|  |
| --- |
| **[511643] 자료구조** |
| **실습 #14 보고서** |

|  |  |
| --- | --- |
| **이름** | 조아현 |
| **학번** | 20215247 |
| **소속**  **학과/대학** | 소프트웨어융합대학  빅데이터학과 |
| **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 두번째 과제 입니다. 최소신장트리(MST) 및 최단경로(Shortest Path) 알고리즘을 구현하는 과제입니다.

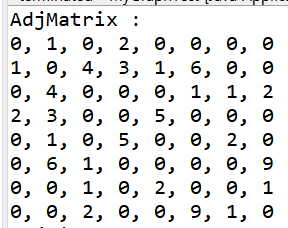
## <실습 과제>

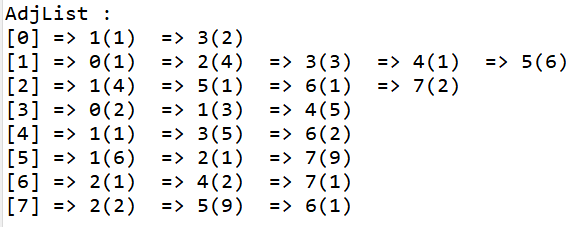
|  |
| --- |
| **[Q 0] 요약 [배점: 10]**  이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3문장으로 요약하세요. |

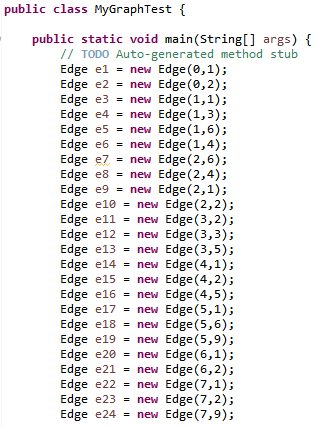
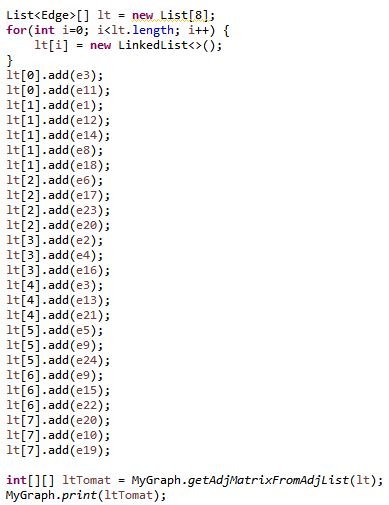
답변:

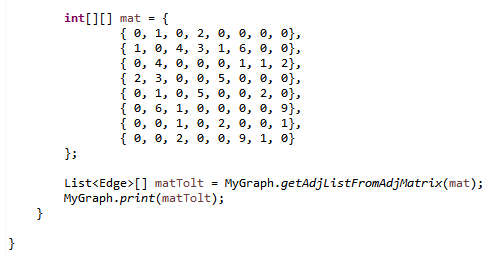
최소신장트리(MST)는 하나의 연결 성분으로 이루어진 무방향 가중치 그래프에서 간선의 가중치 합이 최소인 신장트리를 말한다.  
최단경로(Shortest Path)는 주어진 가중치 그래프에서 출발점ㅇ로부터 도착점까지의 최단경로를 찾는 것이다. 음수 가중치가 없는 그래프에서 동작하는Dijkstra 알고리즘과 음수 가중치가 있어도 동작하는Bellman-Ford 알고리즘이 있다.

