|  |
| --- |
| **[511643] 자료구조** |
| **실습 #13 보고서** |

|  |  |
| --- | --- |
| **이름** | 조아현 |
| **학번** | 20215247 |
| **소속**  **학과/대학** | 소프트웨어융합대학  빅데이터전공 |
| **분반** | 01 (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.
* SmartLEAD 제출 데드라인:
  + 2주 뒤 실습시간 전날(다음 다음번 실습 전날) 23:55까지
  + 데드라인을 지나서 제출하면 0점(예외 없음)
  + 주말/휴일/학교행사 등으로 인한 데드라인 연장 없음
  + 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
* SmartLEAD에 아래의 파일을 제출 해 주세요
  + 보고서(PDF 파일로 변환 후 제출 권장하나, WORD 형식으로 제출도 가능)
  + 보고서 파일명에 이름과 학번을 입력 해 주세요.
  + 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일을 하나의 zip 파일로 압축하여 제출(또는 본 문서에 소스코드 화면 캡쳐해서 붙여넣기)

## <개요>

이번 과제는 그래프 및 탐색 알고리즘을 구현하는 내용입니다.

## <실습 과제>

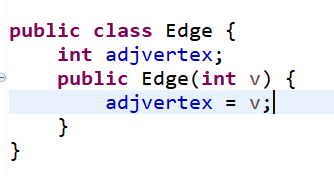
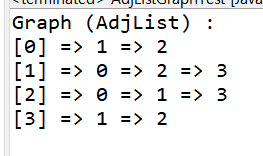
|  |
| --- |
| **[Q 0] 요약 [배점: 10]**  이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3문장으로 요약하세요. |

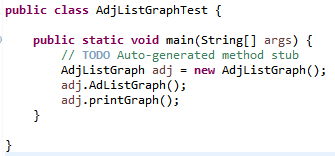
답변:

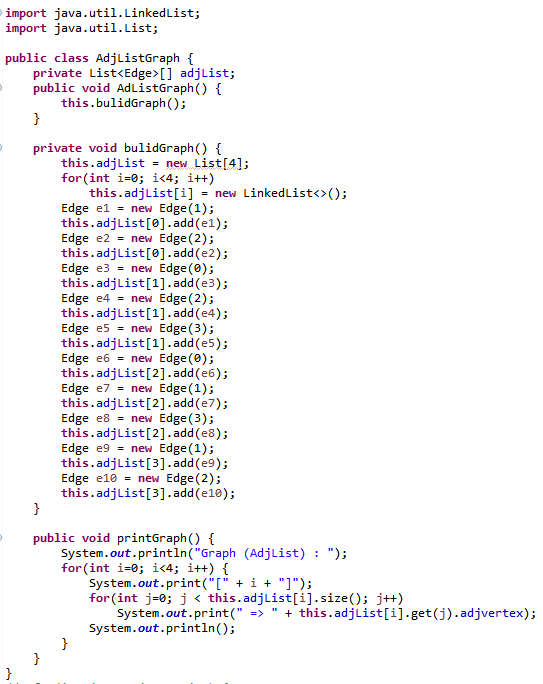
그래프는 정점과 간선을 모아 놓은 자료구조이다. 하나의 간선은 두 개의 정점을 연결하며, 간선에 방향이 있는 방향그래프와 간선에 방향이 없는 그래프인 무방향그래프로 나뉜다.  
그래프 탐색으로는 깊이우선탐색과 너비우선탐색이 있다.

