[제 3 주 실습]

최소비용 확장트리

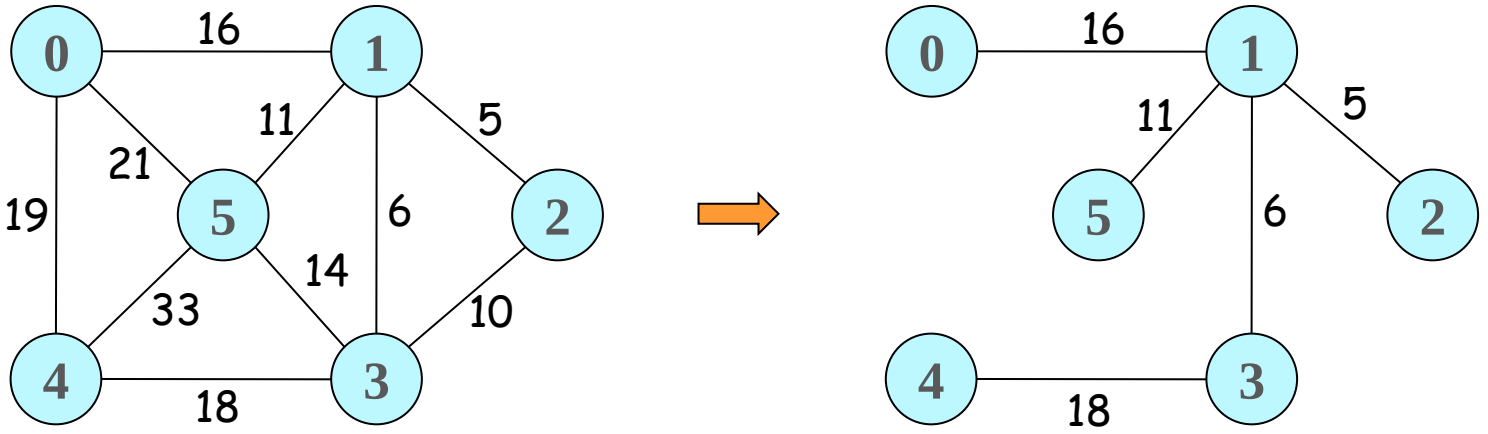
강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

[문제 3]
Finding
Minimum Cost Spanning Trees

문제 개요

- 입력:
  - Weighted undirected graph
- 출력:
  - 주어진 그래프로부터 최소비용확장트리 (Minimum Cost Spanning Tree)를 찾는다.



