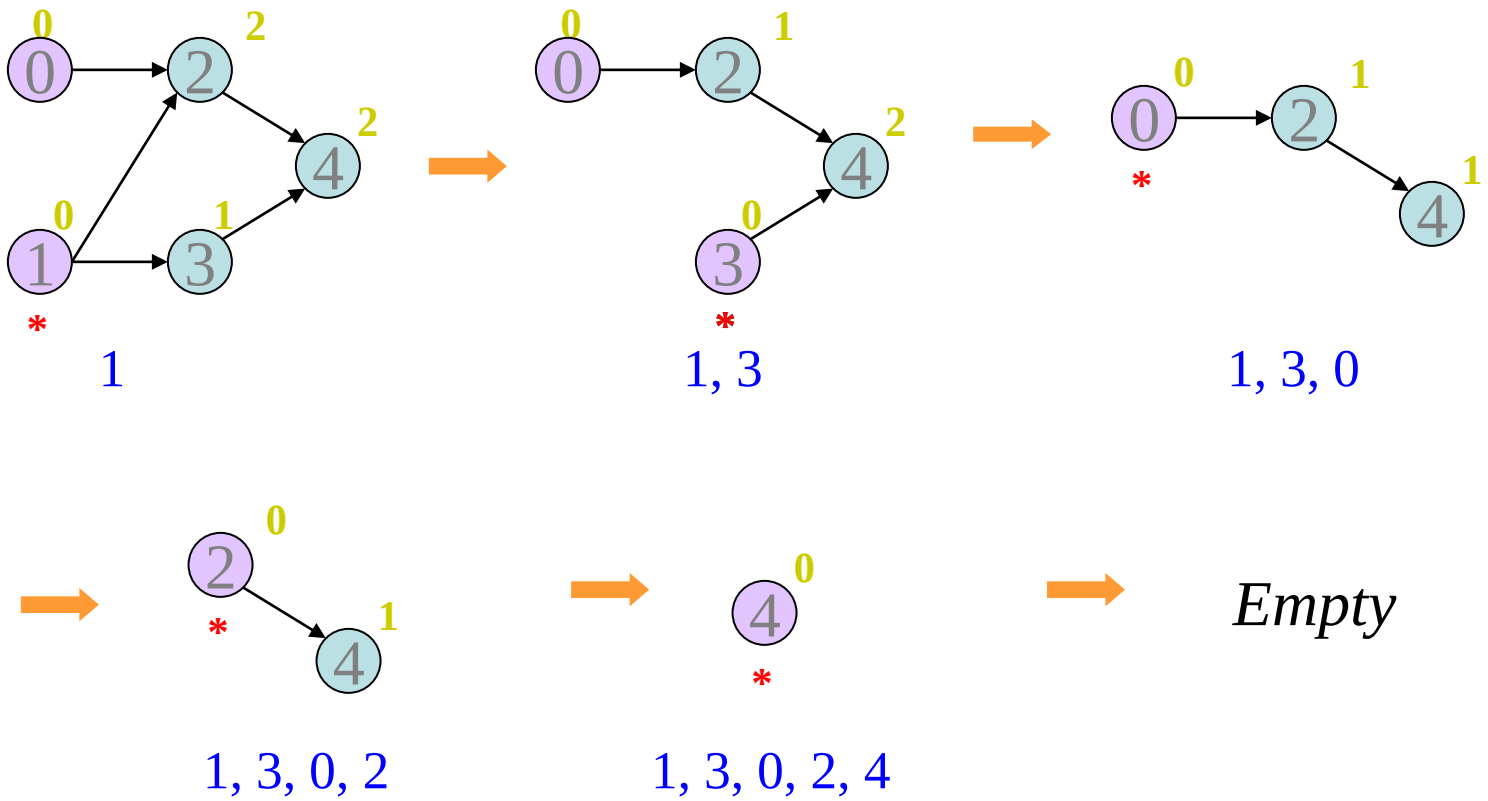
Graph by Adjacency List





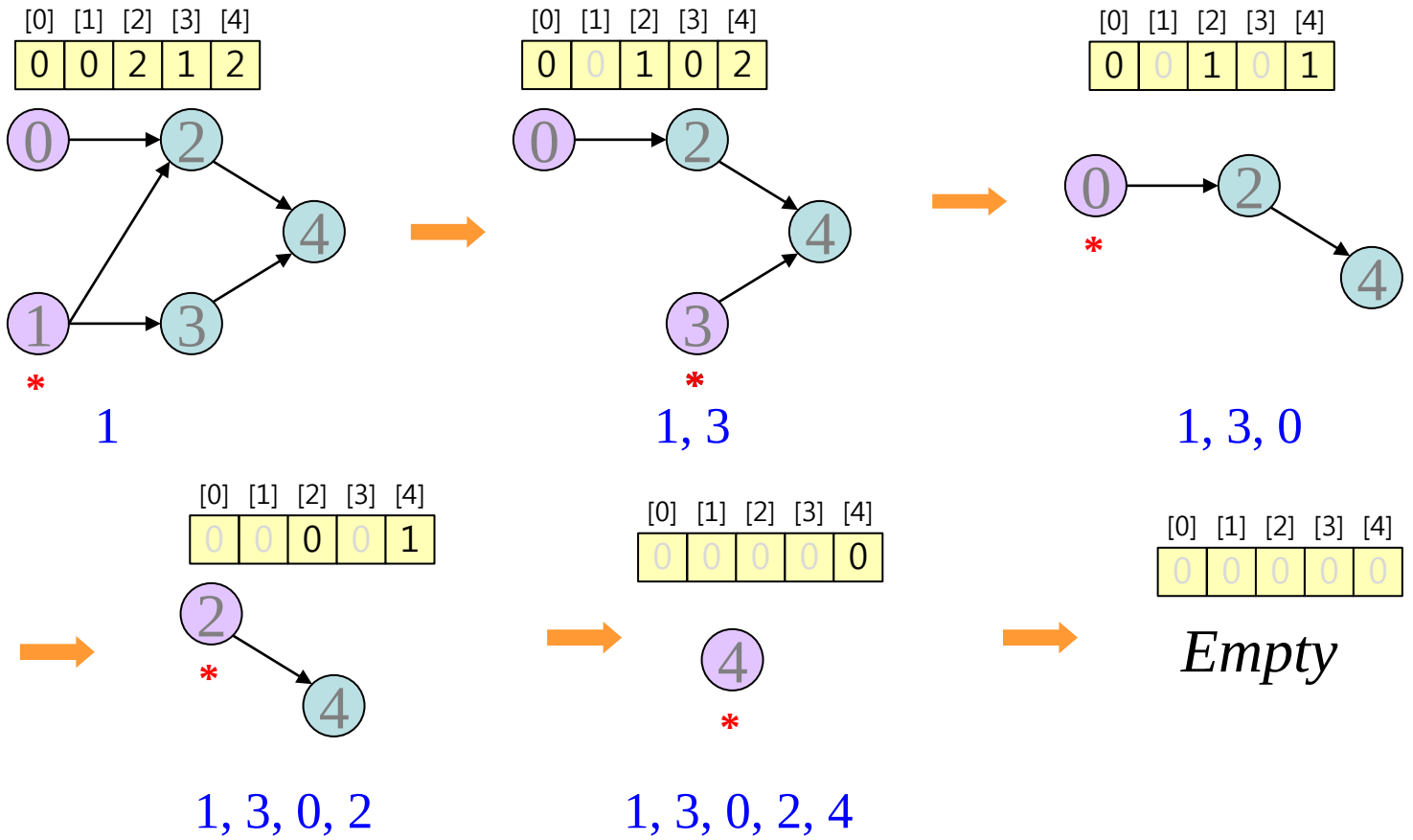
Vertices with No Immediate Predecessors

- How to find and manage the vertices with no immediate predecessors?



