


## Day 5 - Greed Algorithm

트리(DFS, BFS) / 스택 / 순회방법 => 1문제

### 그리드 알고리즘

- 탐욕 알고리즘
  - 현재 시점에서 지금 선택하는 것이 가장 좋은 것이라고 생각
  - 현재 시점에서 가장 좋은 것만 고르는 방법
  - 내가 선택한게 가장 좋은 방법 (내가 현재 보이는게 가장 좋다고 생각하는 알고리즘)
  - 최적은 장담할 수 없다.

예시 문제:


-  Untitled0.ipynb
- 동전 교환
  - 500원, 100원, 50원, 10원 동전을 가지고 있다.
  - 가장 적은 동전의 개수를 교환할 수 있는 방법:
    - 가장 먼저 큰 화폐 단위부터 거슬러 준다.
    - N 거슬러 주어야 하는 화폐  $\Rightarrow$  1260
      1. 500원 2개
      2. 100원 2개
      3. 10원 6개

### 그리드 알고리즘

- 플로이워셜
- 다익스트라 알고리즘 (코딩 문제에 자주 등장)
- 벨만포드 알고리즘

그리드와 백트래킹의 관계

그리드 알고리즘 예시 문제:

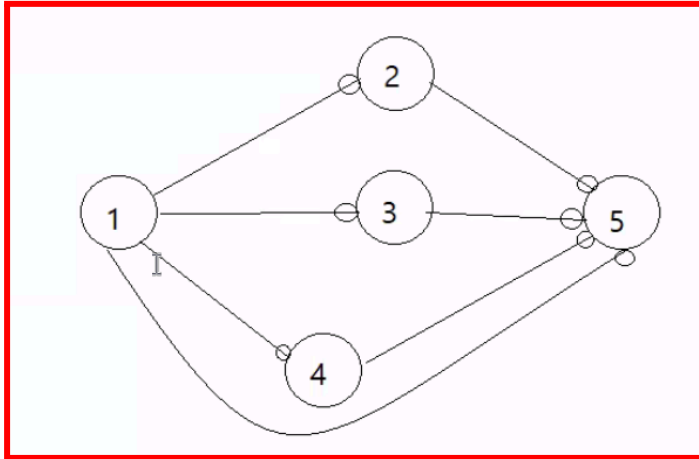
 Greed\_Example.ipynb

## 다익스트라 알고리즘 & 벨만 포드 알고리즘

- 필수적으로 나오는 패턴
- 최단 거리 경로 구하기

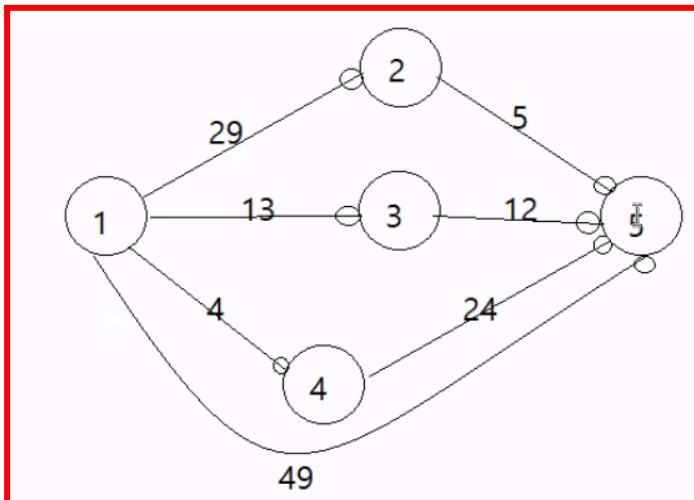
1. 가중치 값이 존재하는 경우 → 가중치 값의 합이 최소
2. 가중치 값이 존재하지 않는 경우 → 간선의 개수가 가장 적은 경로

