그래프를 이용한 기계 학습

#1 그래프란 무엇이고 왜 중요할까?

신기정
(KAIST AI대학원)

강사 소개

신기정(Kijung Shin)

경력:

2019년 2월 ~ 현재 : 카이스트 AI대학원 & 전기 및 전자공학부 조교수
2019년 2월 : Carnegie Mellon University 전산학 박사
2017년 & 2018년 여름 : LinkedIn 인턴연구원

연구 분야: 그래프를 위한/이용한 인공지능/빅데이터 기술

홈페이지: https://kijungs.github.io/ **이메일:** kijungs (AT) kaist.ac.kr

조교 소개

강신환(Shinhwan Kang)

경력:

2021년 3월 ~ : 카이스트 AI대학원 석사과정
2017년 3월 ~ 2021년 2월 : 고려대학교 컴퓨터학과 학사과정

연구 분야: 그래프 알고리즘

이메일: shinhwan.kang (AT) kaist.ac.kr

좋아하는 것: 카메라로 풍경 찍기(필름, 디지털)

조교 소개

윤득렬(Deukryeol Yoon)

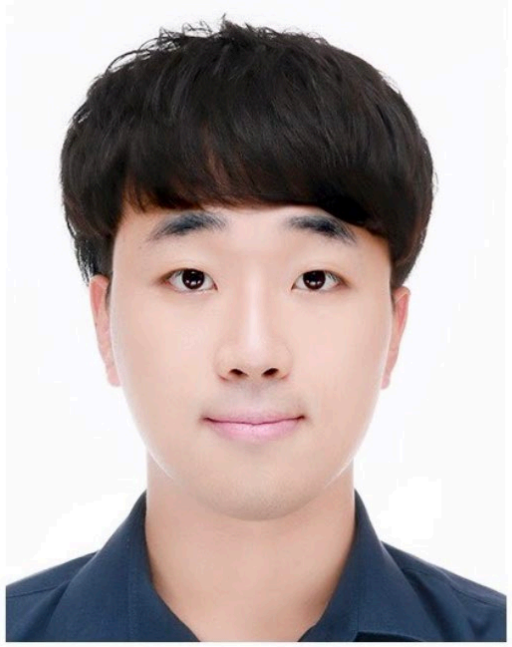
경력:

2021년 3월 ~ : 카이스트 AI대학원 석사과정
2017년 6월 ~ 2020년 5월 : 과학기술전문사관 현역연구원 (중위)

연구 분야: 그래프 네트워크 분석

이메일: deukryeol.yoon (AT) kaist.ac.kr

좋아하는 것: 여행, 맛집 탐방



조교 소개

이현주(Hyeonju Lee)

경력:

2019년 9월 ~ : 카이스트 AI대학원 석사과정
2013년 3월 ~ 2019년 8월 : 카이스트 전자과 학사과정

연구 분야: 그래프 분석 알고리즘

이메일: gladys59 (AT) kaist.ac.kr

좋아하는 것: 맛있는 것 해먹기, 노래 듣기

1. 그래프란 무엇이고 왜 중요할까?

2. 그래프 관련 인공지능 문제

3. 그래프 관련 필수 기초 개념

4. (실습) 그래프의 표현 및 저장

1. 그래프란 무엇이고 왜 중요할까?

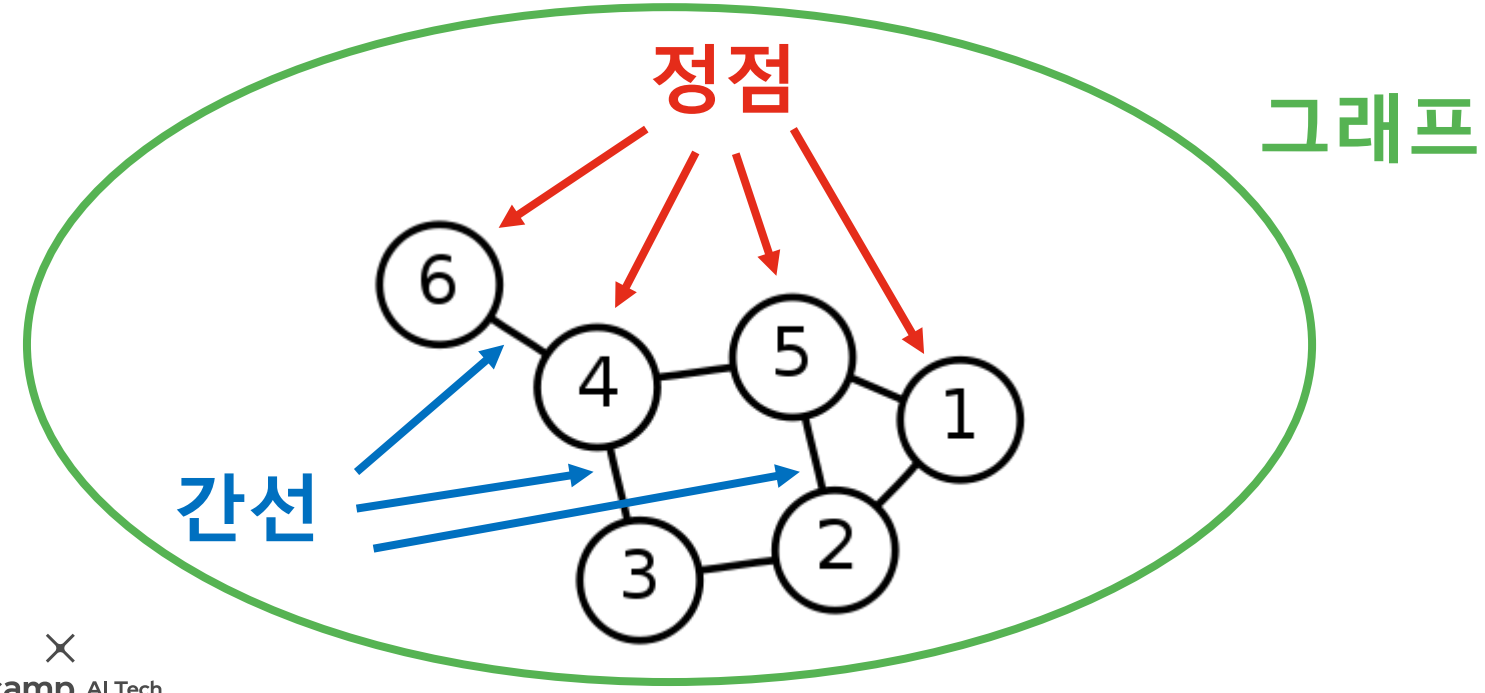
1.1 그래프란 무엇일까?

1.2 그래프는 왜 중요할까?

1.1 그래프란 무엇일까?

