graph - Concept

그래프란?

1. 트리는 비선형 구조 중에서도 위계가 있는 계층 구조를 다루기 위해 특화되었다. 힙 또한 계층 구조이며 이는 우선순위라는 이름에서 알 수 있다.
2. 그래프는 추상화된 개념과의 연결관계를 표현하기 위해 만들어졌다. 가령, 아래의 예시를 보자.

```
가) 사당 – 방배 – 서초 – 교대 – 강남 – 역삼 – 선릉 – 한티 – 도곡 – 구룡 – 개포동
나) 한양 –(41km)– 수원 –(104km)– 아산
다) 왼발 – 오른발
라) 빨강 – 주황 – 노랑 <– 파랑
마) 1 – 2 – 5
   ||__ 6 – 3
   |___ 4 __|
바) A –> B –> C –> D –> E
    |__> D –> F –> G
```

- 이 중에서 그래프가 아닌 건 무엇일까? 전부 그래프다.
- 이처럼 관계가 순환이든 간선에 가중치가 있든 노드를 색깔로 이분을 하든 간선이 방향이 있든 전부 연결관계만 표현하면 그래프다.

그래프의 표현과 분류

1. 그래프는 G(V, E)로 표현한다. G는 graph 이며 V는 vertex(정점, 노드), E는 edge(간선)이다. 즉, 정점들의 집합과 간선의 집합으로 구성된 자료구조다.
2. 위의 예시를 통해 그래프를 분류해 보자.

```
가) 무향 그래프(undirected graph)
나) 가중치 그래프(weighted graph)
```

```
다) 단순 그래프(simple graph)
라) 방향 그래프(directed graph)
마) 이분 그래프(bipartite graph)
바) 비순환 유향 그래프(directed acylic graph)
```

- 무향 그래프는 간선에 방향이 없는 그래프를 말한다. 사당 -> 방배 갔다가 방배 -> 사당으로 가도 상관이 없다.
- 가중치 그래프는 간선마다 특정 수치가 있는 그래프를 말한다. 해당 수치가 긍정적인 수치인지 부정적인 수치인지는 그걸 지정하는 사용자에 따라 다르다.
- 단순 그래프는 두 정점 사이에 최대 한 개의 간선만 있는 그래프를 말한다. 그래서 가) 무향 그래프도 마찬가지로 단순 그래프로 분류할 수도 있다.
- 이분 그래프는 그래프의 정점들을 겹치지 않는 두 개의 그룹으로 구성하는 것이다. 즉, 서로 다른 그룹들 간에만 간선이 존재하도록 만든 것. (1, 5, 3)이 직접적으로 이어지지 않고 (2, 6, 4)가 직접적으로 이어지지 않은 게 보이는가?
- 비순환 유향 그래프는 한 점에서 출발해 출발점으로 되돌아오는 경로가 없는 그래프다. 단, 간선 방향만 없으면 DAG는 출발점으로 되돌아올 수 있는 그래프이다.

3. 그래프가 참 종류도 많다. G(V, E)만 만족하면 다 그래프 아닌가? 맞다. 그리고 잘 보면 알겠지만, 분류법에서 굉장히 간선을 중히 여기는 게 보이는가? 가중치, 방향, 이분 전부 간선이 기준 아닌가? 그렇다. 왜냐하면 그래프에서는 간선을 이어서 만들어지는 경로가 문제 해결의 핵심이기 때문이다.

그래프의 경로

1. 경로(path)는 출발점부터 도착점까지 이어지는 간선들을 순서대로 나열한 것이다.
2. 가령 아래의 예시를 보자

```
1 -> 2 -> 3 -> 4 -> 5
|__> 9 -> 11___^

(1, 2, 3, 4, 5)
(1, 9, 11, 4, 5)
```

- 위 괄호 안에 들어간 내용을 보자. 간선의 방향을 따라 1에서 5까지 순차적으로 들어간 게 보이는가? 이처럼 한 정점을 한 번 만 지나는 경로를 단순 경로라고 한다.

- 만약, 간선에 방향이 없으면 1 - 2 - 3 - 2 - 1 - 9 - 11 - 4 - 5 - 4 - 3 - 2 - 1 같은 정신나간 경로도 가능할 것이다. 현대 그래프 이론에서는 단순 경로를 주로 얘기하기에 위와 같은 정점이 중복되는 경로는 잘 나오지 않겠으나, 엄연히 저것도 경로라는 걸 알아두자.
- 만약 1 - 9 - 11 - 9 - 1 이렇게 출발점으로 다시 돌아오는 경로가 있으면 이를 회로(cycle)라고 한다. 순우리말로 표현하면 도돌이길, 되돌이길이라 표현할 수도 있겠다.

