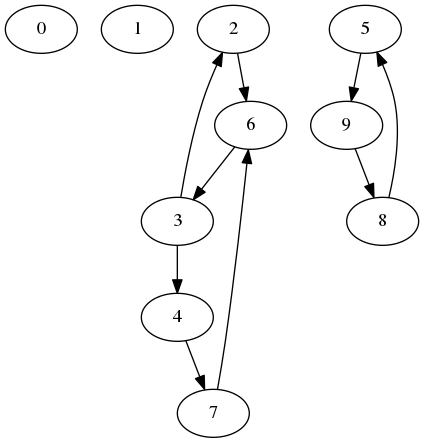
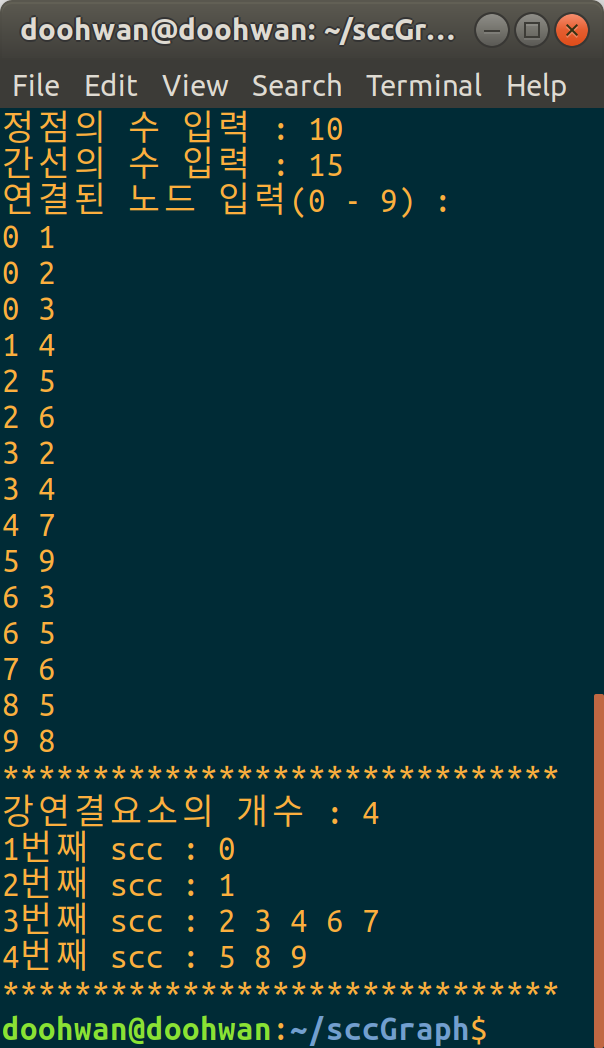