그래프(Graph)는 정점 집합과 간선 집합으로 이루어진 수학적 구조입니다

하나의 간선은 두 개의 정점을 연결합니다
모든 정점 쌍이 반드시 간선으로 직접 연결되는 것은 아닙니다

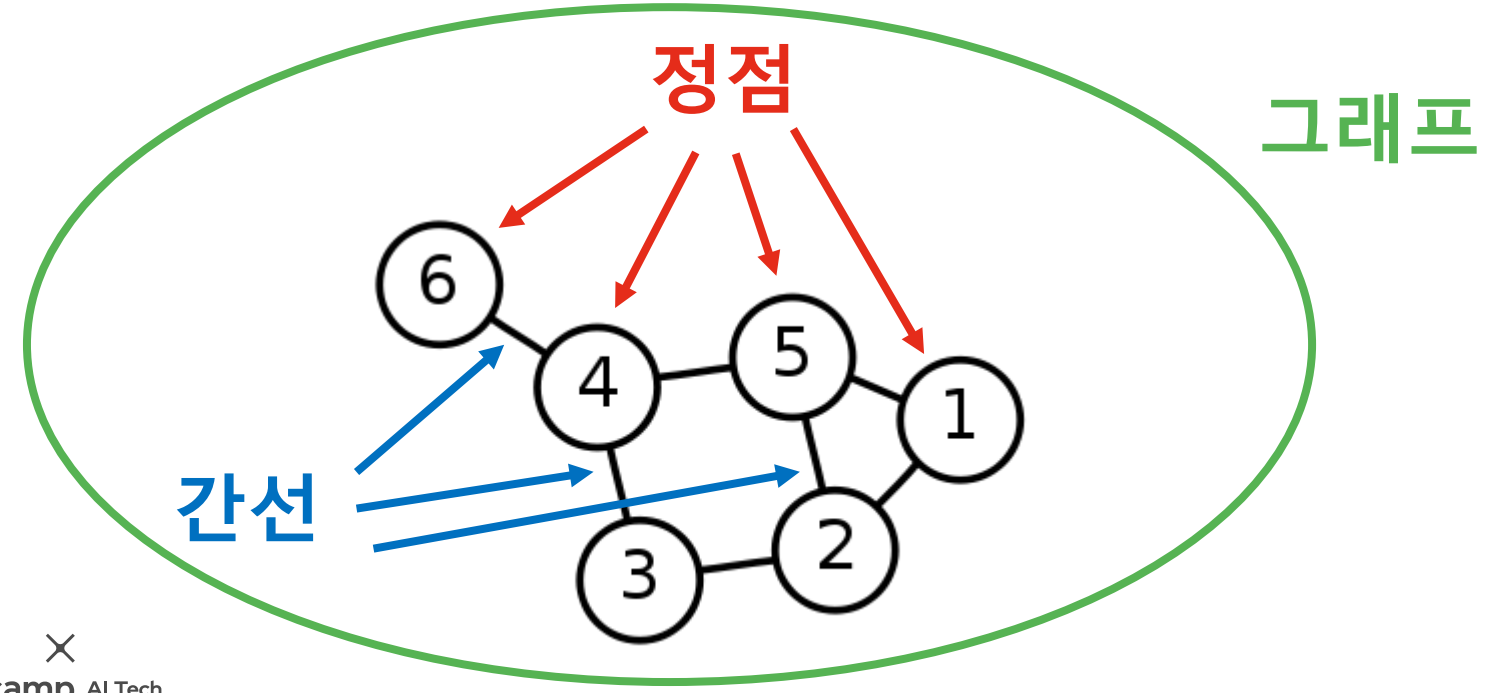


1.1 그래프란 무엇일까?

그래프(Graph)는 정점 집합과 간선 집합으로 이루어진 수학적 구조입니다

그래프는 네트워크(Network)로도 불립니다

정점(Vertex)은 노드(Node)로 간선은 엣지(Edge) 혹은 링크(Link)로도 불립니다



1.2 그래프가 왜 중요할까?

우리 주변에는 많은 복잡계(Complex System)가 있습니다

사회는 70억 인구로 구성된 복잡계입니다
통신 시스템은 전자 장치로 구성된 복잡계입니다
그 밖에도, 정보와 지식, 뇌, 신체 역시 복잡계로 생각할 수 있습니다

1.2 그래프가 왜 중요할까?

우리 주변에는 많은 복잡계(Complex System)가 있습니다

사회는 70억 인구로 구성된 복잡계입니다
통신 시스템은 전자 장치로 구성된 복잡계입니다
그 밖에도, 정보와 지식, 뇌, **신체** 역시 복잡계로 생각할 수 있습니다

Q. 이런 복잡계가 가진 공통적인 특성은 무엇일까요?

1.2 그래프가 왜 중요할까?

우리 주변에는 많은 복잡계(Complex System)가 있습니다

사회는 70억 인구로 구성된 복잡계입니다
통신 시스템은 전자 장치로 구성된 복잡계입니다
그 밖에도, 정보와 지식, 뇌, **신체** 역시 복잡계로 생각할 수 있습니다

Q. 이런 복잡계가 가진 공통적인 특성은 무엇일까요?
A. 구성 요소 간의 복잡한 상호작용입니다

1.2 그래프가 왜 중요할까?

우리 주변에는 많은 복잡계(Complex System)가 있습니다

사회는 70억 인구로 구성된 복잡계입니다
통신 시스템은 전자 장치로 구성된 복잡계입니다
그 밖에도, 정보와 지식, 뇌, **신체** 역시 복잡계로 생각할 수 있습니다

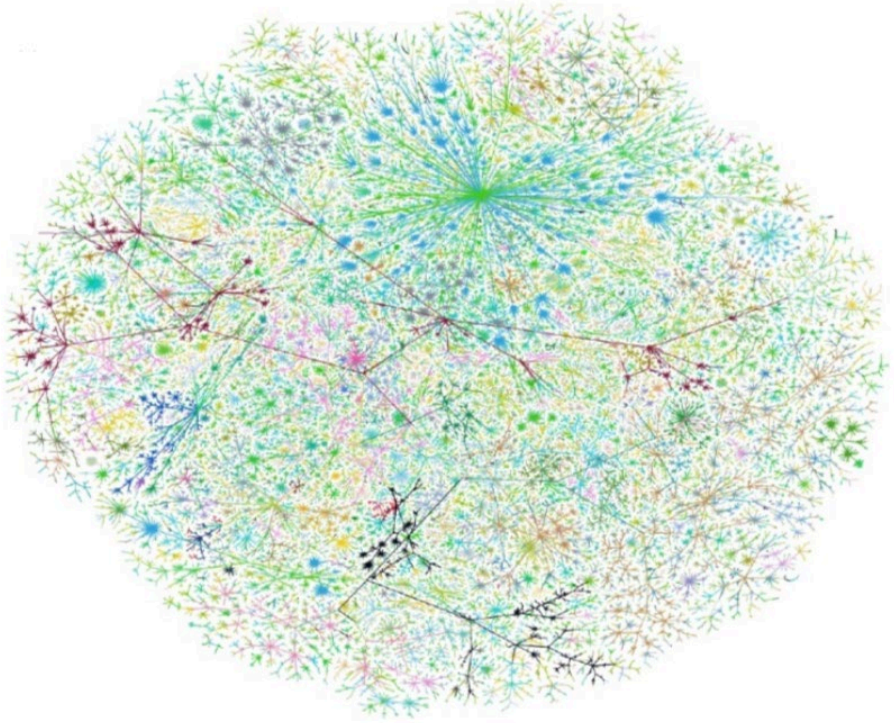
Q. 이런 복잡계가 가진 공통적인 특성은 무엇일까요?
A. 구성 요소 간의 복잡한 상호작용입니다

Q. 이런 복잡계를 어떻게 표현할까요?

1.2 그래프가 왜 중요할까?

정답은 그래프(Graph) 입니다!

