파이썬으로 배우는 알고리즘 기초

Chap 4. 탐욕 알고리즘

4.2 최단 경로와 다익스트라 알고리즘

4.2 최단 경로와 다익스트라 알고리즘

- **최단 경로 문제**: Revisited
  - 모든 정점의 쌍에 대한 최단 경로 구하기
    - 플로이드 알고리즘: 동적 계획법
  - 단일 정점에서 모든 다른 정점으로의 최단 경로 구하기
    - 다익스트라 알고리즘: 탐욕법

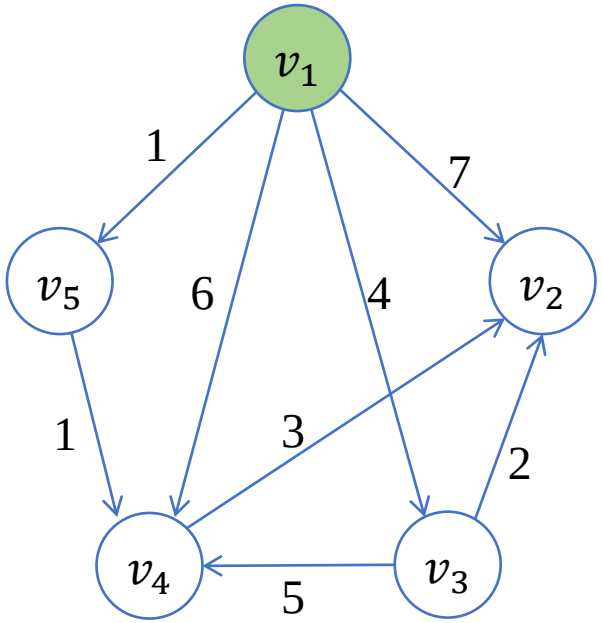
4.2 최단 경로와 다익스트라 알고리즘

- **다익스트라 알고리즘:**
  - 최소비용 신장트리 문제의 프림 알고리즘과 유사

$Y = \{v_1\}$;
$F = \emptyset$;
while (답을 구하지 못했음):
    Y에 속한 정점만 중간에 거쳐 가는 정점으로 하여
        v_1 에서 최단경로를 가진 정점 v를 $V - Y$에서 선택한다.
    새로운 정점 v를 Y에 추가한다.
    (최단경로 상에서) v로 가는 간선을 F에 추가한다.
    if $(Y == V)$
        답을 구했음.

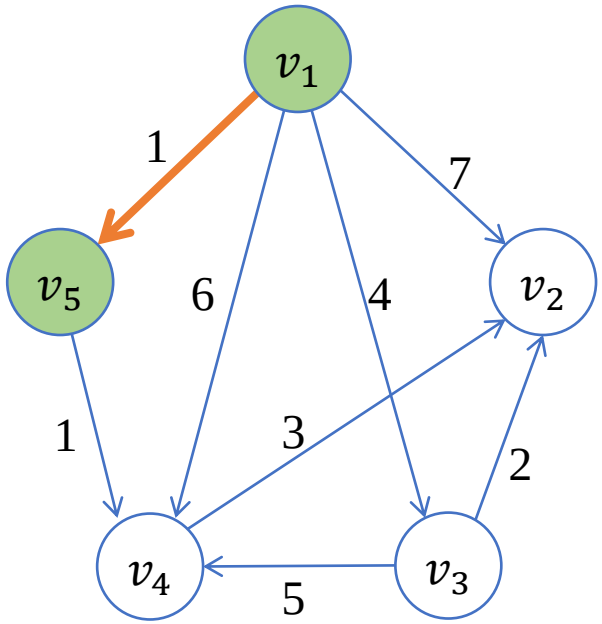
4.2 최단 경로와 다익스트라 알고리즘

- Compute shortest paths from v_1.

