[511643] 자료구조 (2019-2 학기)

실습 #13 보고서

이름	지현한
학번	20165164
소속 학과/ 대학	소프트웨어융합대학 빅데이터전공
분반	03 (담당교수: 김태운)

<주의사항>

- 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
- 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
    - 소스코드/스크립트 등을 작성 한 경우, 해당 파일의 이름도 적어주세요.
- 스마트캠퍼스 제출 데드라인: 2019. 12. 04. (수요일) 23:59
    - 데드라인을 지나서 제출하면 24 시간 단위로 20%씩 감점(5 일 경과 시 0 점)
    - 주말/휴일/학교행사 등 모든 날짜 카운트 함
    - 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0 점 처리함
    - 예외 없음
- 스마트캠퍼스에 아래의 파일을 제출 해 주세요
    - 보고서(PDF 파일로 변환 후 제출)
    - 보고서 파일명에 이름과 학번을 입력 해 주세요.
    - 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일 제출 (미 제출시 감점)

<개요>

이번 과제는 그래프 두번째 과제 입니다. 최소신장트리 및 최단경로 알고리즘을 구현하는 과제입니다. 2 주 과제인 만큼 과제의 양이 조금 많습니다…

<실습 과제>

[Q 0] 요약 [배점: 10]

이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3 문장으로 요약하세요.

답변: 그래프에 대해서 더 깊게 이해하게 되었고, 특징들을 잘 알게 되었습니다.

과제 완성을 위해서 강의 자료를 다시 찾아보았고, 강의자료에 나와있는 코드들을 이해하려고 노력했습니다.

[Q 1] 인접행렬? 인접리스트? 상호 변환하기 [10 점]

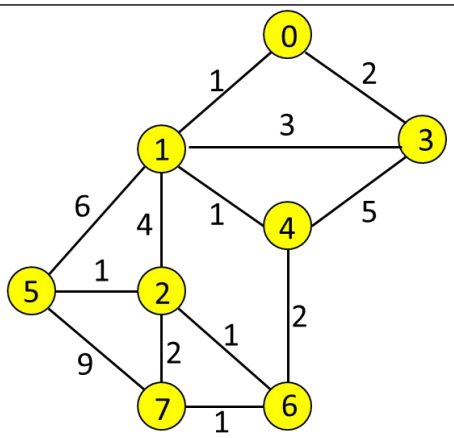
그래프는 인접행렬 또는 인접리스트로 구현(=표현)할 수 있습니다. 둘 간에 변환을 해 주는 메소드를 작성하세요. 참고: 이 문제에서는 무방향 가중치그래프만 고려합니다.

MyGraph 클래스를 만들고, 아래의 메소드를 구현하세요

- `public int[][] getAdjMatrixFromAdjList(List<Edge>[] l);`
  (연결리스트로 표현된 그래프를 연결행렬로 변환)
- `public List<Edge>[] getAdjListFromAdjMatrix(int[][] m);`
  (연결행렬로 표현된 그래프를 연결리스트로 변환)
- `public void print(int[][] m); // 연결행렬을 터미널에 출력`
- `public void print(List<Edge>[] l); // 연결리스트를 터미널에 출력`
- 두개의 print 메소드 출력 예시:

Print(연결행렬) (4x4 매트릭스인 경우)	Print(연결리스트) (출력 예시)
AdjMatrix : 0, 1, 0, 2 1, 0, 4, 3 0, 4, 0, 0 2, 3, 0, 0 참고: 숫자는 가중치를 의미함	AdjList : [0] => 1(1) => 3(2) [1] => 0(1) => 2(4) =>3(3) [2] => 1(4) [3] => 0(2) => 1(3) 참고: 2(4) 에서 2 는 정점의 인덱스 번호, (4) 는 가중치를 의미함.

테스트를 위해 MyGraphTest 클래스를 생성하고, main 메소드에서 아래의 그래프를 사용해서 MyGraph 의 4 가지 메소드를 테스트 하세요.



[Task 1]

- 그래프를 리스트로 구현/표현한 후 print 메소드 호출
- 그래프 리스트를 인자로 하여 getAdjMatrixFromAdjList 메소드 실행
- 리턴 된 행렬을 print

터미널 화면을 캡처하고 본 문서에 첨부하세요.

[Task 2]

- 그래프를 행렬로 구현/표현한 후 print 메소드 호출
- 그래프 행렬을 인자로 하여 getAdjListFromMatrix 메소드 실행
- 리턴 된 리스트를 print

터미널 화면을 캡처하고 본 문서에 첨부하세요.

소스코드도 제출해야 합니다.

