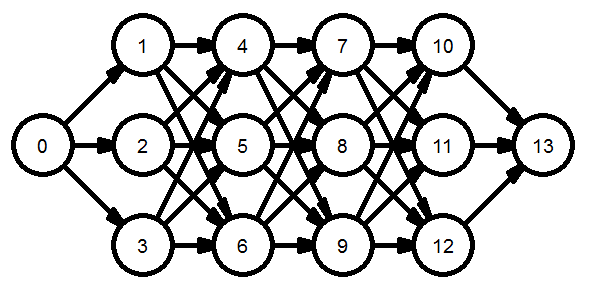
선형대수 프로젝트 보고서

Problem4

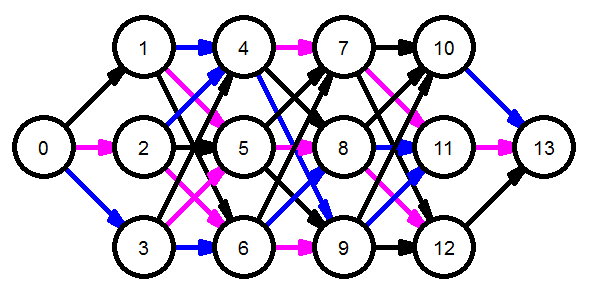
이름: 배지운

학번: 2015004584

1. 문제 톺아보기



선택한 문제는 4번으로 크게 두 가지 문제가 주어진다. 공통적인 개요는 네트워크가 그래프의 형태(V, E)의 꼴로 주어지고 시작점, 끝점 (s, t) 가 주어질 때 s(0)에서 t(13)로 데이터 M을 전송하고자 할 때 각 E마다 최대 통신 가능한 데이터 량(f)이 정해져 있고 통신료(d)가 주어져 있고 데이터 f를 d의 통신료로 전송하면 d\*f 만큼의 비용이 들 때 1) 모든 간선 E의 통신료가 동일 할 때, s에서 t로 데이터M 을 전송할 때 각 V에서 연결된 간선이 위에서부터 차례로 3,5,10의 전송 량을 갖고 t로 연결된 간선도 동일하게 3,5,10의 전송 량을 갖는다고 주어진다면 1초 동안 s에서 t까지의 최대 데이터 통신 량을 구하는 문제와 2) 통신료가 아래의 그림과 같이 검은색 2원, 파란색 4원, 분홍색 5원으로 주어져있고 M=9이며 f가 문제 1) 과 같다고 할 경우 데이터 전송에 드는 최소 비용을 구하는 문제이다.



문제를 처음 보았을 때 든 생각은 MCFP(minimum-cost flow problem)이었다. 최근 ACM-ICPC본선에도 진출했었고, SCPC 예선을 꽤 높은 점수로 통과했을 만큼 평소에 알고리즘 공부를 열심히 하고 있어서 다행히 알고 있는 알고리즘이었다. 하지만 대회에서 문제를 푸는 것과 알고리즘에 대해 자세히 이해하고 개선해보는 것은 다르기 때문에 좀 깊게 알아보고 해당 문제에 대해 적용할 때 개선할 수 있는 부분이 있는지 더 자세히 알아보고자 했다.

1. 선행 지식

먼저 주어진 문제는 가중치 방향 그래프에서 최단경로를 찾는 문제와 유량을 흘릴 수 있는 경로를 찾는 문제가 선행되어야 한다.

최단경로를 구하는 Algorithm은 보통 Floyd-Warshall Algorithm이나 Bellman-Ford Algorithm혹은 Dijkstra Algorithm (가중치가 양수일 경우)을 사용하게 된다. (어떠한 사이클도 가중치의 합이 음수가 없다면 최단경로를 찾을 수 있다. 단, 없다면 최단경로는 존재하지 않는다.)

Floyd-Warshall Algorithm은 time complexity가 으로 매우 간단한 알고리즘이다. 기본적으로 DP(Dynamic Programming)적 접근이 쓰이며, 어떤 점 부터 까지의 최단경로를 모두 구할 수 있다. 거쳐가는 꼭지점, 출발점, 도착점에 대해 반복문을 실행하면서 최단경로를 갱신해주는 것으로 쉽게 이해할 수 있다.

Bellman-Ford Algorithm은 time complexity가 주어진 에 대해 다. 시작점에서 출발하여 최단경로를 계속 갱신하는 작업을 거치면 된다.

Dijkstra Algorithm은 주어진 에서 각각의 v에 대해 s에서 v까지의 최단 경로 d[v]를 저장하면서 작동한다. Greedy Algorithm을 사용하면서 s에서의 최단거리가 확정된 정점의 집합에서 어떤 정점 v까지의 거리가 최소인 정점 v를 선택하면 다음 최단 거리가 확정된 집합에 v를 추가할 수 있고 이런 식으로 반복하다 보면 모든 정점에 대해 최단거리를 구할 수 있다는 사실을 바탕으로 한다.

유량을 어떻게 흘려 보낼 것인가에 대한 문제는 Network Flow에 해당하는 문제로 여기서는 Maximum Flow를 구한다. 기본적으로 Network Flow는 간선에 흐를 수 있는 유량은 간선의 용량보다 작아야 하고 시작 점과 끝 점을 제외한 모든 점에서의 나가는 유량과 들어오는 유량의 합은 같아야 한다는 속성을 가지고 있는다.

이 Maximum Flow(최대 유량)를 구하기 위해서 BFS(Breadth-first search)로 탐색하는 Ford-Fulkerson Algorithm을 사용하면 time complexity 에 구할 수 있고 또는 Dinic’s Algorithm을 사용하면 에 구할 수 있다.

1. 풀어보기 전에

먼저 풀이에 해당하는 알고리즘의 실행시간을 정확히 측정하기 위해 입/출력과 시간 측정하는 부분을 공통되도록 작성하고 100번 실행하여 평균을 측정하는 것으로 알고리즘의 수행속도를 측정하도록 하였다.

여기에 주어진 그래프 에 의해 결과가 달라질 수 있는지 적절히 조절하여 각각의 결과를 또 추가하여 고려하도록 하였다.

실행 시간 측정에 사용되는 함수들은 timeMeasure.h에 작성되어 있고 입/출력은 STL의 입출력 표준인 fstream과 iostream을 사용하였다.

1. 풀어보기

문제 1)은 Maximum Flow를 구하는 문제로 보통 BFS로 해결할 수 있다. 위에서 주어진 그래프 G에 대해 BFS를 적용하는 Ford-Fulkerson Algorithm을 적용하여 maximum flow를 찾는 풀이는 MaximumFlow.cc로 작성해두었다.

간선의 유량을 0으로 두고 시작하여 잔여용량(용량 – 유량)이 남은 간선을 BFS로 탐색해 증가경로를 찾는 것으로 요약할 수 있습니다.

Jim Orlin’s algorithm은 O(VE)에 구할 수 있다는데 짐작도 안 간다.