### 예시1) 가중치 X



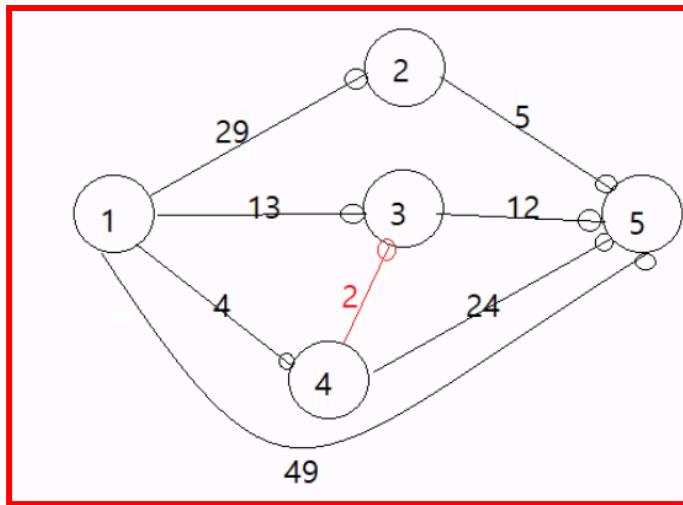
- 가중치 값이 없기때문에 최단거리경로는 1→5 하나밖에 존재하지 않는다.

### 예시2) 가중치 O



1. 1→2→5      가중치값: 34
2. 1→3→5      **가중치값: 25 (최단거리)**
3. 1→4→5      가중치값: 28
4. 1→5          가중치값: 49

번외)



1. 1→2→5      가중치값: 34
2. 1→3→5      가중치값: 25
3. 1→4→5      가중치값: 28
4. 1→5          가중치값: 49
5. 1→4→3→5   가중치값: 18 (최단거리)

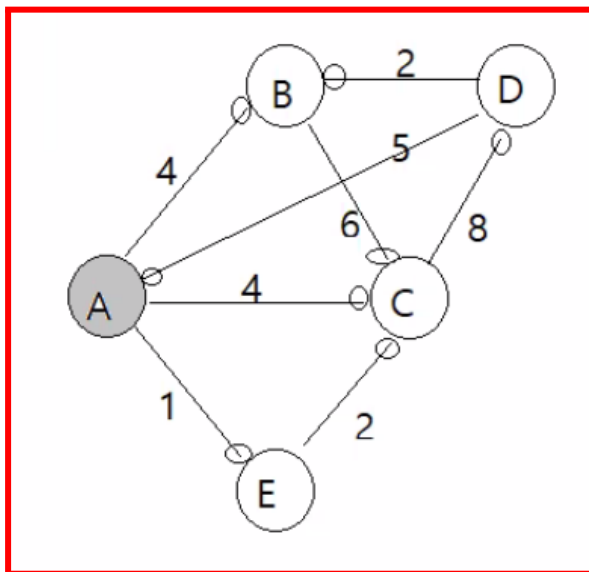
#### 그리드 알고리즘의 단점

- 결과값을 신뢰할 수 없다.

따라서 신뢰하게 만들어주기 위해서 이 두개를 만들었다.

- **다익스트라 알고리즘**
  - 가중치값이 양수일때 사용
- **벨만 포드 알고리즘**
  - 가중치값이 음수일때 사용

## 다익스트라 알고리즘을 이용하여 구현하는 원리



### 1. 시작 노드의 최소 비용은 항상 0이다.

- 그 이외의 최소 비용은 INF 이다.
- 직전 노드는 그 이전 노드로 초기화한다.
- 최소 비용을 구하기 전에 먼저 방문한 노드를 답을 리스트를 선언(빈 리스트)

### 1. 시작노드는 최소비용이 0, 직전노드는 자기자신, 나머지는 INF로 설정한다.

노드	A	B	C	D	E
최소비용	0	INF	INF	INF	INF
직전노드	A	INF	INF	INF	INF

### 2. 시작노드를 기준으로 갈 수 있는 노드의 최소비용과 직전노드를 업데이트한다.

노드	A	B	C	D	E
최소비용	0	4	4	INF	1
직전노드	A	A	A	INF	A

- A → B      가중치: 4
- A → C      가중치: 4
- A → D      가중치: INF (직접 못감)
- A → E      가중치: 1