Q1main.java Mygraph.java

답변 [Task 1]:

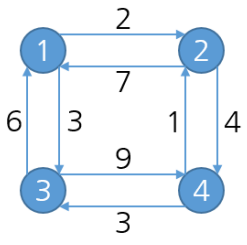
```
AdjMatrix :
0, 1, 0, 2, 0, 0, 0, 0
1, 0, 4, 3, 1, 7, 0, 0
0, 4, 0, 0, 0, 1, 1, 2
2, 3, 0, 0, 5, 0, 0, 0
0, 1, 0, 5, 0, 0, 2, 0
0, 6, 1, 0, 0, 0, 0, 9
0, 0, 1, 0, 2, 0, 0, 1
0, 0, 2, 0, 0, 9, 1, 0
```

답변 [Task 2]:

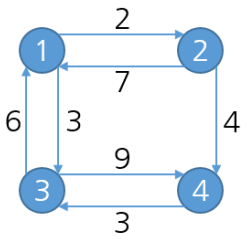
```
AdjList :
[0] => 1(1) => 3(2)
[1] => 0(1) => 2(4) => 3(3) => 4(1) => 5(6)
[2] => 1(4) => 5(1) => 6(1) => 7(2)
[3] => 0(2) => 1(3) => 4(5)
[4] => 1(1) => 3(5) => 6(2)
[5] => 1(6) => 2(1) => 7(9)
[6] => 2(1) => 4(2) => 7(1)
[7] => 2(2) => 5(9) => 6(1)
```

[Q 2] 뒤집어라 엎어라 [10 점]

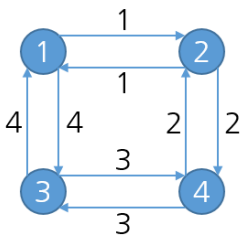
인접행렬 또는 인접리스트로 표현된 방향 또는 무방향 가중치 그래프 G 가 있습니다. G 가 G 를 뒤집은 G^R과 동일한지를 확인하는 메소드를 구현하세요. 아래의 4 가지 그래프 예제에서 <그래프 4>는 G 와 G^R이 동일하고, 나머지 그래프는 G 와 G^R이 서로 다릅니다.



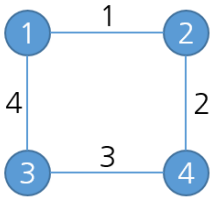
〈그래프 1〉



〈그래프 2〉



〈그래프 3〉



〈그래프 4〉

CheckGraph 클래스를 만들고, 아래의 메소드를 추가하세요

- public boolean isReverseSame(int[][] G); // G 와 G^R이 동일하면

true, 그렇지 않으면 false 를 리턴

- public boolean isReverseSame(List<Edge>[] G); // G 와 G^R이 동일하면 true, 그렇지 않으면 false 를 리턴

[Task 1] <그래프 1>에 해당하는 인접행렬 adjmat1 및 인접리스트 adjlist1 를 구현하세요. 다음으로, isReverseSame (adjmat1) 과 isReverseSame (adjlist1)을 호출하고 리턴되는 결과를 터미널에 출력하세요. 터미널 화면을 캡처하고 본 문서에 첨부하세요.

[Task 2] <그래프 2>에 해당하는 인접행렬 adjmat2 및 인접리스트 adjlist2 를 구현하세요. 다음으로, isReverseSame (adjmat2) 과 isReverseSame (adjlist2)을 호출하고 리턴되는 결과를 터미널에 출력하세요. 터미널 화면을 캡처하고 본 문서에 첨부하세요.

[Task 3] <그래프 3>에 해당하는 인접행렬 adjmat3 및 인접리스트 adjlist3 를 구현하세요. 다음으로, isReverseSame (adjmat3) 과 isReverseSame (adjlist3)을 호출하고 리턴되는 결과를 터미널에 출력하세요. 터미널 화면을 캡처하고 본 문서에 첨부하세요.

[Task 4] <그래프 4>에 해당하는 인접행렬 adjmat4 및 인접리스트 adjlist4 를 구현하세요. 다음으로, isReverseSame (adjmat4) 과 isReverseSame (adjlist4)을 호출하고 리턴되는 결과를 터미널에 출력하세요. 터미널 화면을 캡처하고 본 문서에 첨부하세요.

소스코드도 제출해야 합니다.

Q2main.java CheckGraph.java

답변 [Task 1]:

```
[Task 1]
isReverseSame adjmat1: false
isReverseSame: adjlist1: false
```

답변 [Task 2]:

```
[Task 2]
isReverseSame adjmat2: false
isReverseSame: adjlist2: false
```

답변 [Task 3]:

```
[Task 3]
isReverseSame adjmat3: true
isReverseSame: adjlist3: true
```

답변 [Task 4]:

```
[Task 4]
isReverseSame adjmat1: true
isReverseSame: adjlist1: true
```

[Q 3] PrimMST 구현하기 [10 점]

강의자료를 참고해서 최소신장트리를 찾는 PrimMST 클래스를 구현하세요.

