WordNet, Outcast 탐지 및 SCA/SAP 찾기

WordNet 구현 방법 및 이에 필요한 그래프 탐색, cycle 탐지, symbol table 활용법 이해

01. 퀴즈 풀이 & 예습 내용 복습 (이번 주 #1~3차 답안 공개)
02. WordNet은 무엇이며, 어떻게 사용되는가?
03. Outcast 탐지 방법 개요
04. SCA & SAP 찾기 개요
05. WordNet과 outcast 탐지 구현
06. 구현할 API 및 유의사항
07. 실습: WordNet과 outcast 탐지 구현

이번 시간 수업 자료에 첨부된 코드와 파일을 미리 열어 두세요.
이론 수업 중에 관찰해 볼 예정입니다.

WordNet: 단어 간 의미 관계 나타내는 Digraph

- 정점(**synset**): 유사어(**syn**onym)의 집합(**set**)
- v → w 간선: v is a w 관계 (hyponym → hypernym)
  - "apple" is an "edible fruit"
  - "banana" is an "edible fruit"

motion
movement
move

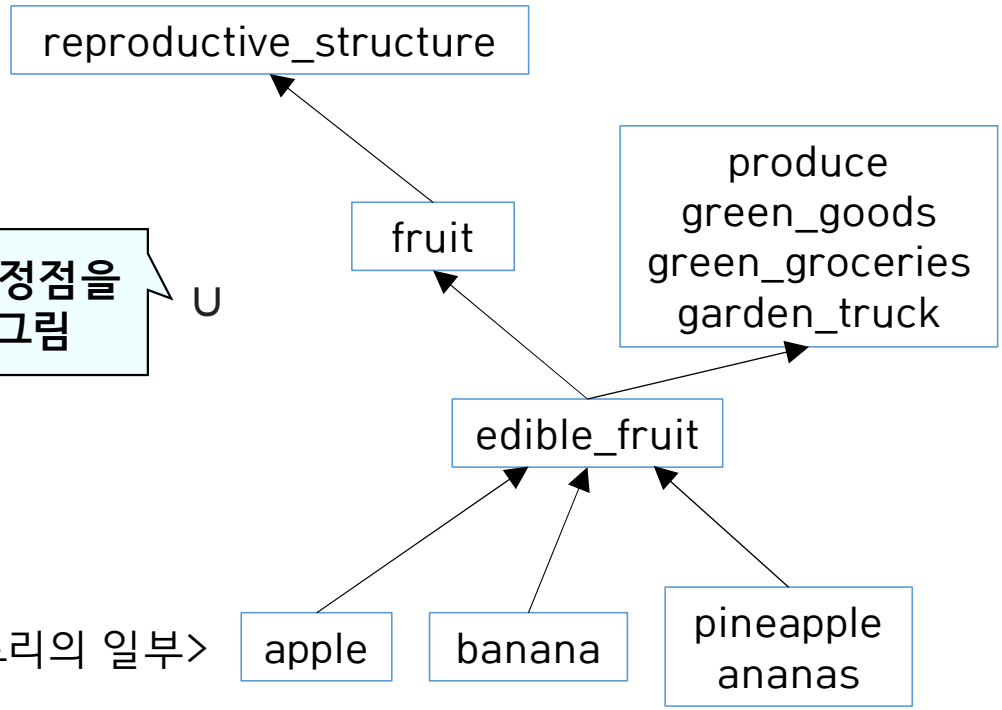
pineapple
ananas

[Q] "love" 와 "passion"은 WordNet에서 같은 정점일까 다른 정점일까?

[Q] "lion"과 "cat(고양이과)" 간 간선의 방향은? lion → cat

v → w
v is a w
v ∈ w

위쪽 정점이 아래쪽 정점을 포함하는 형태로 그림



<WordNet 트리의 일부>

WordNet은 어디에 사용하나?

- (인공지능) **문장 의미** 자동 **분석**, **논리적 추론**에 사용
- 그림: 퀴즈쇼에서 사람 챔피언에게 이긴 IBM의 Watson 컴퓨터가 WordNet 활용
- 예1: "사자의 뒷발에는 몇 개의 발톱이 있는가?"에 대한 문제가 나왔을 때, "사자는 고양이과"이고, "고양이과는 뒷발에 4개의 발톱이 있다"는 사실로부터 4개의 발톱이 있음을 추론
- 예2: "horse zebra cat bear table" 중 가장 관련 적은 것은?"에 대한 문제가 나왔을 때, WordNet에서 홀로 가장 멀리 떨어진 (즉 다른 단어와 의미 상 거리가 가장 먼) "table"을 답함

이번 시간에 만들어볼 기능: WordNet을 그래프 객체로 만든 후 이러한 질문에 답하도록 함

[Q] 빈 칸에 들어갈 단어는?

WordNet의 다른 사용 예

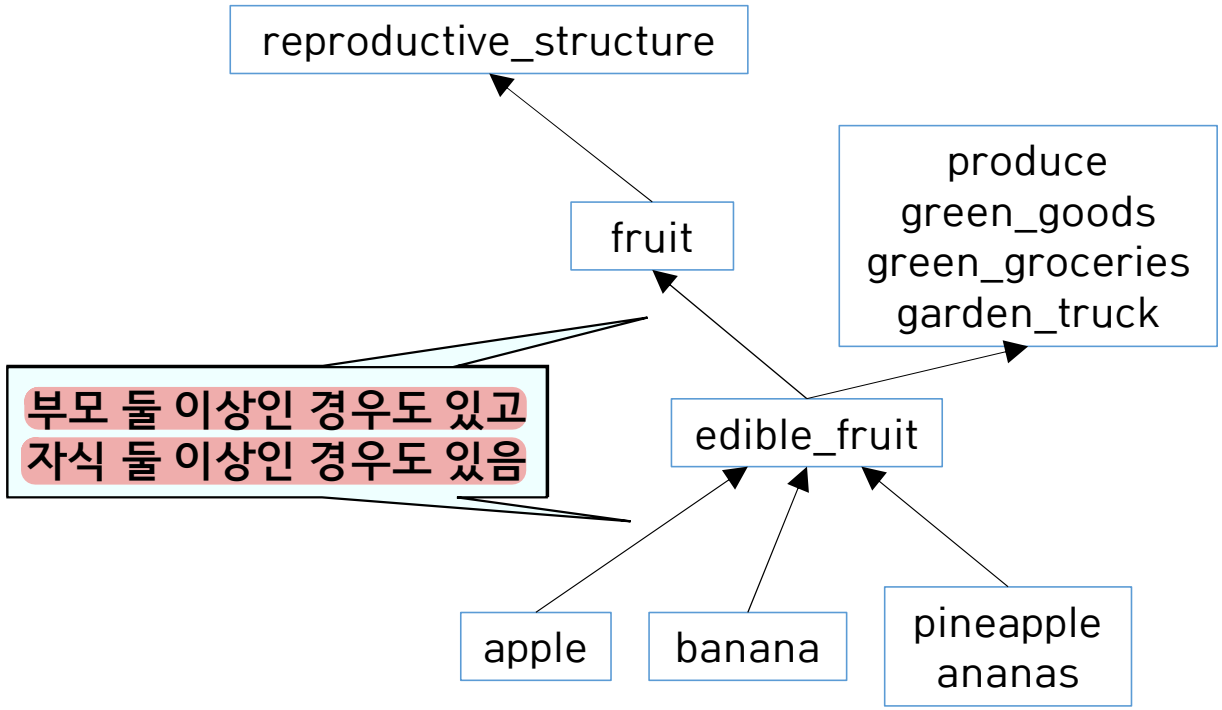
- George W. Bush와 John F. Kennedy는 어떤 관련이 있나?
  - 둘 다 미국의 대통령
- George W. Bush와 chimpanzee는 어떤 관련이 있나?
  - 둘 다 영장류(primates)
- {George W. Bush, John F. Kennedy}와 {George W. Bush, chimpanzee} 중 서로 더 관련 있는 쌍은?
  - {George W. Bush, John F. Kennedy}
- George W. Bush(대통령)와 Eric Arthur Blair (George Orwell, 작가)는 서로 관련이 있는가?
  - 둘 다 유명한 communicator (사람들에게 자신의 생각을 전달함으로써 영향을 끼침)

- 정리: WordNet은 지금까지 본 **다양한 추론 할 수 있도록 만든 자료구조**

WordNet에 대해 기억할 추가 특성 (1)

Cycle 없음 (DAG, Directed Acyclic Graph)

Root는 하나임 ("entity")



<WordNet 트리의 일부>

[Q] WordNet에는 왜 cycle이 없을까?

Cycle이 있어도 포함관계에서 모순

~~작은집합 ∈ 큰집합 & 큰집합 ∈ 작은집합~~

[Q] 빈 칸에 들어갈 단어는?
Every wordnet word is a (entity).
∈

Root of WordNet: "entity"

root : WordNet에서 가장 큰 개념.

- 첨부파일 "synsets.txt" 38004번째 줄 ("38003,"로 검색)

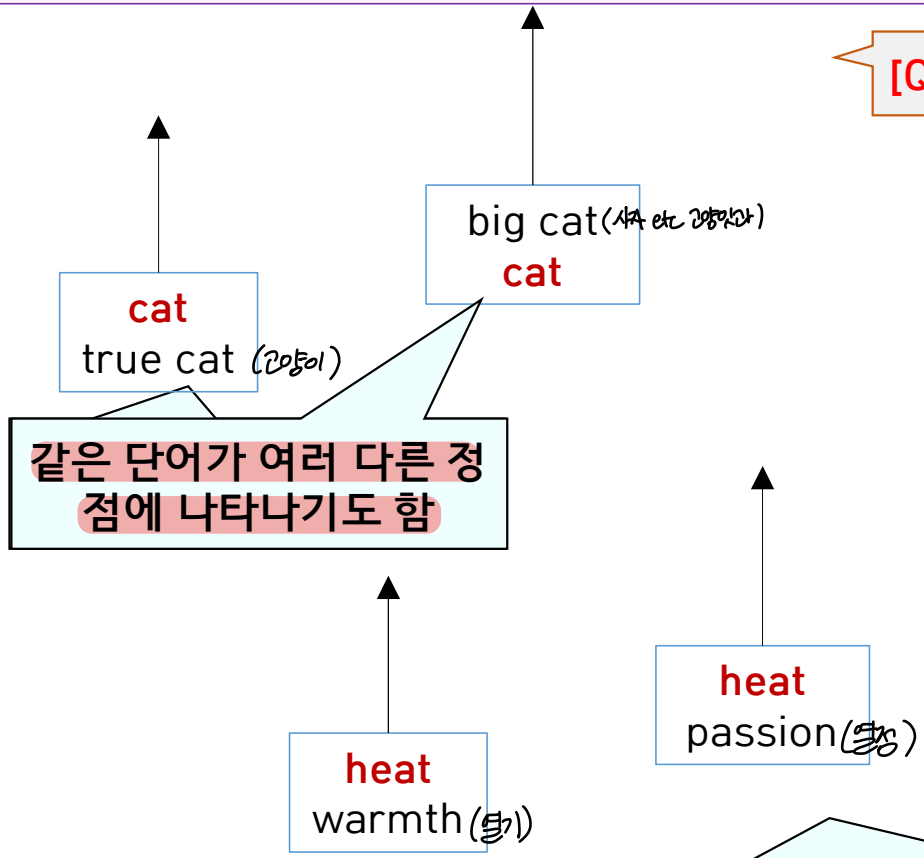
- 38003,**entity**,that which is perceived or known or inferred to have its own distinct existence (living or nonliving)
- "존재하는 무언가"

- WordNet의 자식⊂부모 포함관계 (혹은 "자식 **is a** 부모" 관계에 의해)
- Every WordNet word **is an entity**.

WordNet에 대해 기억할 추가 특성 (2)

[Q] 같은 단어가 여러 다른 정점에 나타나는 이유?

동의어가 있으니까.



[Q] (서로 다른 정점에 있는) 같은 단어라면 부모도 항상 같을까? NO

"horse zebra cat bear table" 중 가장 관련 적은 것은?" 같은 질문 나왔을 때 각 단어의 여러 의미를 다 고려해야 정확히 답할 수 있다는 뜻 (여럿 중 먼저 찾은 하나의 의미만 고려한다면 문제가 의도한 의미가 아닐 수도 있으므로)

[Q] WordNet 관련 아래 문장 각각에 대해 참/거짓을 답하시오.

- (T or F) 한 정점은 여러 단어로 구성될 수 있다. T (동의어끼리 한 묶음)
- (T or F) 같은 단어가 여러 다른 정점에 나타날 수 있다. T
- (T or F) Root는 하나이다. T (entity)
- (T or F) v→w는 v ⊂ w 임을 의미한다. T
- (T or F) 각 정점의 자식은 여럿 있을 수 있지만 부모는 ~~하나이다~~. F (여러개다)

WordNet

WordNet 구현 방법 및 이에 필요한 그래프 탐색, cycle 탐지, symbol table 활용법 이해

01. 퀴즈 풀이 & 예습 내용 복습 (이번 주 #1~3차 답안 공개)

02. WordNet은 무엇이며, 어떻게 사용되는가?

03. Outcast 탐지 방법 개요

04. SCA & SAP 찾기 개요

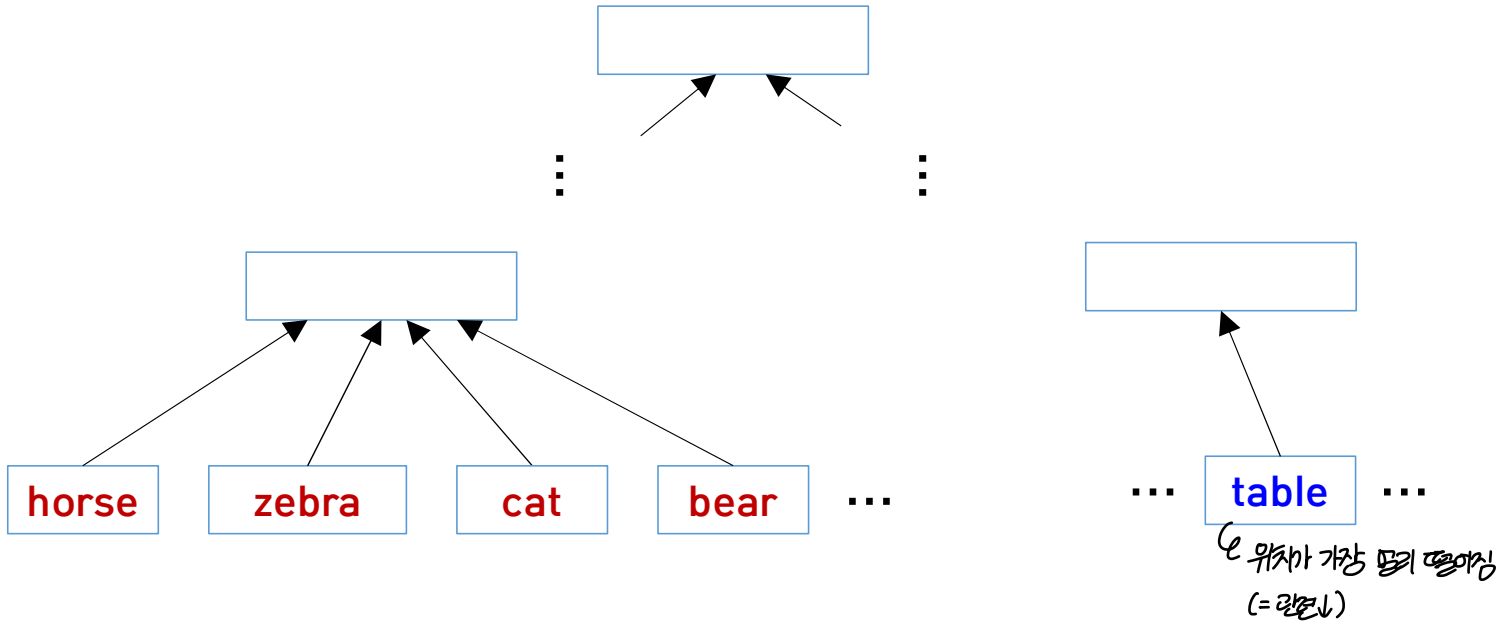
05. WordNet과 outcast 탐지 구현

06. 구현할 API 및 유의사항

07. 실습: WordNet과 outcast 탐지 구현

Outcast: 단어 집합에서 다른 단어와 가장 의미가 다른 단어

- 예: "horse zebra cat bear table" 중 가장 관련 적은 것은?"에 대한 문제에서, WordNet에서 홀로 가장 멀리 떨어진 (즉 다른 단어와 의미 상 거리가 가장 먼) "table"이 답이며 이 집합에서의 outcast



Outcast의 다른 예

- George_W._Bush John_Fitzgerald_Kennedy Eric_Arthur_Blair **chimpanzee**
- coffee, apple_juice, orange_juice, soda, tea, **bed**, milk, water
- mango, peach, banana, strawberry, lemon, pear, apple, lime, blueberry, watermelon, **potato**
- green, white, black, yellow, **fox**, gray, blue, red, pink
- ...