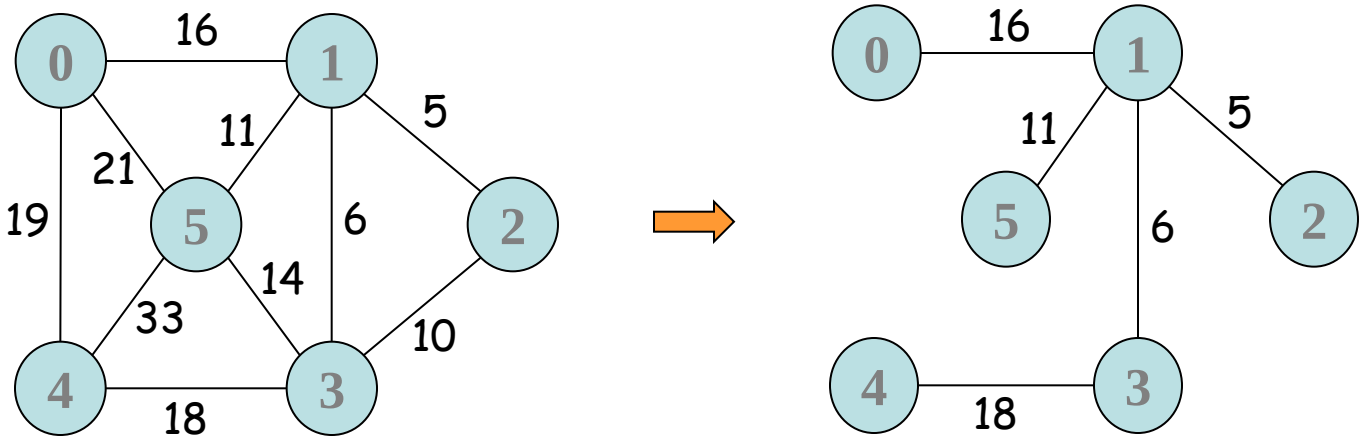
구현에서의 주요 내용

- 그래프의 표현
  - Adjacency Matrix로 구현한다.
  - 선택사항:
    별도의 동일한 프로그램으로 Adjacency List로도 구현한다. (+α)
- Minimum Cost Spanning Trees 찾는 알고리즘
  - Kruskal의 알고리즘
- Union-Find 알고리즘 활용
  - 새로운 edge를 Minimum Cost Spanning Trees에 추가할 때 cycle이 생기는지를 검사

입력

- 키보드로부터 그래프 정보를 입력 받는다.
  - Vertex의 개수
    - Vertex는 0부터 시작하는 정수로 나타낸다.
  - Edge의 개수
    - 입력되는 그래프의 edge의 개수
  - Weighted Edge들
    - cost가 함께 주어져 있는 edge들
    - 미리 주어진 edge 수 만큼 입력 받는다.
    - Edge는 vertex의 쌍과 cost로 나타낸다.
      - (tailVertex, headVertex, cost)

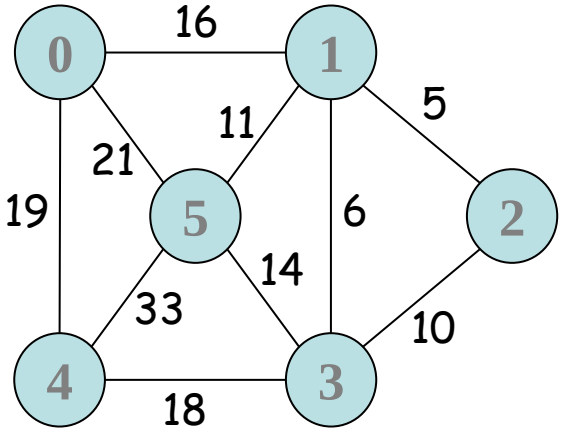
Kruscal's Algorithm



출력

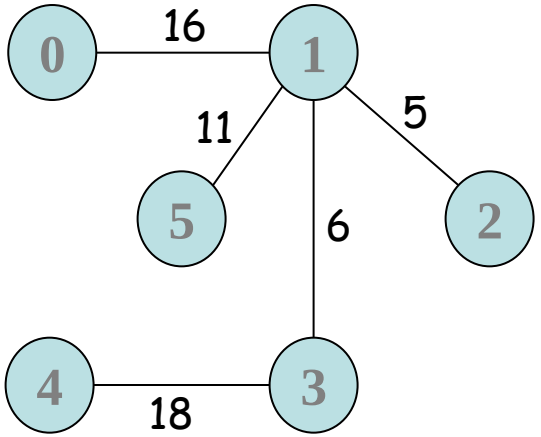
- 입력된 그래프

[0] -> 1(16), 4(19), 5(21)
[1] -> 0(16), 3(6), 2(5), 5(11)
[2] -> 1(5), 3(10)
[3] -> 1(6), 2(10), 4(18), 5(14)
[4] -> 0(19), 3(18), 5(33)
[5] -> 0(21), 1(11), 3(14), 4(33)



- 찾아진 최소비용확장트리의 edge들

(1, 2, (5))
(1, 3, (6))
(1, 5, (11))
(0, 1, (16))
(3, 4, (18))