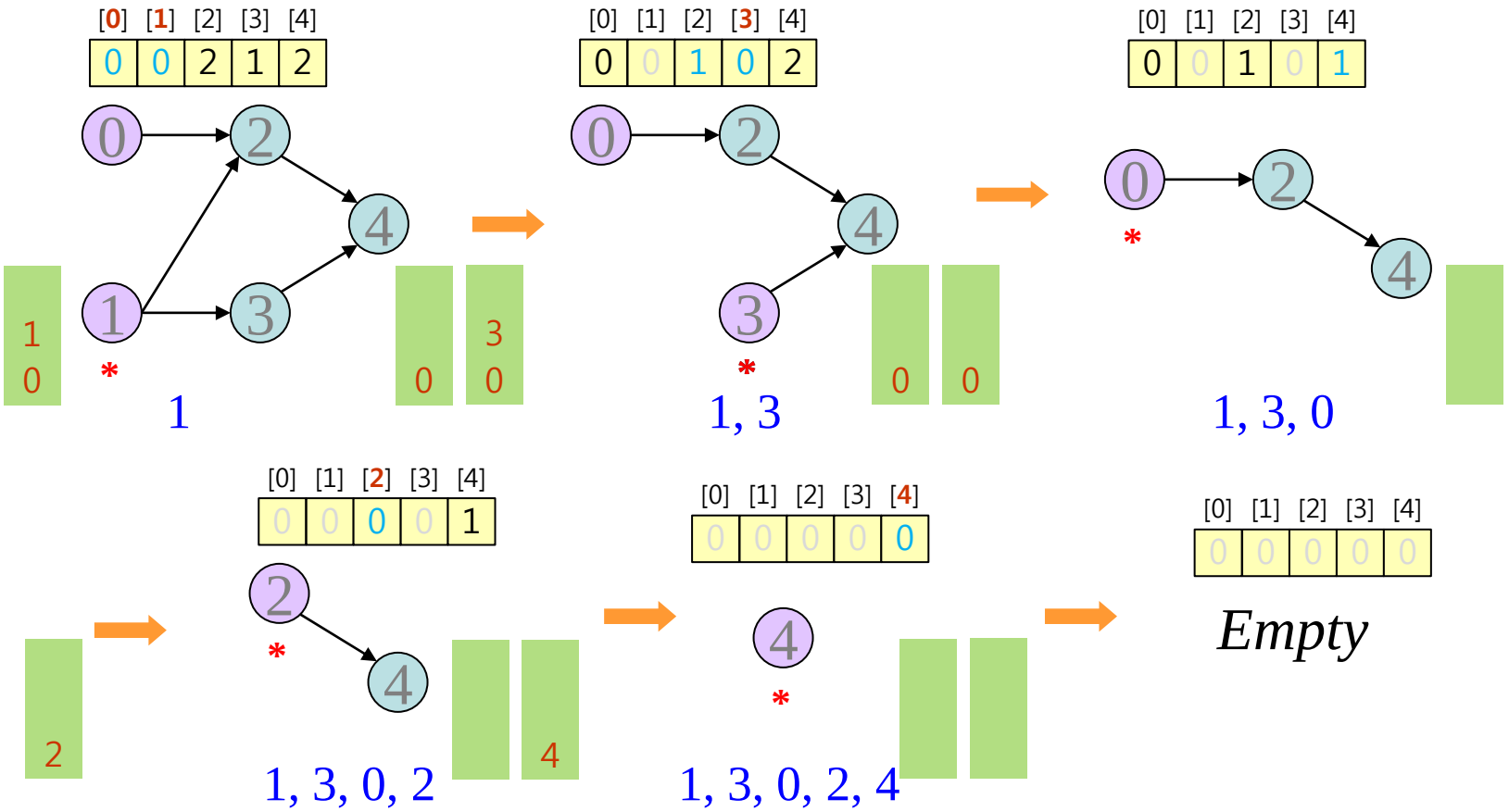
Vertices with No Immediate Predecessors

- How to find and manage the vertices with no immediate predecessors?
  - For each vertex, maintain the number of immediate predecessors.



How to find the vertices with no immediate predecessors efficiently ?

- Use a stack for such vertices! : zeroCountVertices



[문제 5]
Topological Sorting

문제 개요

- Topological Sorting을 구현한다
  - 입력: Dag (Directed Acyclic Graph)
  - 출력: Topologically sorted list

- 구현:
  - 그래프는 adjacency list로 표현한다.

입력

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

출력

- 그래프의 출력

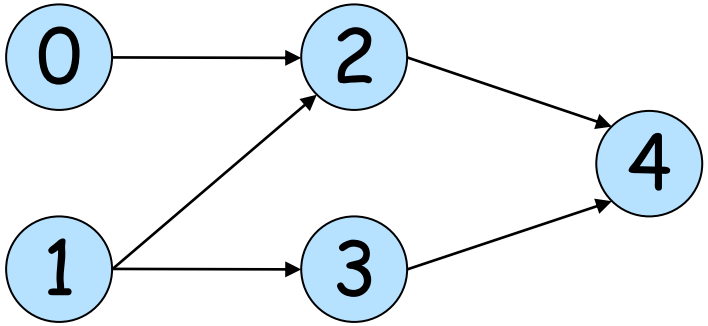
  [0] -> 2

  [1] -> 2 -> 3

  [2] -> 4

  [3] -> 4

  [4]



- 위상 정렬된 리스트 출력

  1 – 3 – 0 – 2 – 4

- 실행의 중간 과정도 <출력의 예>와 같이 보여준다.

❑출력의 예

```
- 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: 1 2
? Edge를 입력하시오: 1 5
[Error] 입력 오류입니다.
? Edge를 입력하시오: 1 3
? Edge를 입력하시오: 2 4
? Edge를 입력하시오: 0 2
[Error] 입력 오류입니다.
? Edge를 입력하시오: 6 3
[Error] 입력 오류입니다.
? Edge를 입력하시오: 3 4

! 입력된 그래프는 다음과 같습니다:
생성된 그래프 :
[0] -> 2(0)
[1] -> 3(0)  2(0)
[2] -> 4(0)
[3] -> 4(0)
[4] ->

- 각 vertex의 선행 vertex 수 :
[0] 0
[1] 0
[2] 2
[3] 1
[4] 2

- Pop & Output : 1
- 각 vertex의 선행 vertex 수 :
[0] 0
[1] 0
[2] 1
[3] 0
[4] 2
- 출력 가능한 vertex 들의 stack : <Top>3 0 <Bottom>
- Pop & Output : 3
- 각 vertex의 선행 vertex 수 :
[0] 0
[1] 0
[2] 1
[3] 0
[4] 1
- 출력 가능한 vertex 들의 stack : <Top>0 <Bottom>
- Pop & Output : 0
- 각 vertex의 선행 vertex 수 :
[0] 0
[1] 0
[2] 0
[3] 0
[4] 1
- 출력 가능한 vertex 들의 stack : <Top>2 <Bottom>
- Pop & Output : 2
- 각 vertex의 선행 vertex 수 :
[0] 0
[1] 0
[2] 0
[3] 0
[4] 0
- 출력 가능한 vertex 들의 stack : <Top>4 <Bottom>
- Pop & Output : 4
- 각 vertex의 선행 vertex 수 :
[0] 0
[1] 0
[2] 0
[3] 0
[4] 0
- 출력 가능한 vertex 들의 stack : <Top><Bottom>
! Topological Sort의 결과는 다음과 같습니다.
1 3 0 2 4
<프로그램을 종료합니다>
```

Algorithm: More Specific

```
int         predecessorCount [] ;
Stack       zeroCountVertices ;

1.  Input the AOV network (i.e. directed graph).
    (Let N be # of vertices)
2.  Set predecesorCount[] ;
3.  Set zeroCountVertices by inserting the vertices with no predecessors ;
4.  i = 0 ;
5.  while ( ! zeroCountVertices .isEmpty() ) {
6.          v = zeroCountVertices.pop() ; // pick a vertex v which has no predecessors ;
7.          output v ;
8.          for (each vertex w adjacent from v) { // delete v and all edges adjacent from v
9.                  predecessorCount[w] -- ;
10.                 if (predecessorCount[w] == 0 )
11.                         zeroCountVertices.push(w) ;
12.         }
13.         i++ ;
14. }
15. if ( i < N ) {
16.         ..... // We cannot find topological order since the graph has a cycle.
17. }
18. else {
19.         ..... // We have got a topological order
20. }
```

How to initialize PredecessorCount[]

- int predecessorCount[] ;
  - size: number of vertices
  - predecessorCount[i]: number of immediate predecessors of vertex i.

- How to set the initial value of predecessorCount[] ?
  - Initially, set all zero.
  - For each edge <i, j> in the graph,
    increment predecessorCount[j] by 1.

- How to maintain?
  - Whenever we delete a vertex v with no immediate predecessors from the graph:
    For each edge <v, w> which starts from v,
    decrement predecessorCount[w] by 1.

이 과제에서 필요한 객체는?

- Application
  - AdjacencyListGraph
    - Node
  - TopologicalSort
    - ArrayList
    - ArrayStack
  - Edge

Application의 공개 함수는?

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

AdjacencyListGraph의 공개 함수는?

- 사용자에게 필요한 함수
  - public AdjacencyListGraph(int givenNumOfVertices)
  - public boolean doesVertexExist (int aVertex)
  - public boolean doesEdgeExist (Edge anEdge)
  - public int numOfVertices ()
  - public int numOfEdges ()
  - public boolean addEdge(Edge anEdge)
  - public AdjacentVerticesIterator adjacentVertircesIterator(int aVertex)
  - public class AdjacentVerticesIterator

Node〈T〉 의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - [문제2]의 것을 복사해서 사용 / 수정 하지 않음
  - public Node()
  - public Node(T givenElement)
  - public Node(T givenElement, Node givenNode)

  - public T element()
  - public void setElement(T anElement)

  - public Node next()
  - public void setNext(Node anNode)

TopologicalSort의 공개 함수는?