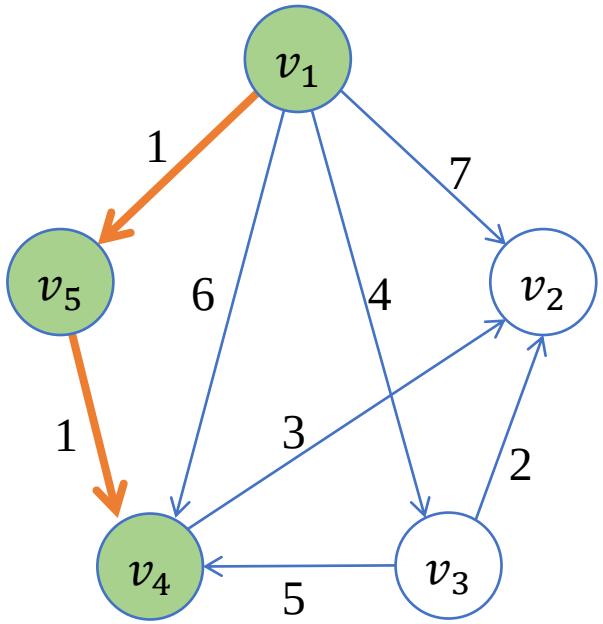


4.2 최단 경로와 다익스트라 알고리즘

1. Vertex v_5 is selected because it is nearest to v_1

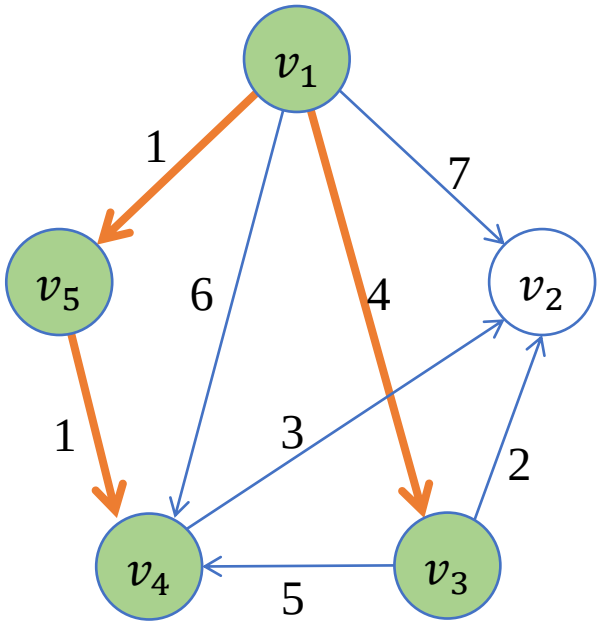


2. Vertex v_4 is selected because it has the shortest path from v_1 using only vertices in $\{v_5\}$ as intermediate

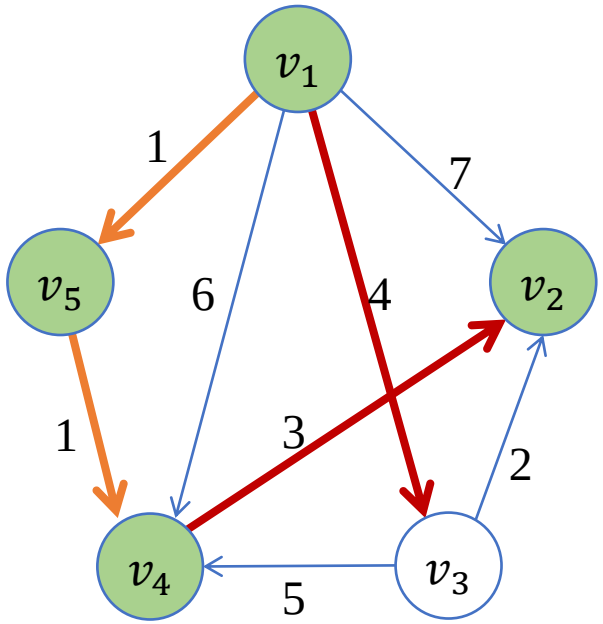


4.2 최단 경로와 다익스트라 알고리즘

3. Vertex v_3 is selected because it has the shortest path from v_1 using only vertices in $\{v_4, v_5\}$ as intermediates