두 단어가 유의어일때 의미관계하게!

Outcast 탐지 방법: WordNet에서 다른 단어와 가장 멀리 떨어진 단어 선정

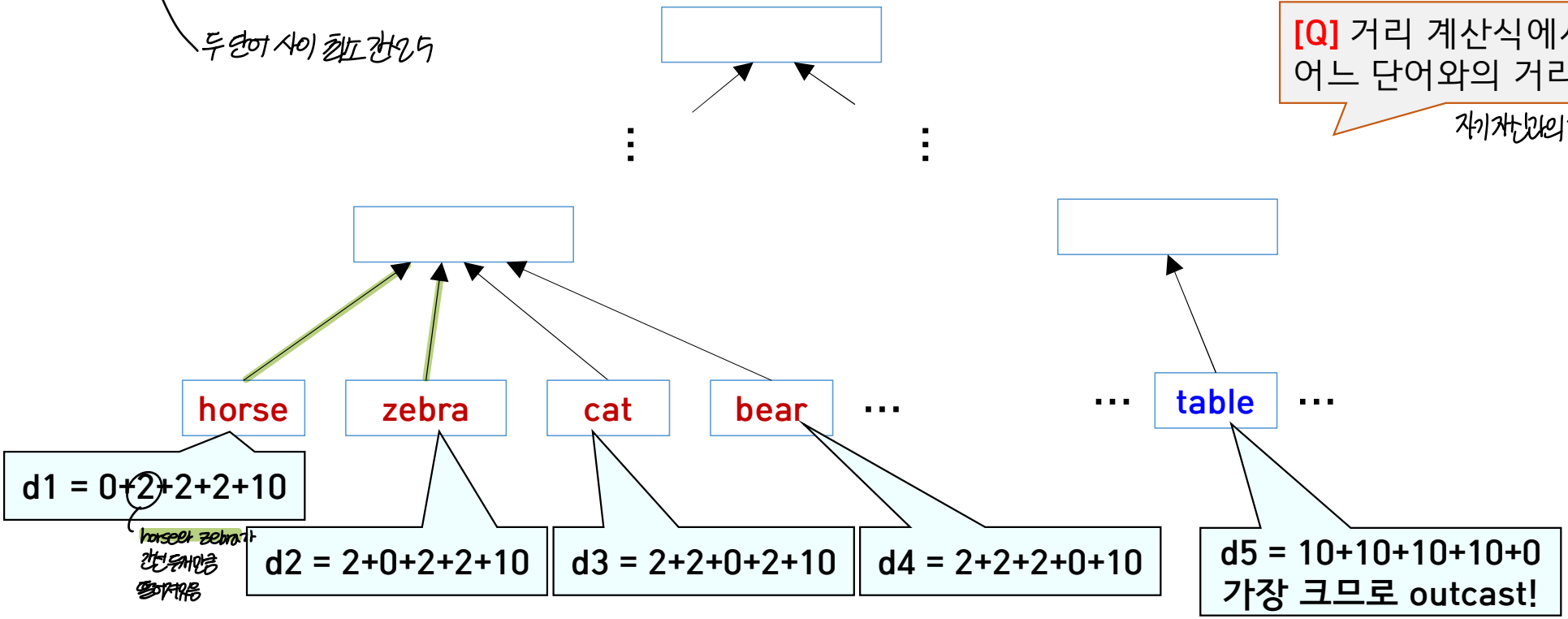
- **단어 집합**이 {w1, w2, … wn}일 때
- wi의 거리 합 di = distance(wi, w1) + distance(wi, w2) + … + distance(wi, wn)
- di가 가장 큰 단어를 outcast로 선정

두 단어 간 거리: 가장 가까운 공통 조상까지의 간선 수

두 단어 사이 의미 관련도

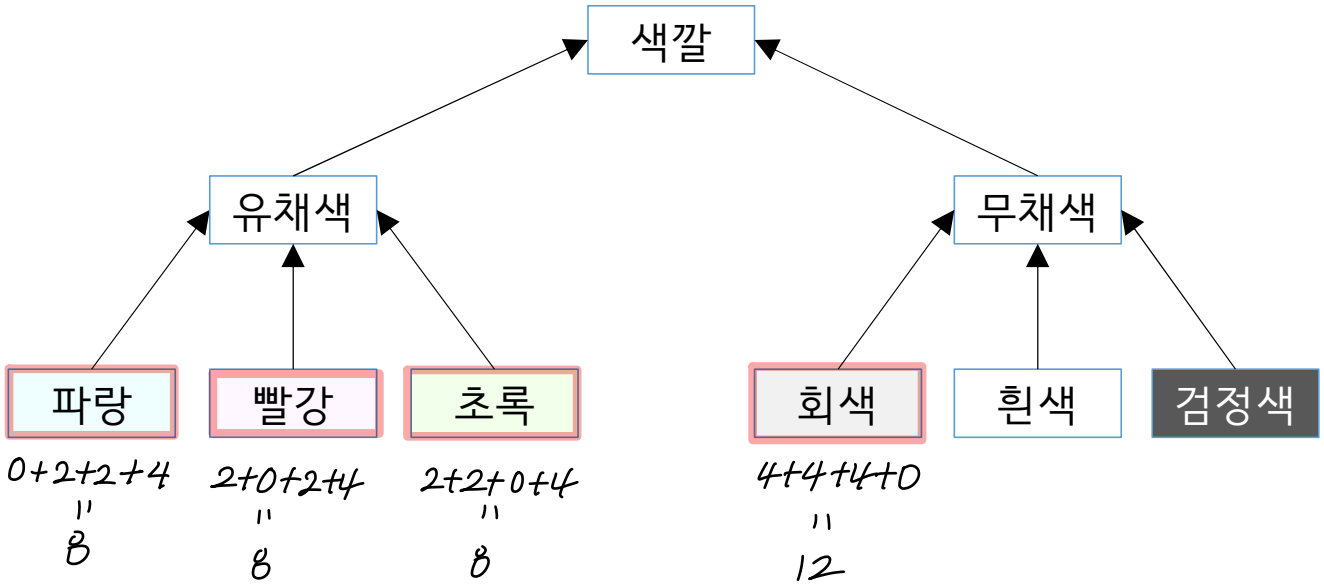
[Q] 거리 계산식에서 0은 어느 단어와의 거리인가?

자기자신과의 거리



[Q] {파랑, 회색, 빨강, 초록} 중 outcast를 찾아야 한다.
(1) 아래 그래프에서 각 단어의 거리 합을 구하시오.
(2) (1)의 결과에 따라 outcast를 선정하시오. 힌트

두 단어 간 거리: 가장 가까운 공통 조상까지의 간선 수

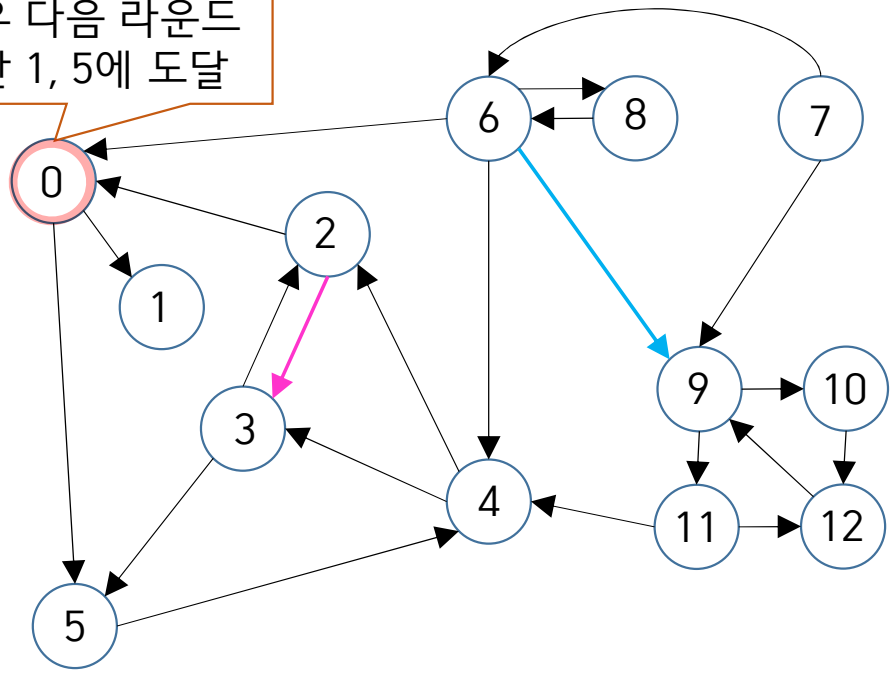


WordNet의 표현(저장) 방식

Adjacency-list: 각 점에서 **나가는 정점**의 목록 저장

왼쪽 그래프를 adjacency-list 방식으로 저장한 결과

정점 0에서 그래프 탐색 시작한 경우 다음 라운드에는 인접한 1, 5에 도달



adj[]

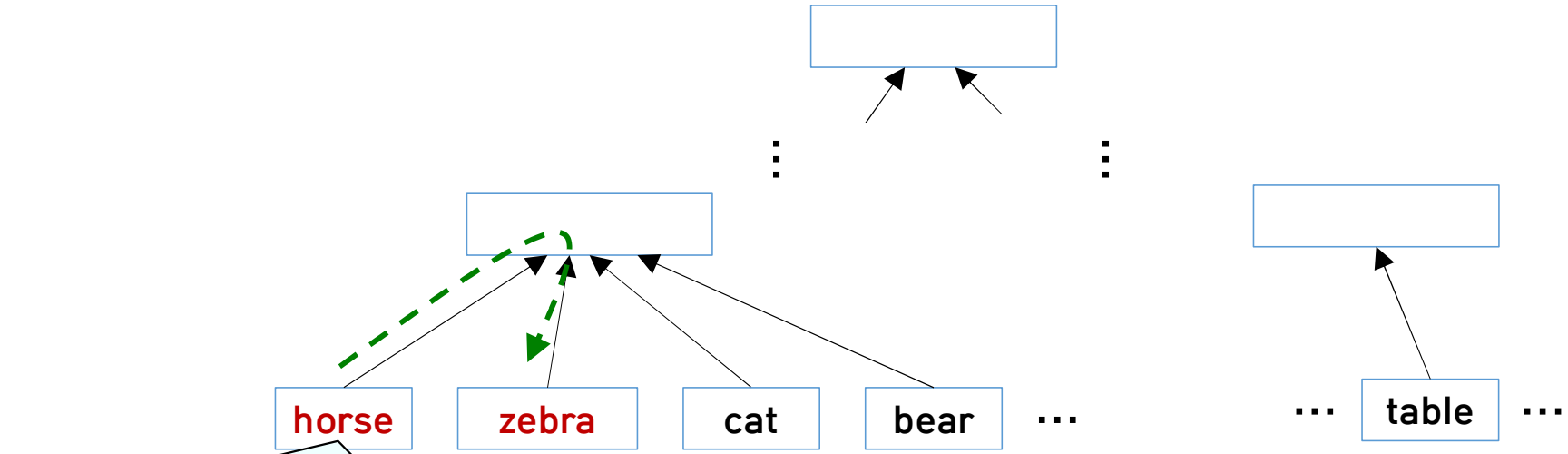
0	[1,5]
1	[]
2	[0,3]
3	[2,5]
4	[2,3]
5	[4]
6	[0,4,8,9]
7	[6,9]
8	[6]
9	[10,11]
10	[12]
11	[4,12]
12	[9]

각 정점에서 나가는 정점의 목록 저장: 각 간선은 **한 번씩**만 목록에 저장됨

왜 adjacency-matrix나 edge-list 방식 아닌 adjacency-list 방식 사용하나?
(1) 각 정점에 인접한 정점을 iterate하는 기능 많이 필요
(2) 많은 실제 문제의 경우 그래프의 정점은 매우 많으나 sparse하므로

distance(wa, wb) 계산 방법:
[Q] 단어1 wa에서 출발해 단어2 wb까지 간선 방향따라 탐색해 갈 수는 없음 (혹은 wb에서 출발해 wa까지 탐색해 갈 수는 없음) Why?

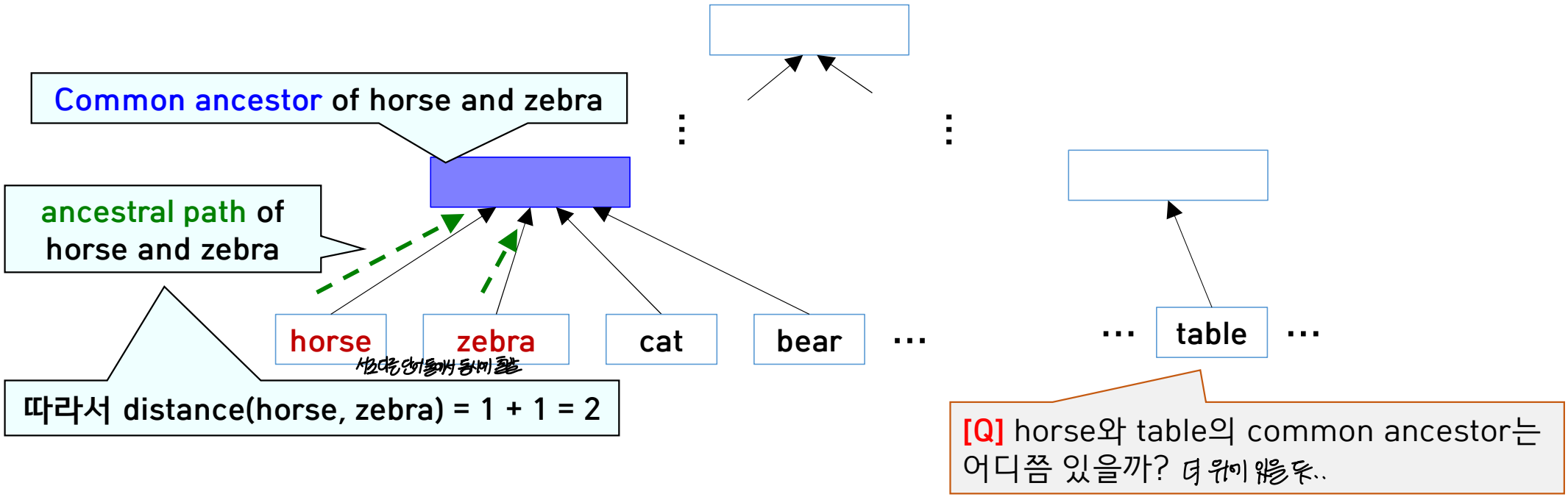
- distance(wa, wb): WordNet 상에서 두 단어 wa와 wb 간 최소 간선 수
- 아래 예제에서 distance(horse, zebra) = 2, distance(horse, table) = 10



distance(wa, wb) 계산 방법: 같은 정점에서 만날 때까지 wa, wb 각각에서 간선 따라 올라간 후, 거쳐온 간선 수 더함

틀린 방법

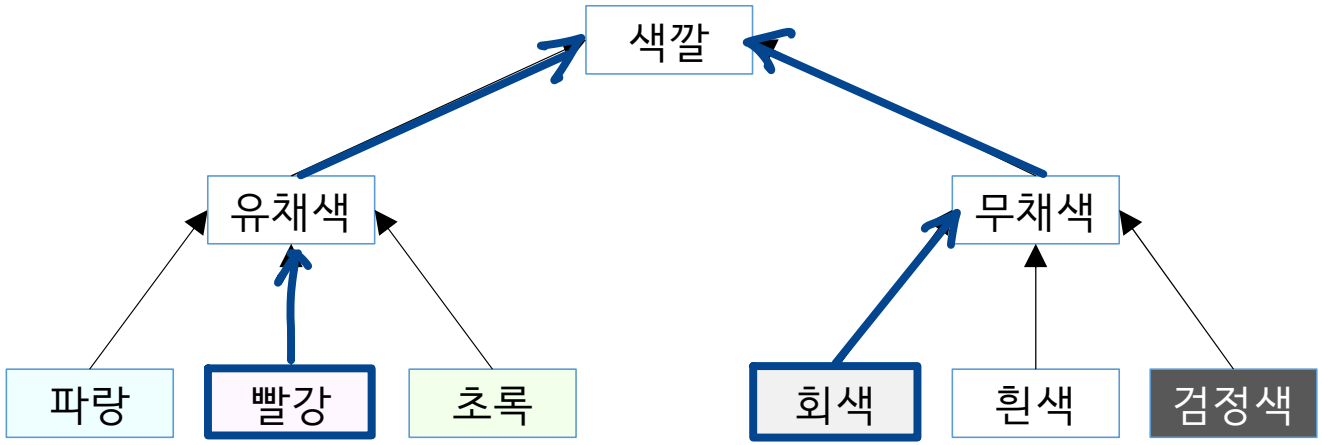
- wa와 wb에서 동시 출발해 만날 때까지 전진한 후, 거쳐온 간선 수 더함
- **common ancestor(공통 조상)**: 두 탐색이 만난 정점
- **ancestral path(공통 조상까지 경로)**: wa, wb에서 공통 조상까지 거쳐온 경로



