알고리즘 실습

181122 - Dijkstra's / Prim's Algorithm

오늘의 목표

- 최단 경로(Shortest Path) 알고리즘
  - 다익스트라(Dijkstra's algorithm)
- 최소 비용 신장 트리(Minimum Spanning Tree) 알고리즘
  - 프림(Prim’s algorithm)

Feedback

지난 시간: 그래프 탐색(BFS/DFS)

- 제출률: 80.49%(33/41)
- 질문: 이메일 56건, 연구실 방문 2건

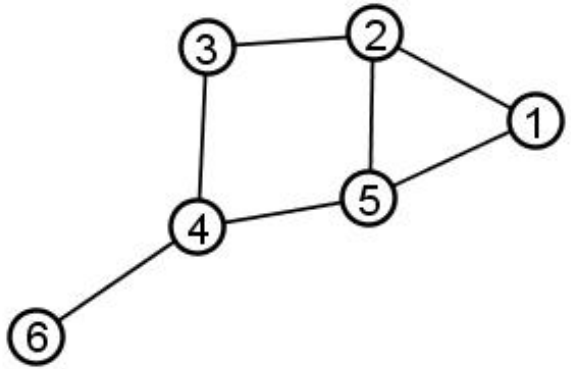
Feedback: 자가 실습 및 과제 방식

- 지난 시간 문제를 해결한 후 퇴실(실습 점수 반영)
- 실습 시간 때 한 것에서 조금 변경하여 과제
- 장점:
- 단점:
    - 분위기가 굉장히 어수선
    - 실습 시간내에 새로 배운 알고리즘을 `구현'하는 부담이 크게 보임

Graph

Graph

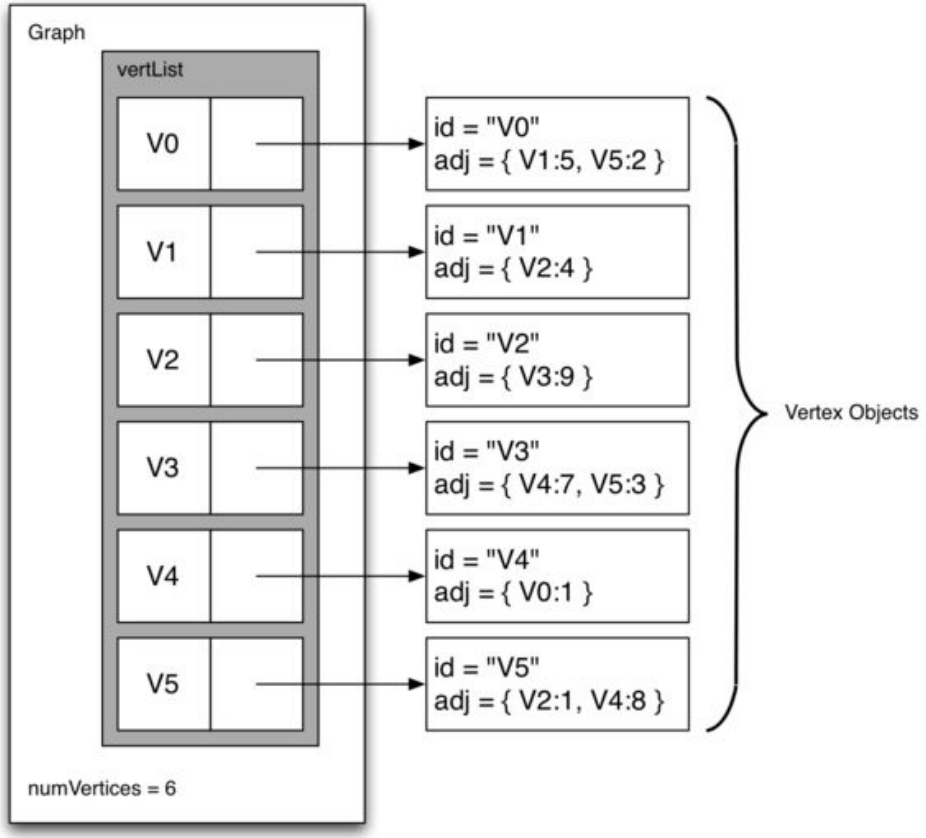
- Node, Edge로 구성
  - Ex) G = (V, E)
    V = {1,2,3,4,5,6}
    E = {{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}