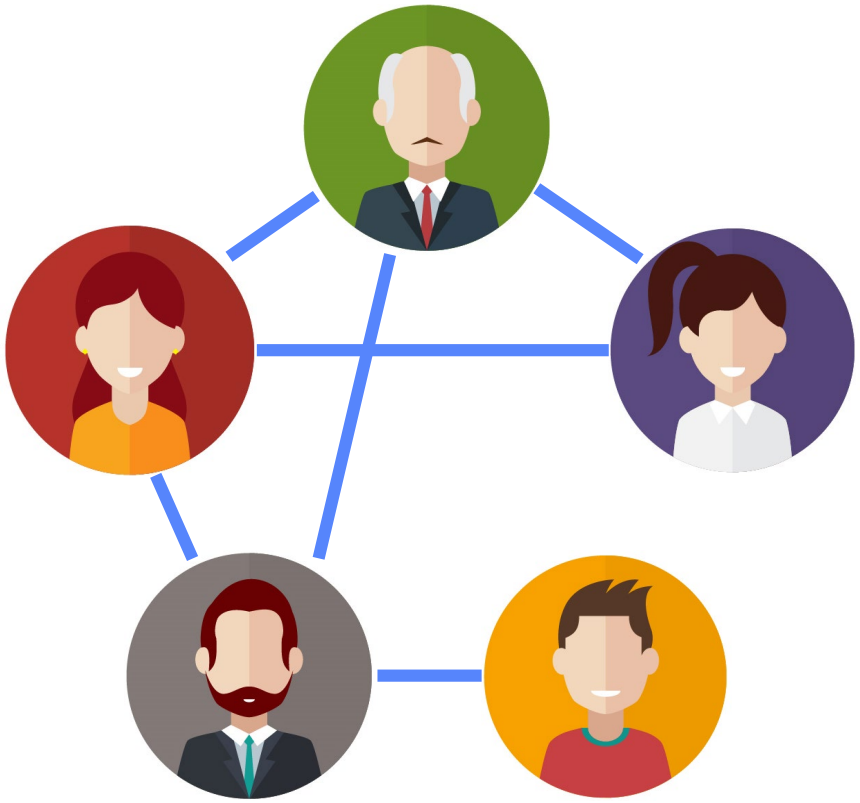
그래프는 복잡계를 표현하고 분석하기 위한 언어입니다



이미지 출처: http://web.stanford.edu/class/cs224w/slides/01-intro.pdf

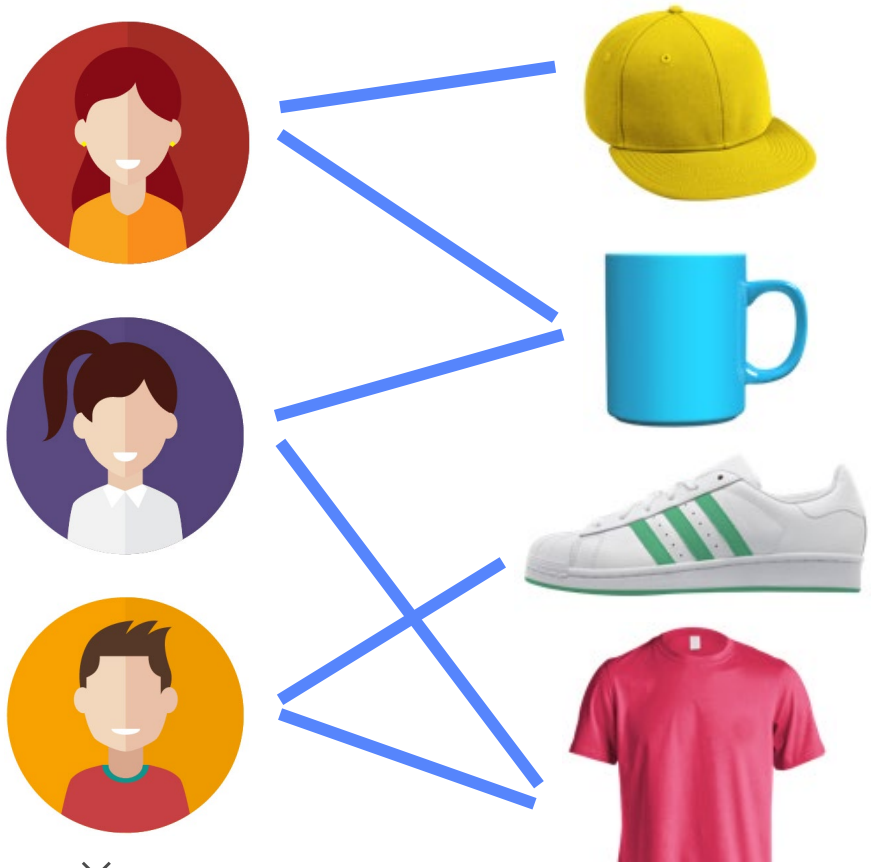
1.2 그래프가 왜 중요할까?

그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다



1.2 그래프가 왜 중요할까?

그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다



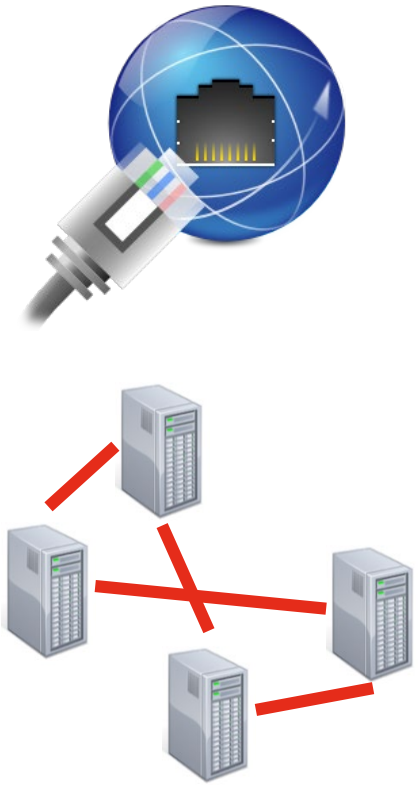
amazon

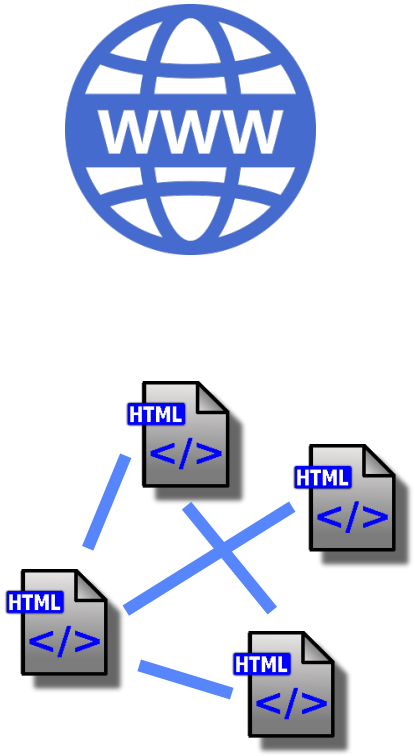
ebay

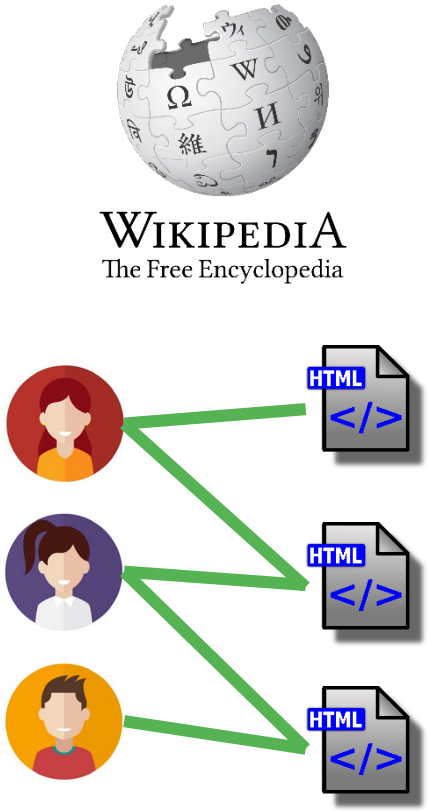
N 쇼핑

1.2 그래프가 왜 중요할까?

그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다

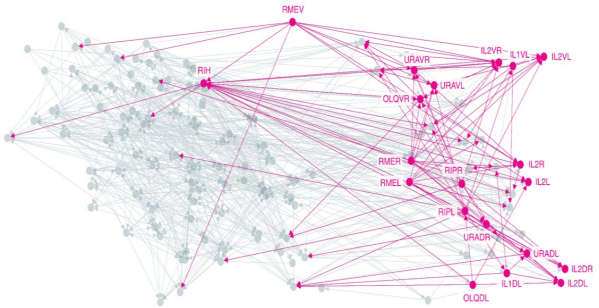




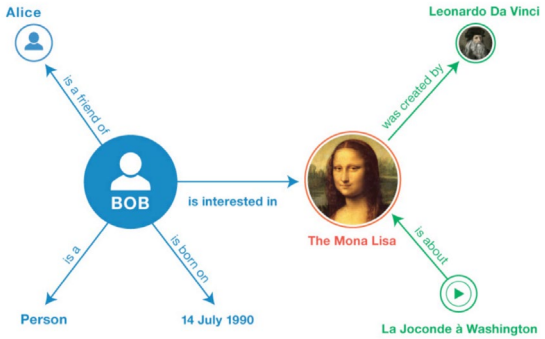


1.2 그래프가 왜 중요할까?

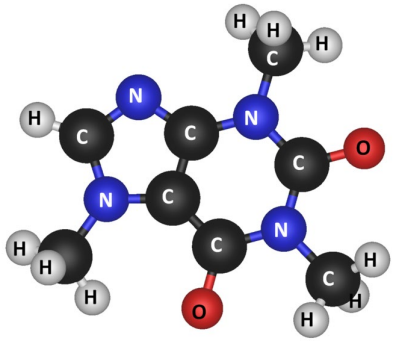
그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다



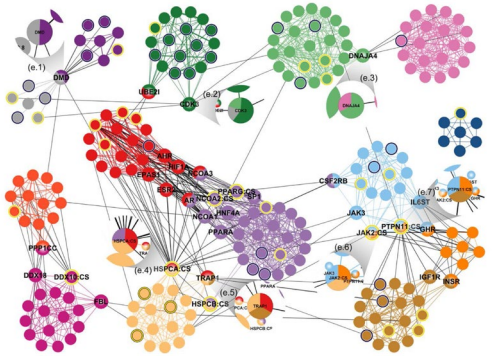
뇌 (뉴런 간 연결)



지식 그래프

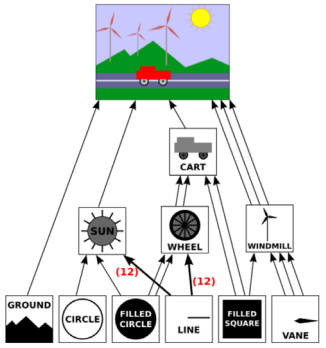


화학 분자



단백질 상호작용

세포간 유사도 그래프



이미지 분해

이미지 출처: http://web.stanford.edu/class/cs224w/slides/01-intro.pdf

1.2 그래프가 왜 중요할까?

그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다

복잡계는 구성 요소들 간의 상호작용으로 이루어집니다
상호작용을 표현하기 위한 수단으로 그래프가 널리 사용됩니다

1.2 그래프가 왜 중요할까?

그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다

복잡계는 구성 요소들 간의 상호작용으로 이루어집니다
상호작용을 표현하기 위한 수단으로 그래프가 널리 사용됩니다

복잡계를 이해하고, 복잡계에 대한 정확한 예측을 하기 위해서는
복잡계 이면에 있는 그래프에 대한 이해가 반드시 필요합니다

1.2 그래프가 왜 중요할까?