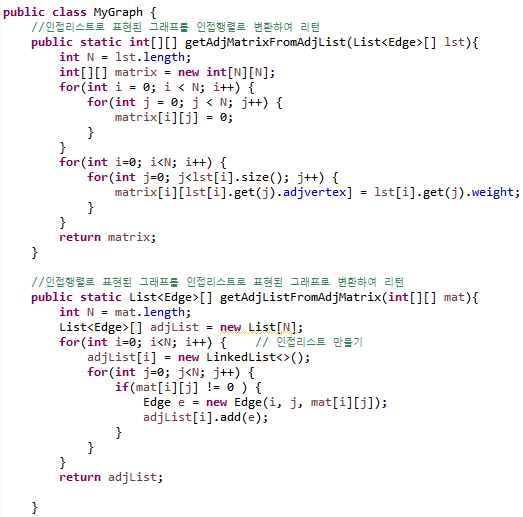
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[Q 1] 인접행렬? 인접리스트? 상호 변환하기 [10점]**  그래프는 인접행렬 또는 인접리스트로 구현(=표현)할 수 있습니다. 둘 간에 변환을 해 주는 메소드를 작성하세요. 참고: 이 문제에서는 무방향 가중치 그래프만 고려합니다.  MyGraph 클래스를 만들고, 아래의 메소드를 구현하세요   * public static int[][] getAdjMatrixFromAdjList(List<Edge>[] lst); (인접리스트로 표현된 그래프를 인접행렬로 변환하여 리턴) * public static List<Edge>[] getAdjListFromAdjMatrix(int[][] mat); (인접행렬로 표현된 그래프를 인접리스트로 표현된 그래프로 변환하여 리턴) * public static void print(int[][] mat); // 인접행렬을 터미널에 출력 * public static void print(List<Edge>[] lst); // 인접리스트를 터미널에 출력 * 두개의 print 메소드 출력 예시 (example): 4개의 정점으로 구성된 그래프  |  |  | | --- | --- | | Print(인접행렬 mat) | Print(인접리스트 lst) | | 참고: 숫자는 가중치를 의미함 | 참고: 2(4) 에서 2는 정점의 인덱스 번호, (4)는 가중치를 의미함. |   테스트를 위해 MyGraphTest 클래스를 생성하고, main 메소드에서 아래의 그래프를 사용해서 MyGraph의 4가지 메소드를 테스트 하세요.    [Task 1]   * 주어진 그래프를 (하드 코딩으로) 인접리스트로 구현/표현한 후 print 메소드 호출 * 인접리스트를 인자로 하여 getAdjMatrixFromAdjList 메소드 실행 * 리턴 된 행렬을 print   터미널 화면을 캡처하고 본 문서에 첨부하세요.  [Task 2]   * 주어진 그래프를 (하드 코딩으로) 인접행렬로 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. |

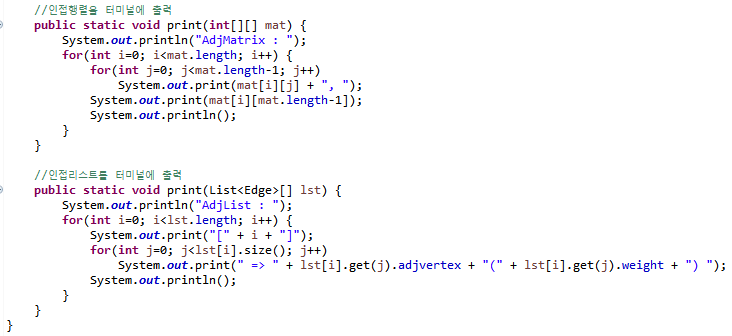
답변 [Task 1]:  


답변 [Task 2]:  
 







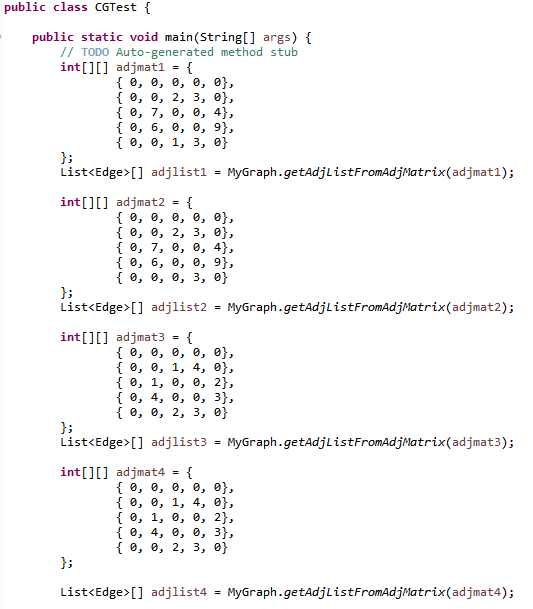
|  |  |  |
| --- | --- | --- |
| **[Q 2] 뒤집어라 엎어라 [10점]**  인접행렬 또는 인접리스트로 표현된 방향 또는 무방향 가중치 그래프 G가 있습니다. G가 G를 뒤집은 GR과 **동일한지를 확인**하는 메소드를 구현하세요. 아래의 4가지 그래프 예제에서 그래프 1과 2는 G와 GR이 서로 다르고, 그래프 3과 4는 G와 GR이 동일합니다.   |  |  | | --- | --- | |  | \* 참고: 무방향 그래프는 양쪽방향으로 간선이 모두 존재하는 것과 같습니다. |   CheckGraph 클래스를 만들고, 아래의 메소드를 추가하세요   * public static boolean isReverseSame(int[][] G); // G와 GR이 동일하면 true, 그렇지 않으면 false를 리턴. 인자는 G만 주어짐(메소드 내에서 GR 을 계산하고 G와 비교하여 결과를 리턴하는 코드를 작성하기) * public static boolean isReverseSame(List<Edge>[] G); // G와 GR이 동일하면 true, 그렇지 않으면 false를 리턴. 인자는 G만 주어짐(메소드 내에서 GR 을 계산하고 G와 비교하여 결과를 리턴하는 코드를 작성하기)   [Task 1] <그래프 1>에 해당하는 인접행렬 adjmat1 및 인접리스트 adjlist1를 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 adjlist2를 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 adjlist3를 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 adjlist4를 구현하세요. 다음으로, isReverseSame (adjmat4) 과 isReverseSame (adjlist4)을 호출하고 리턴되는 결과를 터미널에 출력하세요. 터미널 화면을 캡처하고 본 문서에 첨부하세요.  소스코드도 제출해야 합니다. |