그래프 표현 방법

- Adjacency Matrix
- Adjacency List

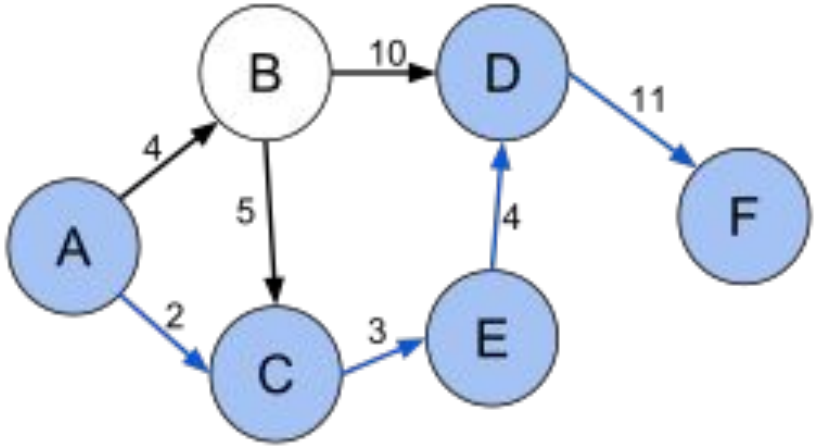
	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	



짧은 길 문제

짧은 길 문제란?

- https://en.wikipedia.org/wiki/Shortest_path_problem
- 시작 노드에서 다른 노드까지 가는 가장 짧은 길을 찾자!

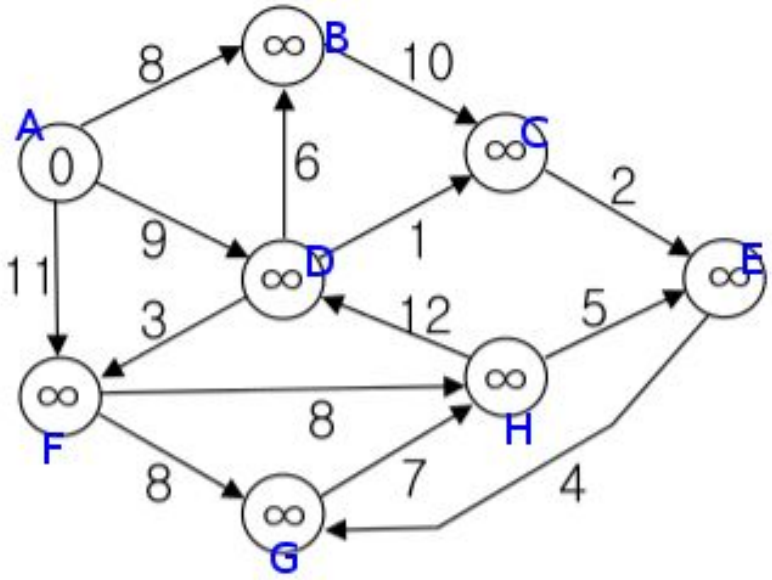


왜 짧은 길을 찾아야할까?

- 최적의 해결법을 알기위해서!
- Open Shortest Path First

가중치가 있는 방향그래프

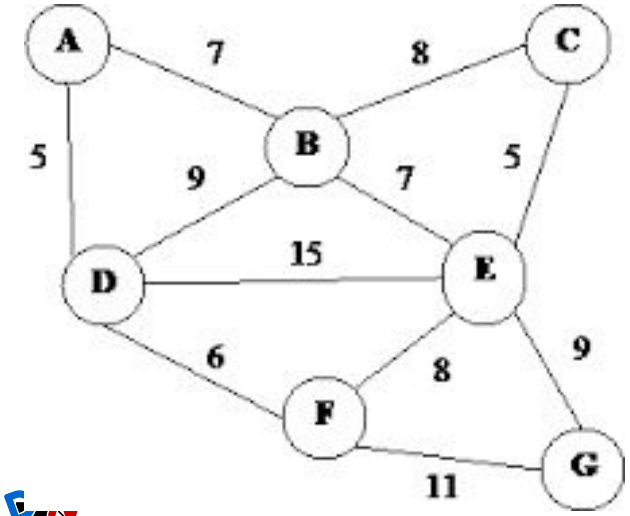
- dict의 연속!




```
 5 WDGRAPH02 = {
 6       'a': {'b': 8, 'd': 9, 'f': 11},
 7       'b': {'c': 10},
 8       'c': {'e': 2},
 9       'd': {'c': 1, 'f': 3},
10       'e': {'g': 4},
11       'f': {'g': 8, 'h': 8},
12       'g': {'h': 7},
13       'h': {'d': 12, 'e': 5}
14       }
15
```

가중치가 있는 무방향그래프

- dict의 연속!