그래프는 복잡계를 효과적으로 표현하고 분석하기 위한 언어입니다

복잡계는 구성 요소들 간의 상호작용으로 이루어집니다
상호작용을 표현하기 위한 수단으로 그래프가 널리 사용됩니다

복잡계를 이해하고, 복잡계에 대한 정확한 예측을 하기 위해서는
복잡계 이면에 있는 그래프에 대한 이해가 반드시 필요합니다

그래프를 공부함으로써 복잡계가 등장하는 수많은 분야에 활용할 수 있습니다
전산학, 물리학, 생물학, 화학, 사회과학 등이 그 예시입니다

2. 그래프 관련 인공지능 문제

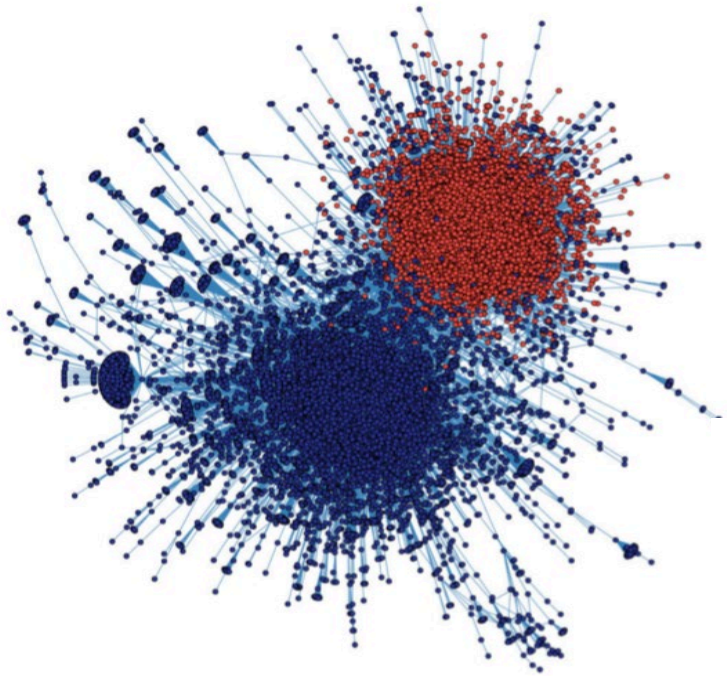
2.1 그래프 관련 인공지능 문제

2.2 우리 수업의 목적 및 범위

2.1 그래프 관련 인공지능 문제

정점 분류(Node Classification) 문제

트위터에서의 공유(Retweet) 관계를 분석하여, 각 사용자의 정치적 성향을 알 수 있을까?



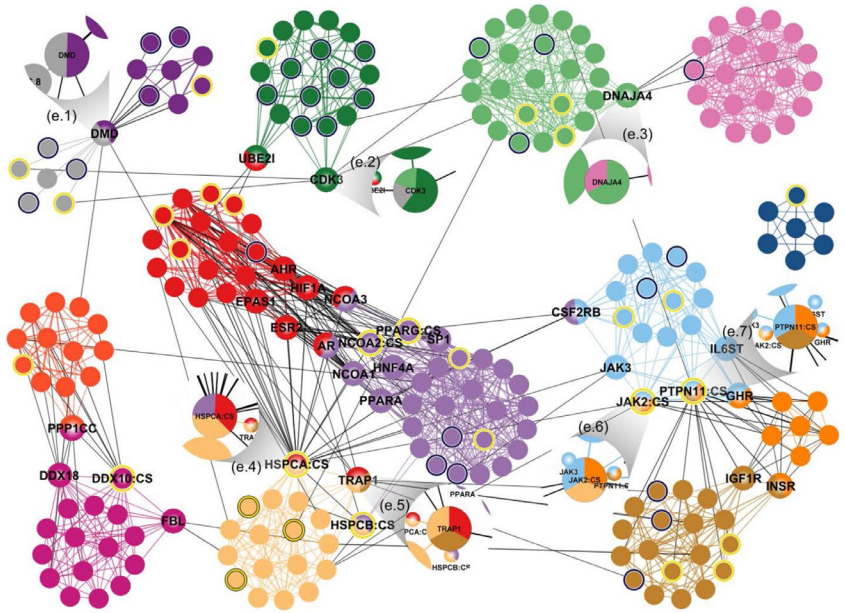
twitter

이미지 출처: http://web.stanford.edu/class/cs224w/slides/01-intro.pdf

2.1 그래프 관련 인공지능 문제

정점 분류(Node Classification) 문제

단백질의 상호작용을 분석하여 단백질의 역할을 알아낼 수 있을까?



이미지 출처: https://www.visual-computing.org/2016/01/18/group-structures-graphs/diss_cv01/

2.1 그래프 관련 인공지능 문제

연결 예측(Link Prediction) 문제

페이스북 소셜네트워크는 어떻게 진화할까?

이미지 출처: https://blog.revolutionanalytics.com/2010/12/facebooks-social-network-graph.html

2.1 그래프 관련 인공지능 문제

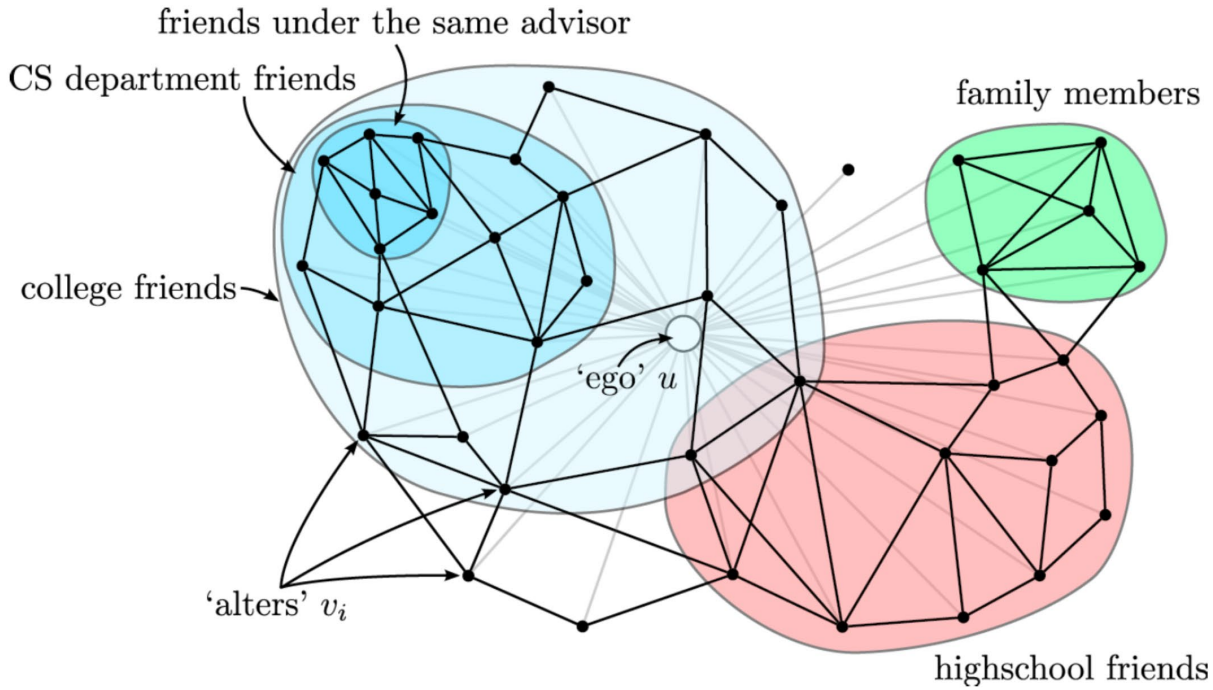
추천(Recommendation) 문제

각자에게 필요한 물건은 무엇일까? 어떤 물건을 구매해야 만족도가 높을까?

2.1 그래프 관련 인공지능 문제

군집 분석(Community Detection) 문제

연결 관계로부터 사회적 무리(Social Circle)을 찾아낼 수 있을까?