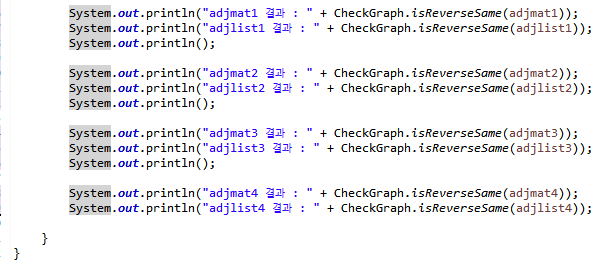
답변 [Task 1]:

답변 [Task 2]:

답변 [Task 3]:

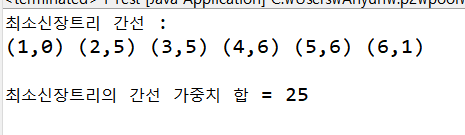
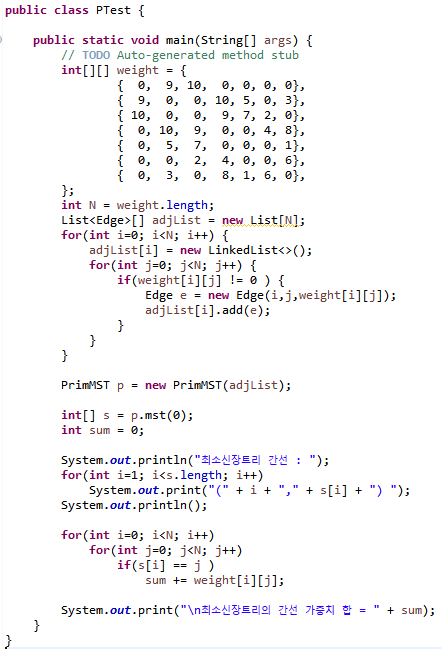
답변 [Task 4]:

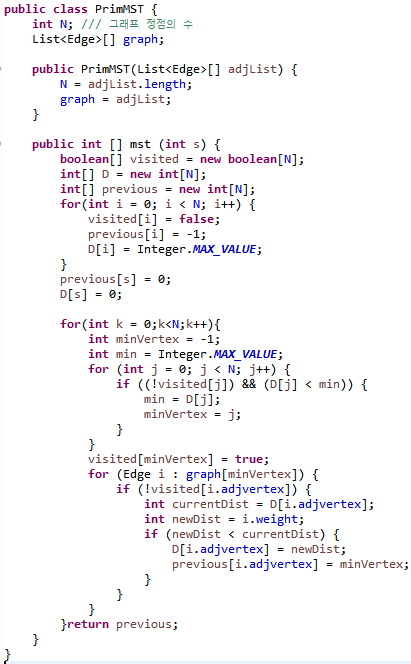




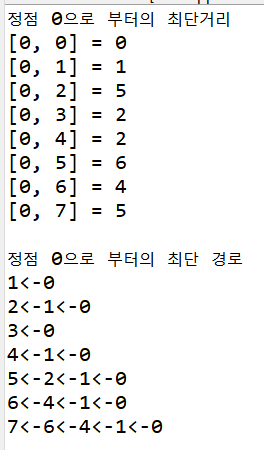
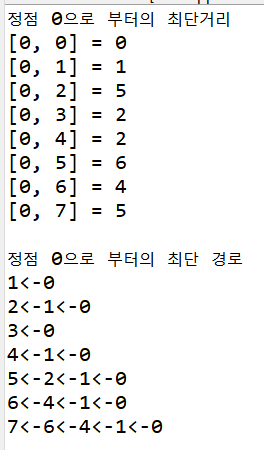
|  |
| --- |
| **[Q 3] PrimMST 구현하기 [20점]**  강의자료와 동일하게, 최소신장트리를 찾는 PrimMST 클래스를 구현하세요.  [Task] 강의자료 p.14에 표시된 입력 그래프를 입력으로하여 PrimMST 클래스를 실행하세요. 강의자료 p.14와 같이 터미널에 출력하도록 main 메소드를 구현하고 실행하세요. 참고로, “최소신장트리 간선”을 열거하는 코드와 “최소신장트리의 간선 가중치 합”을 계산하는 코드는 PrimMST 클래스에 구현해도 되고, 또는 main 메소드에 구현해도 됩니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