- 사용자에게 필요한 함수
  - public TopologicalSort(AdjacencyListGraph givenGraph)
  - public ArrayList sortedList()
  - public boolean perform()

ArrayList〈T〉의 공개 함수는?

- 사용자에게 필요한 함수
  - [문제2]의 것을 복사해서 사용 / 수정 하지 않음
  - public ArrayList()
  - public ArrayList(int givenMaxSize)
  - public boolean isFull()
  - public boolean isEmpty()
  - public int size()
  - public boolean add(T anElement)
  - public ArrayListIterator arrayListIterator()
  - public class ArrayListIterator

ArrayStack〈Element〉의 공개 함수는?

- 사용자에게 필요한 함수
  - public ArrayStack()
  - public ArrayStack(int givenMaxSize)

  - public boolean isEmpty()
  - public boolean isFull()
  - public int length()

  - public boolean push(Element anElement)
  - public Element pop()
  - public Element peek()

  - public ArrayStackIterator arrayStackIterator()
  - public class ArrayStackIterator

Edge의 공개 함수는?

- 사용자에게 필요한 함수
  - [문제4]의 것을 복사해서 사용
  - public Edge (int givenTailVertex, int givenHeadVertex)
  - public Edge (int givenTailVertex, int givenHeadVertex, int givenCost)

  - public void setTailVertex (int aVertex)
  - public int tailVertex ()
  - public void setHeadVertex (int aVertex)
  - public int headVertex ()
  - public void setCost (int aCost)
  - public int cost()

  - public boolean sameEdgeAs (Edge anEdge)

DS2_05_학번_이름 Class의 구조

```
/* 항상 사용하는 main Class 구조
 * 과제 제출시 반드시 포함 되어 있어야 함*/
public class DS2_03_학번_이름 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Application application = new Application();
        application.run ();
    }
}
```

Class “Application“

Application – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class Application {
    private AdjacencyListGraph _graph;
    private Scanner _scanner;
    private TopologicalSort _topologicalSort;
```

Application의 공개 함수 run()의 구현

```
public void run(){
    if ( this.inputAndMakeGraph() ) {
        this.showGraph() ;
        this._topologicalSort = new TopologicalSort(this._graph);
        this._topologicalSort.perform();
        this.showSortedList();
        System.out.println("<프로그램을 종료합니다>");
    }
    else {
        System.out.println("₩n₩n[오류]그래프 삽입 실패") ;
    }
}
```

Application의 Private Method

- Application의 Private Member function의 사용법
  - private boolean inputAndMakeGraph()
    - 키보드로부터 그래프를 입력 받아 그래프 객체로 저장한다.
    - 그래프가 정상적으로 입력되었으면 true, 아니면 false를 얻는다.
    - [문제2]에서 사용한 것을 수정 (이번 Graph는 Cost가 없음)
  - private void showGraph()
    - 입력된 그래프를 보여준다.
    - [문제2]와 동일
  - private void showSortedList()
    - 찾은 정렬 리스트를 출력한다.

Member Functions의 구현

Application의 Private Member function의 구현

private boolean inputAndMakeGraph()

```
private boolean  inputAndMakeGraph()
{
    int countEdges;
    int numOfVertices ;
    int numOfEdges ;

    System.out.println("- 그래프의 vertex수와 edge수를 입력 받아야 합니다.");
    System.out.print("? 그래프의 vertex 수를 입력 하시오:");
    _scanner = new Scanner(System.in);
    numOfVertices = _scanner.nextInt();
    System.out.print("? 그래프의 edge 수를 입력 하시오:");
    numOfEdges = _scanner.nextInt();

    // AdjacencyMatrixGraph 생성
    // 입력받은 numOfVertices와 numOfEdges를 이용하여 생성한다.
    this._graph = new AdjacencyListGraph(numOfVertices);

    countEdges = 0;
    System.out.println("- 그래프의 edge를 반복하여 " + numOfEdges +" 개 입력 받아야 합니다.");
    System.out.println("  하나의 edge는 (vertex1 vertext2)의  순서로 표시됩니다.");
    System.out.print("? Edge를 입력하시오: ");
    while(_scanner.hasNext()) {
        Edge anEdge = new Edge(_scanner.nextInt(), _scanner.nextInt());
        if(this._graph.addEdge(anEdge)){
            countEdges++;
        }
        else
            System.out.println("[Error] 입력 오류입니다.");

        if(countEdges == numOfEdges)
            break;
        System.out.print("? Edge를 입력하시오: ");
    }
    if(countEdges != numOfEdges)
        return false;
    else
        return true;
}
```

Member Functions의 구현

- Application의 Private Member function의 구현
  - private void showGraph()
    - [문제 4]의 showGraph를 그대로 사용 (변동 없음)

Member Functions의 구현

- Application의 Private Member function의 구현
  - private void showSortedList()

```
private void showSortedList()
{
    System.out.println("! Topological Sort의 결과는 다음과 같습니다.");
    ArrayList.ArrayListIterator resultListIterator =
            this._topologicalSort.sortedList().arrayListIterator();
    while(resultListIterator.hasNext())
        System.out.print(resultListIterator.next() + " " );
    System.out.println();
}
```

Class “AdjacencyListGraph”

비공개 인스턴스 변수

```
public class AdjacencyListGraph {
    private static final int NumOfVertices = 50;
    private Node<Edge>[] _adjacency;
    private int _numOfVertices;
    private int _numOfEdges;
```

AdjacencyListGraph 의 멤버 함수는?

- AdjacencyListGraph 의 Public Member function의 사용법
  - public AdjacencyListGraph(int givenNumOfVertices)
    - 생성자
  - public boolean doesVertexExist (int aVertex)
    - Vertex가 삽입 가능한 값인지 확인
  - public boolean doesEdgeExist (Edge anEdge)
    - Edge가 이미 있는지 확인
  - public int numOfVertices ()
    - Vertex 갯수를 확인
  - public int numOfEdges ()
    - 현재 삽입된 Edges의 개수를 확인
  - public boolean addEdge(Edge anEdge)
    - anEdge를 그래프에 삽입

AdjacencyListGraph 의 멤버 함수는?

- AdjacencyListGraph 의 Public Member function의 구현
  - public AdjacencyListGraph(int givenNumOfVertices)
    - this._numOfEdges를 0으로 초기화한다.
    - givenNumOfVertices 를 this._numOfVertices에 저장
    - Node 배열형 this._adjacency를 NumOfVertices만큼 생성
  - public int numOfVertices ()
    - this._numOfVertices를 반환
  - public int numOfEdges ()
    - this._numOfEdges를 반환

AdjacencyListGraph 의 멤버 함수는?

- AdjacencyListGraph 의 Public Member function의 구현
  - public boolean doesEdgeExist (Edge anEdge)

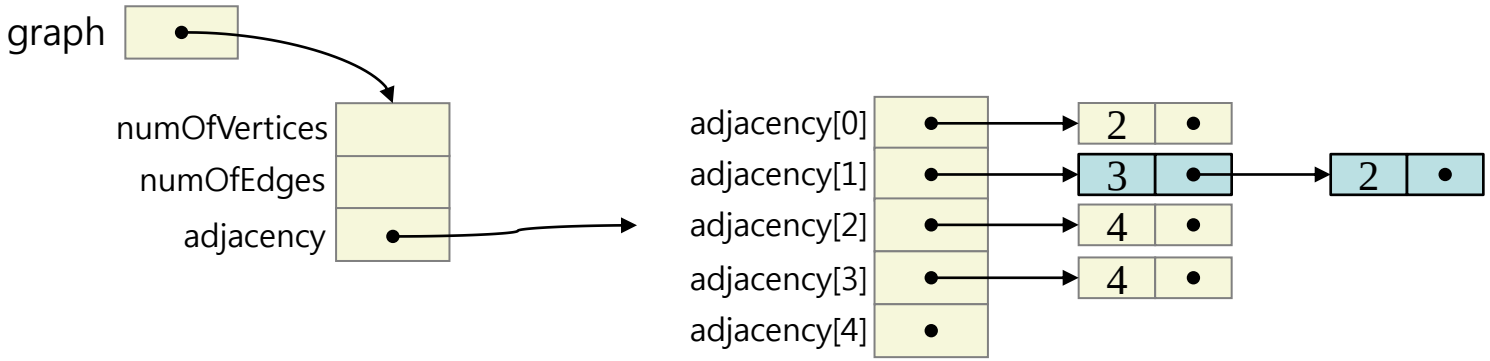
```
public boolean doesEdgeExist(Edge anEdge) {
    Node searchNode = this._adjacency[anEdge.tailVertex()];
    int found = 0;
    while(searchNode != null && found != 1)
    {
        if (anEdge.sameEdgeAs((Edge)searchNode.element()))
            found = 1;
        searchNode = searchNode.next();
    }