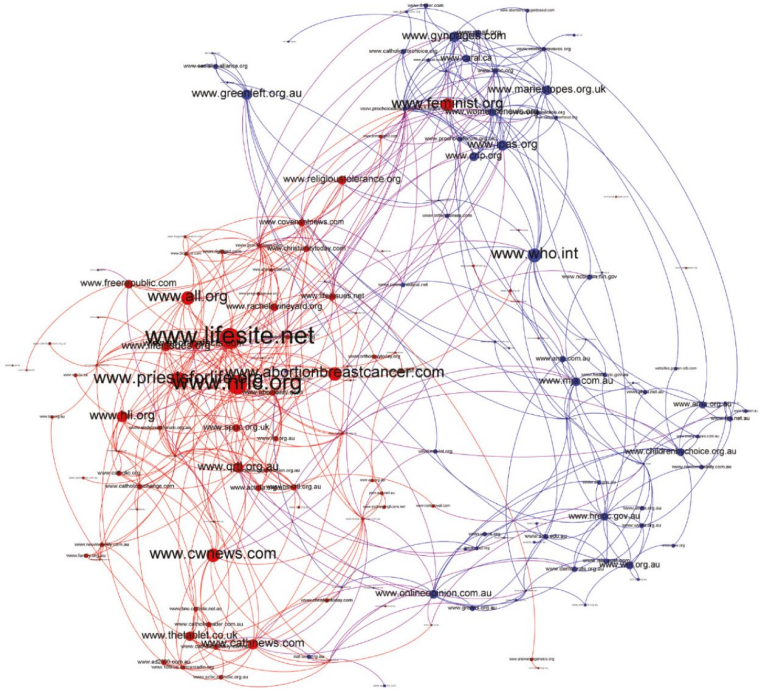


이미지 출처: http://web.stanford.edu/class/cs224w/slides/01-intro.pdf

2.1 그래프 관련 인공지능 문제

랭킹(Ranking) 및 정보 검색(Information Retrieval) 문제

웹(Web)이라는 거대한 그래프로부터 어떻게 중요한 웹페이지를 찾아낼 수 있을까?

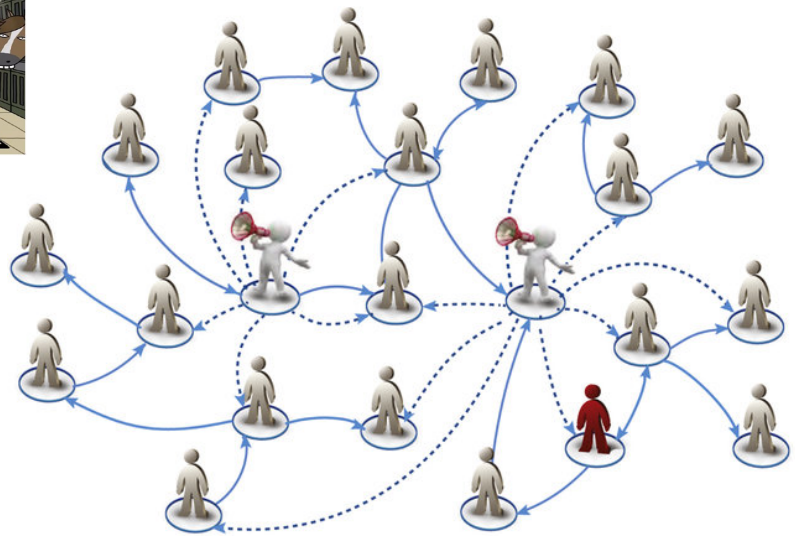


이미지 출처: https://www.ucldigitalpress.co.uk/Book/Article/45/70/3461/

2.1 그래프 관련 인공지능 문제

정보 전파(Information Cascading) 및 바이럴 마케팅(Viral Marketing) 문제

정보는 네트워크를 통해 어떻게 전달될까? 어떻게 정보 전달을 최대화 할 수 있을까?



2.2 우리 수업의 목적 및 범위

앞선 질문과 관련된 인공지능 기술을 배웁니다

정점 분류, 연결 예측, 추천, 군집 분석, 랭킹, 정보 검색, 정보 전파, 바이럴 마케팅 등

2.2 우리 수업의 목적 및 범위

앞선 질문과 관련된 인공지능 기술을 배웁니다

정점 분류, 연결 예측, 추천, 군집 분석, 랭킹, 정보 검색, 정보 전파, 바이럴 마케팅 등

최첨단 기술보다는 기초와 직관적인 방법론에 집중합니다
하지만 그래프 신경망(Graph Neural Networks) 등 최첨단 기술도 일부 소개합니다

3. 그래프 관련 필수 기초 개념

3.1 그래프의 유형 및 분류

3.2 그래프 관련 필수 기초 개념

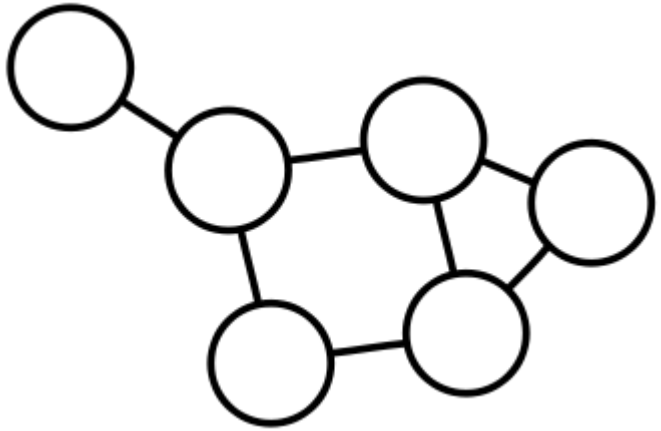
3.1 그래프의 유형 및 분류

방향이 없는 그래프(Undirected Graph) vs 방향이 있는 그래프(Directed Graph)

간선에 방향이 없는 그래프

✓ 협업 관계 그래프

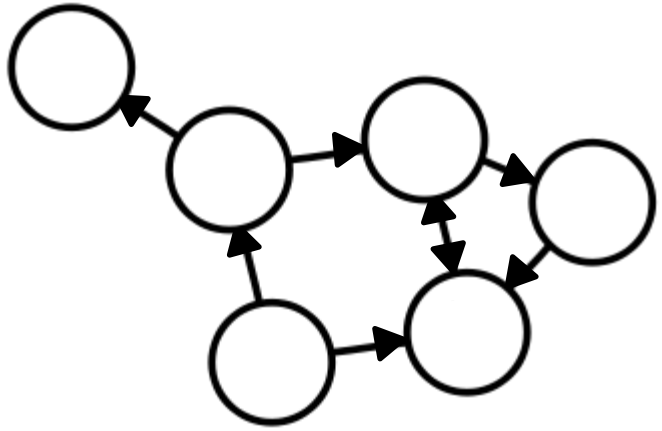
✓ 페이스북 친구 그래프



간선에 방향이 있는 그래프

✓ 인용 그래프

✓ 트위터 팔로우 그래프



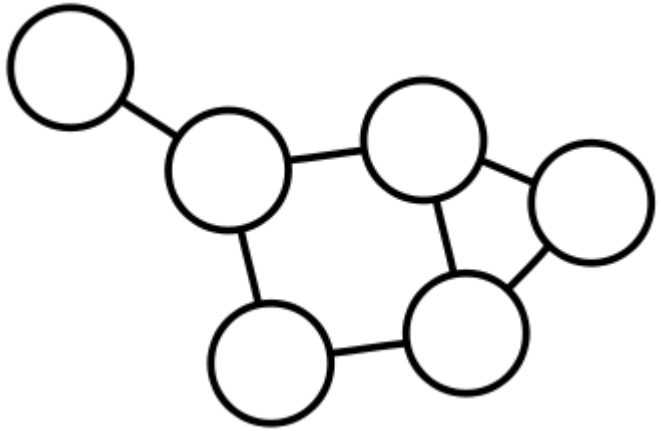
3.1 그래프의 유형 및 분류

가중치가 없는 그래프(Unweighted Graph) vs 가중치가 있는 그래프(Weighted Graph)

간선에 가중치가 없는 그래프

✓ 웹 그래프

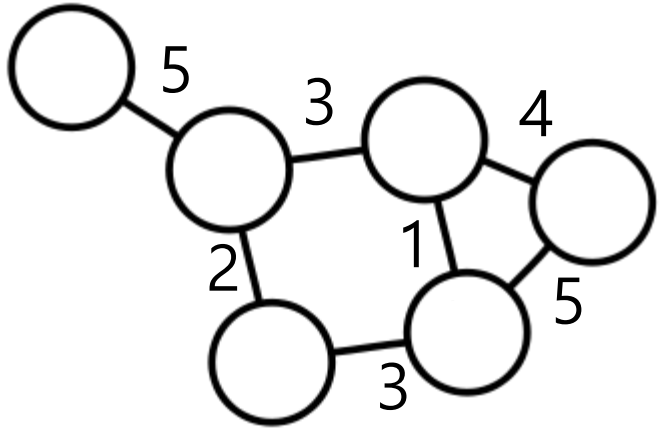
✓ 페이스북 친구 그래프



간선에 가중치가 있는 그래프

✓ 전화 그래프

✓ 유사도 그래프



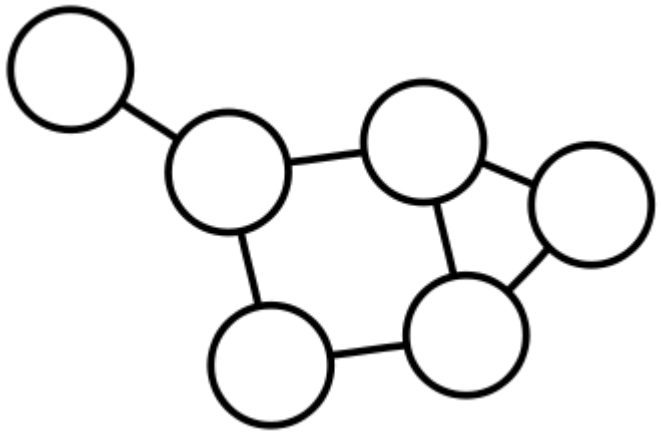
3.1 그래프의 유형 및 분류

동종 그래프(Unpartite Graph) vs 이종 그래프(Bipartite Graph)