[Q] 앞에서 본 방법에 따라 distance(빨강, 회색)을 구하는 과정을 보이시오.

두 단어 간 거리: 가장 가까운 공통 조상까지의 간선 수

4



2 + 2 = 4

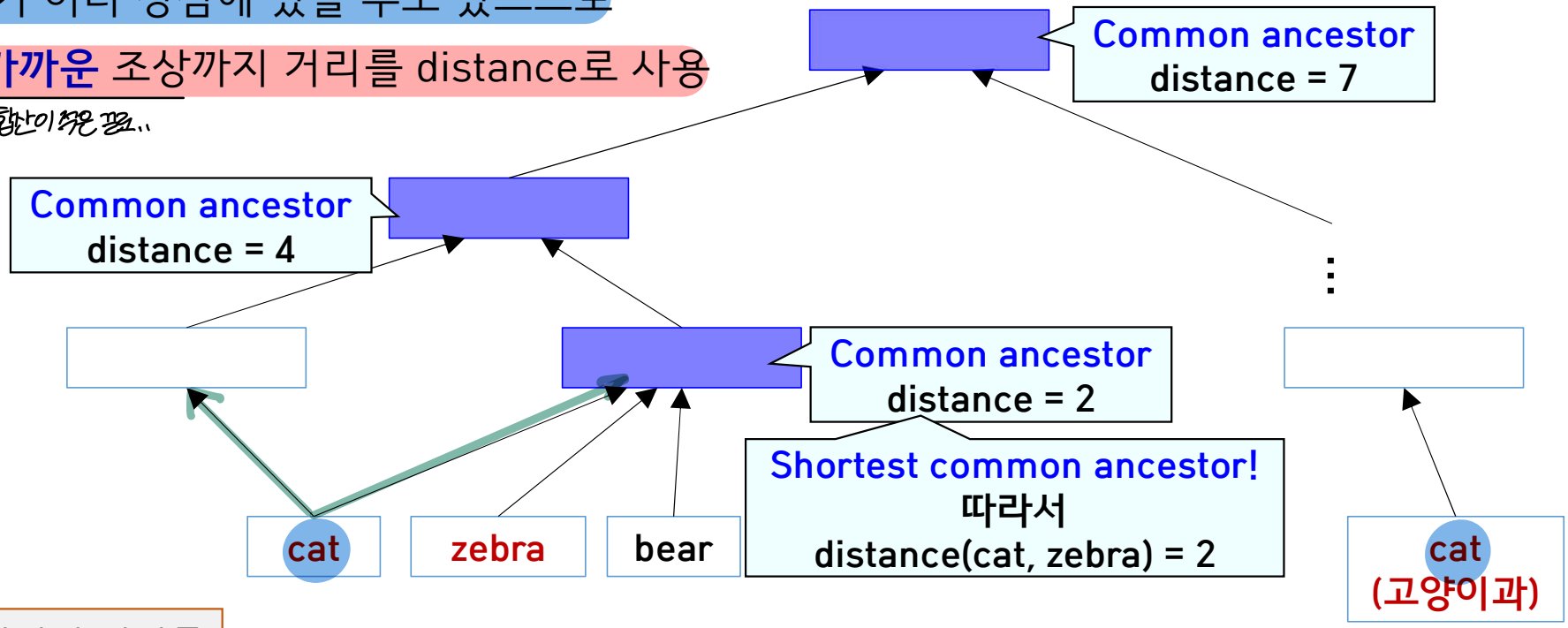
distance(wa, wb) : shortest common ancestor까지 거리

- 같은 단어 wb, wb에 대해
- **common ancestor(공통 조상)**과 ancestral path(조상까지 경로)는 둘 이상일 수 있음

① 부모 둘 이상일 수 있고

② 같은 단어가 여러 정점에 있을 수도 있으므로

- 이 중 가장 **가까운** 조상까지 거리를 distance로 사용

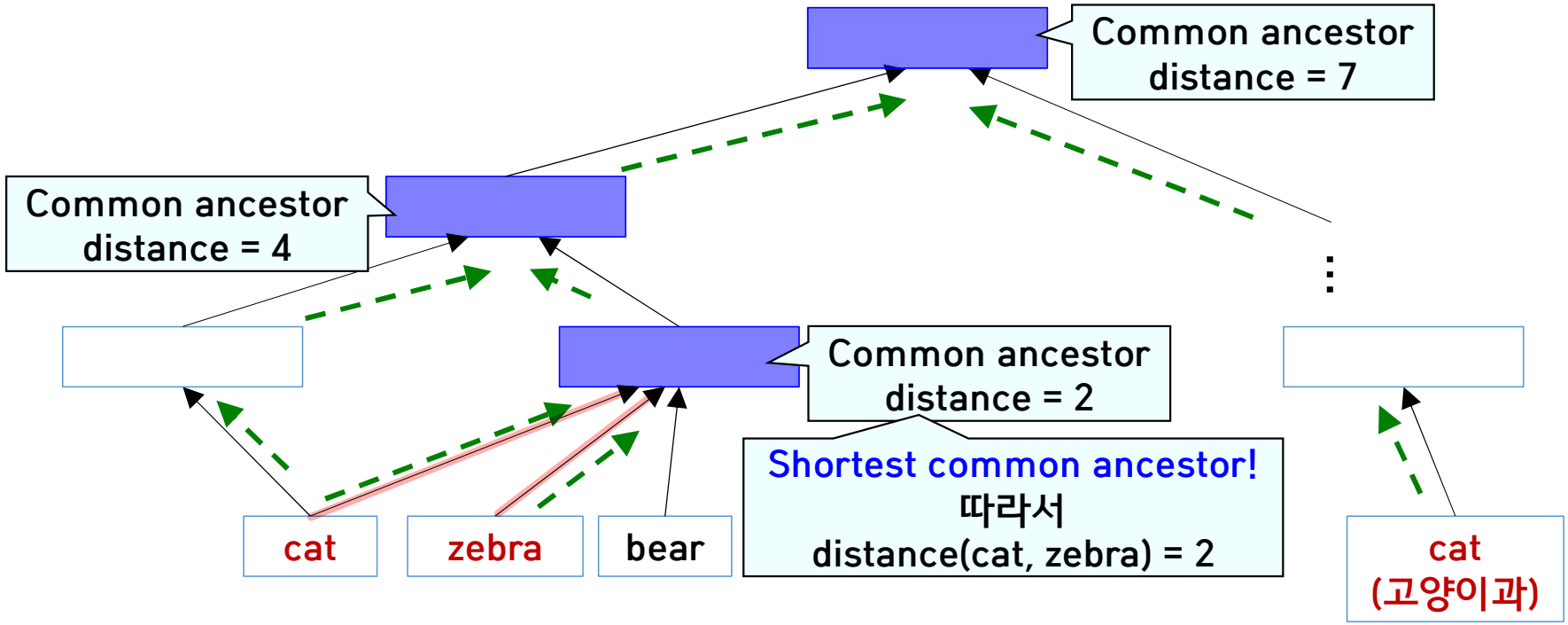


[Q] 왜 가장 가까운 조상까지 거리를 distance로 사용할까?

[방법&용어 정리] SCA, SAP: 가장 가까운 공통 조상과 거기까지 경로
distance(wa,wb): wa,wb에서 SCA까지 거리 or SAP의 길이

- wa와 wb에서 동시 출발해 만날 때까지 전진한 후, 거쳐온 간선 수 더함 (여러 가능성 탐색)
- SCA (shortest common ancestor, 가장 가까운 공통 조상): 두 탐색이 만나는 가장 가까운 정점
- SAP (shortest ancestral path, SCA까지 경로): wa, wb에서 SCA까지 거쳐온 경로

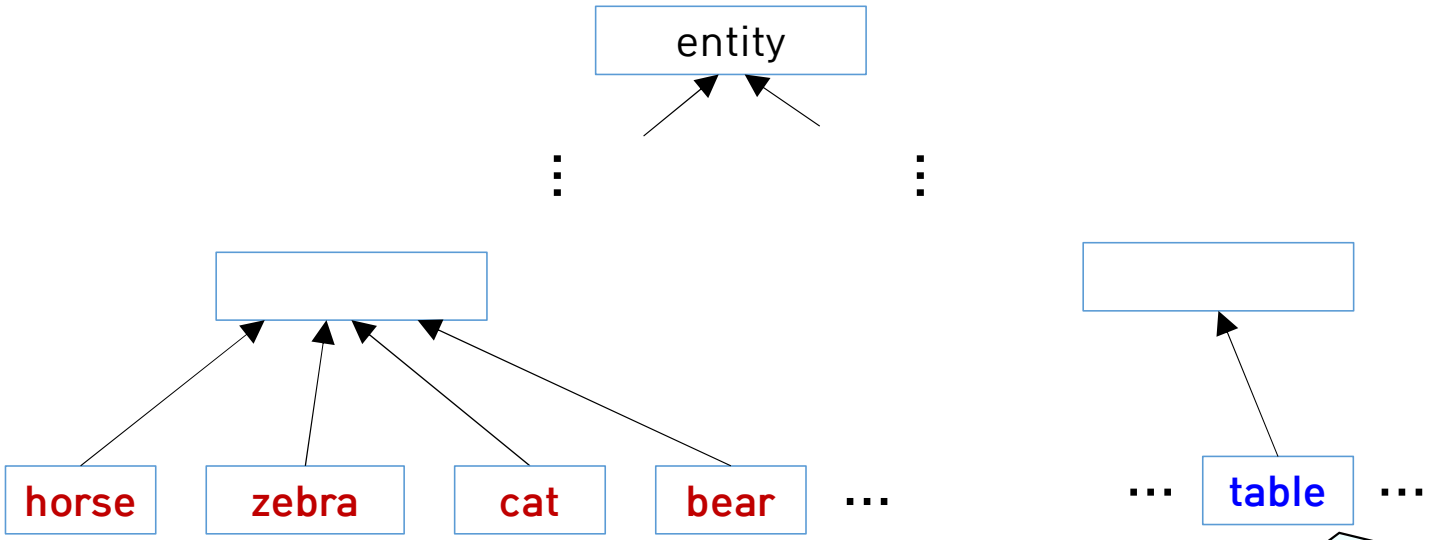
가장 가까운 공통 조상



[Q] WordNet 상 임의의 두 정점 wa, wb 사이에는 항상 공통 조상 존재하나?

공통 조상 없다면 거리 구할 수 없을 것임

반드시 공통 조상이 존재함.



예: table과 bear 사이에 공통 조상이 없다면,
앞에서 본 방법으로 거리도 구할 수 없을 것임

WordNet은 가능하면 항상 동작하도록
여러 특성, 조건 확인해 만들어져 있음

WordNet

WordNet 구현 방법 및 이에 필요한 그래프 탐색, cycle 탐지, symbol table 활용법 이해

01. 퀴즈 풀이 & 예습 내용 복습 (이번 주 #1~3차 답안 공개)
02. WordNet은 무엇이며, 어떻게 사용되는가?
03. Outcast 탐지 방법 개요
04. SCA **(가장 가까운 공통 조상)** & SAP 찾기 개요
05. WordNet과 outcast 탐지 구현
06. 구현할 API 및 유의사항
07. 실습: WordNet과 outcast 탐지 구현

의미가장 먼 단어

Outcast 찾는 과정 정리

예: {“horse”, “zebra”, “cat”, “bear”, “table”}

outcast({w1, w2, …, wn}):
집합의 각 단어 wi에 대해
거리 합 di = distance(wi, w1) + distance(wi, w2) + … + distance(wi, wn) 계산
di 가장 큰 단어를 outcast로 선정