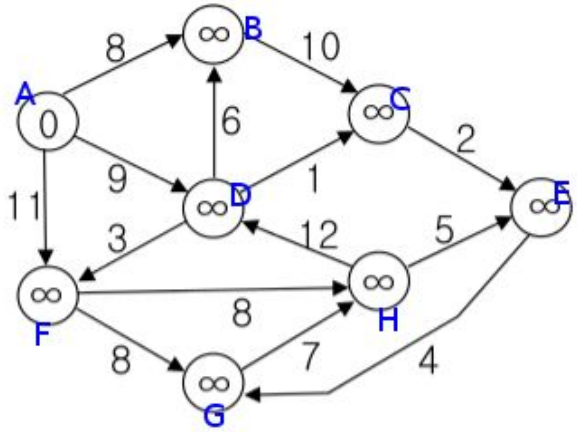



```
4 WUGRAPH01 = {
5     'a': {'b': 7, 'd': 5},
6     'b': {'a': 7, 'c': 8, 'd': 9, 'e': 7},
7     'c': {'b': 8, 'e': 5},
8     'd': {'a': 5, 'b': 9, 'e': 15, 'f': 6},
9     'e': {'b': 7, 'c': 5, 'd': 15, 'f': 8, 'g': 9},
10    'f': {'d': 6, 'e': 8, 'g': 11},
11    'g': {'e': 9, 'f': 11}
12 }
13
```

Networkx

2년 / 4년 전...

- 조교: Dijkstra와 Prim 알고리즘 구현을 끝내면 이런 결과가 나옵니다~
- 결과 보기도 힘들고 재미도 없음

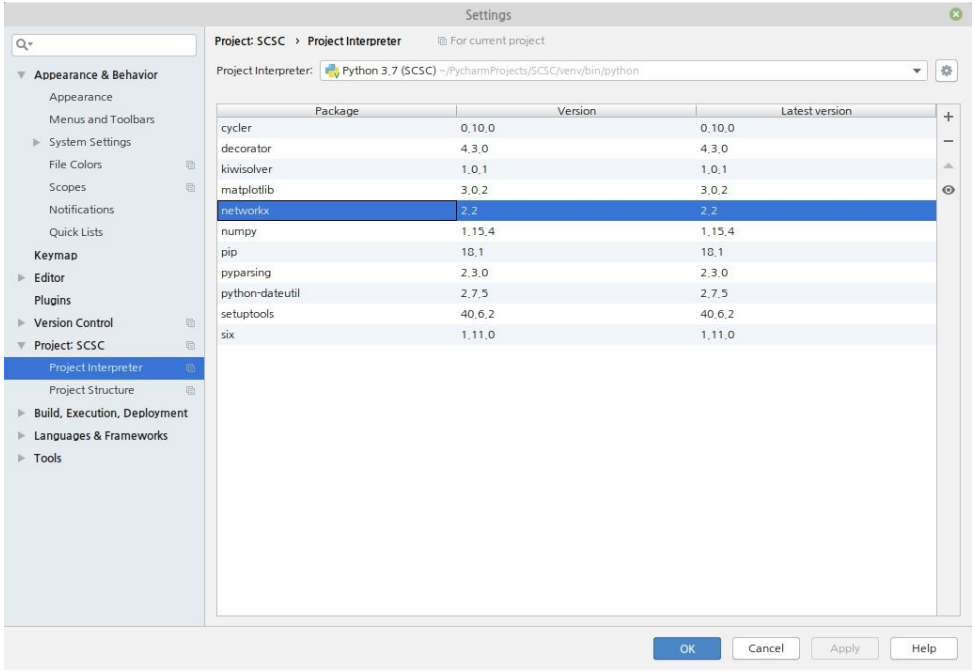



```
62 print('Start A')
63 print(dijkstra(WDGRAPH02, 'a'))
```

```
harny@LuHa-X1-Yoga ~/Downloads $ python3 dijkstra_wiki2.py
Start A
({'c': 10, 'h': 19, 'b': 8, 'g': 16, 'd': 9, 'a': 0, 'f': 11, 'e': 12}, {'c': 'd
', 'h': 'f', 'b': 'a', 'g': 'e', 'd': 'a', 'a': None, 'f': 'a', 'e': 'c'}, {'d':
 {'c': 1}, 'c': {'e': 2}, 'a': {'d': 9, 'f': 11, 'b': 8}, 'f': {'h': 8}, 'e': {'
g': 4}})
```