동종 그래프는 단일 종류의 정점을 가집니다

✓ 웹 그래프

✓ 페이스북 친구 그래프

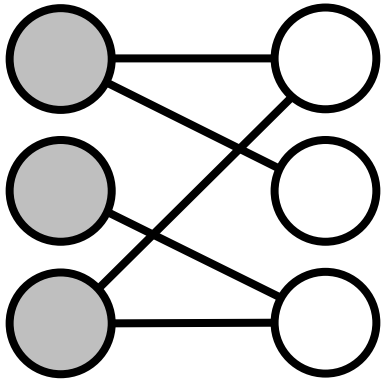


이종 그래프는 두 종류의 정점을 가집니다

다른 종류의 정점사이에만 간선이 연결됩니다

✓ 전자 상거래 구매내역 (사용자, 상품)

✓ 영화 출연 그래프 (배우, 영화)



3.1 그래프의 유형 및 분류

Q. 다음 전자 상거래 구매 내역은 어떤 유형의 그래프일까요?

3.1 그래프의 유형 및 분류

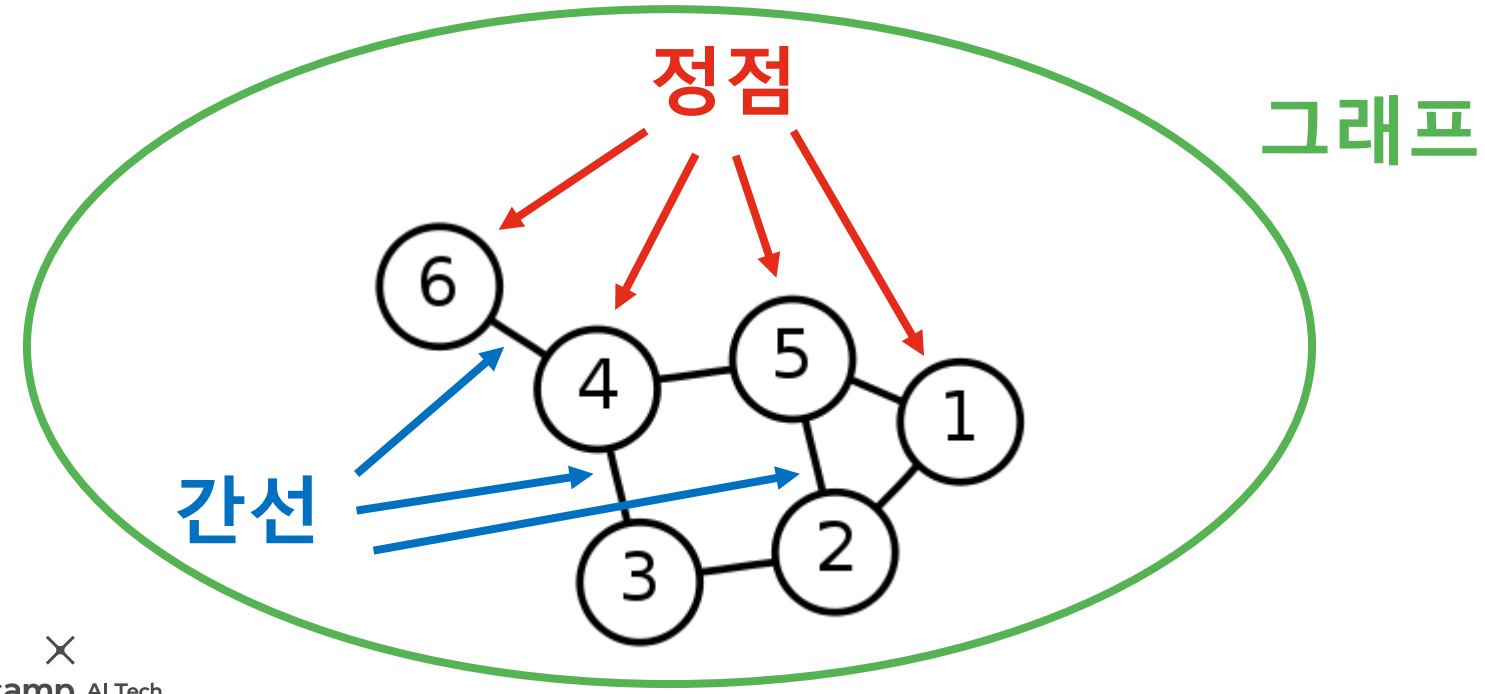
Q. 다음 전자 상거래 구매 내역은 어떤 유형의 그래프일까요?

A. 방향성이 없고, 가중치가 있는 이종 그래프입니다

3.2 그래프 관련 필수 기초 개념

그래프(Graph)는 정점 집합과 간선 집합으로 이루어진 수학적 구조입니다

보통 정점들의 집합을 $\boldsymbol{V}$, 간선들의 집합을 $\boldsymbol{E}$, 그래프를 $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$로 적습니다



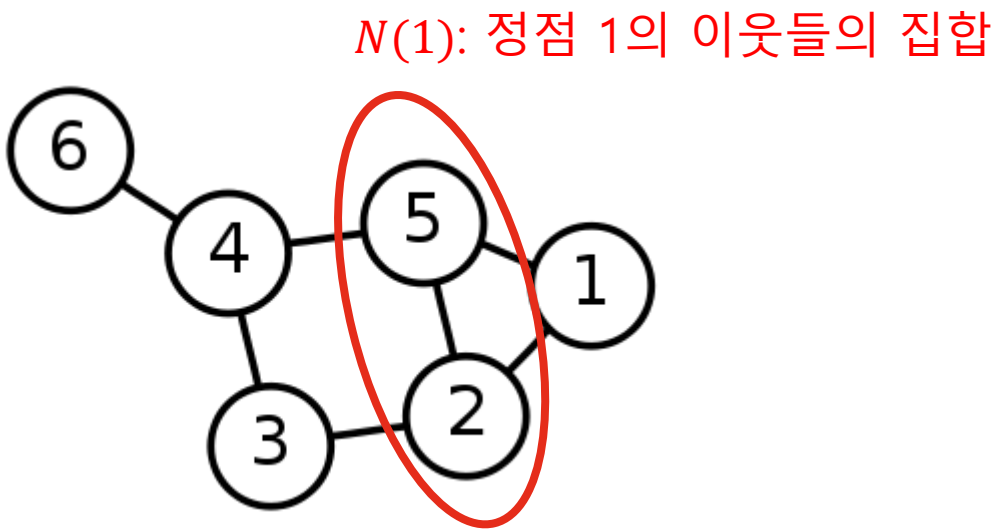
3.2 그래프 관련 필수 기초 개념

정점의 이웃(Neighbor)은 그 정점과 연결된 다른 정점을 의미합니다

정점 v의 이웃들의 집합을 보통 $\boldsymbol{N(v)}$ 혹은 $\boldsymbol{N_v}$로 적습니다

예시:

$N(1) = \{2,5\}$
$N(2) = \{1,3,5\}$
$N(3) = \{2,4\}$
$N(4) = \{3,5,6\}$
$N(5) = \{1,2,4\}$
$N(6) = \{4\}$



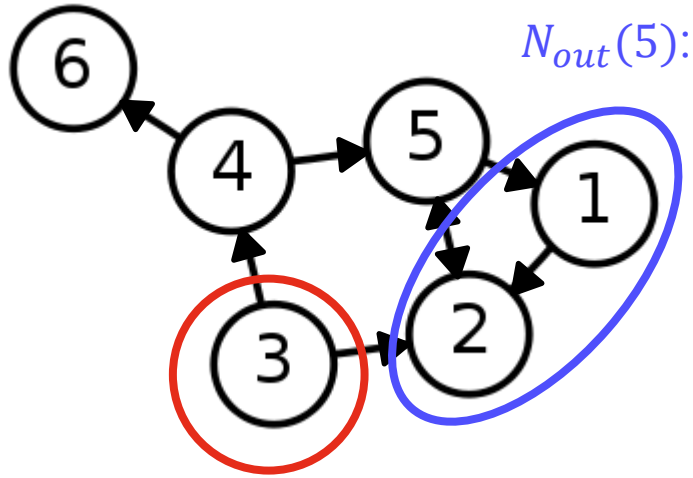
3.3 그래프 관련 필수 기초 개념

방향성이 있는 그래프에서는 나가는 이웃과 들어오는 이웃을 구분합니다

정점 v에서 간선이 **나가는 이웃(Out-Neighbor)**의 집합을 보통 $N_{out}(v)$로 적습니다
정점 v로 간선이 **들어오는 이웃(In-Neighbor)**의 집합을 보통 $N_{in}(v)$로 적습니다