4. The shortest path from v_1 to v_2 is $[v_1, v_5, v_4, v_2]$



4.2 최단 경로와 다익스트라 알고리즘

- `W[i][j]`: 그래프 G의 인접행렬
- `touch[i]`: Y에 속한 정점들만 중간에 거치도록 하여 v_1에서 v_i로 가는 현재 최단경로 상의 마지막 간선을 (v, v_i) 라고 할 때, Y에 속한 정점 v의 인덱스
- `length[i]`: Y에 속한 정점들만 중간에 거치도록 하여 v_1에서 v_i로 가는 현재 최단 경로의 길이

W	1	2	3	4	5
1	0	7	4	6	1
2	∞	0	∞	∞	∞
3	∞	2	0	5	∞
4	∞	3	∞	0	∞
5	∞	∞	∞	1	0

i	1	2	3	4	5
touch[i]		1	1	1	1
length[i]		7	4	6	1
touch[i]		1	1	5	1
length[i]		7	4	2	-1
touch[i]					
length[i]					
touch[i]					
length[i]					
touch[i]					
length[i]					

Algorithm 4.3: Dijkstra's Algorithm

```
def dijkstra (W):
    n = len(W) - 1
    F = []
    touch = [-1] * (n + 1)
    length = [-1] * (n + 1)
    for i in range(2, n + 1):
        touch[i] = 1
        length[i] = W[1][i]
```

Algorithm 4.3: Dijkstra's Algorithm

```
        for _ in range(n - 1):
            minValue = INF
            for i in range(2, n + 1):
                if (0 <= length[i] and length[i] < minValue):
                    minValue = length[i]
                    vnear = i
            edge = (touch[vnear], vnear, W[touch[vnear]][vnear])
            F.append(edge)
            for i in range(2, n + 1):
                if (length[i] > length[vnear] + W[vnear][i]):
                    length[i] = length[vnear] + W[vnear][i]
                    touch[i] = vnear
            length[vnear] = -1
        return F
```

Algorithm 4.3: Dijkstra's Algorithm

```
def length (F):
    total = 0
    for e in F:
        total += e[2]
    return total

def print_tl (F, touch, length):
    print('F = ', end = '')
    print(F)
    print('   touch: ', end = '')
    print(touch)
    print('  length: ', end = '')
    print(length)
```

4.2 최단 경로와 다익스트라 알고리즘


```
INF = 999
W = [
    [-1, -1, -1, -1, -1, -1],
    [-1, 0, 7, 4, 6, 1],
    [-1, INF, 0, INF, INF, INF],
    [-1, INF, 2, 0, 5, INF],
    [-1, INF, 3, INF, 0, INF],
    [-1, INF, INF, INF, 1, 0],
]

F = dijkstra(W)
for i in range(len(F)):
    print(F[i])

print("Shortest Path Length is", length(F))
```

4.2 최단 경로와 다익스트라 알고리즘


```
F = []
    touch: [-1, -1, 1, 1, 1, 1]
   length: [-1, -1, 7, 4, 6, 1]
F = [(1, 5, 1)]
    touch: [-1, -1, 1, 1, 5, 1]
   length: [-1, -1, 7, 4, 2, -1]
F = [(1, 5, 1), (5, 4, 1)]
    touch: [-1, -1, 4, 1, 5, 1]
   length: [-1, -1, 5, 4, -1, -1]
F = [(1, 5, 1), (5, 4, 1), (1, 3, 4)]
    touch: [-1, -1, 4, 1, 5, 1]
   length: [-1, -1, 5, -1, -1, -1]
F = [(1, 5, 1), (5, 4, 1), (1, 3, 4), (4, 2, 3)]
    touch: [-1, -1, 4, 1, 5, 1]
   length: [-1, -1, -1, -1, -1, -1]
(1, 5, 1)
(5, 4, 1)
(1, 3, 4)
(4, 2, 3)
Shortest Path Length is 9
```

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 **구독**과 **좋아요**는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: **구글 드라이브**에서 다운로드
  (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t