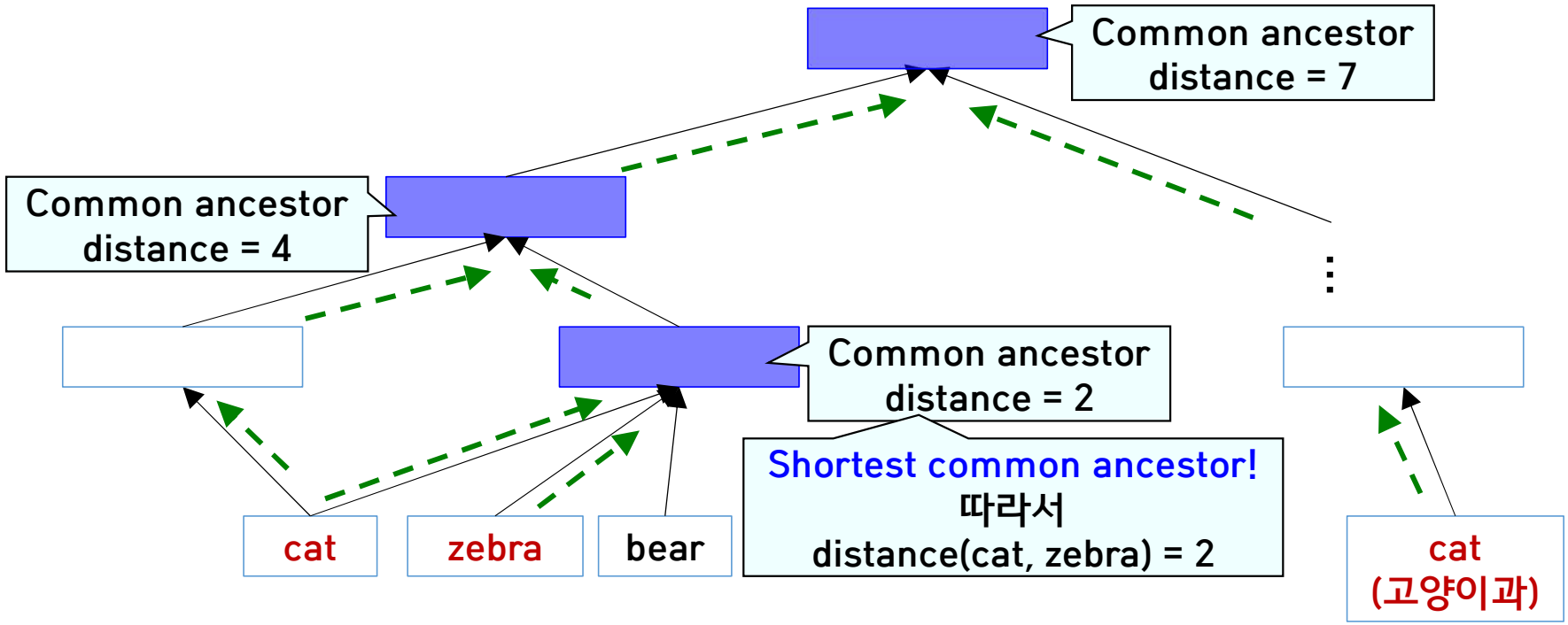
예: di = 16, 16, 16, 16, 50 이므로
outcast는 di 가장 큰 “table”

distance(wa,wb):
wa, wb에서 출발해 가장 가까운 공통 조상(SCA)과 경로(SAP) 찾아 경로 길이 반환

예: “horse”, “cat” 간 SCA는
“animal” 이고 경로 길이는 2

distance(wa,wb):
wa, wb에서 출발해 가장 가까운 공통 조상(SCA)과 경로(SAP) 찾아 경로 길이 반환

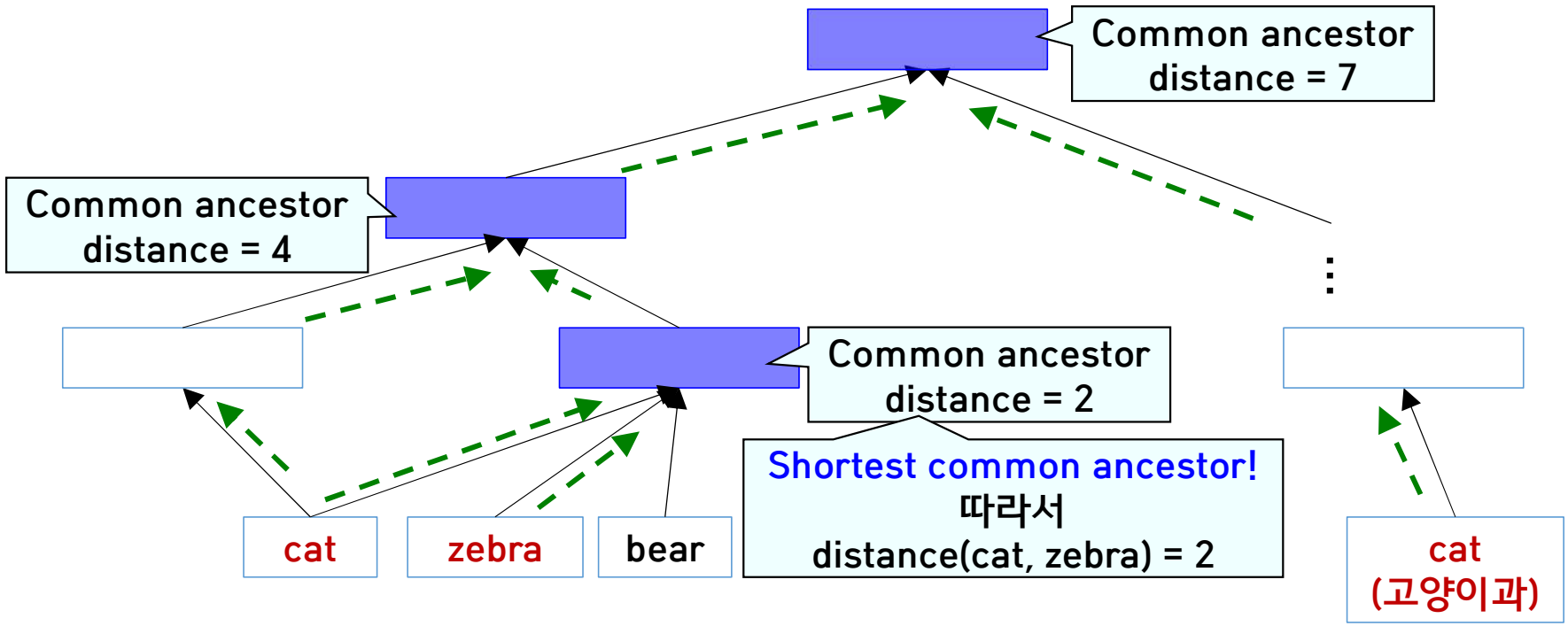
- 가장 가까운 공통 조상(SCA)과 경로(SAP)는 어떻게 찾을 것인가?
- wa, wb에서 출발해 **그래프 탐색**해야 함



distance(wa,wb):
wa, wb에서 출발해 가장 가까운 공통 조상(SCA)과 경로(SAP) 찾아 경로 길이 반환

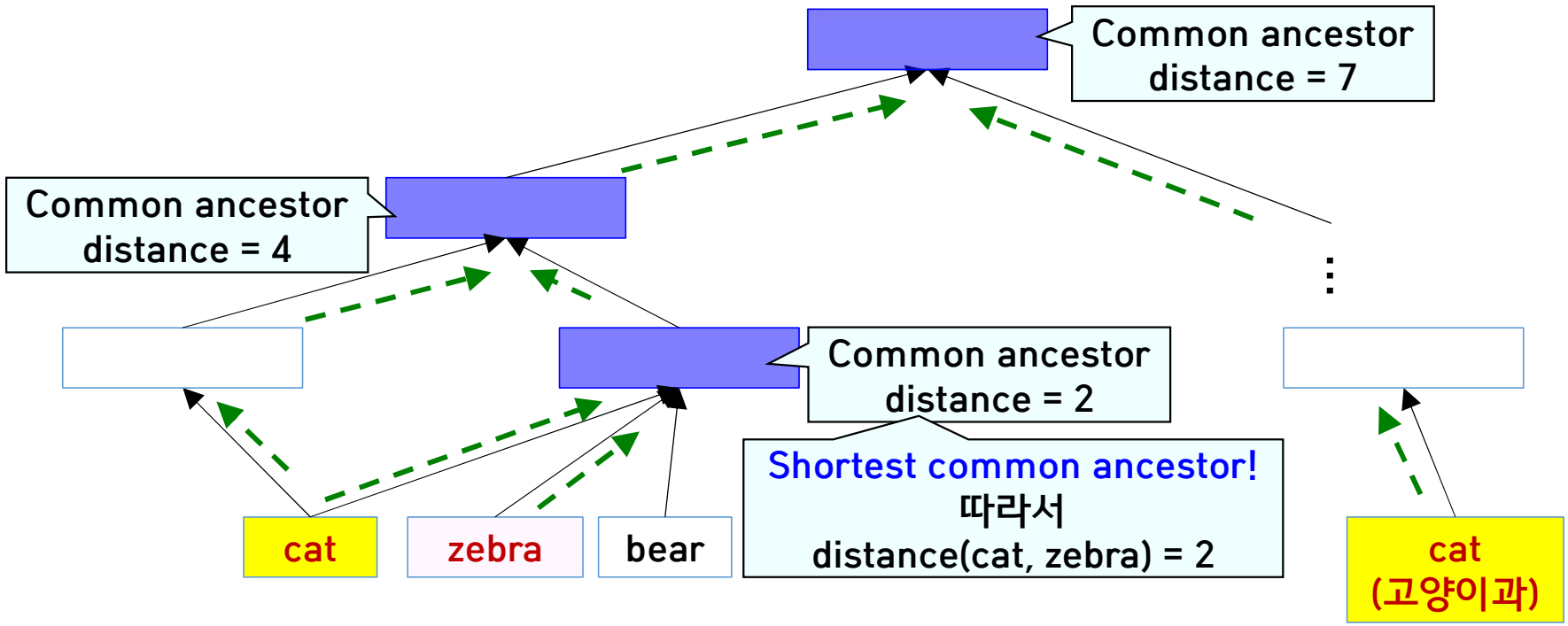
[Q] 가장 가까운 공통 조상(SCA)와 경로(SAP)는 어떤 방법으로 탐색?

DFS vs. BFS



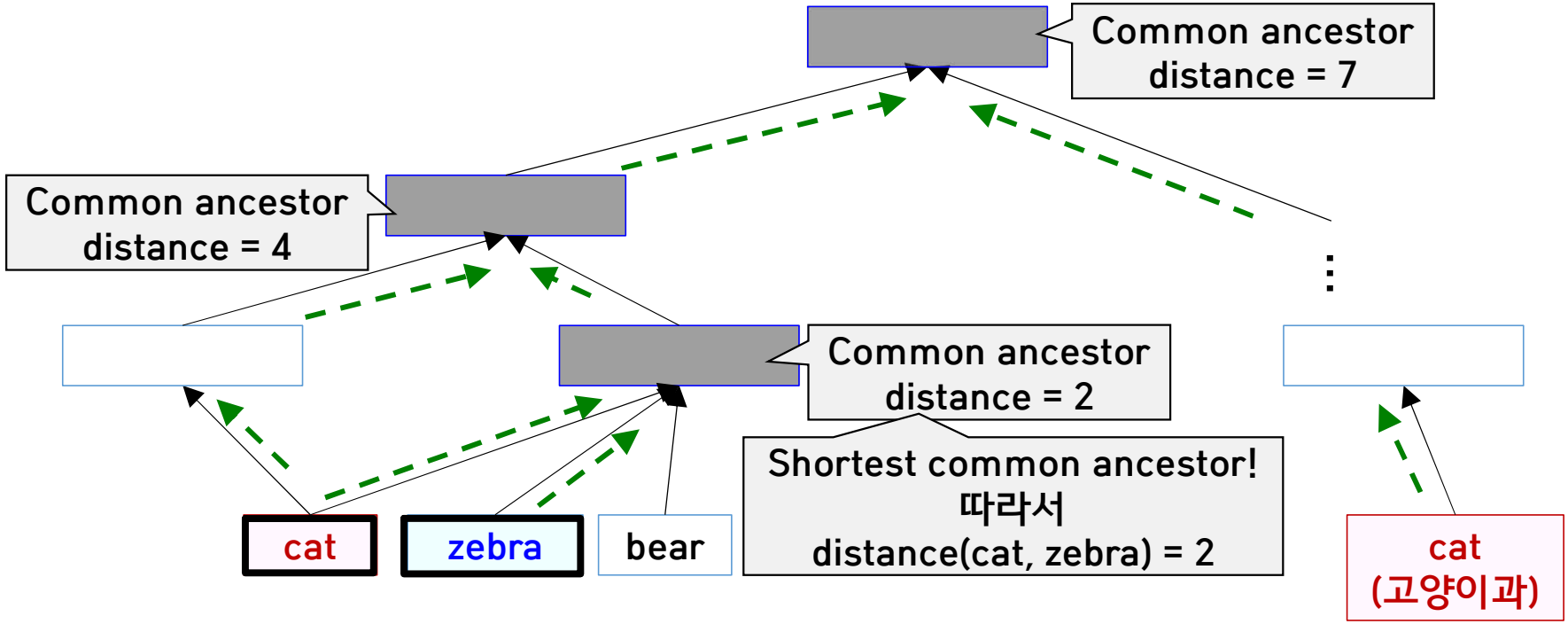
distance(wa,wb):
wa, wb에서 출발해 가장 가까운 공통 조상(SCA)과 경로(SAP) 찾아 경로 길이 반환

[Q] wa와 wb에 해당하는 정점은 각각 둘 이상일 수 있다.
이렇게 **여러 출발지로부터 탐색**하려면 어떤 탐색 방법에 기반해야 하나? ms BFS



distance(wa,wb):
wa, wb에서 출발해 가장 가까운 공통 조상(SCA)과 경로(SAP) 찾아 경로 길이 반환

- 여러 출발지로부터 탐색하면서
- **서로 다른 단어에서 온 경로끼리 만남만 고려**하면 되며,
- **같은 단어의 서로 다른 정점에서 온 경로끼리 만남은 고려할 필요 없음**

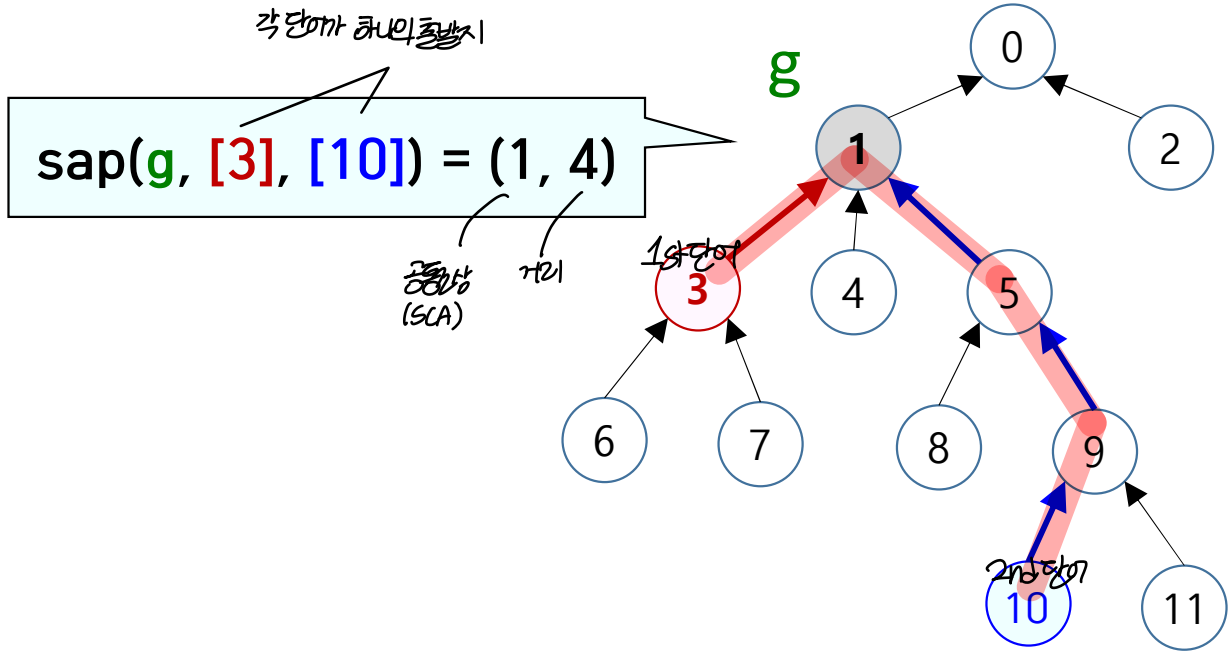


MSBFS + MSBFS → 동시진행
: 고양이 일직 발견됨.
(거리가 멀어지는 단순 탐색 필요)

sap() 함수의 입출력 예: __main__ 아래 테스트 코드 있음

distance(wordA, wordB)