[Task] 강의자료 p.14 에 표시된 입력 그래프를 입력으로 PrimMST 클래스를 실행하세요. 강의자료 p.14 와 같이 터미널에 출력하도록 동작하도록 main 메소드를 구현하고 실행하세요. "최소신장트리 간선"을 열거하는 로직 "최소신장트리의 간선 가중치 합"을 계산하는 로직은 PrimMST 클래스에 구현해도 되고, 또는 main 메소드에 구현해도 됩니다.

터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다.

답변 [Task] : Q3main.java PrimMST.java

```
최소신장트리 간선:
(1,0) (2,5) (3,5) (4,6) (5,6) (6,1)

최소신장트리의 간선 가중치 합 = 25
```

[Q 4] 최단경로 : Dijkstra 알고리즘 구현하기 [15 점]

강의자료를 참고하여 최단경로를 탐색하는 Dijkstra 알고리즘을 구현하는 Dijkstra 클래스를 생성하세요.

[Task] 강의자료를 참고하여 테스트를 위한 main 메소드도 구현하세요(p.26~). 수행 결과가 강의자료 p.28 과 같아야 합니다.
터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다.

답변 [Task] : Q4main.java DijkstraSP.java

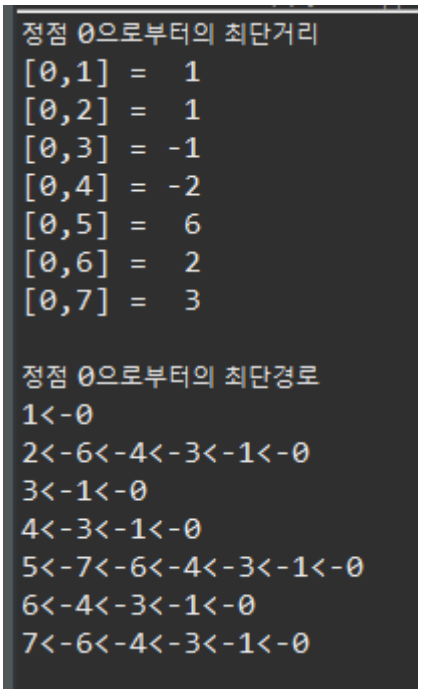
```
정점 0으로부터의 최단거리
[0, 0] = 0
[0, 1] = 1
[0, 2] = 5
[0, 3] = 2
[0, 4] = 2
[0, 5] = 6
[0, 6] = 4
[0, 7] = 5

정점 0으로부터 최단경로
1<-0
2<-1<-0
3<-0
4<-1<-0
5<-2<-1<-0
6<-4<-1<-0
7<-6<-4<-1<-0
```

[Q 5] 최단경로 : Bellman-Ford 알고리즘 구현하기 [15 점]

강의자료를 참고하여 최단경로를 탐색하는 Bellman-Ford 알고리즘을 구현하는 BellmanFord 클래스를 생성하세요.

[Task] 강의자료를 참고하여 테스트를 위한 main 메소드도 구현하세요(p.38~). 수행 결과가 강의자료 p.39 와 같아야 합니다.
터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다.

답변 [Task] : Q5main.java BellmanFord.java



[Q 6] 최단경로 : Bellman-Ford 알고리즘에 음수 싸이클 탐지 로직 구현하기[10 점]

Bellman-Ford 알고리즘으로 최단경로를 찾기 위해선 입력 그래프에 음수 싸이클이 없어야 합니다. 직전 문제[Q 5]에서 구현한 BellmanFord 클래스를 수정한 BellmanFordNew 클래스를 만들고, 입력 그래프에 음수 싸이클이 있는지 없는지를 검사하세요. shortestPath 메소드에서 음수 싸이클이 감지되면 “음수 싸이클이 있습니다”라고 출력하도록 구현하세요. 나머지 동작은 BellmanFord 클래스 구현과 동일합니다.

[Task] 터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다.

답변 [Task] :

[Q 7] 최단경로 : Floyd-Warshall 알고리즘 구현하기 [20 점]

강의자료에 나온 알고리즘 pseudo-code 를 참고하여, 모든 쌍 최단경로를 찾는 Floyd-Warshall 알고리즘을 구현하세요 (클래스 이름: FloydWarshall). 인접행렬을

이용해서 구현해야 합니다. 또한, 그래프를 저장한 인접행렬을 출력하는 printAdjMatrix 메소드도 구현하세요 (print 메소드의 출력 예시는 [Q 1]을 참고하세요).

[Task] 테스트를 위해 FloydWarshallTest 클래스를 만들고, main 메소드에서 아래와 같은 순서로 프로그래밍 하세요.

1) 강의자료 p.44 에 주어진 입력 그래프를 나타내는 행렬 adjMatrix 를 작성하고 D 행렬을 강의자료와 같이 초기화.
2) 행렬 D 를 화면에 출력.
3) Floyd-Warshall 알고리즘 실행.
4) 최종 행렬 D 를 화면에 출력(최종 행렬은, 강의자료 p.46 의 최종 행렬과 같아야 함).

터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다.

답변 [Task]:

끝! 수고하셨습니다 ᴱ