❑출력의 예

```
- 그래프의 vertex수와 edge수를 입력 받아야 합니다.
? 그래프의 vertex 수를 입력 하시오:6
? 그래프의 edge 수를 입력 하시오:10
- 그래프의 edge를 반복하여 10 개 입력 받아야 합니다.
  하나의 edge는 (vertex1 vertext2 cost)의  순서로 표시됩니다.
? Edge를 입력하시오: 0 1 16
? Edge를 입력하시오: 0 4 19
? Edge를 입력하시오: 0 6 19
[Error] 입력 오류입니다.
? Edge를 입력하시오: 0 5 21
? Edge를 입력하시오: 1 5 11
? Edge를 입력하시오: 0 4 19
[Error] 입력 오류입니다.
? Edge를 입력하시오: 4 3 18
? Edge를 입력하시오: 3 4 18
[Error] 입력 오류입니다.
? Edge를 입력하시오: 3 5 14
? Edge를 입력하시오: 3 1 -6
[Error] 입력 오류입니다.
? Edge를 입력하시오: 3 1 6
? Edge를 입력하시오: 2 1 5
? Edge를 입력하시오: -1 2 10
[Error] 입력 오류입니다.
? Edge를 입력하시오: 3 2 10
? Edge를 입력하시오: 4 5 33
```

```
! 입력된 그래프는 다음과 같습니다:
생성된 그래프 :
[0] -> 1(16)  4(19)  5(21)
[1] -> 0(16)  2(5)  3(6)  5(11)
[2] -> 1(5)  3(10)
[3] -> 1(6)  2(10)  4(18)  5(14)
[4] -> 0(19)  3(18)  5(33)
[5] -> 0(21)  1(11)  3(14)  4(33)


! 최소비용확장트리를 찾기 시작합니다:
Edge 1, 2, 5 가 추가 되었습니다.
Edge 1, 3, 6 가 추가 되었습니다.
Edge 2, 3는 cycle을 생성 시킵니다.
Edge 1, 5, 11 가 추가 되었습니다.
Edge 3, 5는 cycle을 생성 시킵니다.
Edge 0, 1, 16 가 추가 되었습니다.
Edge 3, 4, 18 가 추가 되었습니다.

! 최소비용확장트리의 edge들은 다음과 같습니다:
(3, 4, (18))
(0, 1, (16))
(1, 5, (11))
(1, 3, (6))
(1, 2, (5))
```

이 과제에서 필요한 객체는?

- Application
  - AdjacencyMatrixGraph
  - MinCostSpanningTrees
    - PairwiseDisjointSets
    - MinPriorityQ
  - Edge
  - LinkedList
    - Node

Application의 공개 함수는?

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

DS2_03_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 ();
    }
}
```

AdjacencyMatrixGraph의 공개 함수는?

- 사용자에게 필요한 함수
  - [문제2]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 AdjacentVerticesIterator adjacentVertircesIterator(int givenVertex)
  - public class AdjacentVerticesIterator

Edge의 공개 함수는?

- 사용자에게 필요한 함수
  - [문제2]의 것을 복사해서 수정
  - 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)

MinCostSpanningTree의 공개 함수는?

- 사용자에게 필요한 함수
  - public MinCostSpanningTree(AdjacencyMatrixGraph givenGraph, MinPriorityQ givenMinPriorityQ)
  - public boolean perform ()
  - public LinkedList<Edge> minimumCostSpanningTree()

PairwiseDisjointSets 의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - [문제1]의 것을 복사해서 사용 / 수정하지 않음
  - public PairwiseDisjointSets (int givenMaxNumOfMembers)
  - public int find(int aMember)
  - public void union(int idOfSet1, int idOfSet2)

MinPriorityQ의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public MinPriorityQ (int givenMaxSize)
  - public boolean isEmpty ()
  - public boolean isFull ()
  - public boolean add (Edge anEdge)
  - public Edge deleteMin ()

LinkedList〈T〉의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - [문제2]의 것을 복사해서 사용 / 함수 이름만 수정
  - public LinkedList()
  - public LinkedList(int givenMaxSize)

  - public void clear()
  - public boolean isFull()
  - public boolean isEmpty()
  - public boolean isIndexOK(int anIndex)
  - public boolean doesExist(T anElement)
  - public int size()

  - public boolean add(T anElement)

  - public LinkedListIterator linkedListIterator()
  - public class LinkedListIterator

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)

Class “Application“

Application – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class Application {
    private Scanner                 _scanner ;

    private AdjacencyMatrixGraph    _graph ;
    private MinCostSpanningTree     _minCostSpanningTree ;
    private LinkedList<Edge>        _minCostSpanningTreeEdges ;
    private MinPriorityQ  _minPriorityQ;
    ……
```