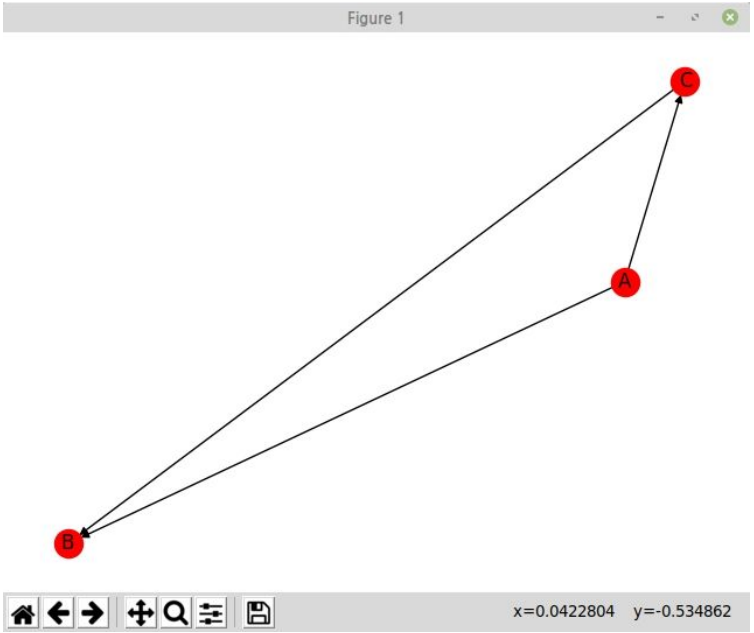
Graph visualization: networkx

- Python 3에서 Graph를 다루고 시각화하는데에 많은 도움을 주는 도구
- matplotlib 설치한 것과 같은 방법으로 networkx 검색하여 설치



networkx simple example

- Graph 시각화

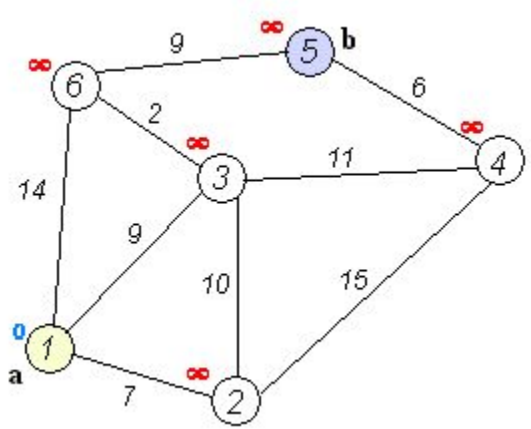



```python
import matplotlib.pyplot as plt
import networkx as nx

def simple_main():
    graph = nx.DiGraph()

    graph.add_node('A')
    graph.add_node('B')
    graph.add_node('C')

    graph.add_edge('A', 'B', weight=3)
    graph.add_edge('A', 'C', weight=5)
    graph.add_edge('C', 'B', weight=1)

    nx.draw(graph, with_labels=True)

    plt.show()
```

Dijkstra’s Algorithm

Dijkstra’s Algorithm

- 짧은 길 찾기 알고리즘 중 가장 유명한 알고리즘
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm



Dijkstra 수도코드

- 출처: wikipedia

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:             // Initialization
6          dist[v] ← INFINITY                  // Unknown distance from source to v
7          prev[v] ← UNDEFINED                 // Previous node in optimal path from source
8          add v to Q                          // All nodes initially in Q (unvisited nodes)
9
10     dist[source] ← 0                        // Distance from source to source
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]    // Source node will be selected first
14         remove u from Q
15
16         for each neighbor v of u:           // where v is still in Q.
17             alt ← dist[u] + length(u, v)
18             if alt < dist[v]:               // A shorter path to v has been found
19                 dist[v] ← alt
20                 prev[v] ← u
21
22     return dist[], prev[]
```

dict형 graph용 Dijkstra

- 총 5개의 FILL_HERE를 수정해야 함
- 수도코드를 읽고 알고리즘에 맞게 고쳐보자!

```python
def dijkstra(wdgraph, start_node):
    qnode = set()
    dist = dict()
    prev = dict()

    # FILL_HERE: initialize variables(multi lines)

    # FILL_HERE: set distance 0 from start node to start node

    while len(qnode) != 0:
        udist = sys.maxsize
        unode = None
        for node in qnode:
            if dist[node] < udist:
                udist = dist[node]
                unode = node
        # FILL_HERE: remove node from queue