    if (found == 1)
        return true;
    else
        return false;
}
```

  - public boolean doesVertexExist (int aVertex)
    - AdjacencyMatrixGraph와 동일

AdjacencyListGraph 의 멤버 함수는?

- AdjacencyListGraph 의 Public Member function의 구현
  - public boolean addEdge(Edge anEdge)
    - anEdge의 tailVertex와 headVertex가 적합한 값(doesVertexExist 사용)인지 확인
      - 적합하지 않을 경우 false를 반환
    - anEdge가 이미 존재 하는 Edge(doesEdgeExist 사용) 인지 확인
      - 적합하지 않을 경우 false를 반환
    - this._adjacency의 anEdge.tailVertex에 anEdge를 값으로 하는 Node를 삽입
    - this._numOfEdges 하나 증가
    - 모든 작업이 끝나면 true를 반환



AdjacencyListGraph 의 멤버 함수는?

- AdjacencyListGraph 의 Public Member function의 구현
  - public AdjacentVerticesIterator adjacentVertircesIterator(int givenVertex) {
    - AdjacentVerticesIterator를 givenVertex로 새로 생성하여 반환
  - public class AdjacentVerticesIterator
    - 비공개 맴버 변수
      - private Node<Edge> _nextNode;
    - private AdjacentVerticesIterator(int givenVertex)
      - _adjacency의 givenVertex의 값을 this._nextNode에 저장
    - public boolean hasNext()
      - this._nextNode가 null인지 확인하여 null이 아닐 경우 true 반환
    - public Edge next()
      - 임시 변수 Edge e에 this._nextNode의 element()를 저장
      - this._nextNode에 this._nextNode의 next()를 저장
      - 임시변수 e를 반환

Class “TopologicalSort”

비공개 인스턴스 변수

```
public class TopologicalSort {
    private int   [] _predecessorCount ; // 각 vertex의 직전 선행자의 개수
    private ArrayStack<Integer>     _zeroCountVertices ; // 선행자가 없는 vertex의 리스트
    private ArrayList<Integer>  _sortedList ; // sort된 vertex 순서를 저장하는 리스트
    private AdjacencyListGraph _graph;
```

TopologicalSort의 멤버 함수는?

- TopologicalSort 의 Public Member function의 사용법
  - public TopologicalSort(AdjacencyListGraph givenGraph)
    - 생성자
    - 위상 정렬할 준비를 한다.
    - 각 속성들을 초기화 한다.
  - public ArrayList sortedList()
    - 정렬된 sortedList를 반환
  - public boolean perform()
    - 위상정렬을 실행
    - 사이클이 존재하여, 정상적인 완료를 할 수 없게 되면 FALSE로 종료
      - 이 검사를 위하여, 맨 마지막 문장으로 다음 return 문을 실행
      - return (
        (List_size(_this->sortedList) == Graph_numOfVertices(_this->graph)
        );

TopologicalSort 의 멤버 함수는?

- TopologicalSort 의 Public Member function의 구현
  - public TopologicalSort(AdjacencyListGraph givenGraph)

```
public TopologicalSort(AdjacencyListGraph givenGraph)
{
    this._graph = givenGraph;
    // predecessorCount[] 배열을 생성하고,
    // 선행 vertex 개수로 초기화한다.
    createAndSetPredecessorCount() ;
    // 선행 vertex가 없는 출력 가능한
    // vertex들을 유지하기 위한 stack을 생성힌다.
    int numOfVerticesInGraph = givenGraph.numOfVertices();
    _zeroCountVertices = new ArrayStack(numOfVerticesInGraph) ;
    // predecesorCount[]에서 값이 0인 vertex,
    // 즉 선행자가 없는 vertex를 찾아 stack에 삽입한다.
    pushVerticesWithNoPredecessors() ;
    // 위상 정렬되는 순서,
    // 즉 출력되는 순서대로 vertex를 보관하기 위한 리스트를 생성한다.
    _sortedList = new ArrayList(numOfVerticesInGraph) ;
}
```

  - public ArrayList sortedList()
    - this._sortedList를 반환

TopologicalSort 의 멤버 함수는?

- TopologicalSort 의 Public Member function의 구현
  - public boolean perform()

```
public boolean perform()
{
    int     tailVertex, headVertex;
    showPredecessorCount() ;
    while ( ! this._zeroCountVertices.isEmpty()) {
        tailVertex= this._zeroCountVertices.pop();
        this._sortedList.add(tailVertex);
        //List_insertIntoLast(_this->sortedList, fromVertex) ;
        System.out.println("- Pop & Output : " + tailVertex) ;
        //출력 예:  "- Pop & Output: 1"
        AdjacencyListGraph.AdjacentVerticesIterator adjacentVerticesIterator =
                this._graph.adjacentVertircesIterator(tailVertex);
        while (adjacentVerticesIterator.hasNext()) {
            headVertex = adjacentVerticesIterator.next().headVertex();
            _predecessorCount[headVertex]--;
            if ( _predecessorCount[headVertex] == 0 )
                this._zeroCountVertices.push(headVertex);
        }
        showPredecessorCount() ;
            // 출력 예: "각 vertex의 선행 vertex 수는 다음과 같습니다: ....."
        showZeroCountVertices() ;
            // 출력 예: "- 출력 가능한 vertex들의 stack: <Top> 1, 0 <Bottom>"
    }
    return ( this._sortedList.size() == this._graph.numOfVertices() );
}
```

디버깅용 출력

TopologicalSort 의 멤버 함수는?

- TopologicalSort 의 private Member function의 구현
  - private void createAndSetPredecessorCount()
    - _PredecessorCount를 생성 _graph의 numOfVertices만큼 생성
    - 배열 PredecessorCount[]를 초기화
    - 그래프 전체 edge를 탐색 하면서 Edge의 값 tailVertex, headVertex 에 대해, headVertex의 PredecessorCount, 즉 _predecessorCount[headVertex]를 하나 증가시킨다.
  - private void pushVerticesWithNoPredecessors()
    - 그래프를 탐색하여 설정한 _predecessorCount[]에서 값이 0인 vertex, 즉 직전선행자가 존재하지 않는 vertex들을 찾아 stack에 삽입한다.

TopologicalSort 의 멤버 함수는?

- TopologicalSort 의 private Member function의 구현
  - 디버깅용 함수들 – 실습을 위하여 확인 용으로만 사용
  - private void showPredecessorCount()
    - 각 vertex의 선행 vertex 수를 출력하는 함수
    - this._predecessorCount의 모든 값을 출력
  - private void showZeroCountVertices()
    - 출력 가능한 vertex 들의 stack내부 값을 출력
    - ArrayStack의 ArrayStackIterator를 이용하여 Stack 내부 값을 모두 출력

Class “ArrayStack<Element>”

비공개 인스턴스 변수