3. 그럼 도대체 이 경로를 이용해서 무슨 문제를 풀 것인가? 아래 예시에서 살펴보도록 하자.

그래프 사용 예시

1. 철도망의 안전성 분석
   - 서울역에서 천안역으로 가는 철도가 중간에 끊어졌다. 그래서 우회로를 찾고자 하려면 어떻게 하는가? 서울역에서 천안역으로 가는 정점과 간선을 찾고 그 중에 이어진 간선을 찾아 경로를 이으면 된다.
2. 소셜 네트워크 분석
   - SNS 사이트에서 회원을 정점으로 하고 친밀도를 가중치로 두어 간선을 구축한다. 그리고 가중치가 높은 순서대로 그래프를 이으면 그래프의 깊이(트리에서 거론한 그 깊이다)가 얕을 수록 나와 사이가 가까운 회원이 나올 것이다.
3. 인터넷 전송 속도 계산
   - 인터넷과 연결된 라우터와 컴퓨터를 정점으로 하고 케이블을 간선으로 표현하면 그래프가 하나 만들어진다. 여기서 간선이 된 케이블에 케이블 별 전송용량을 가중치로 두면 전송 속도가 계산이 될 것이다. 이걸 응용한 게 Webserv, ft_irc와 관련하여 네트워크 공부할 때 잠깐 나오는 스패닝 트리다.
4. 한 붓 그리기
   - 연필을 떼지 않고 주어진 도형을 그리되 모든 선을 한 번 씩만 지나는 퍼즐이다. 도형의 꼭지점을 정점으로 삼고 변을 간선으로 삼아서 단순 경로를 찾아내면 된다. 이 문제를 오일러 경로(Eulerian path)라고도 부른다.
5. 외환 거래
   - 각 통화(원, 달러, 위안, 엔, 유로 등)를 정점으로 두고 통화들 사이에 간선을 둔 뒤에 가중치를 통화 간 교환 비율이라 해보자.
   - 하나의 통화를 출발점으로 하여 여러 간선을 거쳐간 뒤에 다시 출발점의 통화로 되돌아 왔을 때 더 수익을 벌 수 있는 경로를 찾아보자. 즉, 환차익이 날 수 있는 곳을 계산하는 것이다. 어떻게 계산할 수 있을까?
   - 출발점 -> 출발점으로 돌아오는 회로(cycle)를 구성했을 때, 회로에 들어간 간선의 가중치를 전부 곱하면 1이 초과하는 순간 이득을 버는 것이다. 이를 통해 가장 값이 큰 경로를 찾으면 가장 환차익을 많이 벌 수 있다.

암시적 구래프 구조

1. 위의 예시처럼 딱 그래프인 것처럼 보이지 않더라도 그래프로 구성하면 쉽게 풀 수 있는 문제들이 존재한다. 이를 보도록 하겠다.
2. 할 일 목록 정리
    - 외출을 한다 -> 외출복을 입어야 한다 -> 외출복을 빨래 해야 한다 -> 빨래하기 위해 세제가 필요하다
    - 이처럼 어떠한 일에 의존 관계가 있을 시, 이 일을 한 번에 하나씩 해나갈 가장 빠른 방법을 구하려면 각 일을 정점으로 삼고 순서를 방향 삼아 간선으로 이은 뒤, 하나의 일에 걸릴 시간을 가중치로 삼아 가장 가중치가 작은 경로를 계산하면 된다.
3. 15-퍼즐
    - 4 x 4 게임판에 1부터 15까지의 숫자가 적힌 열다섯 개의 타일이 끼워져 있고 이 타일들을 움직여 원래 있던 순서대로 정렬해보자.
    - 현재 타일의 배치를 하나의 정점으로 하고 한 배치에서 한 타일을 움직였을 때, 얻을 수 있는 배치를 정점으로 하여 정점과 정점을 잇는 간선으로 구성한다. 각 정점을 만들고 원래 배치로 되돌아 가는 회로를 만든다.
    - 회로에서 가장 짧은 횟수로 원래 배치로 돌아가는 걸 찾으려면 당연히 경로가 가장 짧은 것이 정답이다.
4. 게임판 덮기
    - 가로 N, 세로 N으로 나뉘어져 있는 정사각형의 게임판을 1x2 크기의 블록으로 덮는 문제를 생각해 보자.
    - 게임판의 일부는 막혀 있으며 블록을 겹쳐서 놓을 수 없을 때, 막히지 않은 모든 칸에 블록을 놓을 수 있는 방법이 있는가?
    - 게임판의 막히지 않은 칸을 정점으로 하고 사방에 인접한 칸들 사이에 간선을 연결하는 그래프를 구성한다. 이는 이분 그래프로 구성할 수 있다.
5. 회의실 배정
    - N개의 팀이 각각 회의를 하려고 하는데 회의실이 하나 뿐이다. 각 팀은 회의실을 사용하고 싶은 시간을 두 개 씩 적어 낸다. 이 때, 모든 팀이 겹치거나 분할되지 않고 온전히 전부 회의를 할 수 있게 회의실을 배정할 수 있는가?
    - 어느 팀이 회의를 할 때와 하지 않을 때를 각각 정점으로 두고 회의 하지 않는 경우에 그 때 시간을 점유할 수 있는 팀을 각각 정점으로 삼아 간선으로 잇는다. 회의를 할 때는 회의가 끝나고 점유할 수 있는 팀을 정점으로 삼아 간선으로 잇는다. 이렇게 만든 그래프에서 모든 팀이 속하는 경로를 찾아내면 된다.