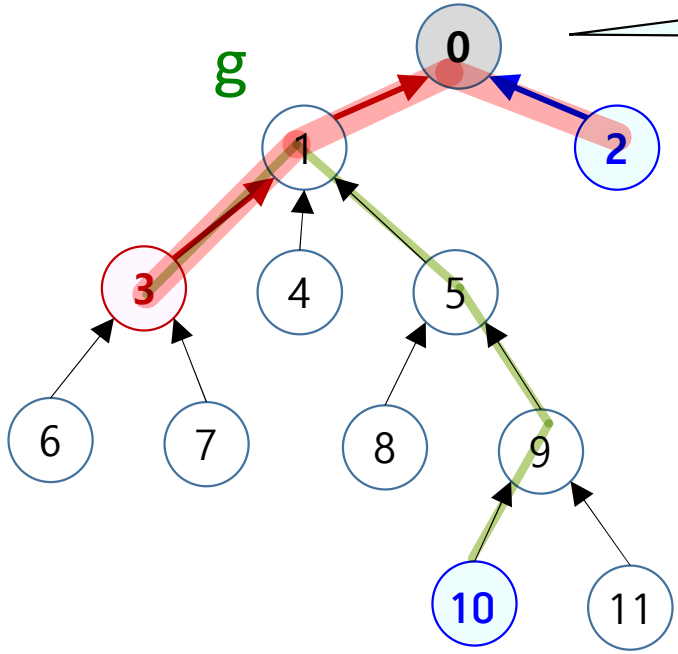
```
# 구현해야 하는 기능
def sap(g, aList, bList):
    # g: Digraph 객체, aList: 정점 번호 목록, bList: 정점 번호 목록
    # Digraph g 탐색해 aList 속한 정점과 bList 속한 정점 간
    # SCA(가장 가까운 조상) 하나와 SCA까지 거리 반환
```



sap() 함수의 입출력 예: __main__ 아래 테스트 코드 있음

distance(wordA, wordB)

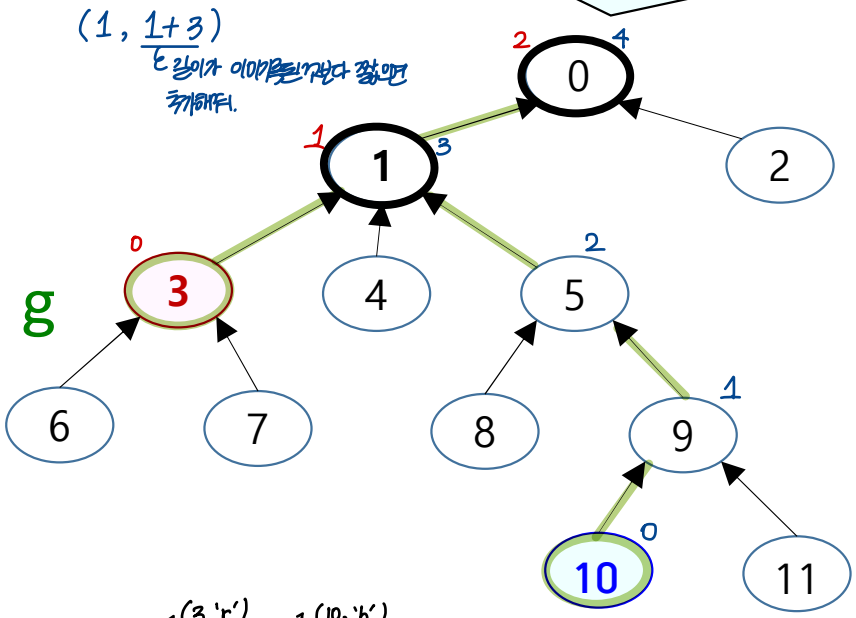
```
# 구현해야 하는 기능
def sap(g, aList, bList):
    # g: Digraph 객체, aList: 정점 번호 목록, bList: 정점 번호 목록
    # Digraph g 탐색해 aList 속한 정점과 bList 속한 정점 간
    # SCA(가장 가까운 조상) 하나와 SCA까지 거리 반환
```



sap(g, [3], [10, 2]) = (0, 3)

조상정점 거리(가장 가까운 애들)

sap(g, [3], [10]) = (1, 4) 구하는 과정



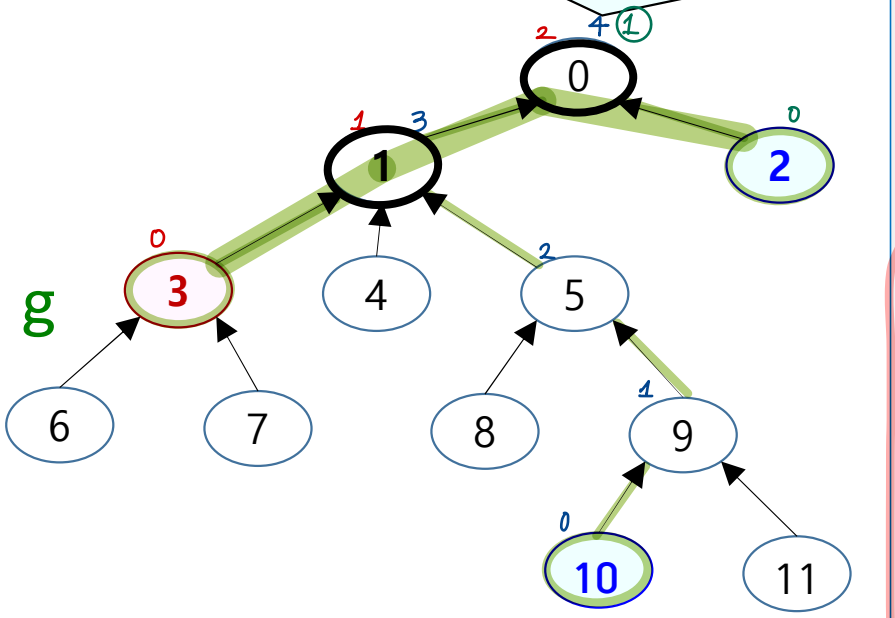
depth	탐색한 정점 (Q에 추가한 정점)
0	3 10
1	1 9
2	0 5
3	1
4	0
5	

탐색 끝

```
sapLength = math.inf  # 지금까지 찾은 SAP의 최소 길이
                      # 무한대로(∞) 초기화
aList, bList 원소 모두 Q에 추가
while Q is not empty:
    v = Q.get()
    v까지 거쳐온 거리 >= sapLength: break  # 탐색 중단
    for w in g.adj[v]: # v→w 간선 있는 각 정점 w에 대해
        if v에 aList로부터 왔다면:
            w에 aList로부터 도달한 적 없다면
            w를 aList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 bList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update

        elif v에 bList로부터 왔다면:
            w에 bList로부터 도달한 적 없다면
            w를 bList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 aList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update
```

같은 정점에 (예: 정점 1)
aList, bList로부터 다 올 수 있음에 유의!

sap(g, [3], [10,2]) = (0, 3) 구하는 과정

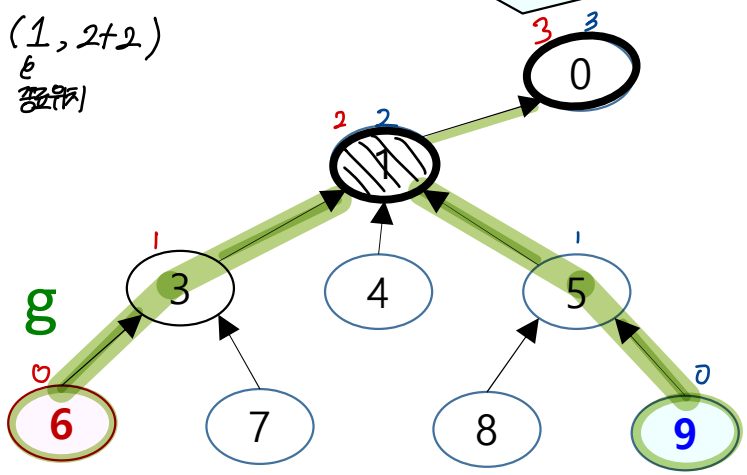


depth	탐색한 정점 (Q에 추가한 정점)
0	3 10 2
1	1 9 0
2	0 5
3	1
4	
5	

```
sapLength = math.inf  # 지금까지 찾은 SAP의 최소 길이
                      # 무한대로(∞) 초기화
aList, bList 원소 모두 Q에 추가
while Q is not empty:
    v = Q.get()
    v까지 거쳐온 거리 >= sapLength: break  # 탐색 중단
    for w in g.adj[v]: # v→w 간선 있는 각 정점 w에 대해
        if v에 aList로부터 왔다면:
            w에 aList로부터 도달한 적 없다면
            w를 aList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 bList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update

        elif v에 bList로부터 왔다면:
            w에 bList로부터 도달한 적 없다면
            w를 bList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 aList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update
```

multi-source BFS 수행하므로
~(V+E) 시간 소요

sap(g, [6], [9]) 구하는 과정

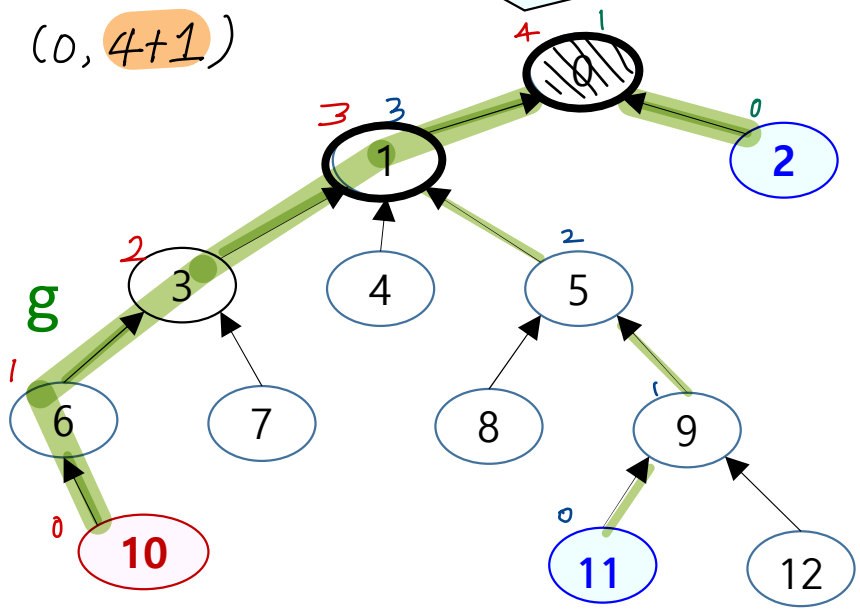


depth	탐색한 정점 (Q에 추가한 정점)
0	6 9
1	3 5
2	1 1
3	0 0
4	
5	

```
sapLength = math.inf  # 지금까지 찾은 SAP의 최소 길이
                      # 무한대로(∞) 초기화
aList, bList 원소 모두 Q에 추가
while Q is not empty:
    v = Q.get()
    v까지 거쳐온 거리 >= sapLength: break  # 탐색 중단
    for w in g.adj[v]: # v→w 간선 있는 각 정점 w에 대해
        if v에 aList로부터 왔다면:
            w에 aList로부터 도달한 적 없다면
            w를 aList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 bList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update

        elif v에 bList로부터 왔다면:
            w에 bList로부터 도달한 적 없다면
            w를 bList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 aList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update
```

multi-source BFS 수행하므로
~(V+E) 시간 소요

sap(g, [10], [11,2]) 구하는 과정



depth	탐색한 정점 (Q에 추가한 정점)
0	10 11 2
1	6 9 0
2	3 5
3	1 1
4	
5	

```
sapLength = math.inf  # 지금까지 찾은 SAP의 최소 길이
                      # 무한대로(∞) 초기화
aList, bList 원소 모두 Q에 추가
while Q is not empty:
    v = Q.get()
    v까지 거쳐온 거리 >= sapLength: break  # 탐색 중단
    for w in g.adj[v]: # v→w 간선 있는 각 정점 w에 대해
        if v에 aList로부터 왔다면:
            w에 aList로부터 도달한 적 없다면
            w를 aList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 bList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update

        elif v에 bList로부터 왔다면:
            w에 bList로부터 도달한 적 없다면
            w를 bList로부터 방문한 것으로 표기하고 Q.put(w)
            w에 aList로부터 방문했었다면 SAP 길이 구해서
              sapLength보다 더 작다면 update
```

multi-source BFS 수행하므로
~(V+E) 시간 소요

SAP&SCA 탐색 기능 생각할 때 유의사항 (구현 파트에서도 다시 언급)

- 거리 같은 SCA가 여럿이면, 그 중 하나만 반환 (예: 먼저 발견한 하나)

- SAP&SCA 탐색은
- 다른 방법 사용해도 되나
- **(worst case) V+E에 비례한 시간**에 수행해야 함

WordNet

WordNet 구현 방법 및 이에 필요한 그래프 탐색, cycle 탐지, symbol table 활용법 이해

01. 퀴즈 풀이 & 예습 내용 복습 (이번 주 #1~3차 답안 공개)
02. WordNet은 무엇이며, 어떻게 사용되는가?
03. Outcast 탐지 방법 개요
04. SCA & SAP 찾기 개요
05. WordNet과 outcast 탐지 구현
06. 구현할 API 및 유의사항
07. 실습: WordNet과 outcast 탐지 구현

디버깅 위해서는 오늘 작성해야 하는 부분 외에도
기 제공되는 코드의 기능도 이해 필요

WordNet 그래프 구현: synsets.txt(정점), hypernyms.txt(간선) 읽어 구현

WordNet 그래프가 준비되어야 탐색도 할 수 있음

- synsets.txt: WordNet의 정점에 대한 정보 (유사어 집합들, **syn**onym **sets**)
- 한 라인에 정점 하나의 정보 포함
- 라인 형식: [정점번호],[정점에 포함되는 유사어 집합(공백으로 구분)],[의미]
- [정점번호]: 0~82191
- [정점에 포함되는 유사어 집합(공백으로 구분)]: 1~28개
- [의미]: 이번 시간 실습에서는 사용하지 않음

유사어 간에는 공백으로 구분

```
...
24084,big_cat cat,any of several large cats typically able to roar and ···
...
27900,cat true_cat,feline mammal usually having thick soft fur and ···
...
```