**컴퓨터과학과 알고리즘 과제2**

**강연결요소의 이해**

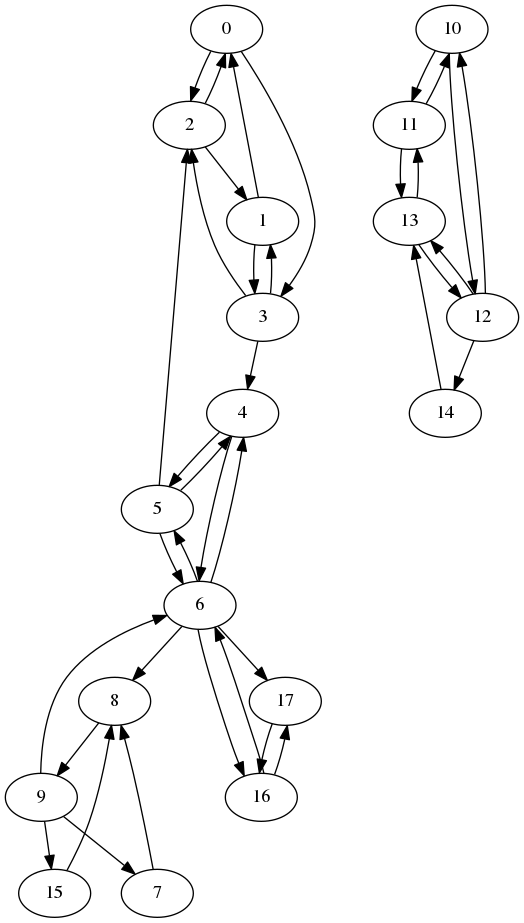
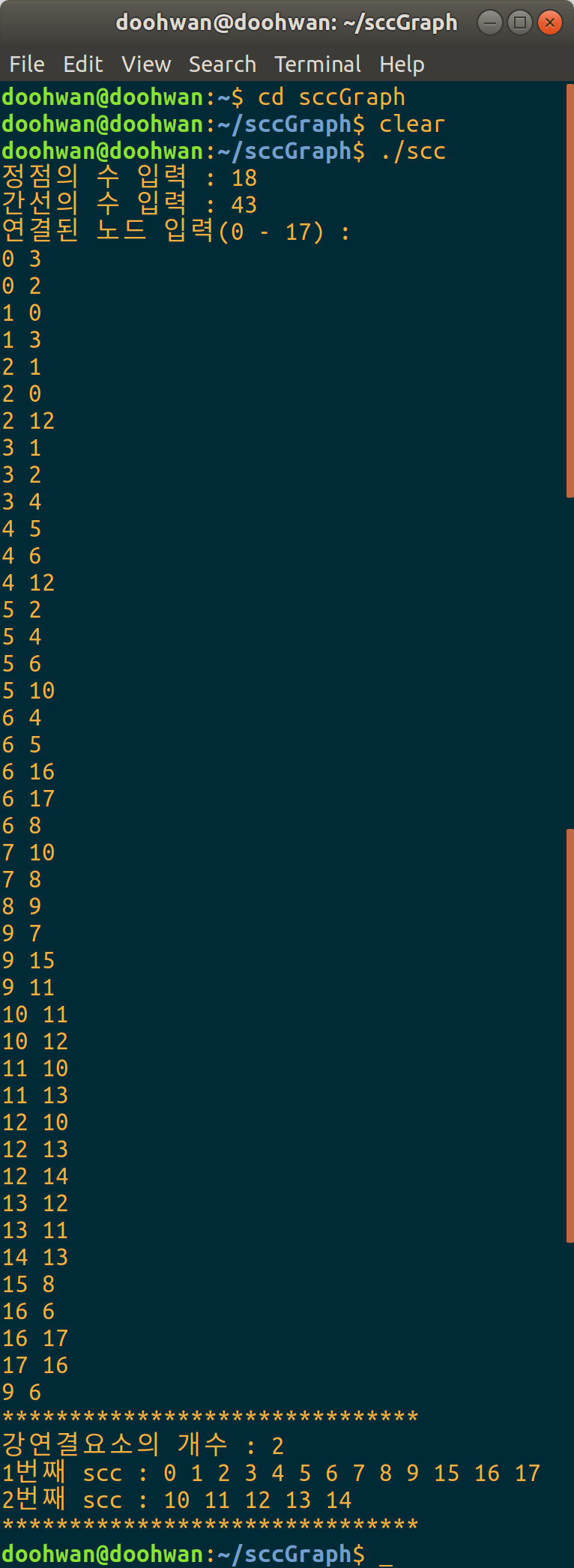
**2017080066 김두환**