그래프의 표현 방법

1. 행렬
    - 행렬로 표현하는 경우, 행은 현재 정점을, 열은 행과 연결된 다른 정점을 상징한다. 가령, (1, 2)는 1번 정점에 연결된 2번 정점이며 (1, 2)에 있는 값은 연결된 간선이 존재하는 지의 여부다. 물론, 이는 설정하는 사람에 따라 달라질 수 있다. 정점이 있는지와 더불어 가중치도 나타내는 역할도 할 수 있으니까.
2. 리스트

- 리스트는 각 정점을 기준으로 연결된 간선을 리스트로 연결한 것이다. 즉, 각 정점을 기준으로 하는 배열을 구성하고 해당 정점과 이어진 정정만 각 정점의 원소로 갖는 것이다. 쉽게 말하면 각각의 원소 개수가 다른 이차원 배열을 생각하면 되겠다.

3. 행렬과 리스트의 차이
    - 행렬은 행이 5개면 열도 5개다. 즉, 정점의 개수^2 만큼의 메모리 공간을 차지한다. 만약, 정점의 개수가 기하급수적으로 커지면 메모리가 터지지 않겠는가? 단, 인덱스로 바로 접근해서 쉬이 이어진 간선과 가중치를 알아낼 수 있는 점은 장점이라 할 수 있다.
    - 리스트는 각 정점에 존재하는 실제 간선의 개수만큼 원소를 갖는다. 즉 정점x원소이다. 행렬보다 덜 차지하지만 리스트는 각 정점과 연결된 원소의 인덱스와 원소의 정점값이 일치하지 않는다.
    - 보통, 위의 장단점을 취합하면 간선의 개수가 정점의 개수^2보다 현저히 적으면 리스트를 택하며 거의 비슷하면 행렬을 택한다.
4. 암시적 그래프 표현
    - 굳이 그래프로 그리지 않고 그래프 적인 방법만 채택해서 사용하는 것.
    - 가령, 미로에서 최단경로를 찾는다 생각해보자. 미로가 막힌 곳과 막히지 않은 곳이 간단하게 숫자로 표현되어 배열처럼 들어오면 이를 그래프로 만드는 데 수고로움이 들 것이다. 그럴 바에야 그냥 빈 칸의 위치를 기점으로 상하좌우를 보면서 인접한 칸으로 옮겨가며 탈출구를 찾으면 된다.
    - 즉, 간선의 목록을 찾는 대신에 현재 칸의 상하좌우를 점검하는 식이며 두 정점이 인접해 있는 지 확인하는 건 두 칸이 인접하는 걸로 갈음한다.
    - 위와 같이 그래프로 직접적으로 표현하는 대신 그래프의 표현 방식을 들고와서 간접적으로 사용하는 걸 암묵적 그래프 표현이라 한다.
    - 이와 같은 방법을 사용하면 그래프를 만들지 않기에 메모리 소모가 덜하고 마찬가지로 그래프를 만들고 그래프를 뒤지는 시간을 절약했기에 시간도 줄어든다.
    - 단, 그래프의 표현 방식을 차용해서 표현했기에 어떤 방식으로 차용했는 지 모르면 코드가 이해하기 어렵고 복잡해지는 단점이 있다.

그래서 그래프를 언제 쓰는데요?

- 위에서 언급한 예시에서도 나오기는 했으나 상세한 내용은 깊이 탐색, 너비 탐색, 최단 경로 알고리즘을 설명하는 주차에 차근차근 언급하도록 하겠다.