예시:

$N_{in}(1) = \{5\},\ N_{out}(1) = \{2\}$
$N_{in}(2) = \{1,3,5\},\ N_{out}(1) = \{5\}$
$N_{in}(3) = \{\},\ N_{out}(3) = \{2,4\}$
$N_{in}(4) = \{3\},\ N_{out}(4) = \{5,6\}$
$N_{in}(5) = \{2,4\},\ N_{out}(5) = \{1,2\}$
$N_{in}(6) = \{4\},\ N_{out}(6) = \{\}$



$N_{out}(5)$: 정점 5의 나가는 이웃들의 집합

$N_{in}(4)$: 정점 4의 들어오는 이웃들의 집합

4. (실습) 그래프의 표현 및 저장

4.1 파이썬 라이브러리 NetworkX 소개

4.2 그래프의 표현 및 저장

4.1 파이썬 라이브러리 NetworkX 소개

본 수업에서는 그래프를 다루기 위해 파이썬 라이브러리인 NetworkX를 사용합니다.

NetworkX를 이용하여, 그래프를 생성, 변경, 시각화할 수 있습니다
그래프의 구조와 변화 역시 살펴볼 수 있습니다

자세한 정보는 아래 링크에서 찾을 수 있습니다.
https://networkx.org/documentation/stable/index.html

본 수업에서는 사용하지 않지만, Snap.py 라는 라이브러리도 많이 사용됩니다
https://snap.stanford.edu/snappy/

4.1 파이썬 라이브러리 NetworkX 소개

실습에 필요한 라이브러리를 불러옵니다

```
[ ]  # 실습에 필요한 library를 임포트합니다.
     import networkx as nx                          # NetworkX
     import numpy as np                             # 선형대수를 위한 라이브러리
     import matplotlib.pyplot as plt                # 그림을 그리기 위한 라이브러리
```

4.1 파이썬 라이브러리 NetworkX 소개

그래프를 초기화합니다

```
[ ]  print("###### Graph Init ######")
     G= nx.Graph()                                  # 방향성이 없는 그래프 초기화
     DiGraph = nx.DiGraph()                         # 방향성이 있는 그래프 초기화
```

```
###### Graph Init ######
```

4.1 파이썬 라이브러리 NetworkX 소개

정점을 추가하고, 정점의 수를 세고, 목록을 반환합니다

```
[ ]  print("###### Add Node to Graph ######")
     print("# Add node 1")
     G.add_node(1)                                          # 정점 1 추가
     print("Num of nodes in G : " + str(G.number_of_nodes()))   # 정점의 수 반환
     print("Graph : " + str(G.nodes)+ "\n")                 # 정점의 목록 반환
```

```
###### Add Node to Graph ######
# Add node 1
Num of nodes in G : 1
Graph : [1]
```

4.1 파이썬 라이브러리 NetworkX 소개

더 많은 정점을 추가합니다

```
[ ]  print("# Add vertex 2 ~ 10")                                    # 정점 2 ~ 10 추가
     for i in range (1, 11):
         G.add_node(i)
     print("Num of nodes in G : " + str(G.number_of_nodes()))
     print("Graph : " + str(G.nodes) + "\n")
```

```
# Add vertex 2 ~ 10
Num of nodes in G : 10
Graph : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

4.1 파이썬 라이브러리 NetworkX 소개

간선을 추가하고, 목록을 반환합니다

```
[ ]  print("###### Add Edge to Graph ######")
     print("#Add edge (1, 2)")
     G.add_edge(1, 2)                          # 정점 1과 2 사이에 간선 추가
     print("Graph : " + str(G.edges) + "\n")   # 간선의 목록을 반환
```

```
###### Add Edge to Graph ######
#Add edge (1, 2)
Graph : [(1, 2)]
```

4.1 파이썬 라이브러리 NetworkX 소개

더 많은 간선을 추가합니다

```
[ ]  print("#Add edge (1, i) for i = 2 ~ 10")          # 정점 1과 다른 정점 사이의 간선 추가
     for i in range (2, 11):
         G.add_edge(1, i)
     print("Graph : " + str(G.edges) + "\n")
```

```
#Add edge (1, i) for i = 2 ~ 10
Graph : [(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10)]
```

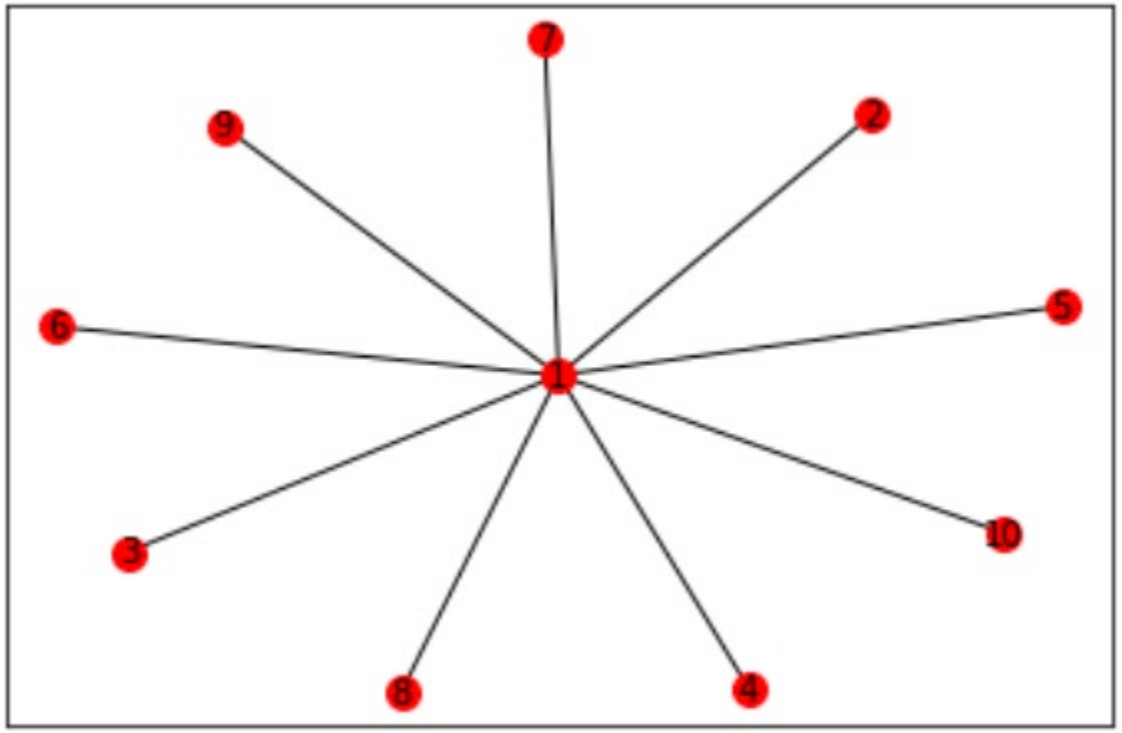
4.1 파이썬 라이브러리 NetworkX 소개

만들어진 그래프를 시각화합니다