**1. 교재 p323 그래프**

****

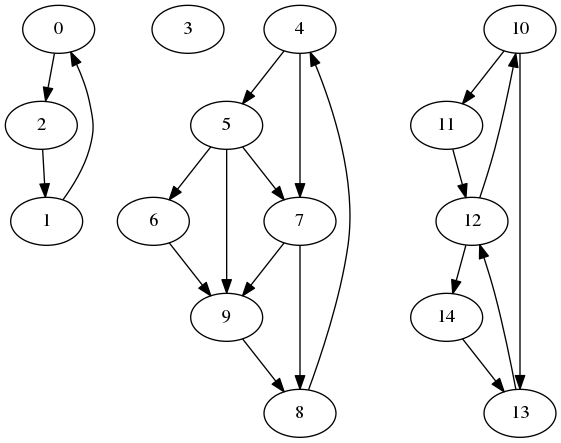
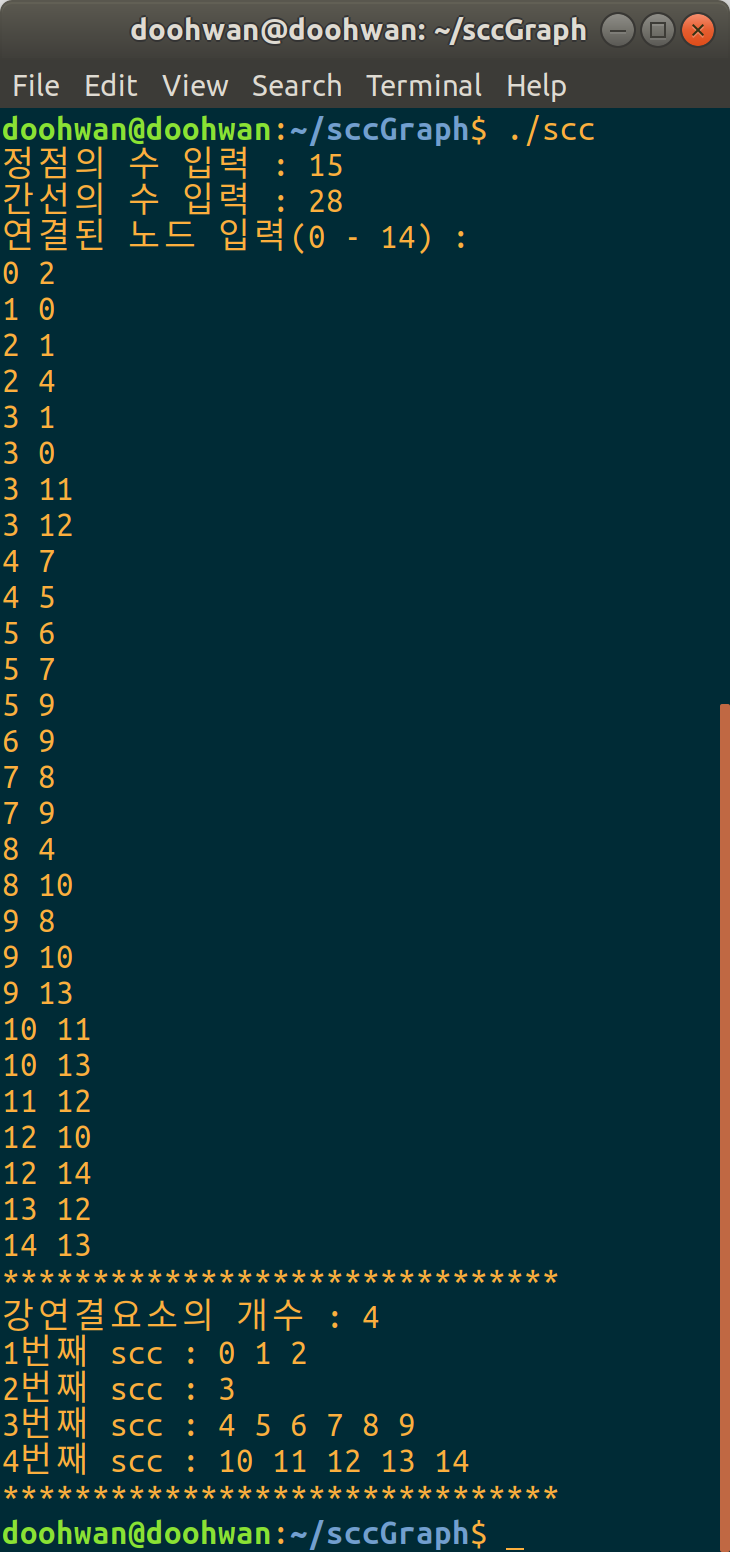
**(터미널 출력) (출력 이미지)**

**2. 임의의 그래프**

****

**(터미널 출력) (출력 이미지)**

**3. 임의의 그래프**

****

**(터미널 출력) (출력 이미지)**

**문제 해결 방법**

**가)그래프를 어떻게 구현하였는가?**

**· 정점, 간선의 입력과 간선의 방향 입력**

**프로그램 시작 시 입력 정점과 간선의 총 개수를 입력 받으며 간선의 수만 큼 x-> y의 연결된 정점의 번호를 입력 받아 처리한다. 여기서 x->y는 정점x에서 정점y로 가는 진출 간선이므로 역방향 간선 그래프를 만들기 위해 y->x의 그래프 또한 만들어 주었다.**