        # (Most important!) FILL_HERE: compute new distance(multi lines)

    return dist, prev
```

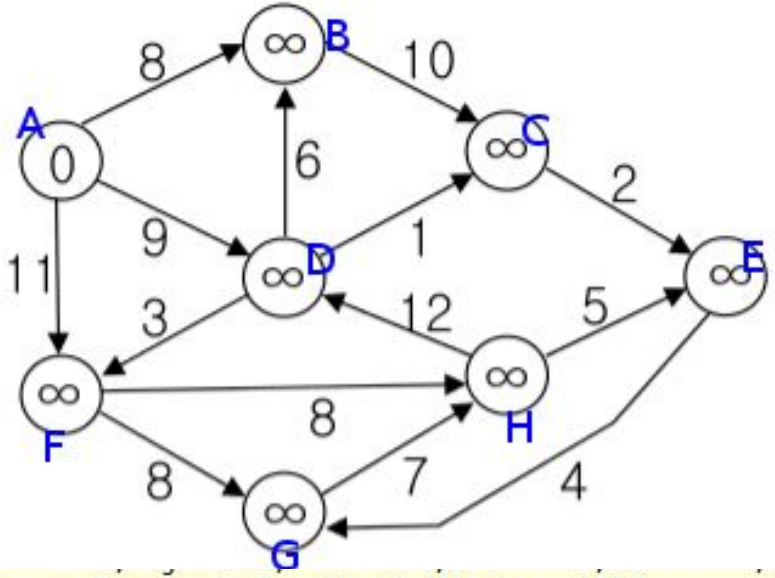
dict형 graph용 Dijkstra

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:
6          dist[v] ← INFINITY
7          prev[v] ← UNDEFINED
8          add v to Q
9
10     dist[source] ← 0
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]
14         remove u from Q
15
16         for each neighbor v of u:
17             alt ← dist[u] + length(u, v)
18             if alt < dist[v]:
19                 dist[v] ← alt
20                 prev[v] ← u
21
22     return dist[], prev[]
```

```
8   def dijkstra(wdgraph, start_node):
9       qnode = set()
10      dist = dict()
11      prev = dict()
12
13      # FILL_HERE: initialize variables(multi lines)
14
15      # FILL_HERE: set distance 0 from start node to start node
16
17      while len(qnode) != 0:
18          udist = sys.maxsize
19          unode = None
20          for node in qnode:
21              if dist[node] < udist:
22                  udist = dist[node]
23                  unode = node
24          # FILL_HERE: remove node from queue
25
26          # (Most important!) FILL_HERE: compute new distance(multi lines)
27
28      return dist, prev
```

Dijkstra 구현 결과

- 최소거리와 연결된 노드를 출력




```
harny@LuHa-X1-Yoga ~/Downloads $ python3 dijkstra_wiki2.py
Start A
({'g': 16, 'd': 9, 'e': 12, 'b': 8, 'a': 0, 'h': 19, 'f': 11, 'c': 10}, {'g': 'e
', 'd': 'a', 'e': 'c', 'b': 'a', 'a': None, 'h': 'f', 'f': 'a', 'c': 'd'})
```

Prim’s Algorithm

MST

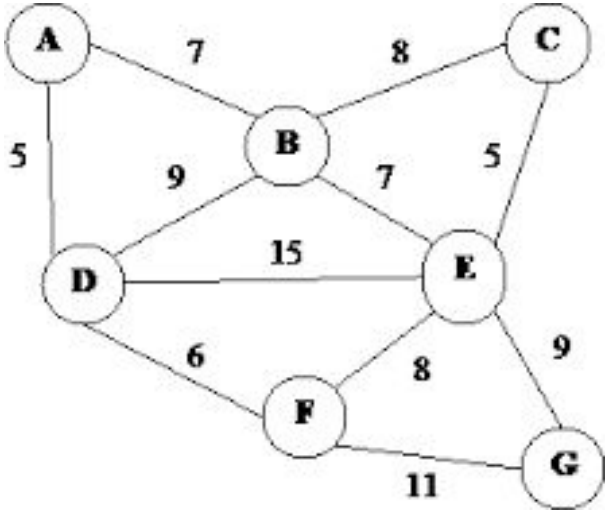
- Minimum Spanning Tree
- 모든 노드가 포함된 최소거리 합 트리!

Prim’s Algorithm

- 가장 기본적인 MST 구하기 알고리즘
  - 방향이 없는 그래프에서 동작!
- https://en.wikipedia.org/wiki/Prim%27s_algorithm

Prim’s Algorithm 수도코드

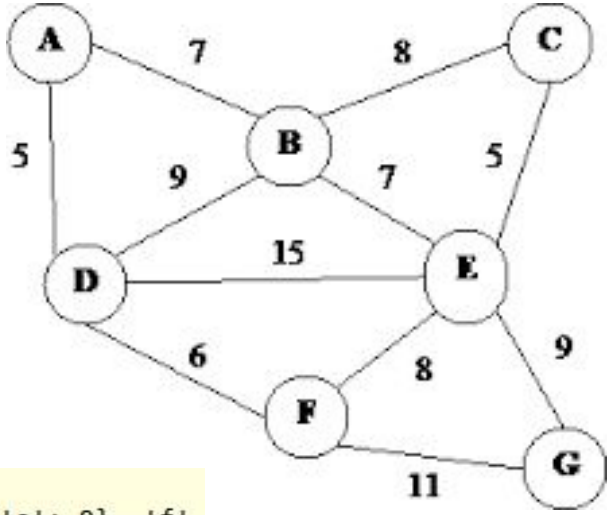
1. dict() - MST, set() - AN, set() - RN 초기화
    a. MST: 결과 - 모든 Key-Value 생성
    b. AN: MST에 추가된 Node Set - 빈 Set
    c. RN: MST에 연결 안 된 Node Set - 모든 Node Set
2. 임의의 노드를 선택 및 AN, RN 추가/삭제