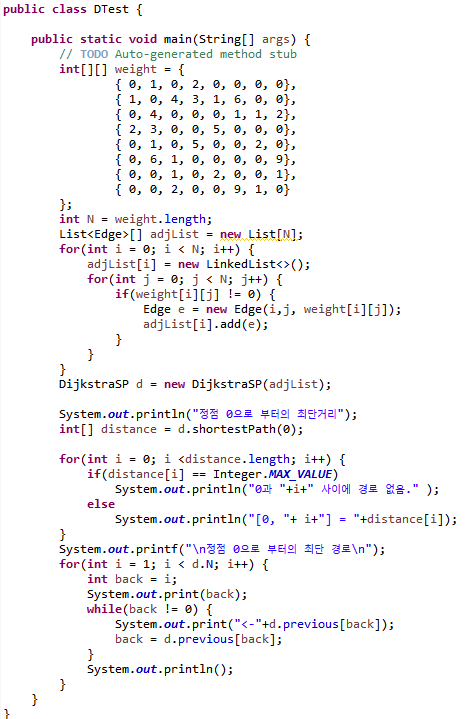
답변 [Task] :

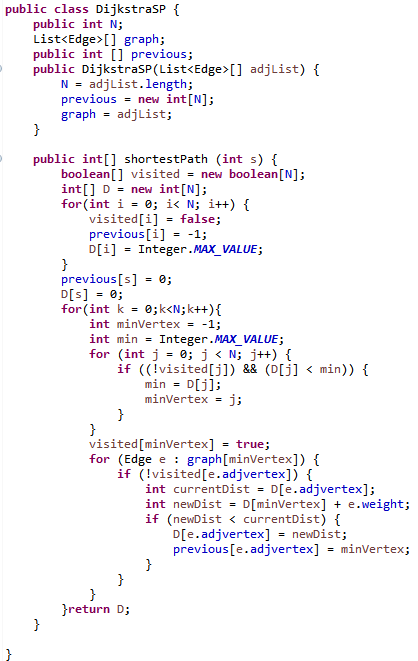
  




|  |
| --- |
| **[Q 4] 최단경로 : Dijkstra 알고리즘 구현하기 [25점]**  강의자료와 동일하게, 최단경로를 탐색하는 Dijkstra 알고리즘을 구현하는 Dijkstra 클래스를 생성하세요.  [Task] 강의자료를 참고하여 테스트를 위한 main 메소드도 구현하세요(p.25~). 수행 결과가 강의자료 p.27과 같아야 합니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

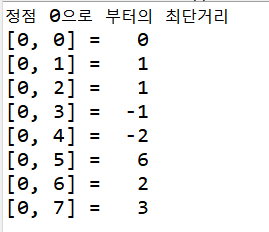
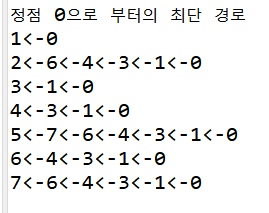
답변 [Task] :   
 

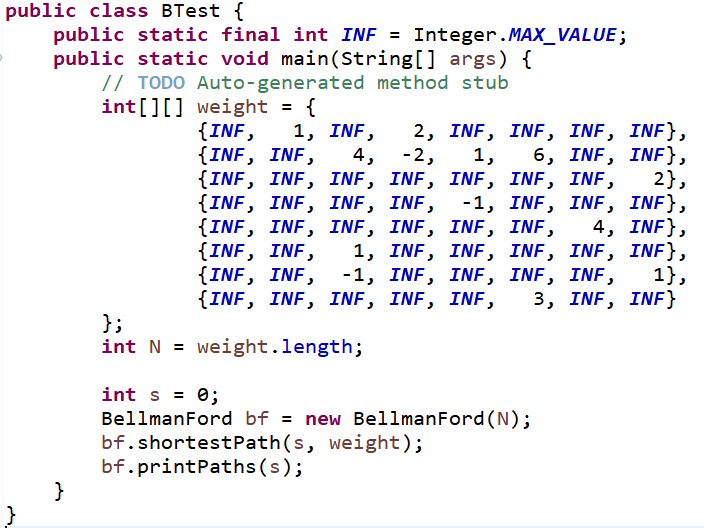




|  |
| --- |
| **[Q 5] 최단경로 : Bellman-Ford 알고리즘 구현하기 [25점]**  강의자료와 동일하게, 최단경로를 탐색하는 Bellman-Ford 알고리즘을 구현하는 BellmanFord 클래스를 생성하세요.  [Task] 강의자료를 참고하여 테스트를 위한 main 메소드도 구현하세요(p.37~). 수행 결과가 강의자료 p.38과 같아야 합니다.  터미널 화면을 캡처하고 본 문서에 첨부하세요. 소스코드도 제출해야 합니다. |

답변 [Task] :

****

****

**끝! 수고하셨습니다 ☺**