|  |  |  |
| --- | --- | --- |
| **[Q 1] 인접리스트를 이용한 그래프 표현 [20점]**  아래와 같이 동작하는 **AdjListGraph** 클래스를 코딩하세요.  강의자료 p.12 (Edge 클래스), p.13(인접 리스트 생성 코드)을 참고하여 p.11에 표시된 그래프를 인접리스트로 구현하세요[[1]](#footnote-1). AdjListGraph 클래스의 기본 생성자(= 아무런 인자를 받지 않는 생성자)가 호출되면, 생성자는 멤버 메소드인 private void buildGraph()를 호출하고, buildGraph 메소드는 항상 p.11 과 동일한 인접 리스트를 생성하여 this.adjList에 저장합니다. p.13의 adjList 변수를 AdjListGraph 클래스의 멤버 변수로 선언하세요: private List<Edge>[] adjList;  다음으로, public void printGraph() 메소드를 구현하여 this.adjList를 출력하도록 코딩하세요. 출력 결과는 아래와 같아야 합니다.   |  |  | | --- | --- | | Graph (AdjList):  [0] => 1 => 2  [1] => 0 => 2 => 3  [2] => 0 => 1 => 3  [3] => 1 => 2 | \* 참고1: Edge 클래스를 수정해서 사용해도 됩니다.  \* 참고2: 출력 결과는 왼쪽의 출력 예시와 완벽히 동일해야 합니다 (즉, 정점이 출력되는 순서도 동일해야 함) |   [Task] 테스트를 위해 AdjListGraphTest 클래스를 생성하고, main 메소드에서 AdjListGraph 클래스의 생성자를 호출한 뒤, printGraph 메소드를 호출하세요. 터미널/콘솔 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

답변 [Task]:



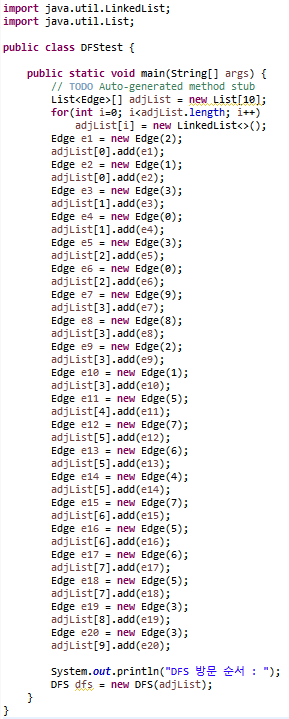


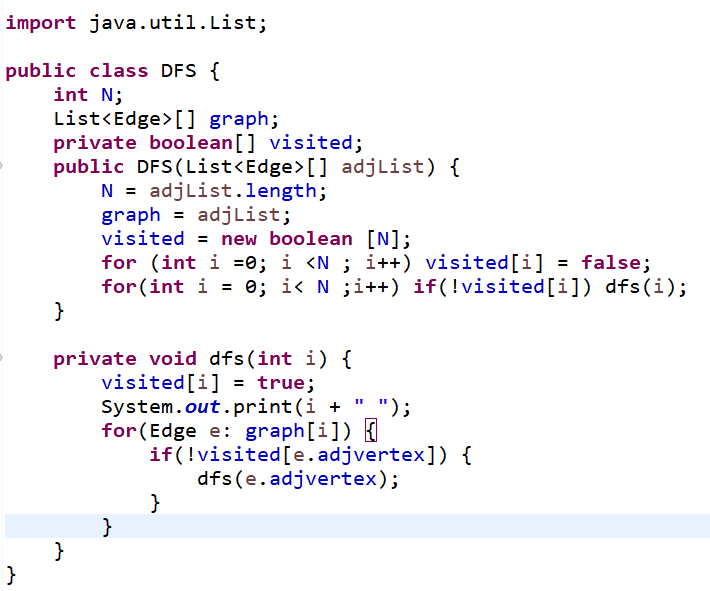


|  |
| --- |
| **[Q 2] 깊이우선탐색(DFS) [20점]**  강의자료와 동일하게 **DFS** 클래스를 구현하세요. 강의자료 p.19와 같이 그래프를 구성하고(즉, 주어진 그래프를 표현한 인접리스트로 구현), 깊이우선탐색을 실행하세요. 터미널 출력 결과가 p.19의 결과와 동일해야 합니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

답변:



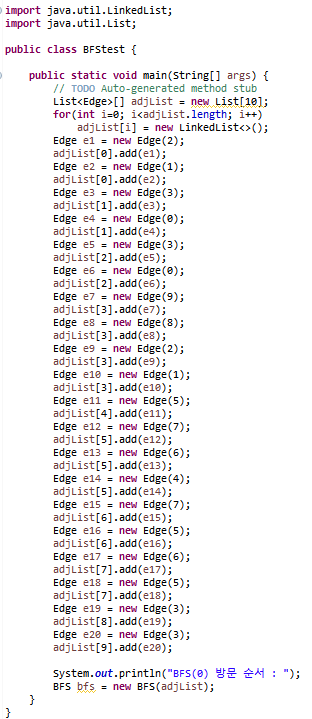




|  |
| --- |
| **[Q 3] 너비우선탐색(BFS) [20점]**  강의자료와 동일하게 **BFS** 클래스를 구현하세요. 강의자료 p.25와 같이 그래프를 구성하고(즉, 주어진 그래프를 표현한 인접리스트로 구현), 너비우선탐색을 실행하세요. 터미널 출력 결과가 p.25의 결과와 동일해야 합니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

답변:



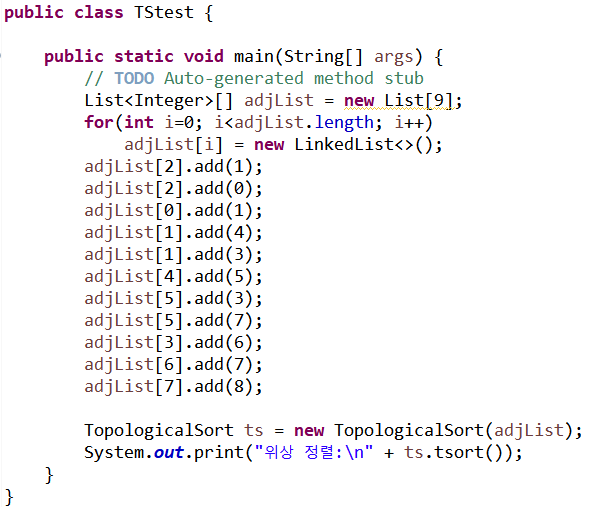


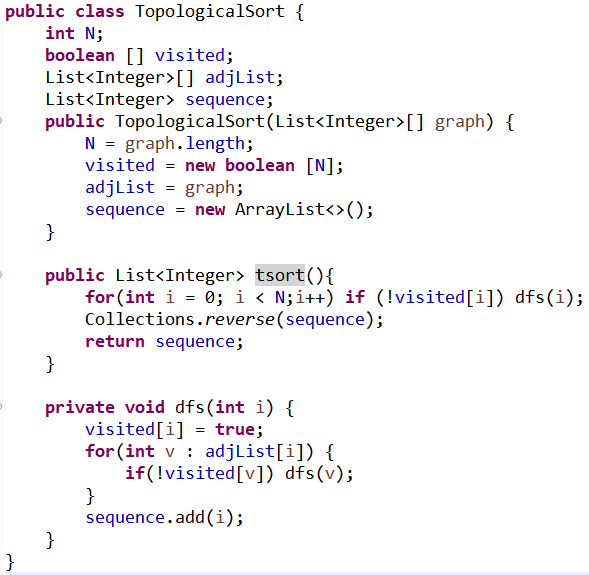


|  |
| --- |
| **[Q 4] 위상정렬 [20점]**  강의자료와 동일하게 **TopologicalSort** 클래스를 구현하세요. 강의자료 p.33과 같이 그래프를 구성하고, 위상정렬을 실행하세요. 터미널 출력 결과의 예시는 p.33을 참고하세요. 출력 결과가 p.33의 결과와 같아도 되고, 또는 결과가 다르더라도 위상정렬 순서에 맞기만 하면 됩니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

답변:

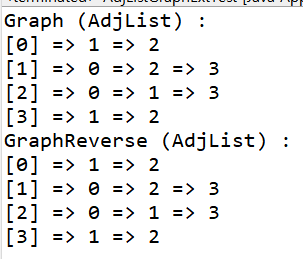


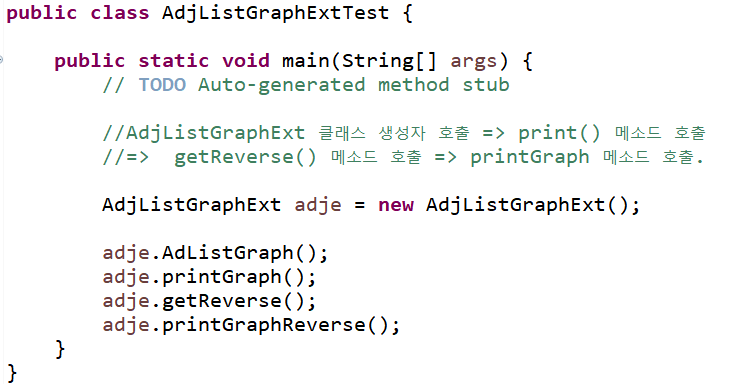


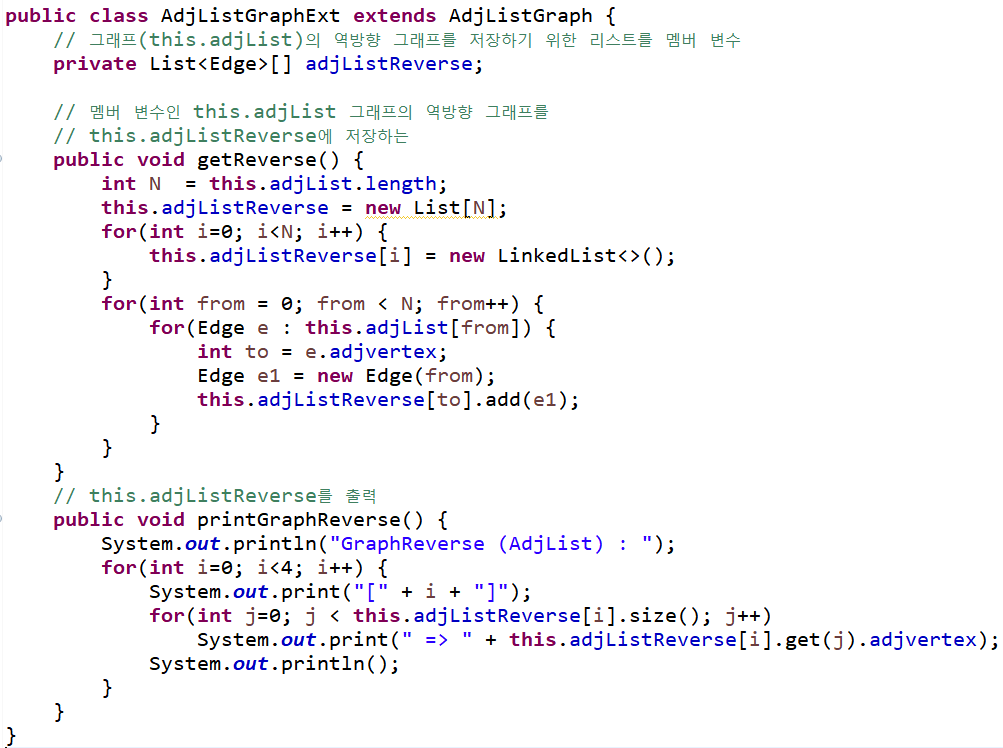


|  |
| --- |
| **[Q 5] 인접리스트 그래프의 역방향 그래프 표현 [5점]**  [Q 1] 에서 작성한 AdjListGraph 클래스를 상속받아 **AdjListGraphExt** 클래스를 아래와 같이 코딩하세요.  인접리스트 형태로 주어진 그래프(this.adjList)의 역방향 그래프를 저장하기 위한 리스트를 멤버 변수로 선언하세요: private List<Edge>[] adjListReverse; 즉, adjListReverse는 this.adjList에 해당하는 그래프의 역방향 그래프를 인접리스트 형태로 저장하는 멤버 변수 입니다.  멤버 변수인 this.adjList 그래프의 역방향 그래프를 this.adjListReverse에 저장하는 public void getReverse() 메소드를 구현하세요.  다음으로, public void printGraphReverse() 메소드를 구현하여 this.adjListReverse를 출력하도록 코딩하세요. 출력 결과에 대한 요구조건은 [Q 1]과 같습니다.  [Task] 테스트를 위해 AdjListGraphExtTest 클래스를 생성하고, main 메소드를 다음과 같이 코딩하세요: AdjListGraphExt 클래스 생성자 호출 => print() 메소드 호출 => getReverse() 메소드 호출 => printGraph 메소드 호출.  터미널/콘솔 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다.  \* 참고: AdjListGraphExt 클래스의 생성자는 AdjListGraph와 같이 buildGraph를 호출하여 p.11의 그래프를 this.adjList에 저장합니다. p.11의 그래프는 무방향 그래프 이지만, 무방향인 각각의 간선이 방향을 가지는 두개의 간선으로 표현된 것과 같습니다. 따라서, 이번 문제에서는 원래의 그래프를 표현한 인접리스트와 역방향 그래프를 표현한 인접리스트는 동일합니다. 즉, printGraph 결과와 printGraphReverse 결과가 동일합니다 (단, 출력되는 순서는 달라도 됩니다) |

