# 그리디 알고리즘

# 현재 상황에서

# 당장 좋은 것만 고르는 알고리즘

# 탐욕법 # 탐욕 알고리즘

# 만원 - 2650원(샌드위치) = 7350원(거스름돈)




# 7350원(거스름돈) - 500원 = 6850원

# 7350원(거스름돈) - 500원 - 500원 = 6350원



• • •

# 7350원(거스름돈) - 500*14원= 350원


X 14

# 7350원(거스름돈) - 500*14원 - 300원= 50원


X 14



# 7350원(거스름돈) - 500*14원 - 300원 - 50원 = 0원


X 14




# 현재 상황에서

# 당장 좋은 것만 고르는 알고리즘

# 탐욕법 # 탐욕 알고리즘

# 그리디 알고리즘은 최적의 해를 보장할 수 X

# 따라서 최적의 해를 보장하는지 확인이 필요!!!

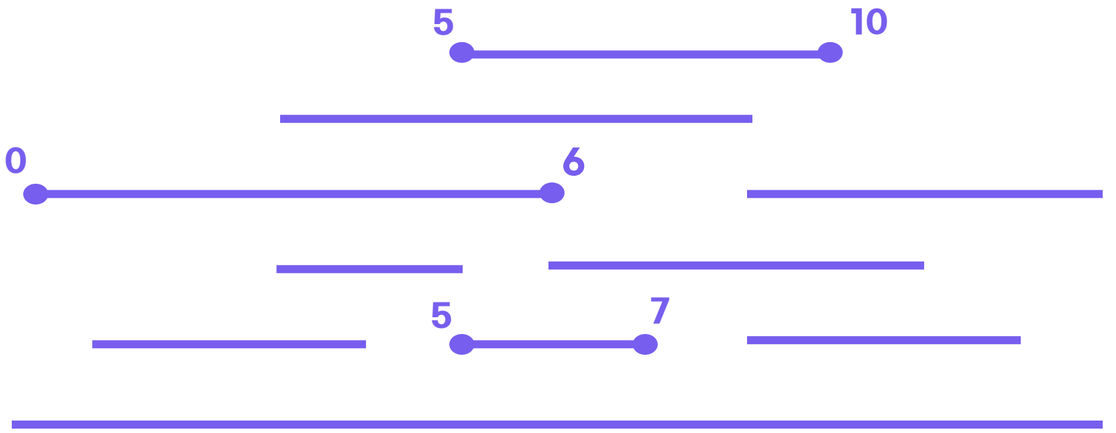
# 1. 탐욕 선택 속성

**답의 모든 부분을 고려하지 않고 탐욕적으로만 선택하더라도 최적해를 구할 수 있다는 속성**

# 2. 최적 부분 구조

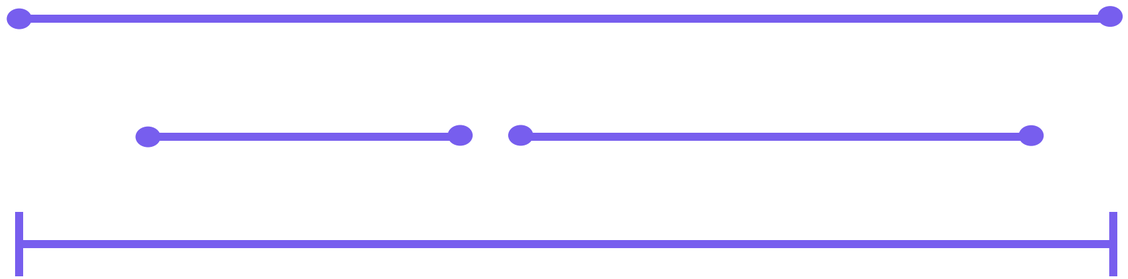
**부분 문제의 최적해에서 전체 문제의 최적해를 만들 수 있다.**

# 백준 1931번



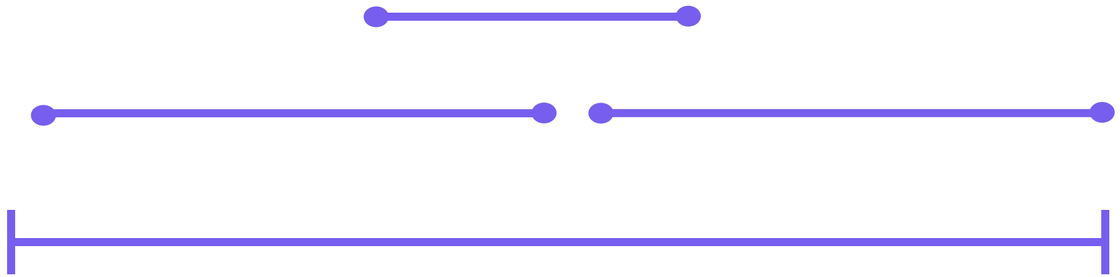
# 빨리 시작하는 회의부터 선택한다면?