Application의 공개 함수 run()의 구현

```
public void run(){
        if ( this.inputAndMakeGraph() ) {
            this.showGraph() ;
            this._minCostSpanningTree = new MinCostSpanningTree(this._graph,
this._minPriorityQ);
            System.out.println("최소비용확장트리를 찾기 시작합니다:") ;
            this._minCostSpanningTree.perform() ;
            this.showMinCostSpanningTree() ;
        }
        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 것에, Cost 부분을 입력 받는 것을 추가하여 만든다.
  - private void showGraph()
    - 입력된 그래프를 보여준다.
    - [문제2]와 동일
  - private void showMinCostSpanningTree()
    - 찾은 최소비용확장트리를 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 AdjacencyMatrixGraph(numOfVertices);
    this._minPriorityQ = new MinPriorityQ(numOfEdges);

    countEdges = 0;
    System.out.println("- 그래프의 edge를 반복하여 " + numOfEdges +" 개 입력 받아야 합니다.");
    System.out.println("  하나의 edge는 (vertex1 vertext2 cost)의  순서로 표시됩니다.");
    System.out.print("? Edge를 입력하시오: ");
```

Member Functions의 구현

Application의 Private Member function의 구현

```
        while(_scanner.hasNext()) {
            Edge anEdge = new Edge(_scanner.nextInt(), _scanner.nextInt(), _scanner.nextInt());
            if(anEdge.cost() > 0 && this._graph.addEdge(anEdge)){
                countEdges++;
                this._minPriorityQ.add(anEdge);
            }
            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()
    - Vertex가 아닌 Edge를 받아서 cost도 출력 할 수 있도록 수정

Member Functions의 구현

- Application의 Private Member function의 구현
  - private void showMinCostSpanningTree()
    - LinkedList<Edge>의 Iterator를 사용하여 찾아진 최소비용확장트리의 edge들을 출력

```
private void showMinCostSpanningTree()
{
    System.out.println("\n! 최소비용확장트리의 edge들은 다음과 같습니다:");
    @SuppressWarnings("unchecked")
    LinkedList<Edge> _result = this._minCostSpanningTree.result();

    LinkedList<Edge>.LinkedListIterator it = _result.linkedListIterator();
    while(it.hasNext()){
        Edge tmpEdge = it.next();
        System.out.println("(" + tmpEdge.tailVertex() + ", " +
        tmpEdge.headVertex() + ", (" + tmpEdge.cost() + "))");
    }
    System.out.println();
}
```

Class "AdjacencyMatrixGraph"

AdjacencyMatrixGraph의 수정

- [문제2] 에서 사용한 것을 복사해 와서 Cost 추가
  - 기존 함수에 추가
    - public boolean addEdge(Edge anEdge)
      - 실제 내용 저장 시 1 값이 아닌 cost를 저장하도록 수정
    - public class AdjacentVerticesIterator의 내부 함수 수정
      - public Edge next() : Edge형태로 반환하도록 수정

Class “Edge”

Edge의 수정

- [문제2]에서 사용한 것을 복사해 와서 Cost 관련 부분을 추가
  - 비공개 인스턴스 변수 추가
    - private int _cost;
  - 함수 수정/추가
    - public Edge(int givenTailVertex, int givenHeadVertex, int givenCost)
      - 생성자 수정
      - givenCost의 값을 this._cost에 저장
    - public void setCost(int aCost)
      - Cost를 저장 하는 함수 추가
      - aCost를 this._cost에 저장
    - public int cost()
      - Cost를 반환하는 함수 추가
      - this._cost를 반환

Edge의 수정

- 함수 추가
  - public boolean sameEdgeAs(Edge anEdge)
    - 전달 받은 anEdge와 현재 Edge의 값이 같은지 확인

```
public boolean sameEdgeAs(Edge anEdge)
{
    if(this._tailVertex == anEdge.tailVertex() && anEdge.headVertex() == this._headVertex)
        return true;
    else
        return false;
}
```

Class “MinCostSpanningTree”

비공개 인스턴스 변수

```
public class MinCostSpanningTree {
    MinPriorityQ              _minPriorityQ ;
    AdjacencyMatrixGraph      _graph ;
    LinkedList<Edge>          _result ;
```

MinCostSpanningTree 의 멤버 함수는?

- MinCostSpanningTree의 Public Member function의 사용법
  - public MinCostSpanningTree(AdjacencyMatrixGraph givenGraph, MinPriorityQ givenMinPriorityQ)
    - MinCostSpanningTree 초기화
  - public boolean perform()
    - MinCostSpanningTree를 찾는다.
  - public LinkedList<Edge> result() ;
    - 현재까지 찾아진 MinCostSpanningTree의 결과를 반환한다.

MinCostSpanningTree 의 멤버 함수는?

- MinCostSpanningTree의 Public Member function의 구현
  - public MinCostSpanningTree(AdjacencyMatrixGraph givenGraph, MinPriorityQ givenMinPriorityQ)
    - _graph를 givenGraph로 초기화
    - _minPriorityQ를 givenMinPriorityQ로 초기화
    - _result를 LinkedList<Edge>로 _graph의 numOfVertices()-1 만큼 생성
  - public LinkedList<Edge> result()
    - _result를 리턴

MinCostSpanningTree 의 멤버 함수는?

- MinCostSpanningTree의 Public Member function의 구현