```
[ ]  # 그래프를 시각화
     # 정점의 위치 결정
     pos = nx.spring_layout(G)
     # 정점의 색과 크기를 지정하여 출력
     im = nx.draw_networkx_nodes(G, pos, node_color="red", node_size=100)
     # 간선 출력
     nx.draw_networkx_edges(G, pos)
     # 각 정점이 라벨을 출력
     nx.draw_networkx_labels(G, pos, font_size=10, font_color="black")
     plt.show()
```

4.1 파이썬 라이브러리 NetworkX 소개

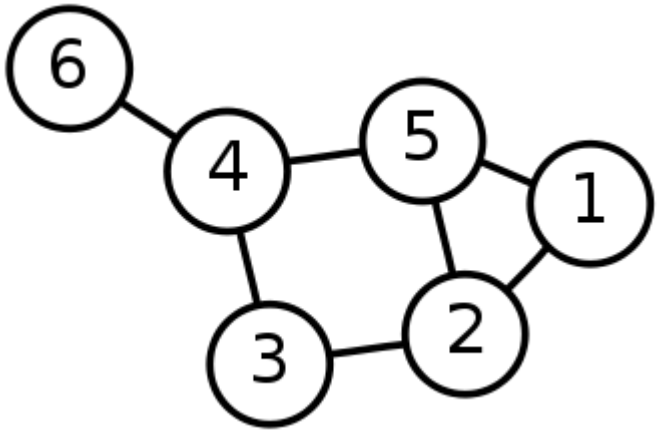
만들어진 그래프를 시각화합니다



4.2 그래프의 표현 및 저장

간선 리스트(Edge List): 그래프를 간선들의 리스트로 저장

각 간선은 해당 간선이 연결하는 두 정점들의 순서쌍(Pair)으로 저장됩니다

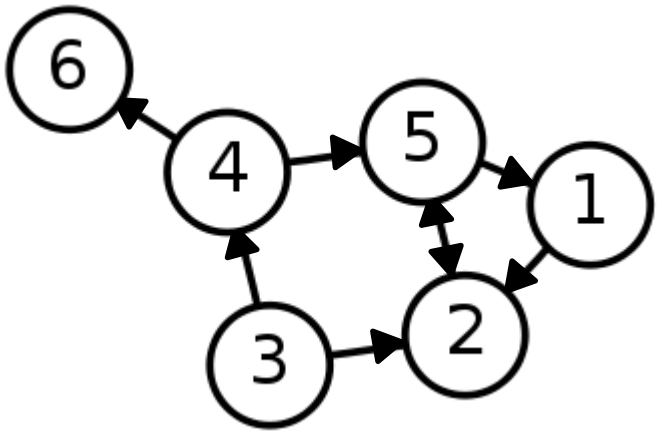


(1,2)
(1,5)
(2,3)
(2,5)
(3,4)
(4,5)
(4,6)

4.2 그래프의 표현 및 저장

간선 리스트(Edge List): 그래프를 간선들의 리스트로 저장

방향성이 있는 경우에는 (출발점, 도착점) 순서로 저장됩니다

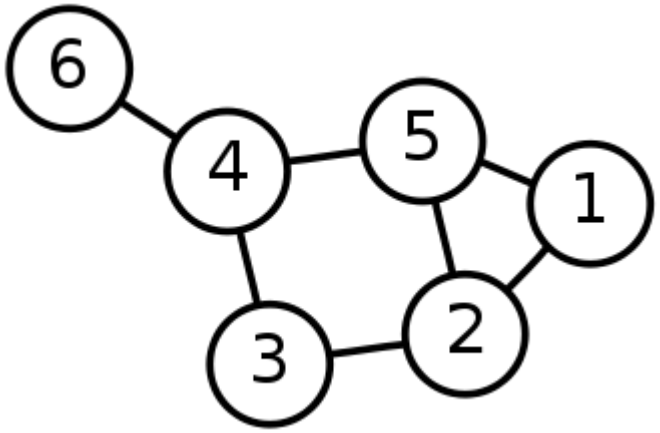


(1,2)
(2,5)
(3,2)
(3,4)
(4,5)
(4,6)
(5,1)
(5,2)

4.2 그래프의 표현 및 저장

인접 리스트(Adjacent list) – 방향성이 없는 경우:

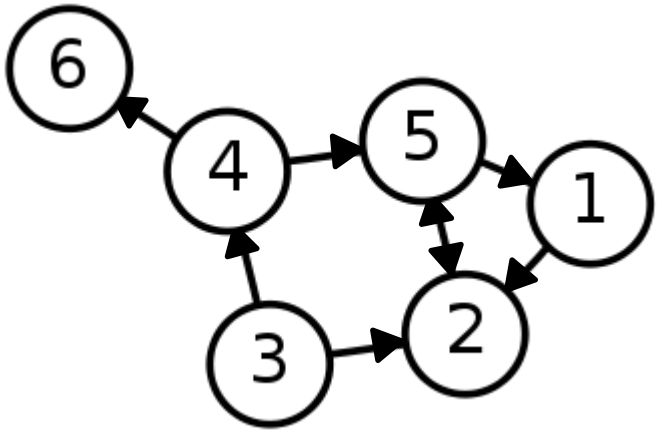
각 정점의 이웃들을 리스트로 저장



1: 2, 5
2: 1, 3, 5
3: 2, 4
4: 3, 5, 6
5: 1, 2, 4
6: 4

인접 리스트(Adjacent list) – 방향성이 있는 경우

각 정점의 나가는 이웃들과 들어오는 이웃들을 각각 리스트로 저장

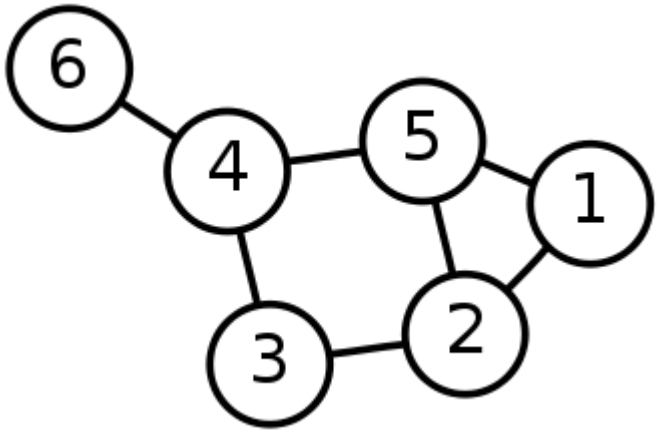


나가는 이웃들	들어오는 이웃들
1: 2	1: 5
2: 5	2: 1, 3, 5
3: 2, 4	3:
4: 5, 6	4: 3
5: 1, 2	5: 2, 4
6:	6: 4

4.2 그래프의 표현 및 저장

인접 행렬(Adjacency Matrix) – 방향성이 없는 경우

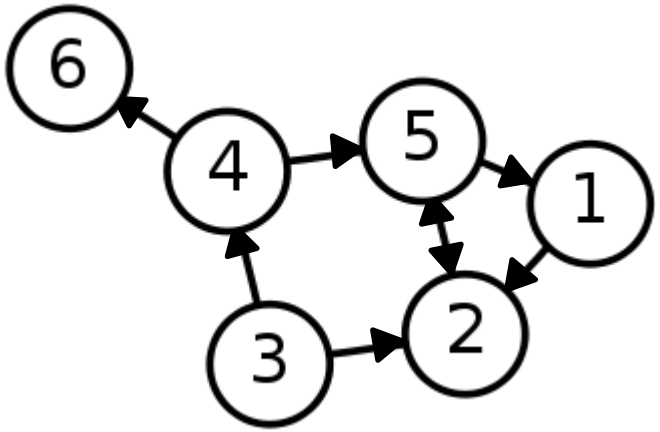
정점 수 × 정점 수 크기의 행렬
정점 i와 j 사이에 간선이 **있는** 경우, 행렬의 i행 j열 (그리고 j행 i열) 원소가 **1**
정점 i와 j 사이에 간선이 **없는** 경우, 행렬의 i행 j열 (그리고 j행 i열) 원소가 **0**



	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

인접 행렬(Adjacency Matrix) – 방향성이 있는 경우

정점 수 × 정점 수 크기의 행렬
정점 i에서 j로의 간선이 **있는** 경우, 행렬의 i행 j열 원소가 **1**
정점 i에서 j로의 간선이 **없는** 경우, 행렬의 i행 j열 원소가 **0**



	1	2	3	4	5	6
1	0	1	0	0	0	0
2	0	0	0	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	0	0	0
6	0	0	0	0	0	0

4.2 그래프의 표현 및 저장

NetworkX를 이용하여 그래프를 표현하고 저장하기

```
[ ]  # 그래프를 인접 리스트로 저장
     nx.to_dict_of_lists(G)
```

```
[ ]  # 그래프를 간선 리스트로 저장
     nx.to_edgelist(G)
```

```
[ ]  # 그래프를 인접 행렬(일반 행렬)로 저장
     nx.to_numpy_array(G)
```

```
[ ]  # 그래프를 인접 행렬(희소 행렬)로 저장
     nx.to_scipy_sparse_matrix(G)
```

일반 행렬은 전체 원소를 저장하므로
정점 수의 제곱에 비례하는 저장 공간을 사용

희소 행렬은 0이 아닌 원소만을 저장하므로
간선의 수에 비례하는 저장 공간을 사용

예시) 정점의 수가 10만, 간선의 수가 100만이라면

정점의 수의 제곱 (100억) >> 간선의 수 (100만)

1강 정리

1. 그래프란 무엇이고 왜 중요할까?

- 그래프는 정점 집합과 간선 집합으로 이루어진 수학적 구조입니다
- 그래프는 복잡계를 표현하고 분석하기 위한 언어입니다

2. 그래프 관련 인공지능 문제

- 정점 분류, 연결 예측, 추천, 군집 분석, 랭킹, 정보 검색, 정보 전파, 바이럴 마케팅 등

3. 그래프 관련 필수 기초 개념

- 방향성이 있는/없는 그래프, 가중치가 있는/없는 그래프, 동종/이종 그래프
- 나가는/들어가는 이웃

4. (실습) 그래프의 표현 및 저장

- 파이썬 라이브러리 NetworkX
- 간선 리스트, 인접 리스트, 인접 행렬