e 정점번호 e 유사어들

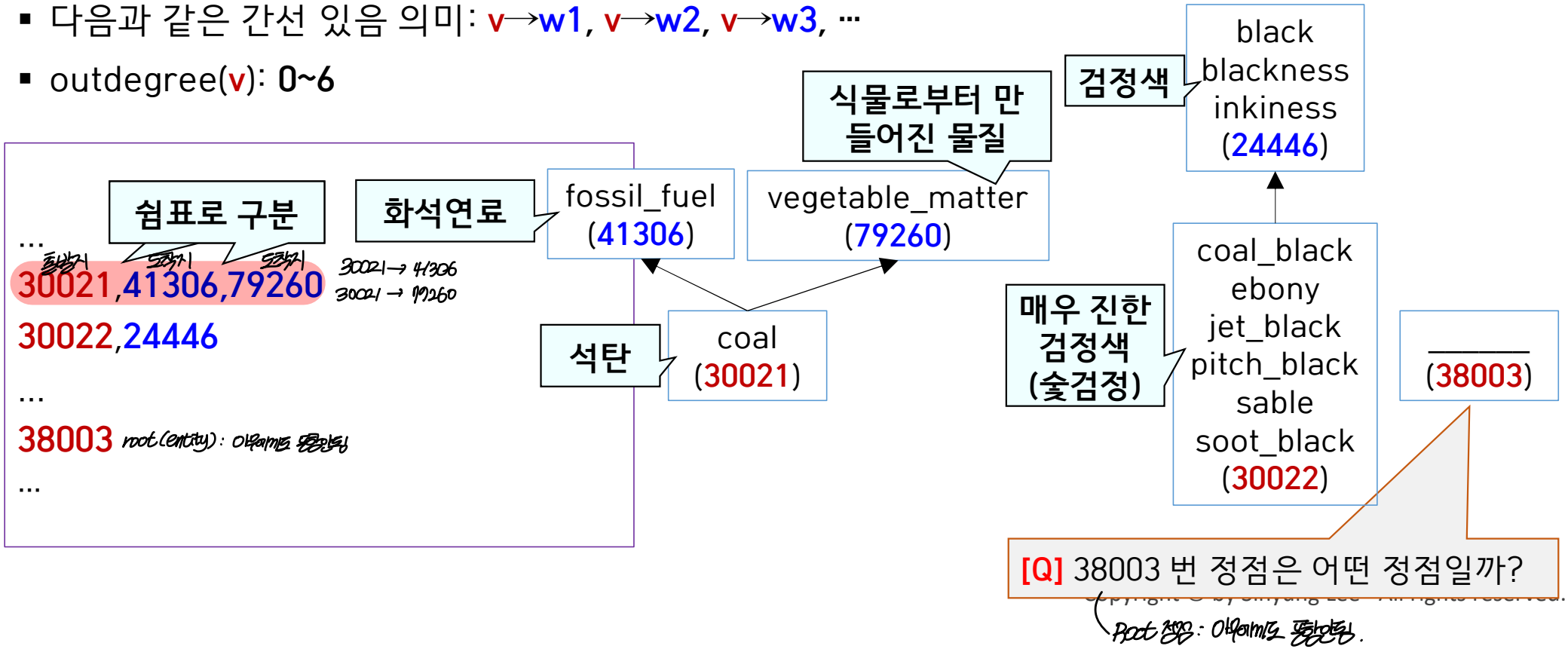
쉼표로 구분

big_cat cat

cat true_cat

WordNet 그래프 구현: synsets.txt(정점), **hypernyms.txt(간선)** 읽어 구현

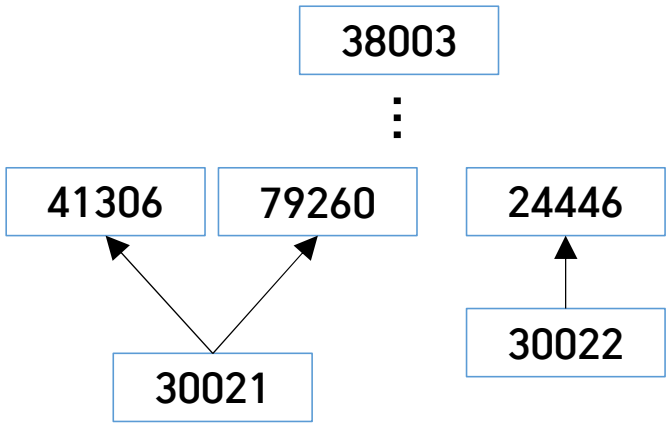
- hypernyms.txt: WordNet의 간선 정보. hyponym(부분 집합) → hypernym(더 큰 집합) 관계
- 한 라인에 정점 하나에서 나가는 모든 간선 (outgoing edges) 정보 포함
- 라인 형식: [정점 v 번호],[정점 w1 번호],[정점 w2 번호],[정점 w3 번호], …
- 다음과 같은 간선 있음 의미: v→w1, v→w2, v→w3, …
- outdegree(v): 0~6



WordNet 그래프 표현: ① Digraph ② 숫자-영문 목록 ③ 영문-숫자 심볼테이블

① Digraph self.g

- 기존 Digraph 클래스로 숫자 정점 가진 그래프 표현
- 기존에 Digraph에 대해 작성한 기능 재활용 가능하므로 작성할 코드 줄어들고, 숫자 그래프에 대한 SCA&SAP 탐색도 가능



② 리스트 self.synsets

- synset[v]: 정점번호 v 의미하는 유사어 집합(synset) 저장
- 정점 번호 주어졌을 때 대응되는 단어 찾기 위함

v	synset[v]
24446	black blackness …
30021	coal
30022	coal_black ebony …
38003	entity
41306	fossil_fuel
79260	vegetable_matter
…	…

유의어 집합이 한 정점 구성

③ 심볼테이블 self.nounToIndex

- nounToIndex[n]: 단어 n에 대응되는 정점번호 **목록**
- 단어 주어졌을 때 대응되는 정점 찾기 위함

n	nounToIndex[n]
black	[24444,24445,24446]
coal	[30021]
entity	[38003]
coal_black	[30022]
ebony	[30022,37052,…]
fossil_fuel	[41306]
…	…

[Q] 왜 한 숫자 아닌 숫자 **목록** 필요한가?

WordNet 자료구조 ①~③ 사용한 outcast 탐색 과정

outcast({"horse", "cat", "zebra", "bear", "table"}):
집합의 각 단어 wi에 대해
거리 합 di = distance(wi, w1) + distance(wi, w2) + … + distance(wi, wn) 계산
di 가장 큰 단어를 outcast로 선정

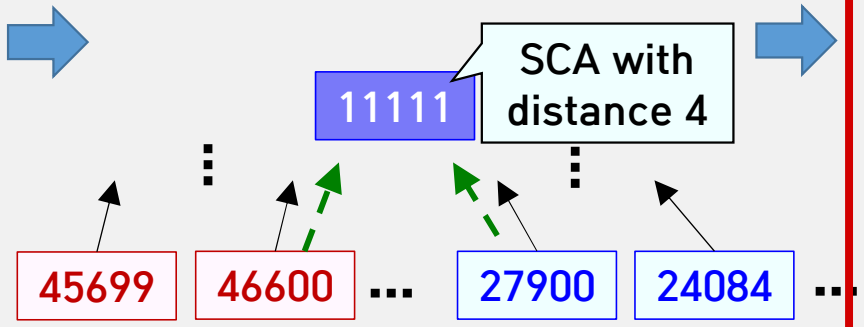
distance("horse", "cat"):
"horse", "cat"에서 출발해 가장 가까운 공통 조상(SCA)과 경로(SAP) 찾아 경로 길이 반환

③ 심볼 테이블 사용해 단어 wa, wb에 대응되는 정점 집합 찾기

n	nounToIndex[n]
horse	[46599,46600,…]
zebra	[82086]
cat	[27900,24084,…]
bear	[23558,23559]
table	[75055,75056,…]
…	…

SAP

① 숫자 정점 그래프 탐색해 SAP&SCA 찾기



② 목록 사용해 숫자 SCA를 영문으로 변환, 거리와 함께 반환

v	synset[v]
…	…
11111	animal
…	…

SCA="animal" with distance 4

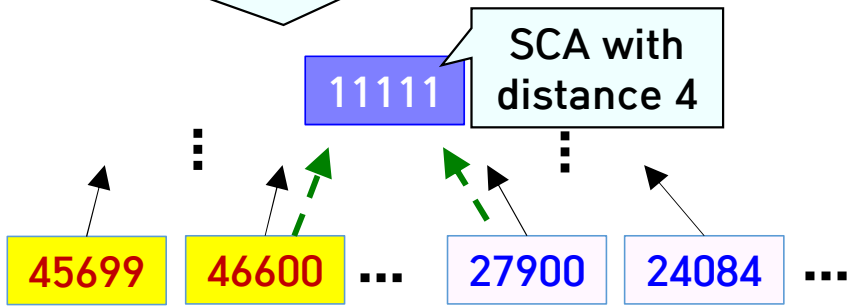
① 숫자 그래프에서 최단거리 탐색 위해 전후로 ③,② 사용해 영문→숫자, 숫자→영문 대응시킴

숫자 그래프에서 SAP&SCA 찾는 기능 구현

- 모든 탐색은 숫자 그래프에서 이루어지므로
- 숫자 그래프에서 SAP&SCA 찾는 기능만 구현하면 됨
- 이후 WordNet 아닌 다른 그래프에서도 같은 기능 필요하면 재활용 가능 (모듈화)

```
sap(g, aList, bList):
    # Digraph g에서 정점 집합 aList와 정점 집합 bList간 SAP과 SCA 찾아 반환
    # aList에서 multi-source BFS, bList에서 multi-source BFS 해서
    # 서로 만나는 가장 가까운 지점 찾기
```

예: sap(WordNet.g, [45699,46600,…], [27900,24084,…])



WordNet

WordNet 구현 방법 및 이에 필요한 그래프 탐색, cycle 탐지, symbol table 활용법 이해

01. 퀴즈 풀이 & 예습 내용 복습 (이번 주 #1~3차 답안 공개)

02. WordNet은 무엇이며, 어떻게 사용되는가?

03. Outcast 탐지 방법 개요

04. SCA & SAP 찾기 개요

05. WordNet과 outcast 탐지 구현

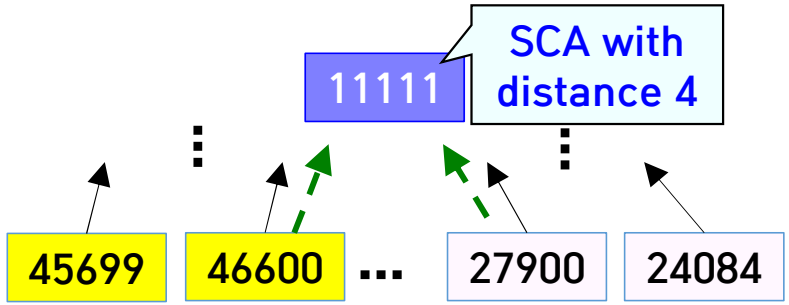
06. 구현할 API 및 유의사항

07. 실습: WordNet과 outcast 탐지 구현

실습 목표: 숫자 정점 그래프 탐색해 SAP&SCA 찾는 기능 작성

- 기본 그래프 탐색 방법(DFS, BFS, multi-source BFS)을 적재 적소에 활용
- 위 기능을 제외한 기능은 구현된 코드 제공됨

↳ DFS, BFS를 알아둘 필요X

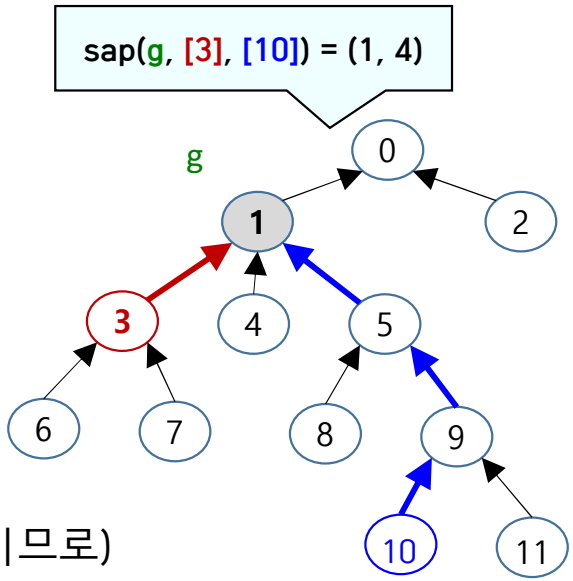


Unit test for sap( )
사용.

프로그램 구현 조건

- 숫자 그래프가 입력으로 주어졌을 때, SAP과 거리 찾는 함수 구현

```
def sap(g, aList, bList):
    # 입력 g: Digraph 객체, aList: 정점 번호 리스트, bList: 정점 번호 리스트
    # Digraph g 탐색해 aList 속한 정점과 bList 속한 정점 간
    # SCA(가장 가까운 조상) 하나와 SCA까지 거리 반환
```

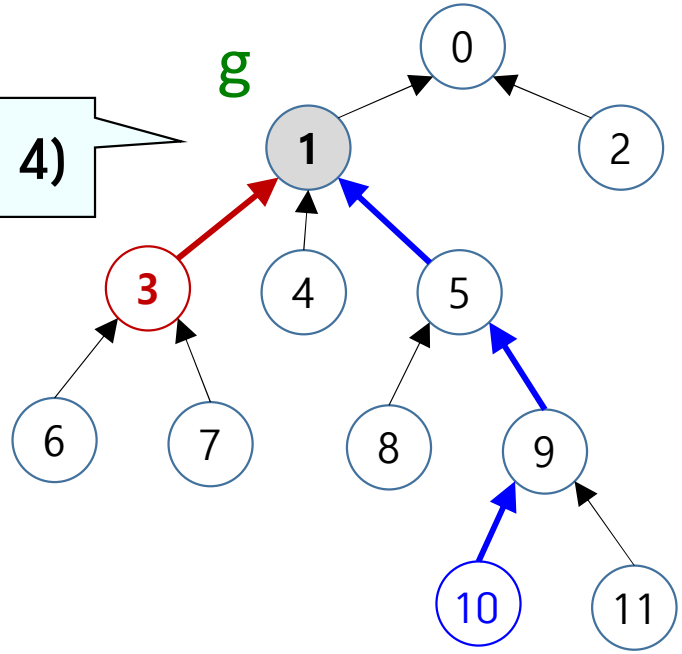


- 이번 시간에 제공한 코드 Digraph.py에 위 함수에 대한 코드 작성해 제출
  - 위 파일에 포함된 Digraph 클래스는 반드시 사용해야 함 (sap 함수의 입력이므로)
  - 위 파일에 포함된 코드 중 sap 함수 이외의 코드는 변경하면 안 됨
  - 이미 import된 모듈 외 추가로 import할 수 없음

sap() 함수의 입출력 예: __main__ 아래 테스트 코드 있음

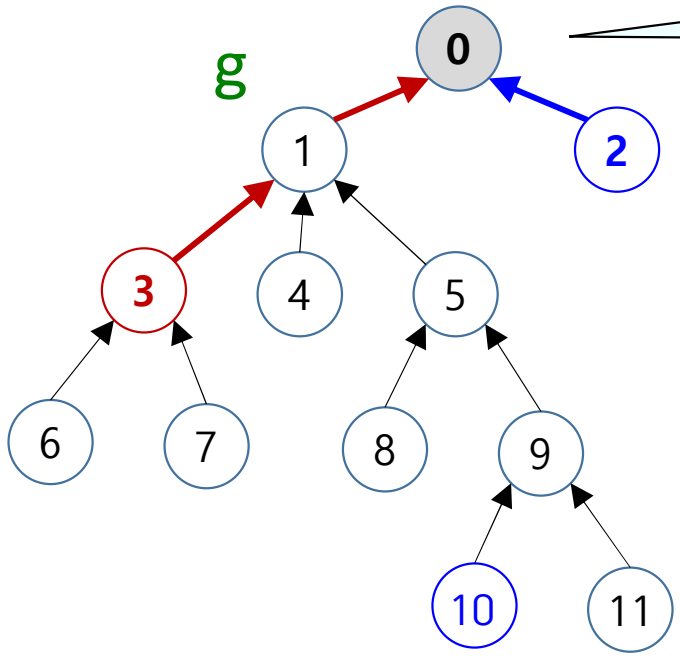
```
# 구현해야 하는 기능
def sap(g, aList, bList):
    # g: Digraph 객체, aList: 정점 번호 목록, bList: 정점 번호 목록
    # Digraph g 탐색해 aList 속한 정점과 bList 속한 정점 간
    # SCA(가장 가까운 조상) 하나와 SCA까지 거리 반환
```

sap(g, [3], [10]) = (1, 4)



sap() 함수의 입출력 예: __main__ 아래 테스트 코드 있음

```
# 구현해야 하는 기능
def sap(g, aList, bList):
    # g: Digraph 객체, aList: 정점 번호 목록, bList: 정점 번호 목록
    # Digraph g 탐색해 aList 속한 정점과 bList 속한 정점 간
    # SCA(가장 가까운 조상) 하나와 SCA까지 거리 반환
```



sap(g, [3], [10, 2]) = (0, 3)

구현할 API 정리: sap(g, aList, bList)

```
# 구현해야 하는 기능
def sap(g, aList, bList):
    # g: Digraph 객체, aList: 정점 번호 목록, bList: 정점 번호 목록
    # Digraph g 탐색해 aList 속한 정점과 bList 속한 정점 간
    # SCA(가장 가까운 조상) 하나와 SCA까지 거리 반환

    # Queue 클래스 객체 Q만들어 aList와 bList의 모든 정점 추가
    # Queue에 정점 추가할 때는 3-tuple로 추가: (정점번호, aList/bList 중 어디에서 왔는지, 거쳐온 거리)
    # aList와 bList에 있는 정점은 모두 시작점이므로 거쳐온 거리 = 0으로 추가
    # 만약 aList와 bList 모두에 속하는 정점 v 있다면 (v, 0)을 바로 반환하며 이후로 진행 안 함

    # sapLength = math.inf 로 초기화
    # while not Q.empty():
    #     v = Q.get()
    #     if v까지 거쳐온 거리 > sapLength: break # 탐색 중단
    #     for w in g.adj[v]:  # v->w 간선 있는 각 정점 w에 대해
    #         앞 페이지에서 본 방법대로 w의 방문 여부 표기하고 sapLength 업데이트
    #         …
```