```
public boolean perform()
{
    PairwiseDisjointSets _pairwiseDisjointSets;
    int setOfTailVertex, setOfHeadVertex;
    Edge    anEdge;
    int numOfVertices = this._graph.numOfVertices();
    int maxTreeEdges = numOfVertices -1;

    _pairwiseDisjointSets = new PairwiseDisjointSets(numOfVertices);
    while((this._result.size() < maxTreeEdges) && !this._minPriorityQ.isEmpty())
    {
        anEdge = this._minPriorityQ.deleteMin();
        setOfTailVertex = _pairwiseDisjointSets.find((int)anEdge.tailVertex());
        setOfHeadVertex = _pairwiseDisjointSets.find((int)anEdge.headVertex());
        if(setOfTailVertex == setOfHeadVertex){
            System.out.println("Edge " + anEdge.tailVertex() + ", " +
        anEdge.headVertex() + "는 cycle을 생성 시킵니다.");
        }
        else
        {
            _pairwiseDisjointSets.union(setOfTailVertex, setOfHeadVertex);
            this._result.add(anEdge);
            System.out.println("Edge " + anEdge.tailVertex() + ", " +
            anEdge.headVertex() + ", " + anEdge.cost()+ " 가 추가 되었습니다.");
        }
    }
    return (this._result.size() == maxTreeEdges);
}
```

디버깅을 위한 출력

MinPriorityQ Class

MinPriorityQ – 비공개 인스턴스 변수

```
public class MinPriorityQ {
    private int       _size;
    private int       _maxSize;
    private Edge[]    _heap;
```

MinPriorityQ의 멤버 함수는?

- MinPriorityQ의 Public Member function의 사용법
  - public MinPriorityQ(int givenMaxsize)
    - MinPriorityQ의 생성자
    - Size를 전달 받아 MinPriorityQ를 givenMaxsize만큼 생성한다.
  - public boolean isEmpty()
    - PriorityQ가 비어있는지 확인 한다.
  - public boolean isFull()
    - PriorityQ가 가득 차 있는지 확인 한다.
  - public boolean doesEdgeExist(Edge anEdge)
    - 전달 받은 anEdge가 이미 존재하는지 확인 한다.
  - public boolean add(Edge anEdge)
    - anEdge를 Queue에 삽입한다.
  - public Edge deleteMin()
    - 가장 작은 값을 Queue에서 삭제한다.

Member Functions의 구현

- MinPriorityQ의 Public Member function의 구현
  - MinPriorityQ은 지난 학기 자료 강의자료 [13주 우선순위 큐] 와 실습자료 [14주 우선순위 큐] 를 참고한다.
  - public MinPriorityQ(int givenMaxsize)
    - _heap을 givenMaxsize+1만큼 생성
    - givenMaxsize를 _maxSize에 저장
    - _size를 0으로 초기화
  - public boolean isEmpty()
    - _size가 0와 같으면 true를 리턴
  - public boolean isFull()
    - _size와 _maxSize가 같으면 true를 리턴

Member Functions의 구현

MinPriorityQ의 Public Member function의 구현

```
public boolean doesEdgeExist(Edge anEdge)
{
    boolean found = false;
    if(this.isEmpty())
        return found;
    for(int i = 0; i < this._maxSize; i++)
    {
        Edge tmpEdge = this._heap[i];
        if(tmpEdge != null) {
            Edge reverseEdge = new Edge(anEdge.headVertex(),anEdge.tailVertex(),anEdge.cost());
            if(tmpEdge.sameEdgeAs(anEdge) || tmpEdge.sameEdgeAs(reverseEdge))
            {
                found = true;
                break;
            }
        }
    }
    return found;
}
```

Member Functions의 구현

MinPriorityQ의 Public Member function의 구현