답변 [Task]:

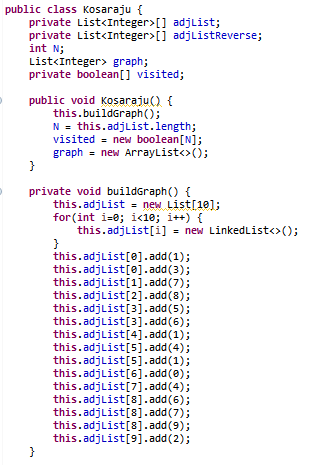


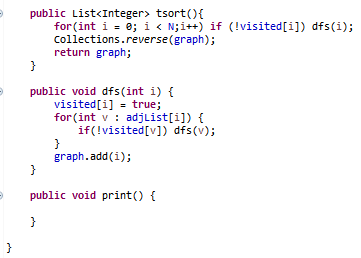


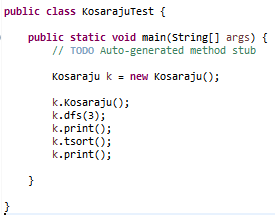


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[Q 6] Kosaraju 알고리즘을 이용한 강연결성분 찾기 [5점]**  강의자료를 참고하여 Kosaraju 클래스를 구현하세요. 강의자료 p.41과 같이 그래프G를 구성하고, 강연결성분을 모두 찾아 터미널에 출력하세요. 터미널 출력 결과의 예시는 아래와 같습니다.   |  |  | | --- | --- | | <터미널 출력 예시, Ex 1> | <터미널 출력 예시, Ex 2> | | 강 연결 성분:  [2,8,9], [0,6,3], [5], [1,4,7] | 강 연결 성분:  [1] 2,9,8  [2] 0,6,3  [3] 5  [4] 1,4,7 |   \* 참고: 어떤 강연결성분이 먼저 출력되는지, 그리고 각각의 강 연결성분에서 어떤 정점이 먼저 출력되는지는 위의 예시와 달라도 됩니다. 즉, [5], [8,9,2], [7,1,4], [3,6,0] 으로 출력되어도 됩니다. (정답만 맞으면 OK). 출력 형식은 예시와 달라도 됩니다. 하지만 발견한 강연결성분은 정확해야 합니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

답변:

****

****

****

**끝! 수고하셨습니다 ☺**

1. 강의노트 p.13에서, 반복문을 사용해서 인접리스트를 생성하고 있음. 하지만, 본 문제에서는 p.13과 같이 반복문을 사용해서 인접 리스트를 만들 수 없음. 왜냐 하면, Line:05의 if 문을 평가할 수 있는 방법이 없기 때문. 따라서, p.11의 그래프를 인접리스트로 표현하기 위해서는 모든 인접한 노드에 대해서 직접(= 하드코딩으로...) Edge 클래스를 생성하여 adjList[i].add를 호출해 주어야 함. 만약, p.11의 그래프를 인접행렬로 표현한다면(=이때도 인접 행렬은 하드코딩으로 직접 생성해 주어야 함) p.13의 Line:05를 평가할 수 있으나, 이 때, 인접 리스트에 추가되는 Edge 클래스의 순서가 강의자료 p.11과 달라질 수 있고, 이 경우 출력 순서가 위와 달라질 수 있음. [↑](#footnote-ref-1)