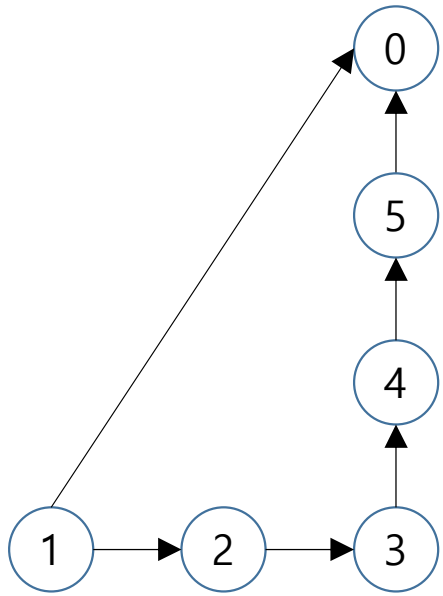
이미 구현되어 활용 가능한 기능

```
# Digraph를 나타내는 클래스
class Digraph:
    @staticmethod
    def digraphFromFile(fileName):
        # 파일로부터 그래프 정보 읽어와 Digraph 객체 만들어 반환
```

- 그래프 파일 형식
- **첫 줄: 정점 개수**
- 다음 줄부터 한 줄에 간선 하나 정보 있음: v w
- 간선 v→w 있음 의미
- 그래프 파일은 Digraph.py와 같은 디렉토리에 있어야 함

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

digraph6.txt



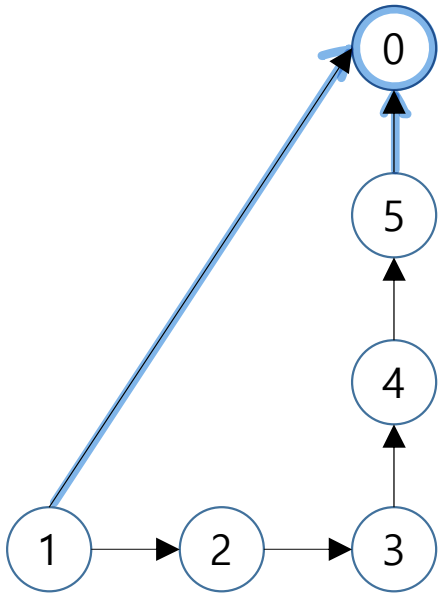
Digraph.digraphFromFile(fileName)와 sap(g, aList, bList) 활용 예

```
d6 = Digraph.digraphFromFile('digraph6.txt')
print(sap(d6, [1], [5])) # (0,2)
print(sap(d6, [1], [1])) # (1,0)
print(sap(d6, [1], [4])) # Either (0,3) or (4,3)
print(sap(d6, [1], [3])) # (3,2)
```

거리 같은 SCA 둘 이상이면 하나만 반환

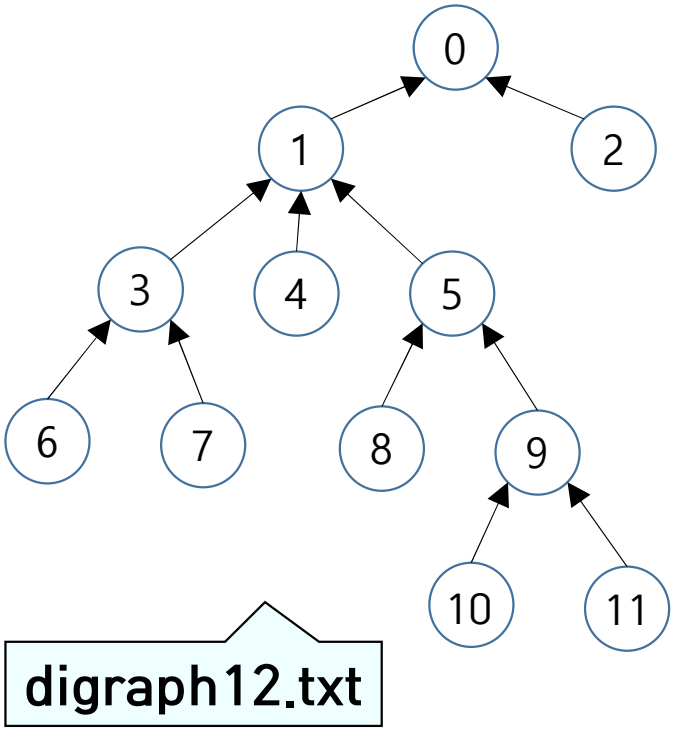
```
6
1 0
1 2
2 3
3 4
4 5
5 0
```

digraph6.txt



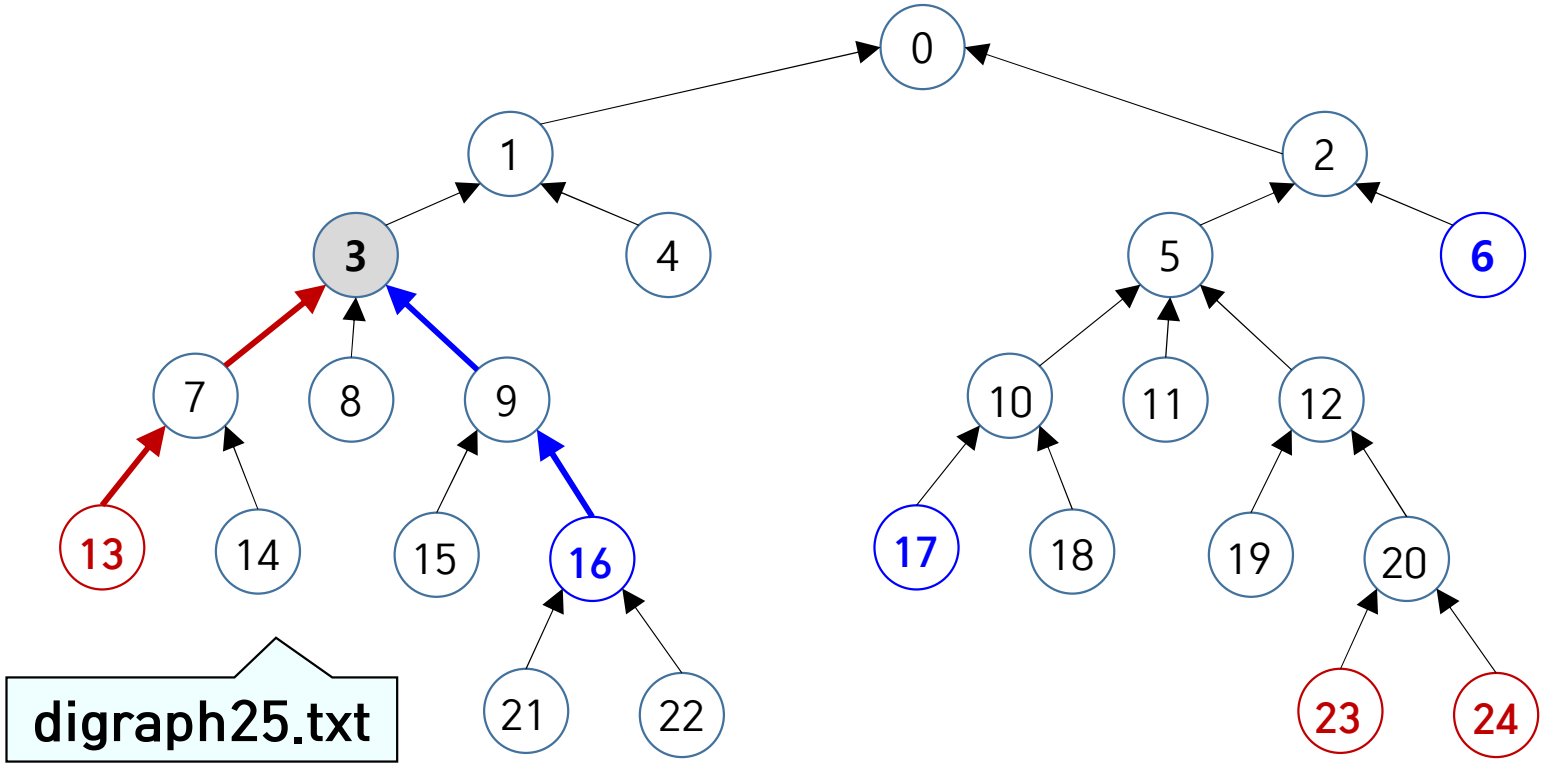
Digraph.digraphFromFile(fileName)와 sap(g, aList, bList) 활용 예

```
d12 = Digraph.digraphFromFile('digraph12.txt')
print(sap(d12, [3], [10]))      # (1,4)
print(sap(d12, [3], [10, 2]))   # (0,3)
```



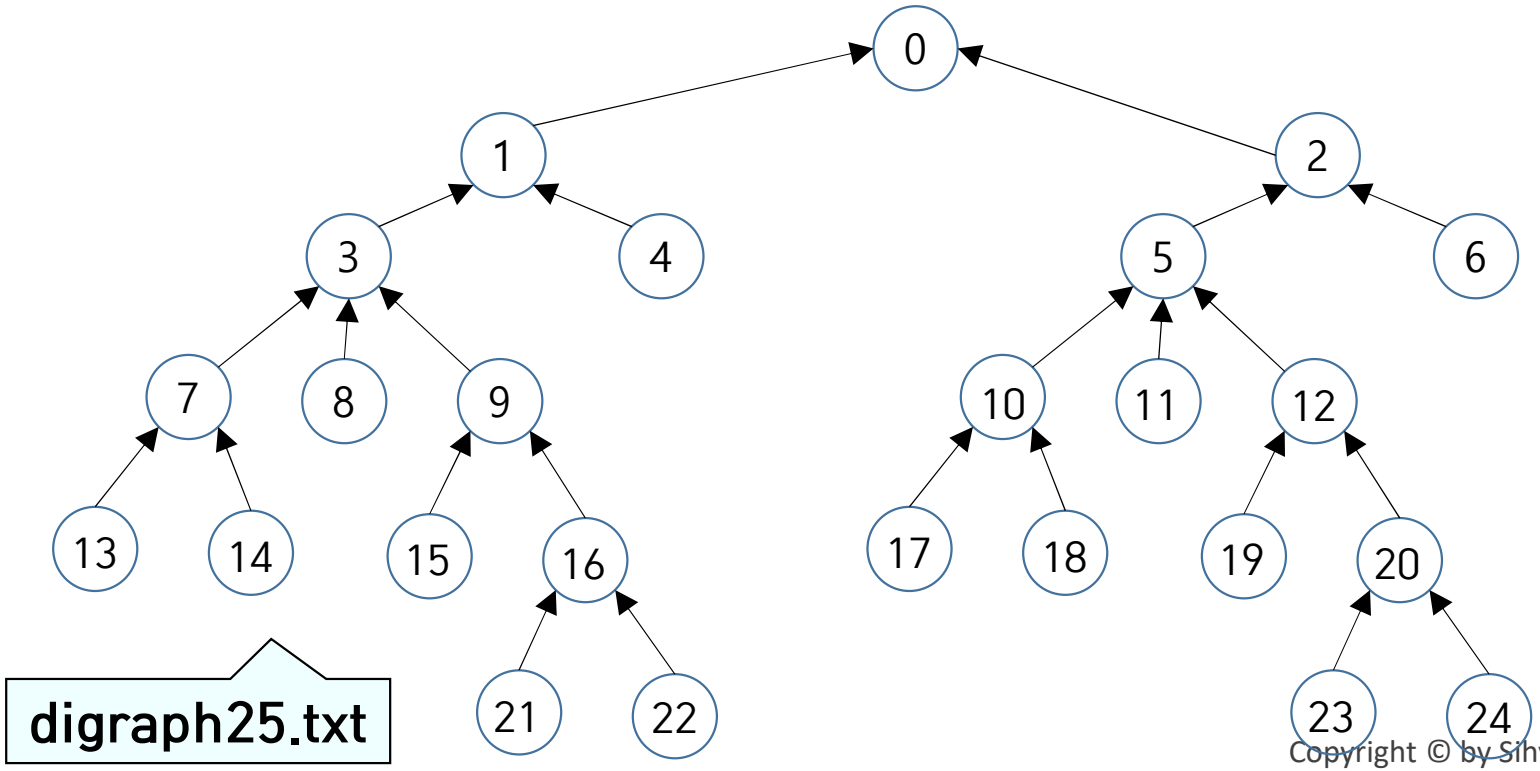
Digraph.digraphFromFile(fileName)와 sap(g, aList, bList) 활용 예

```
d25 = Digraph.digraphFromFile('digraph25.txt')
print(sap(d25, [13,23,24], [6,16,17]))  # (3,4)
```



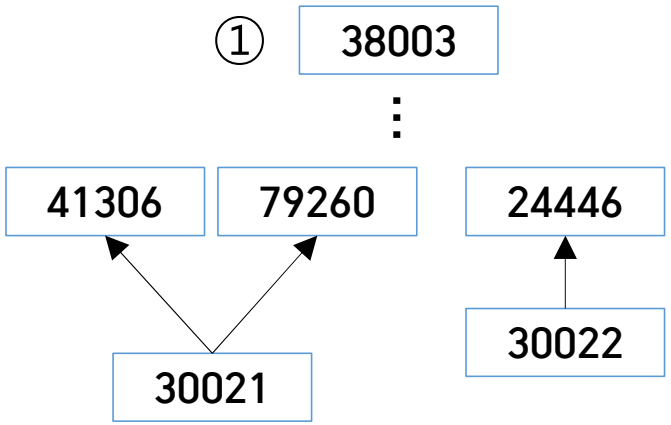
Digraph.digraphFromFile(fileName)와 sap(g, aList, bList) 활용 예

```
d25 = Digraph.digraphFromFile('digraph25.txt')
print(sap(d25, [13,23,24], [6,16,17]))  # (3,4)
print(sap(d25, [13,23,24], [6,16,17,4]))  # (3,4) or (1,4)
print(sap(d25, [13,23,24], [6,16,17,1]))  # (1,3)
print(sap(d25, [13,23,24,17], [6,16,17,1]))  # (17,0)
```



그 외 구현되어 활용 가능한 기능 (검증, 디버깅에 필요)
WordNet 표현하는 클래스: 생성자

```
# 구현되어 활용 가능한 기능
class WordNet:
    def __init__(self, synsetFileName, hypernymFileName):
        # 생성자. synset과(정점) hypernym(간선) 정보 저장한 파일 이름 받아
        # WordNet 표현하는 다음 3개 자료구조를 멤버변수로 생성
        # ① self.g: 숫자 정점 가진 그래프로 Digraph 클래스 객체
        # ② self.synsets: 정점번호→유사어집합(문자열) 관계 저장하는 리스트
        # ③ self.nounToIndex: 단어→정점번호리스트 관계 저장하는 심볼테이블
```



②

v	synset[v]
24446	black blackness …
30021	coal
30022	coal_black ebony …
38003	entity
41306	fossil_fuel
79260	vegetable_matter
…	…

③

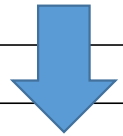
n	nounToIndex[n]
black	[24444,24445,24446]
coal	[30021]
entity	[38003]
coal_black	[30022]
ebony	[30022,37052,…]
fossil_fuel	[41306]
…	…

그 외 구현되어 활용 가능한 기능 (검증, 디버깅에 필요)
WordNet 표현하는 클래스: 생성자 외 멤버함수

```
# 구현되어 활용 가능한 기능
class WordNet:
    def __init__(self, synsetFileName, hypernymFileName): # 생성자

    def nouns(self): # WordNet에 속하는 모든 단어를 리스트로 반환

    def isNoun(self, word):
        # word가 WordNet에 속하는 단어인지 True/False로 알려줌

    def sap(self, nounA, nounB):
        # WordNet 상의 두 단어 nounA와 nounB 간 SCA와 거리 찾아 반환하며
        # 실습 과제로 구현해야 하는 sap(g, aList, bList)함수 호출
        # 따라서 sap(g, aList, bList)함수가 제대로 구현되어야 동작
```