```
public Edge deleteMin()
{
    int         parent, smallerChild ;
    Edge        root, last ;
    // Queue는 empty가 아니라고 가정한다.
    // 그러므로 deleteMax를 call 하기 전에 empty 검사를 해야 한다.
    root = this._heap[1];
    this._size--;
    if ( this._size > 0 ) {
        // 삭제 한 후에 적어도 하나의 원소가 남아 있다.
        // 마지막 위치 (_this-<size+1)의 원소를 떼어내어,
        // root 위치 (1)로부터 아래쪽으로 새로운 위치를 찾아 내려간다.
        last = this._heap[this._size +1];
        parent = 1 ;
        while ( (parent*2) <= this._size ) {
            // child 가 존재한다.
            // left, right 중에서 더 작은 cost 값을 갖는 child를 smallerChild로 한다.
            smallerChild = parent * 2 ;
            if ( (smallerChild < this._size) &&
                    (this._heap[smallerChild].cost() > this._heap[smallerChild+1].cost())) {
                // right child가 존재하고, 그 cost 값이 더 작으므로,
                // right child를 smallerChild로 한다.
                smallerChild++ ;
            }
            if ( last.cost() <= this._heap[smallerChild].cost() ) {
                // last 원소는 더 이상 아래로 내려갈 필요가 없다.
                // 현재의 parent 위치에 삽입하면 된다.
                break ;
            }
            // child 원소를 parent 위치로 올려보낸다.
            // child 위치는 새로운 parent 위치가 된다.
            this._heap[parent] = this._heap[smallerChild] ;
            parent = smallerChild;
        }  // end while
        this._heap[parent] = last ;
    }
    return  root ;
}
```

Member Functions의 구현

- MinPriorityQ의 Public Member function의 구현
  - public boolean add(Edge anEdge)
    - 보고서에 구현 알고리즘 반드시 명시 할 것

```
public boolean add(Edge anEdge)
{
    int i ;

    if ( this._size >= this._maxSize ) {
        return false; // Heap is full.
    }
    else if(this.doesEdgeExist(anEdge))
        return false;
    else {
        i = ++(this._size);
        while ( ( i > 1 ) && (anEdge.cost() < this._heap[i/2].cost()) ) {
            this._heap[i] = this._heap[i/2];
            i = i / 2 ;
        }
        this._heap[i] = anEdge;
        return true;
    }
}
```

[문제 3] 요약

- Finding Minimum Cost Spanning Trees를 [AdjacencyMatrixGraph, PairwiseDisjointSets, MinPriorityQ, ArrayList]를 이용하여 구현하여 본다.

- Finding Minimum Cost Spanning Trees가 실제 어떻게 처리 되는 지에 대하여 확인한다.
  - Test Data는 최소 10개 이상의 vertex와 20 개 이상의 edge를 가지고 있는 graph를 사용할 것
  - 보고서에 test용 그래프를 PPT로 그려서 첨부할 것

- 확인 해본 내용은 보고서에 포함이 되어야 한다.

과제 제출

과제 제출

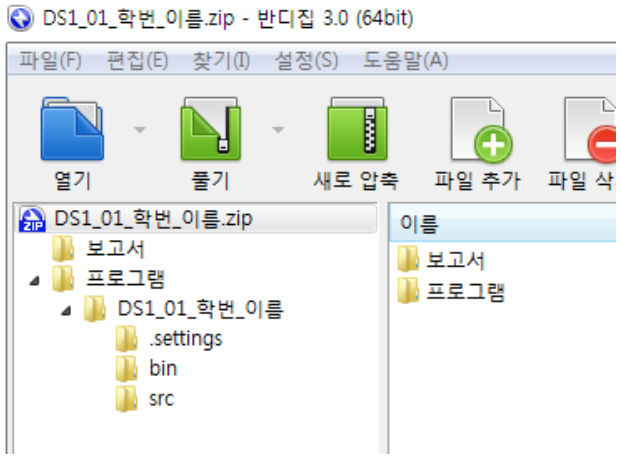
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_03_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 9월 17일(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_03_학번_이름.zip
  - 폴더의 구성
    - DS2_03_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_03_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 3 주 실습] 끝