AN : Added Nodes
RN : Remain Nodes

Prim's Algorithm 수도코드

3. RN가 비워질 때 까지 Loop
   a. AN에서 RN까지의 거리중 가장 짧은 거리를 계산
      - Src Node, Dst Node, Dist 모두 반환
   b. MST에 a)에서 반환된 Node 및 Edge 추가
   c. AN, RN 추가/삭제
4. MST 반환




```
69
70 print(prim(WUGRAPH01))
harny@LuHa-X1-Yoga ~/Downloads $ python3 prims_good.py
{'d': {'f': 6, 'a': 5}, 'b': {'e': 7, 'a': 7}, 'c': {'e': 5}, 'g': {'e': 9}, 'f'
: {'d': 6}, 'e': {'b': 7, 'c': 5, 'g': 9}, 'a': {'b': 7, 'd': 5}}
harny@LuHa-X1-Yoga ~/Downloads $
```

dict형 graph용 Prim

- FILL_HERE 4개
- 제공된 search_min() 함수 활용할 것!

```python
def search_min(wugraph, an, rn):
    dist = float('inf')
    for src in an:
        for dst in wugraph[src]:
            if dst in rn:
                if (wugraph[src])[dst] < dist:
                    dist = (wugraph[src])[dst]
                    src_node = src
                    dst_node = dst

    return (src_node, dst_node, dist)
```

```python
def prim(wugraph):
    mst = dict()
    added_node = set()
    remain_node = set()

    for node in wugraph.keys():
        mst[node] = dict()

    # FILL_HERE: initialize remain_node

    import random
    start_node = random.choice(list(wugraph.keys()))

    # FILL_HERE: handle start_node - add to added_node, remove from remain_node

    # (Most important!) FILL_HERE: compute new distance, and handle sets - HINT: while loop

    return mst
```

과제: networkx기반 Dijkstra, Prim

과제! Going Home

- 슬라이드에 있는 그래프 (방향/무방향)에 대하여 Dijkstra, Prim 시각화
- 임의의 그래프 각 알고리즘별 1개씩 시각화
  - Dijkstra: 집으로 돌아가는 길을 찾자!
  - Prim: 내가 우리학교 네트워크 관리자라면?

networkx Graph 시각화 예제 - 생성

- 그래프 생성
  - data를 따로 구분해서 그래프를 만들면 추후 시각화할 때 편리함
  - 메인/서브 그래프 구분하여 시각화 가능

```python
import matplotlib.pyplot as plt
import networkx as nx


def main():
    graph = nx.DiGraph()

    data = {('A', 'B'): 4,
            ('A', 'C'): 2,
            ('B', 'C'): 5,
            ('B', 'D'): 1,
            ('C', 'E'): 3,
            ('E', 'D'): 4,
            ('D', 'F'): 11}

    for key in data.keys():
        u, v = key
        weight = data[key]
        graph.add_edge(u, v, weight=weight)
```

networkx Graph 시각화 예제 - 연산

- 알고리즘 연산
  - 각 알고리즘 연산 결과로 서브 그래프의 에지 정보가 리스트 형태로 나와야 함(path 변수)
  - 연산 결과 그래프와 이외의 그래프를 구분(에지 리스트)

```
21      path = [('A', 'C'),
22              ('C', 'E'),
23              ('E', 'D'),
24              ('D', 'F')]
25      non_path = list(set(graph.edges) - set(path))
```