```
public class ArrayStack<Element> {
    private static final int   DEFAULT_MAX_STACK_SIZE = 100 ;
    private int                _maxSize ;
    private int                _top ;
    private  Element[]         _elements ;
```

Member Functions의 구현

- ArrayStack의 Public Member function의 구현
  - Stack의 구조 및 이해는 지난 학기 강의 자료구조및실습 7주차와 실습 7주차를 참고하여 작성하세요.
    - http://winslab.cnu.ac.kr/lecture/2013/spring/ds/slides/lec/[DS1-06-1]Stack.pdf
    - http://winslab.cnu.ac.kr/lecture/2013/spring/ds/slides/prac/DS1_07.pdf

Member Functions의 구현

- ArrayStack의 Public Member function의 구현
  - public ArrayStack()
  - public ArrayStack(int givenMaxSize)
    - _maxSize를 원하는 max 값으로 초기화
    - _top은 -1로 초기화
    - ArrayStack이 Element 형태로 선언 되어 있으므로 _element 배열은 Object로 초기화 해야 함
      - this._elements =

        (Element[]) new Object[this.DEFAULT_MAX_STACK_SIZE];

Member Functions의 구현

- ArrayStack의 Public Member function의 구현
  - public boolean isEmpty()
    - _top을 확인하여 비어있는지 확인
  - public boolean isFull()
    - _top을 확인하여 가득차 있는지 확인
  - public int length()
    - _top을 확인하여 삽입된 원소의 길이를 확인

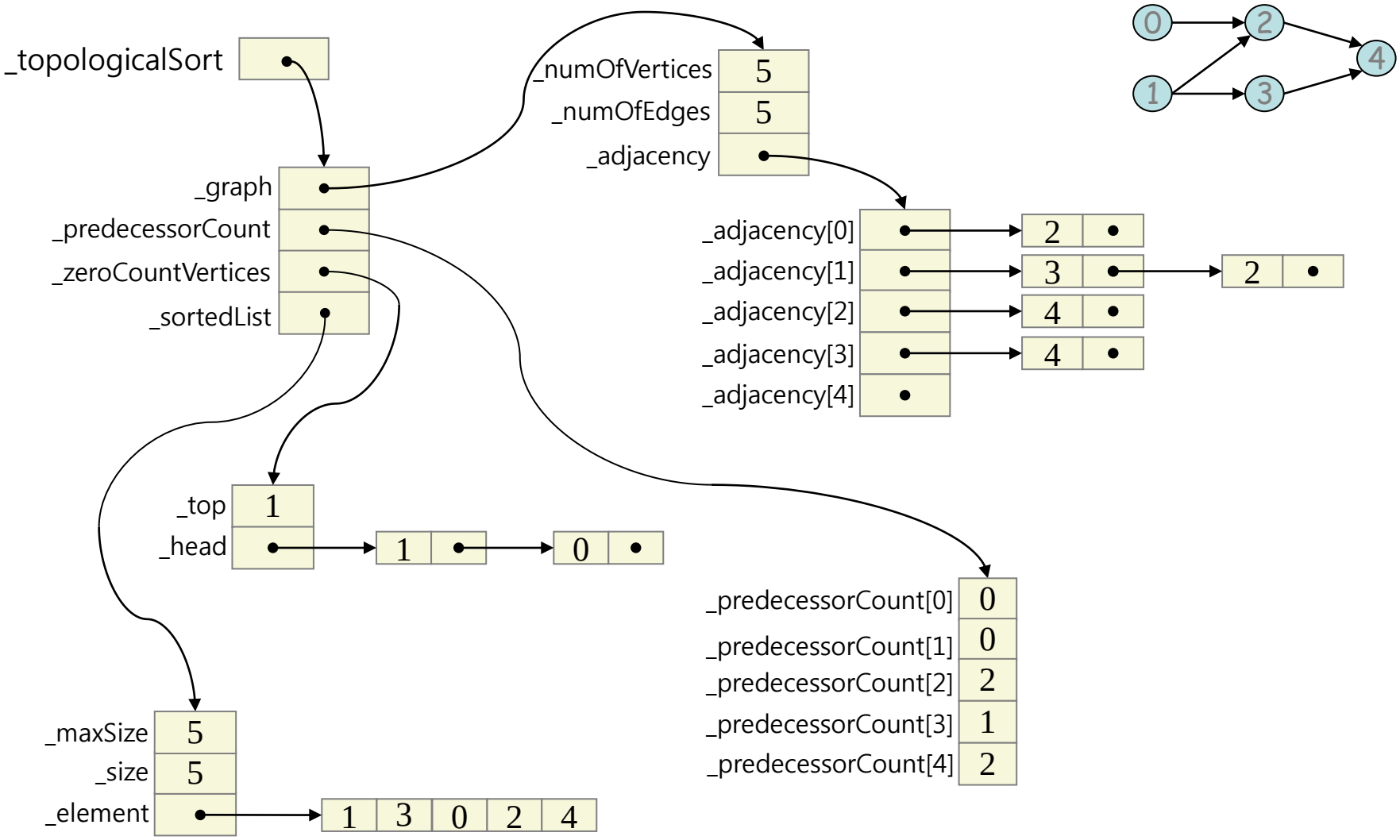
Member Functions의 구현

- ArrayStack의 Public Member function의 구현
  - public boolean push(Element anElement)
    - 가득차 있을 경우 false를 반환
    - 가득차 있지 않을 경우
      - _top을 1 증가
      - _element에 anElement를 삽입
  - public Element pop()
    - 비어 있으면 null을 반환
    - 비어 있지 않으면
      - _top에 있는 _element의 원소를 반환
      - _top을 1 감소

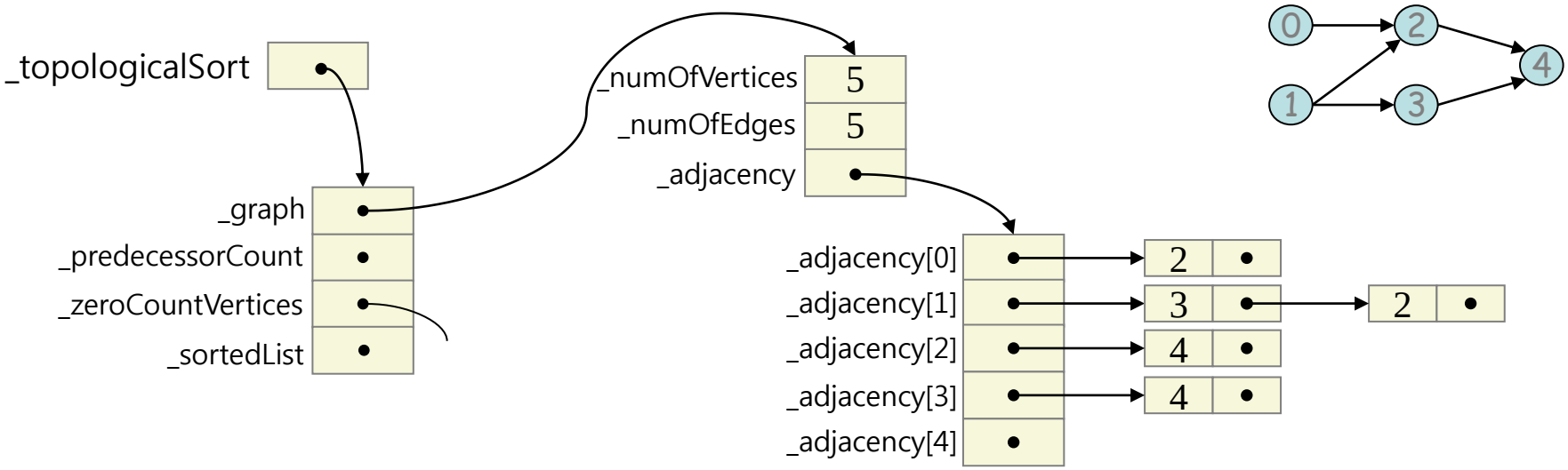
Member Functions의 구현

- ArrayStack의 Public Member function의 구현
  - public Element peek()
    - 비어 있으면 null을 반환
