알고리즘 과제 2 보고서

이름: 박재언
학번: 2019-12551

이번 과제의 목적은 방향 그래프에서 강하게 연결된 컴포넌트(Strongly Connected Components, SCC)를 찾는 것이다. 주어진 입력은 정점 수와 간선 수, 그리고 각 간선의 방향이 주어지는 형식이다. 본인은 Kosaraju 알고리즘을 사용하여 두 번의 깊이 우선 탐색(DFS)을 통해 SCC 를 구하였다. 과제의 요구 사항에 따라 그래프를 인접 행렬, 인접 리스트, 인접 배열의 세 가지 형태로 구현하고 각각에 대해 동일한 알고리즘을 적용하였다.

본 과제에서는 Java 언어(OpenJDK 23)를 사용하였으며, macOS 환경의 MacBookPro 에서 실험을 진행하였다. DFS 연산의 소요 시간만을 측정하기 위해 `System.nanoTime()`을 사용하였으며, 입력 파싱이나 그래프 변환, 결과 출력 등은 시간 측정에서 제외하였다.

예시 입력(example.txt)을 기준으로 실험한 결과는 다음과 같다. 인접 행렬을 사용했을 때는 0.181625 밀리초, 인접 리스트에서는 0.398250 밀리초, 인접 배열에서는 0.177333 밀리초가 소요되었다. 출력 결과는 세 방식 모두 정확히 일치하였으며, SCC 는 [0 1 2], [3 4 5], [6 7] 총 3 개의 컴포넌트로 구성되었다.

인접 행렬 방식은 구현이 직관적이지만 메모리 사용량이 많고, 특히 희소 그래프(sparse graph)에서는 비효율적이다. 인접 리스트 방식은 대부분의 경우에 균형 잡힌 성능을 보이며, 디버깅이 쉬운 장점이 있다. 인접 배열 방식은 배열을 기반으로 압축된 형태로 그래프를 표현하므로 메모리 사용 효율이 높고, 실행 속도도 가장 빠른 결과를 보였다. 단, 구현이 다소 복잡하고 정점과 간선 인덱스를 정확히 관리해야 한다는 점에서 주의가 필요하다.

전체적으로 Kosaraju 알고리즘은 세 표현 방식 모두에서 안정적으로 동작하였으며, 성능 차이는 표현 방식에 따라 미세하게 나타났다. 후속 연구로는 Tarjan 알고리즘과의 성능 비교, 다양한 밀도(density)의 대형 실세계 데이터셋에 대한 실험, 병렬화 가능성 탐색 등을 진행하면 유의미할 것으로 판단된다.