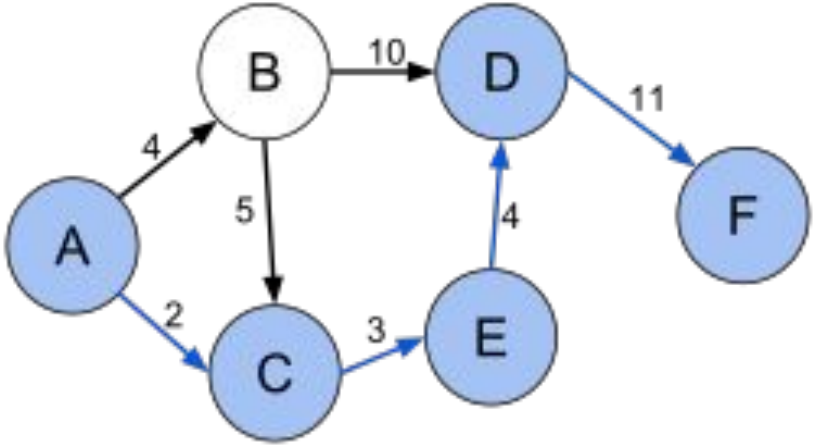
networkx Graph 시각화 예제 - 시각화

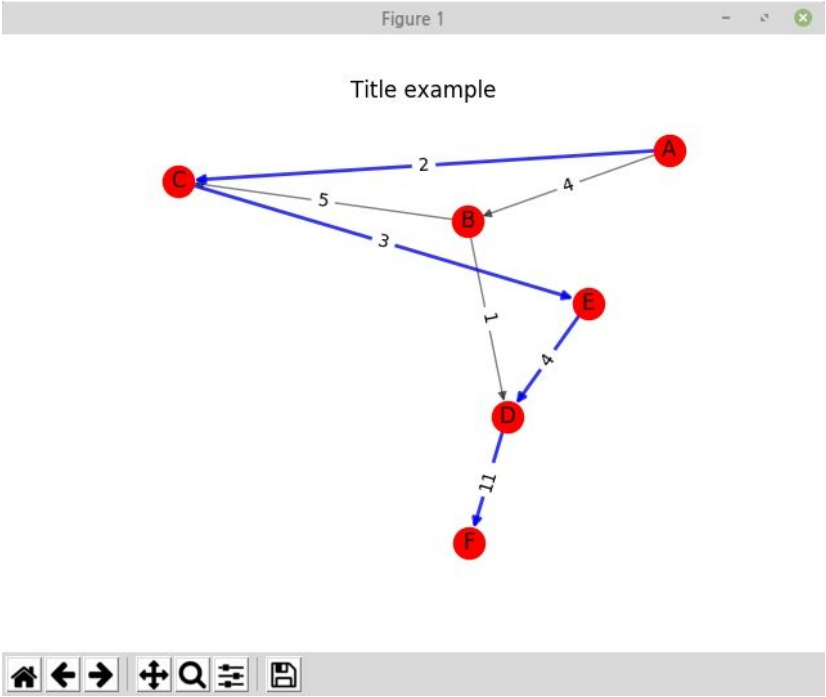
- 연산 결과 시각화
  - 궁금하면 networkx 공식 문서 찾아볼 것
  - 아래와 같은 방법으로 시각화 가능(그대로 복사 붙여넣기로 사용해도 무방!)

```
27      position = nx.spring_layout(graph)
28      # position = nx.circular_layout(graph)
29      nx.draw_networkx_nodes(graph, position)
30      nx.draw_networkx_labels(graph, position)
31      nx.draw_networkx_edges(graph, position, edgelist=non_path, alpha=0.5)
32      nx.draw_networkx_edges(graph, position, edgelist=path, alpha=0.8, edge_color='blue', width=2)
33      nx.draw_networkx_edge_labels(graph, position, edge_labels=data)
34
35      plt.title('Title example')
36      plt.axis('off')
37      plt.show()
```

networks Graph 시각화 예제 결과

- position layout에 따라서 모양이 달라짐, 매 실행시마다도 달라짐





과제 할 때 도움이 될 정보

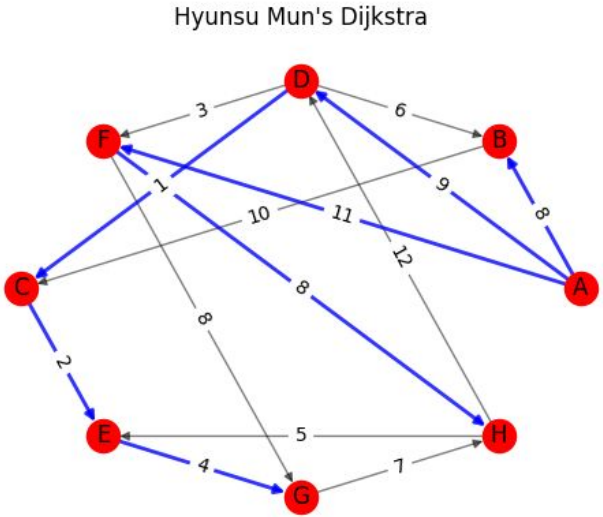
- networkx 의 Graph 클래스는 __getitem__()이 구현되어 있음
  - 따라서 GRAPH[KEY]와 같은 연산이 가능!
- networkx 의 Graph 클래스는 nodes, edges 정보를 멤버 변수로 가지고 있음
  - nodes는 노드 키값의 리스트
  - edges는 (시작 노드 키, 종료 노드 키) 튜플의 리스트
- networkx 의 Graph 클래스에서 edge의 무게를 접근하는 방법(예제와 같이 생성했을 시)
  - GRAPH[시작][끝]['weight']