    - 비어 있지 않으면
      - _top에 있는 _element의 원소를 반환
  - public ArrayStackIterator arrayStackIterator()
  - public class ArrayStackIterator
    - 이전에 구현한 Iterator를 참고하여 StackIterator를 구현
    - Stack은 Top element가 가장 마지막에 있으므로 마지막부터 출력하도록 Iterator를 구현 하여야 함

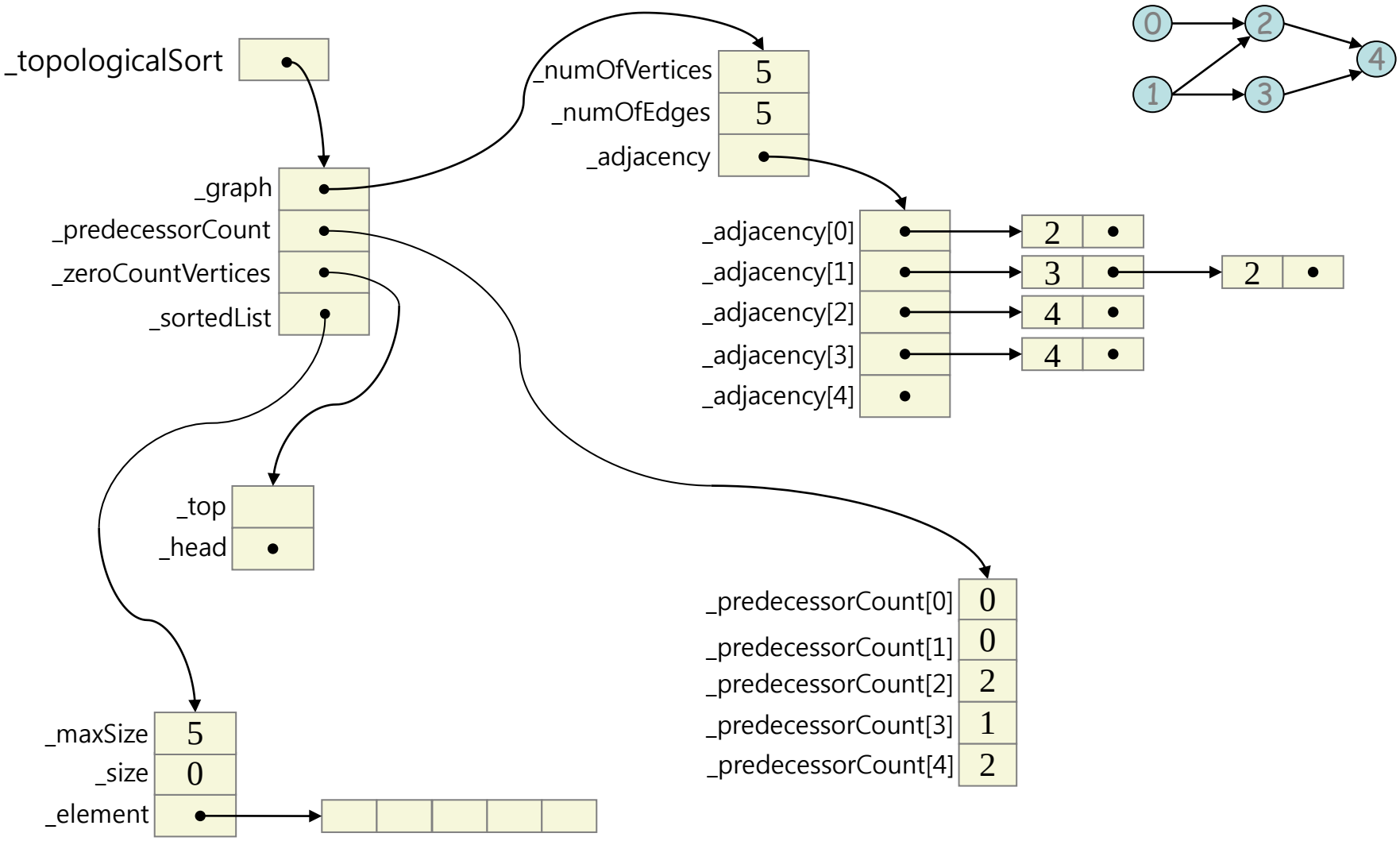
예: TopologicalSort 객체



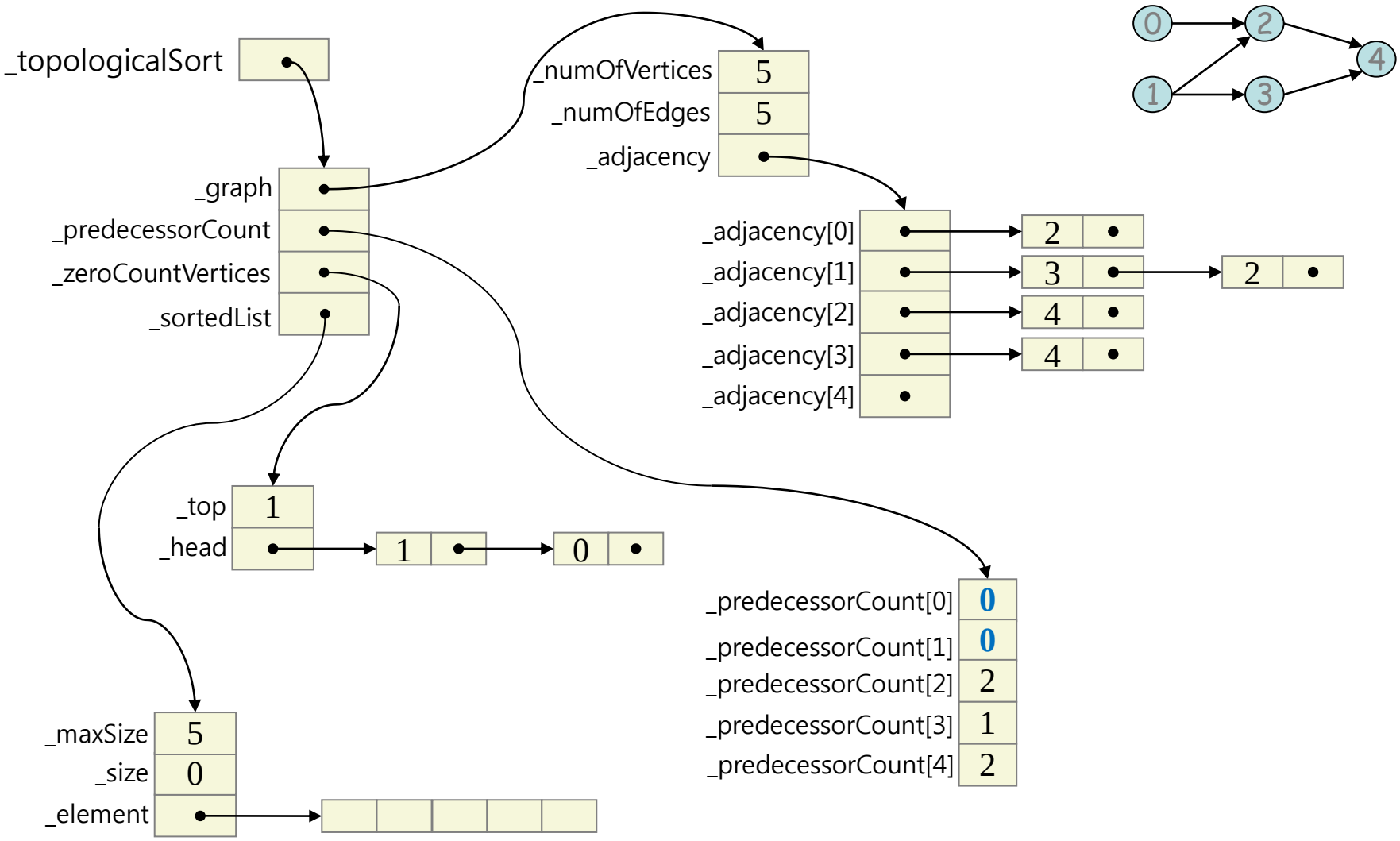
inputAndMakeGraph() 실행 직후



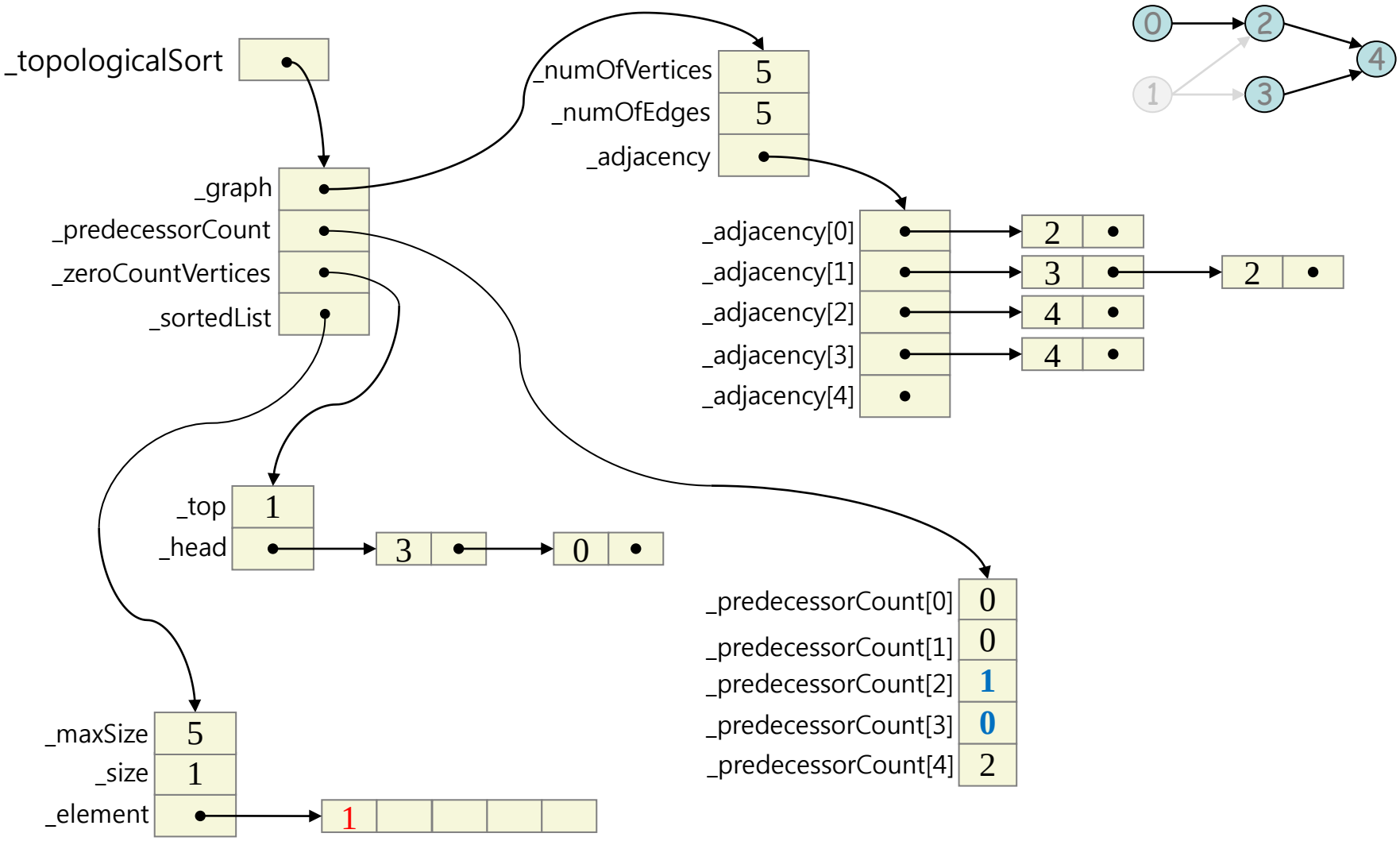
예: createAndSetPredecessorCount() 실행 직후



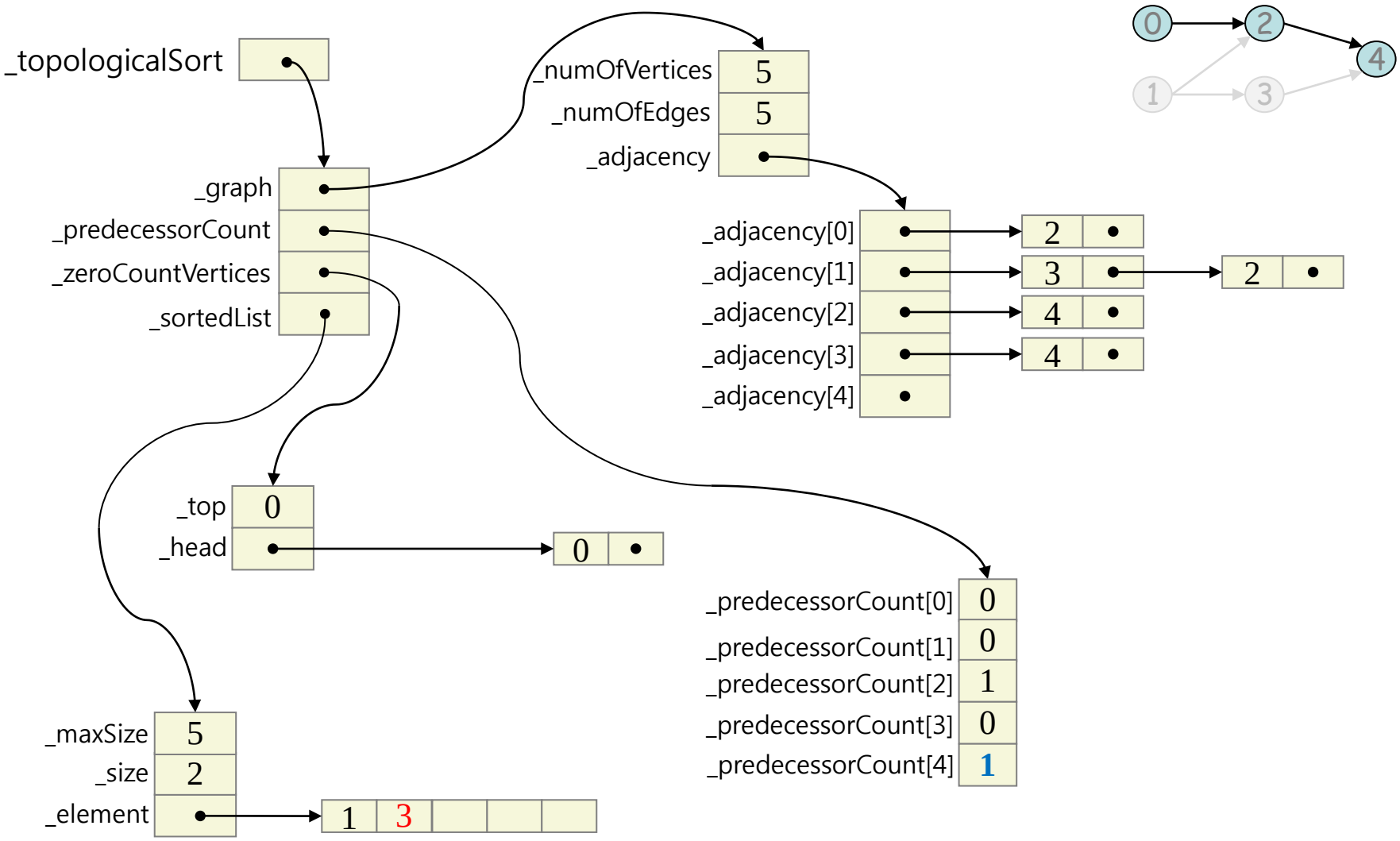
pushVerticesWithNoPredecessors() 실행 직후



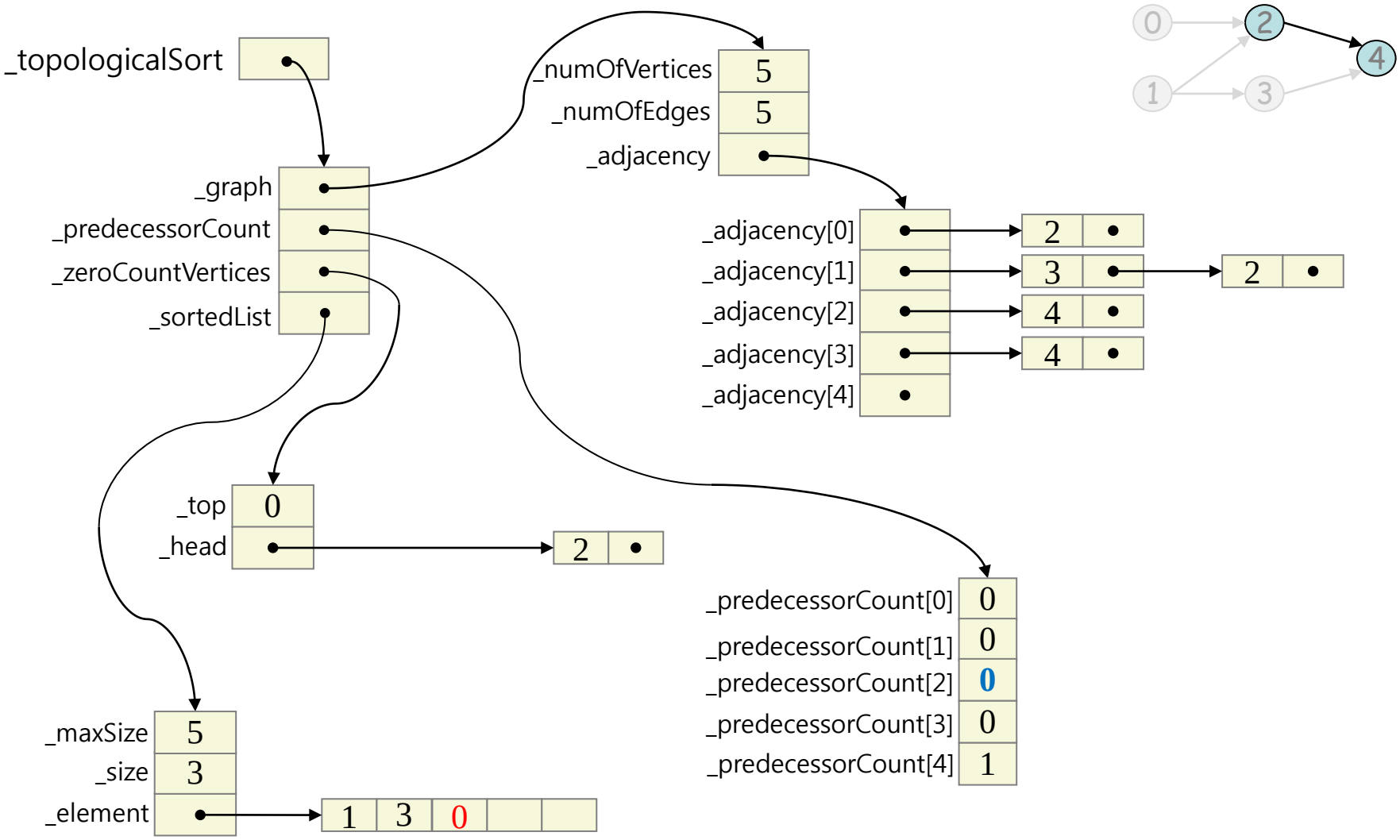
Vertex 1 을 삭제한 후



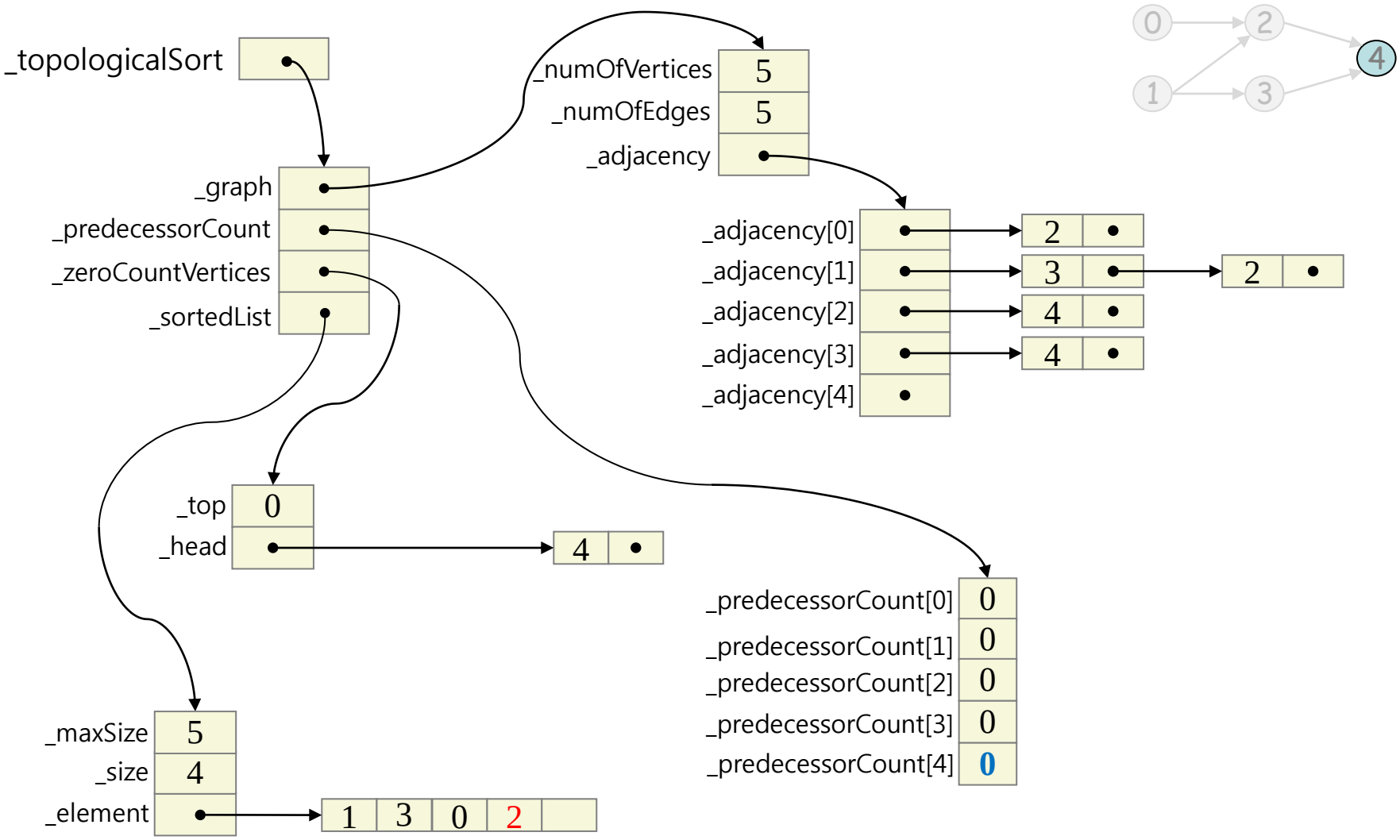
Vertex 3 을 삭제한 후



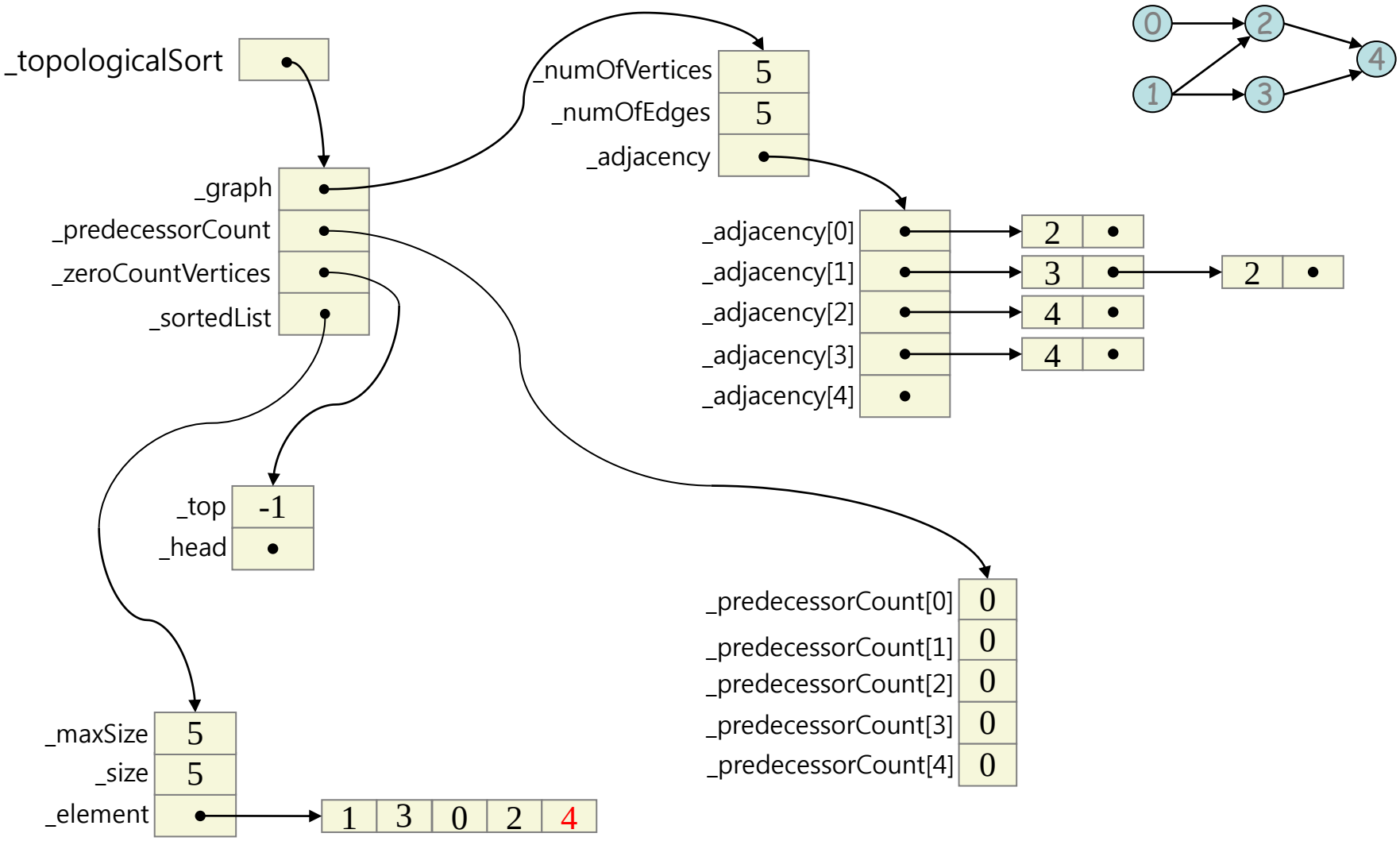
Vertex 0 을 삭제한 후



Vertex 2 를 삭제한 후



Vertex 4 를 삭제한 후



[문제 5] 요약