**· 정점들의 번호는?**

**정점들의 번호는 결과 그래프를 이미지로 출력하기 위한 파이프라인을 고려하여 인덱스 0번을 시작으로 하였으며 입력 받는 총 정점의 개수에 따라 0 - (총 정점들의 개수 -1)로 할당된다.**

**나)그래프는 어떻게 내부적으로 표현하였는가? (자료구조)**

**그래프 G와 역그래프 Gr은 모두 2차원 배열(cpp에서 2d vector를 사용)을 이용하여 구현하였으며 처리된 강연결요소를 구하기 위해서도 2차원 배열을 이용하였다. 중간에 DFS를 이용하여 정점 v의 완료시간 f[v]를 계산하기 위해서는 스택을 사용하였고 방문 된 정점을 확인하기 위해 1차원 배열을 사용 하였다.**

**즉 그래프는 인접행렬로 구현하였으며, scc를 구하기 위해 추가적인 각 1개의 스택과 1차원 배열을 사용하였다.**

**다)그래프의 결과를 보이기 위하여 어떻게 출력할 것인가?**

**강연결요소를 구하기 위한 모든 실행이 끝난 후 2차원 배열 scc에 이중루프를 이용하여 각 강연결요소 집합의 정점들을 콘솔화면으로 출력하였다. 아래는 코드와 설명이다.**

|  |  |
| --- | --- |
| **for(int i = 0; i < r; i++){**  **printf("%d번째 scc : ", i+1 );**  **for(int x : scc[i]){**  **printf("%d ", x);**  **}**  **printf("\n");**  **}** | **r은 2차원 배열 scc의 길이이다.**  **scc[i]에 있는 값들은 각 강연결요소의 집합이다.** |

**(계속)**

**이미지출력을 위해서 C++의 boost 라이브러리와 Graphviz(Third Party software package)를 이용하였다. 아래는 주요 소스코드와 설명이다**

|  |  |
| --- | --- |
| **for(int i = 0; i < r; i++){**  **if(scc[i].size() != 1){**  **for(int x : scc[i]){**  **for(int y : vt[x])**  **for(int z : scc[i])**  **if(z != x and y == z)**  **used\_by.push\_back(make\_pair(x, z));**  **}}}** | **Scc[i]의 배열에는 각 강연결요소 집합의 정점 번호밖에 없기 때문에 이 문제를 해결하기 위하여 각 강연결요소의 정점에 대응 되는 원래의 인덱스 정점 정보(vt 배열)를 사용하여 연결시켜 주었다. 자기경로를 제외 하기 위해 추가적으로 z != x로 처리를 하였다.**  **used\_by 변수는 정점들의 연결 관계를 나타내기 위하여 사용하는 배열이다.** |
| **const int n\_edges = used\_by.size();**  **vector<int> weights(n\_edges, 1);**  **typedef boost::adjacency\_list<**  **boost::vecS,**  **boost::vecS,**  **boost::directedS,**  **boost::property< boost::vertex\_color\_t, boost::default\_color\_type >,**  **boost::property< boost::edge\_weight\_t, int >**  **> Graph;**  **Graph g(used\_by.begin(), used\_by.end(), weights.begin(), v);** | **인접리스트를 나타내기 위한 adjacency\_list 자료형을 typedef로 기본값(여기서 directedS는 방향성 그래프를 의미)을 설정하여 Graph 자료형으로 선언하였고 앞서 연결 관계를 나타낸 used\_by를 정점들의 관계로 Graph 변수 g를 만들었다.** |
| **{**  **std::ofstream f("scc.dot");**  **boost::write\_graphviz(f, g, boost::make\_label\_writer(&names[0]));**  **f.close();**  **}**  **system("dot scc.dot -Tpng -o scc.png")** | **Boost의 graphviz라이브러리는 그래프를 시각하기 위해 .dot 포맷의 변환을 지원한다.**  **.dot 파일을 이미지파일로 변환하기 위해 추가적으로 소스코드 내에서 시스템콜을 사용하여 .png 이미지로 변환하였다(system("dot scc.dot -Tpng -o scc.png")).** |

**경상대학교 컴퓨터과학과 2017080066 김두환**