### 3. A → C 로 가는 방법

노드	A	B	C	D	E
최소비용	0	4	3	INF	1
직전노드	A	A	E	INF	A

1. A → B → C   가중치값 10
2. A → C       가중치값 4
3. **A → E → C   가중치값 3**

### 4. A → D 로 가는 방법

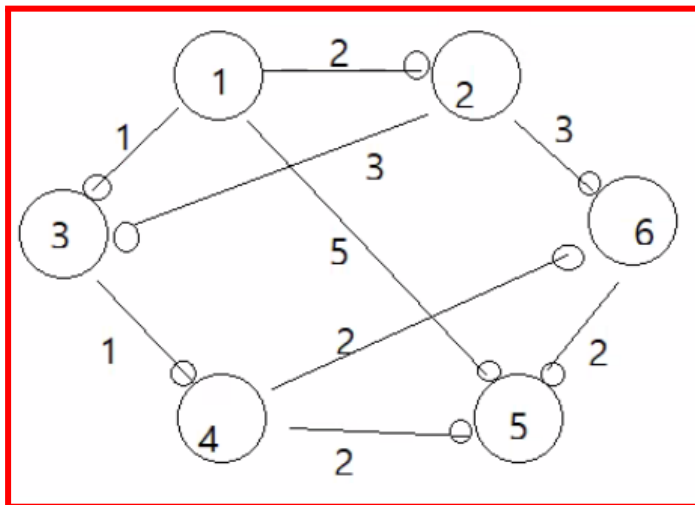
노드	A	B	C	D	E
최소비용	0	4	3	11	1
직전노드	A	A	E	C	A

1. A → C → D       가중치값 12
2. **A → E → C → D   가중치값 11**

최소 거리

**A → E → C → D**

Ex1)



1. 시작노드는 최소비용이 0, 직전노드는 자기자신, 나머지는 INF로 설정한다.

노드	1	2	3	4	5	6
최소비용	0	INF	INF	INF	INF	INF
직전노드	1	INF	INF	INF	INF	INF

2. 시작노드를 기준으로 갈 수 있는 노드의 최소비용과 직전노드를 업데이트한다.

노드	1	2	3	4	5	6
최소비용	0	2	1	INF	INF	INF
직전노드	1	1	1	INF	INF	INF

1 → 4 로 가는 방법

노드	1	2	3	4	5	6
최소비용	0	2	1	2	INF	INF
직전노드	1	1	1	3	INF	INF

1. 1→3→4      가중치: 2

2. 1→2→3→4      가중치: 6

## 1→5 로 가는 방법

노드	1	2	3	4	5	6
최소비용	0	2	1	2	4	INF
직전노드	1	1	1	3	4	INF

1.  $1 \rightarrow 5$                       가중치 5
2.  $1 \rightarrow 3 \rightarrow 4 \rightarrow 5$           가중치 4
3.  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$       가중치 8
4.  $1 \rightarrow 2 \rightarrow 6 \rightarrow 5$             가중치 7

## 1→6 로 가는 방법

노드	1	2	3	4	5	6
최소비용	0	2	1	2	4	4
직전노드	1	1	1	3	4	4

1.  $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$       가중치 4

## 다익스트라 알고리즘 예시

co 다익스트라 알고리즘문제.ipynb

### 벨만포드 알고리즘:

- 가중치 값이 음의 숫자라면 무조건 벨만 포드 알고리즘으로 구현
- 다익스트라 알고리즘은 가중치가 음수가 존재하지 않음으로 최소값을 구할 때 모든 간선을 다시 확인 할 필요가 없다.
- 벨만 포드 알고리즘은 음의 가중치 값을 가질 수 있음으로 매 단계 마다 모든 간선의 가중치를 다시 확인하여 최소 비용을 갱신하여야 한다.

### 프로그래밍 문법적인 형식:

최소비용 == 최소비용 + 간선

- 갱신을 하지 않는다.

최소비용 > 최소비용 + 간선

- 갱신을 한다.

최소비용 < 최소비용 + 간선

- 갱신하지 않는다.