- Test Data
  - Test Data는 최소 10개 이상의 vertex와 15 개 이상의 edge를 가지고 있는 graph를 사용할 것
    - 보고서에 test용 그래프를 그려서 첨부할 것
    - 실제 실행된 결과의 화면을 함께 첨부할 것

- 앞의 요약에서 언급한 사항에 대해, 자신의 프로그램에서는 어떻게 되어 있는지 각자의 의견을 논하시오.

보강 일정

- 10월 9일 한글날(휴일)
  - 00반 (수요일 오전 10시 수업) – 10일(목) 저녁 7시
  - 01반 (수요일 오후 4시 수업) – 11일 (금) 저녁 7시

과제 제출

과제 제출

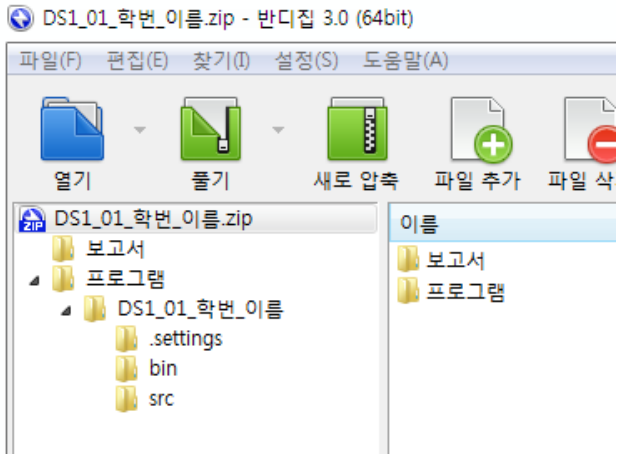
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_05_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 10월 8일(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_05_학번_이름.zip
  - 폴더의 구성
    - DS2_05_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_05_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

     ----- 이번 주는 그림 첨부가 필수이므로 분량 제한 없음 -----
  4. 소스코드 : 화면 capture가 아닌 소스를 붙여넣을 것

     소스는 장수 제한이 없음.

[제 5 주 실습] 끝