WordNet 클래스 활용 예

```
wn = WordNet("synsets.txt", "hypernyms.txt")
print(wn.isNoun("blue"))
print(wn.isNoun("fox"))
print(wn.isNoun("lalala"))
print(wn.sap("blue", "red"))
print(wn.sap("blue", "fox"))
print(wn.sap("apple", "banana"))
```



```
True
True
False
('chromatic_color chromatic_colour spectral_color spectral_colour', 2)
('material stuff', 8)
('edible_fruit', 2)
```

그 외 구현되어 활용 가능한 기능 (검증, 디버깅에 필요)
Outcast 찾는 기능

```
# 구현되어 활용 가능한 기능
def outcast(wordNet, wordFileName):
    # WordNet 클래스 객체 wordNet에서
    # wordFileName 내의 단어들 중 outcast 찾기
    # 3-tuple 반환:
    #    (1) outcast 단어 (2) 단어 (1)의 다른 단어들과 거리 합
    #    (3) wordFileName 내 단어들로 만든 집합 words
    #
    # 수업에서 배운 대로 각 단어 쌍 간의 SCA와 거리 구해 합산하여
    # 합산 거리가 가장 먼 단어 찾아 반환
    #
    # WordNet 클래스의 sap 함수 호출하므로
    # 실습 과제로 구현해야 하는 sap(g, aList, bList)함수 제대로 구현되어야 동작
```

outcast 함수 활용 예

```
wn = WordNet("synsets.txt", "hypernyms.txt") # WordNet 객체 생성
print(outcast(wn, "outcast5.txt"))
```

outcast5.txt

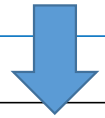
```
horse zebra cat bear table
```

```
('table', 39, {'bear', 'zebra', 'horse', 'cat', 'table'})
```

- outcast
- words 집합
- outcast와 다른 단어 간 거리 합

outcast 함수 활용 예

```
wn = WordNet("synsets.txt", "hypernyms.txt") # WordNet 객체 생성
print(outcast(wn, "outcast5.txt"))
print(outcast(wn, "outcast8.txt"))
print(outcast(wn, "outcast11.txt"))
print(outcast(wn, "outcast9.txt"))
```



```
('table', 39, {'bear', 'zebra', 'horse', 'cat', 'table'})
('bed', 60, {'bed', 'coffee', 'apple_juice', 'orange_juice', 'soda', 'tea', 'milk', 'water'})
('potato', 48, {'mango', 'peach', 'banana', 'strawberry', 'lemon', 'pear', 'apple', 'lime',
'blueberry', 'watermelon', 'potato'})
('fox', 80, {'green', 'white', 'black', 'yellow', 'fox', 'gray', 'blue', 'red', 'pink'})
```

그 외 sap() 함수 구현 시 유의사항

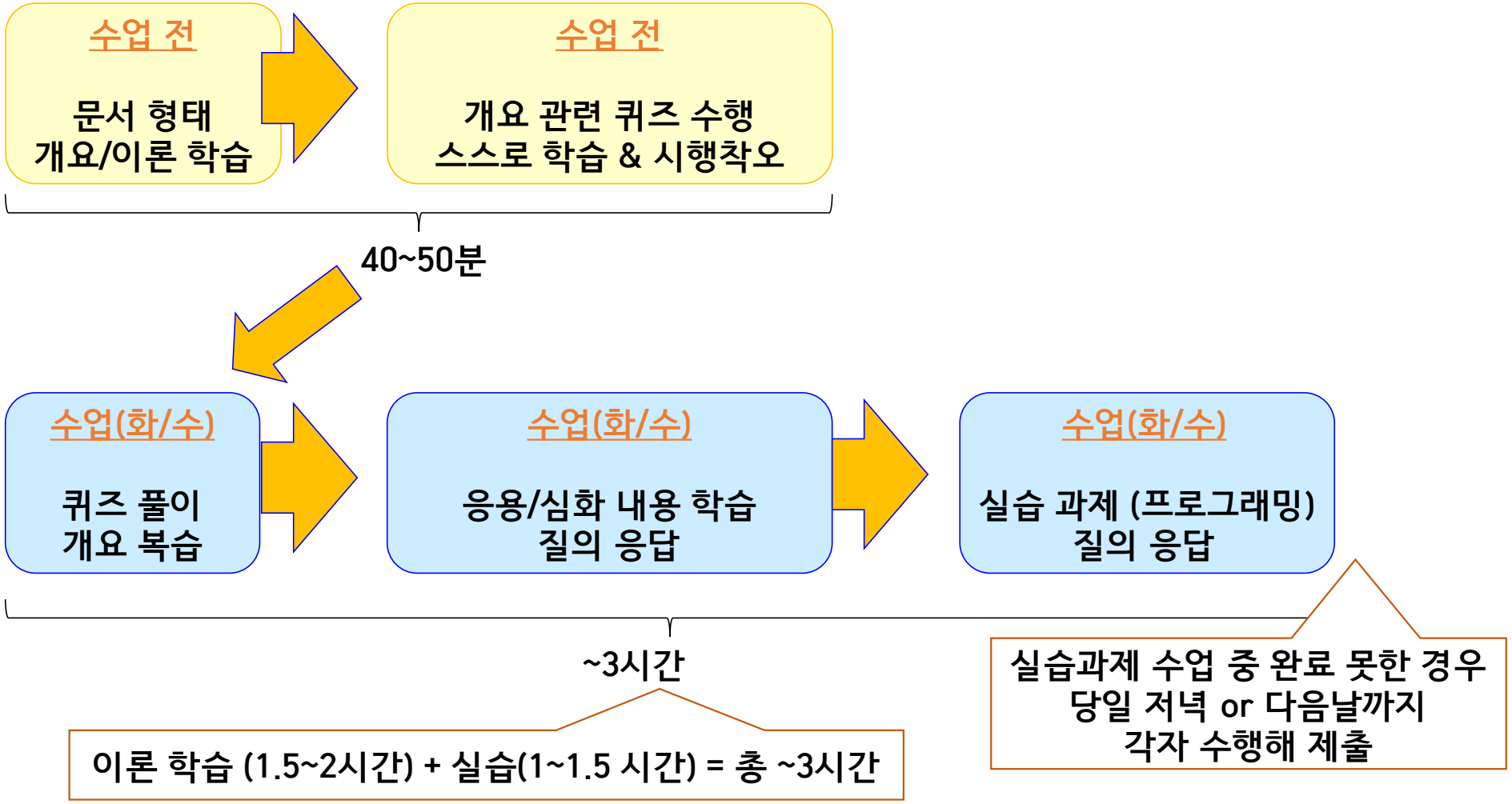
- sap() 함수 구현 시 Digraph.py 내의 BFS 클래스를 호출하지 말고 (참조만 하세요)
- sap() 내부에 직접 BFS 루틴을 다시 구현해 넣는 것이 더 편리합니다.

- 거리 같은 SCA가 여럿이면, 그 중 하나만 반환 (예: 먼저 발견한 하나)

- 앞에서 본 입출력 예제들은 모두 __main__ 아래에 있으므로 검증에 활용

- sap()의 수행 시간은 다음 조건을 지켜야 함
- sap() 함수의 수행 시간은 최대 ~V+E이어야 하며, 이보다 더 오래 걸려서는 안 됨

스마트 출결



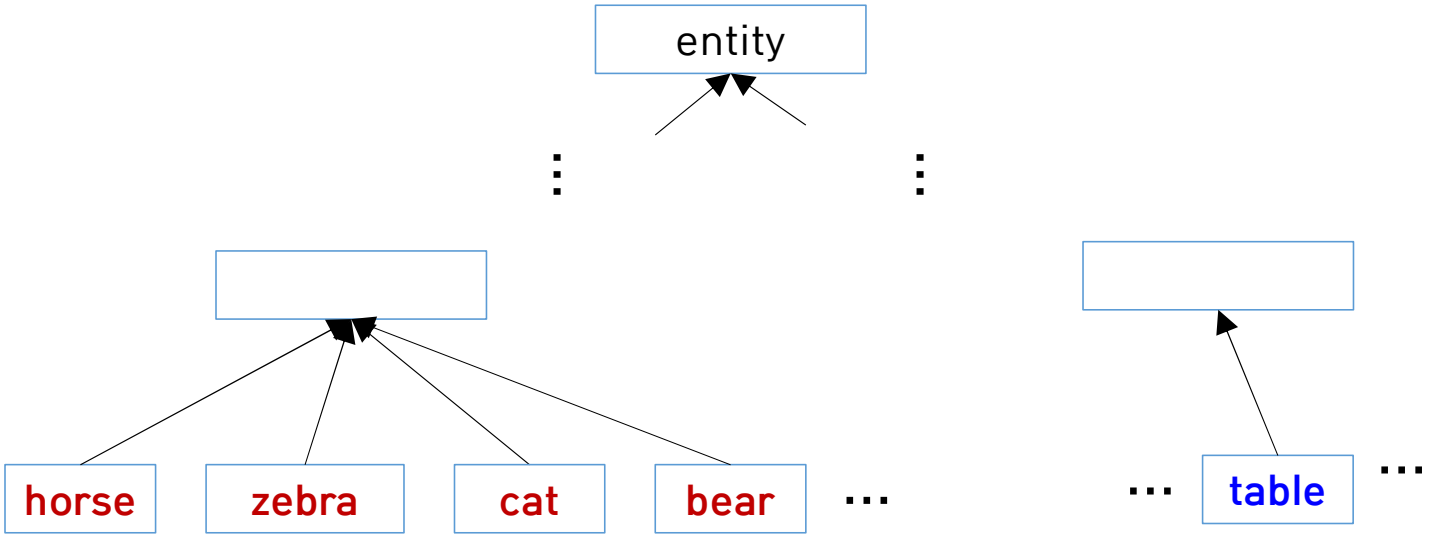
12:00까지 실습 & 질의응답

- 작성한 코드는 lms > 강의 콘텐츠 > 오늘 수업 > 실습 과제 제출함에 제출

- 시간 내 제출 못한 경우 내일 11:59pm까지 제출 마감
- 마감 시간 후에는 제출 불가하므로 그때까지 작성한 코드 꼭 제출하세요.

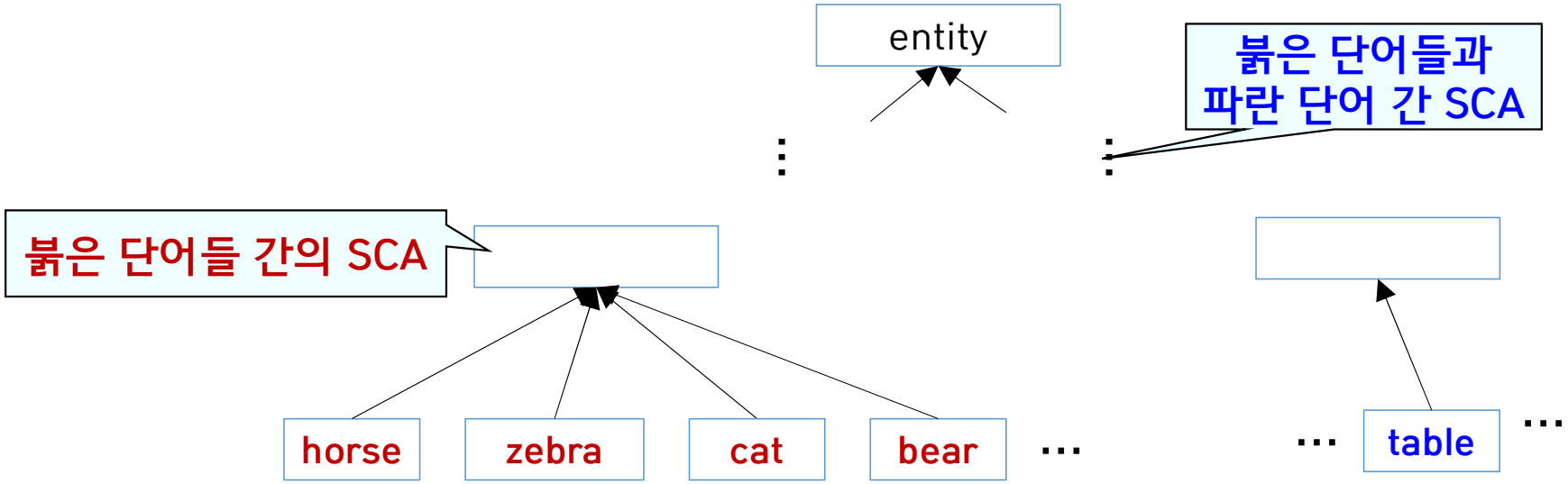
WordNet 정리: WordNet 정의와 활용도

- 단어 간 의미 상 포함 관계 나타낸 Digraph
  - 정점: 유사어 집합, 간선: "is a" 관계 (포함 관계)
- 자연어 입력의 의미 분석해 논리적 추론하는데 사용
  - 사자는 고양잇과. 고양잇과는 뒷발톱 4개. 따라서 사자는 뒷발톱 4개
  - horse, zebra, table, monkey 중 의미상 가장 다른 단어는 table
  - …



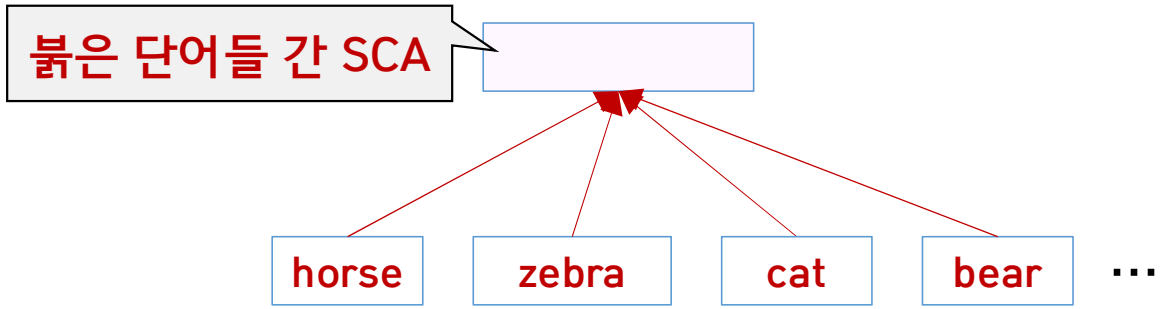
WordNet 정리: Outcast 탐지

- Outcast: 주어진 단어 집합 중 의미상 가장 다른 단어
- WordNet 상에서 단어간 거리가 가장 먼 단어를 outcast로 선정
- 이를 위해 **그래프 탐색** 활용해
- 단어들 간 가장 가까운 조상(SCA) 및 조상까지 경로(SAP)의 거리 계산

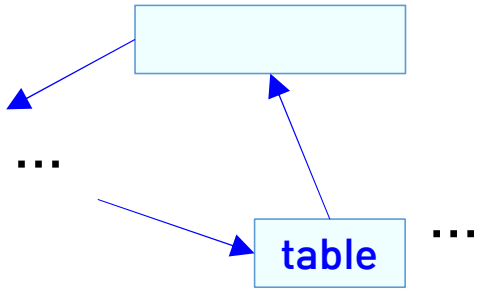


WordNet 정리: BFS, DFS 활용

- WordNet 상 여러 단어들 간 가장 가까운 조상(SCA) 찾기 위해
- multi-source BFS 활용



- WordNet 상 사이클 탐지 위해 DFS 활용 (↱ 사이클 있어야함)
- 사이클 존재하면 단어 간 포함 관계 잘못 정의된 것이므로



- 이처럼 대부분의 그래프 탐색은 BFS, DFS에 기반