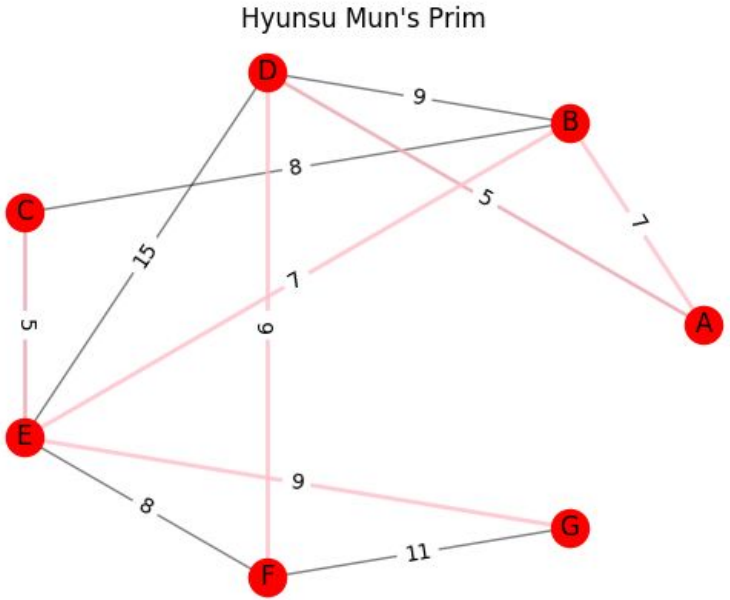
기타 유용한 정보

과제 요약: networkx 기반 Graph 알고리즘 구현

- networkx기반 Graph에 대한 Dijkstra, Prim 알고리즘 구현
  - 지난 시간 class 기반 그래프 탐색을 완료하였어야 해결하기 쉬움
  - 주어진 그래프(각 알고리즘별 1개)를 생성하고 알고리즘 수행 및 시각화
- 각 알고리즘에 대하여 임의의 그래프 1개 이상 실험(보고서에 설명)
  - 지나치게 단순한 그래프는 감점 대상

조교가 한 것


Hyunsu Mun's Dijkstra
D
F
B
C
A
E
H
G
3
6
1
9
8
10
11
12
8
8
2
5
4
7


Hyunsu Mun's Prim
D
B
C
A
E
F
G
9
8
5
7
15
7
5
6
9
8
11

실습 숙제 제출

- 숙제 제출 기한: 2018. 11. 28. 23:59:59
  - 실습 전 날
- 파일 제목: AL_학번_이름_11.zip
  - 파일 제목 다를 시 채점 안 합니다.
  - .egg 안 됨!

실습 숙제 제출할 것

- 2가지 파일을 제출
- AL_학번_이름_숙제번호.zip
  - 소스코드
    - Pycharm을 사용했을 경우 Project 디렉터리에 .idea, venv 같은 디렉터리는 제외
    - Jupyter + IPython을 사용했을 경우 ‘File - Download as’ 에서 .py 다운로드 가능
  - AL_학번_이름_숙제번호.pdf
    - 보고서는 무조건 .pdf
    - .hwp, .doc 등 채점 안 함

실습 보고서에 들어가야할 것

- 목표(할 일)
- 과제를 해결하는 방법
  - 알아야 할 것
- 과제를 해결한 방법
  - 주요 소스코드: 굳이 소스코드 전체를 붙일 필요는 없음
- 결과화면
  - 결과화면 설명(해석), 테스트코드 통과
- 보고서는 기본적으로 '내가 숙제를 했음'을 보이는 것
  - 지나치게 대충 작성하면 의심하게 됨

출석부 및 실습 점수가 궁금하다면?

- 출석부 및 실습 채점표
  - 수업 시작 후 30분까지 지각, 이후 결석
  - 실습 딜레이 1일당: -2점
    - 딜레이 2일까지: -2
    - 이후 -1씩 추가
- 튜터의 테스트 결과

질문이 생기면?

- 이름: 문현수
- 전공: 통신및보안
- 과정: 석박사통합과정 8학기
- 연구실: 데이터네트워크연구실(공5633)
- 메일: munhyunsu@cs-cnu.org
- 알고리즘은 함께 해결해가는 과목이므로 과감하게 연락
- 이메일로 처리가 안 되는 급한일: 문자/전화 등