## 반례)

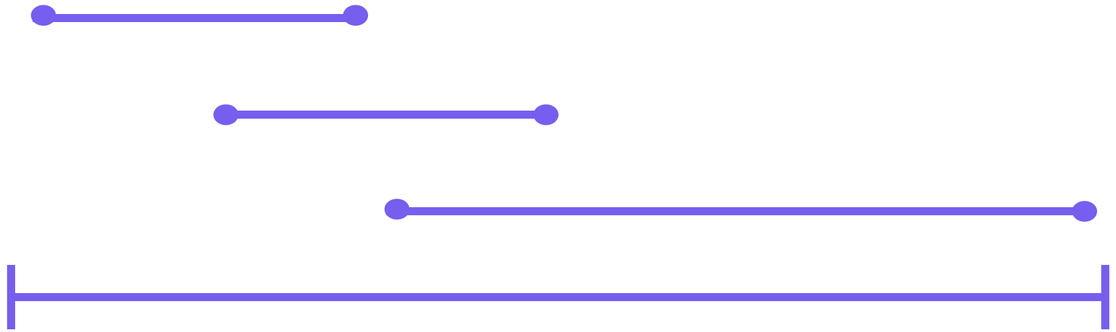


# 회의시간이 짧은 회의부터 선택한다면?

## 반례)



# 가장 빨리 끝나는 회의부터 선택한다면?



# 귀류법

명제에서 결론을 부정한다음,이것의 모순을 이끌어내

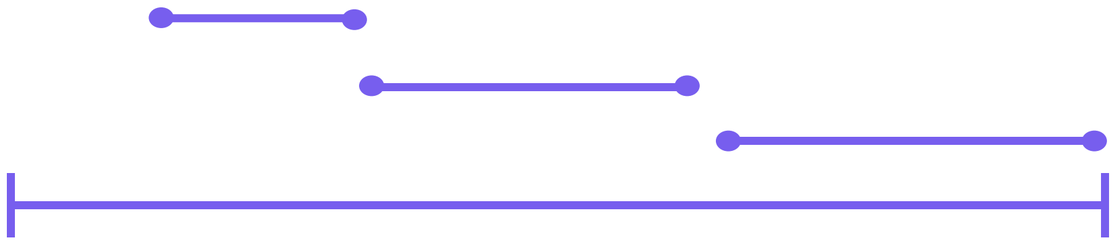
원래 명제가 맞다는 것을 보이는 증명 방법

가장 빨리 끝나는 회의부터 선택한다면
최적의 해가 반드시 존재한다.

가장 빨리 끝나는 회의부터 선택한다면
**최적의 해가 존재하지 않는다.**

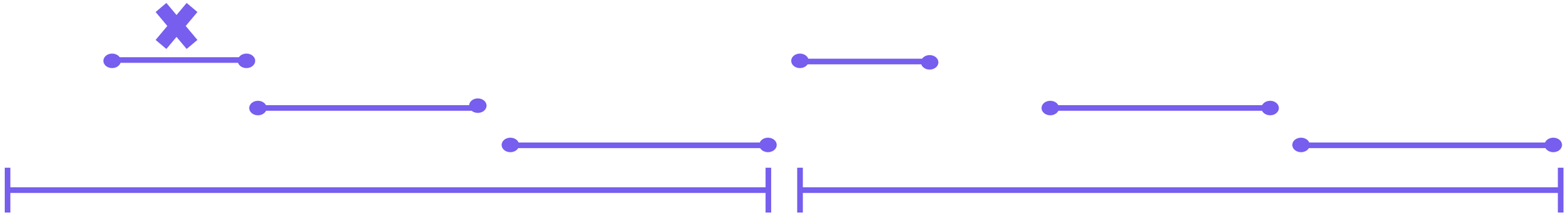
**가장 빨리 끝나는 회의부터 선택한다면**
**최적의 해가 존재하지 않는다.**

**가장 빨리 끝나는 회의를 포함하지 않는 최적해가 있다고 가정**



**가장 빨리 끝나는 회의부터 선택한다면**
**최적의 해가 존재하지 않는다.**

**가장 빨리 끝나는 회의를 포함하지 않는 최적해가 있다고 가정**



# 1. 탐욕 선택 속성

답의 모든 부분을 고려하지 않고 **탐욕적으로만 선택하더라도 최적해를 구할 수 있다는 속성**

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
    int n, i;
    cin>>n;
    vector<pair<int, int>> meeting(n);
    for(i=0; i<n; i++)
        cin>>meeting[i].second>>meeting[i].first;

    sort(meeting.begin(), meeting.end());

    int BeginTime, EndTime, earlist=0, selected=0;
    for(i=0; i<n; i++) {
        BeginTime=meeting[i].second;
        EndTime=meeting[i].first;
        if(earlist<=BeginTime) {
            earlist=EndTime;
            selected++;
        }
    }
    cout<<selected;
    return 0;
}
```

O(nlogn)

회의 시간을 입력 받는다.

입력 받은 회의 시간들을
종료 시간을 기준으로 정렬한다.

for문으로 회의 시간을 하나씩 방문하면서
선택된 회의의 종료시간이 방문한 회의
시작시간보다 적다면, 선택 수를 늘린다.


## 타 알고리즘 보다

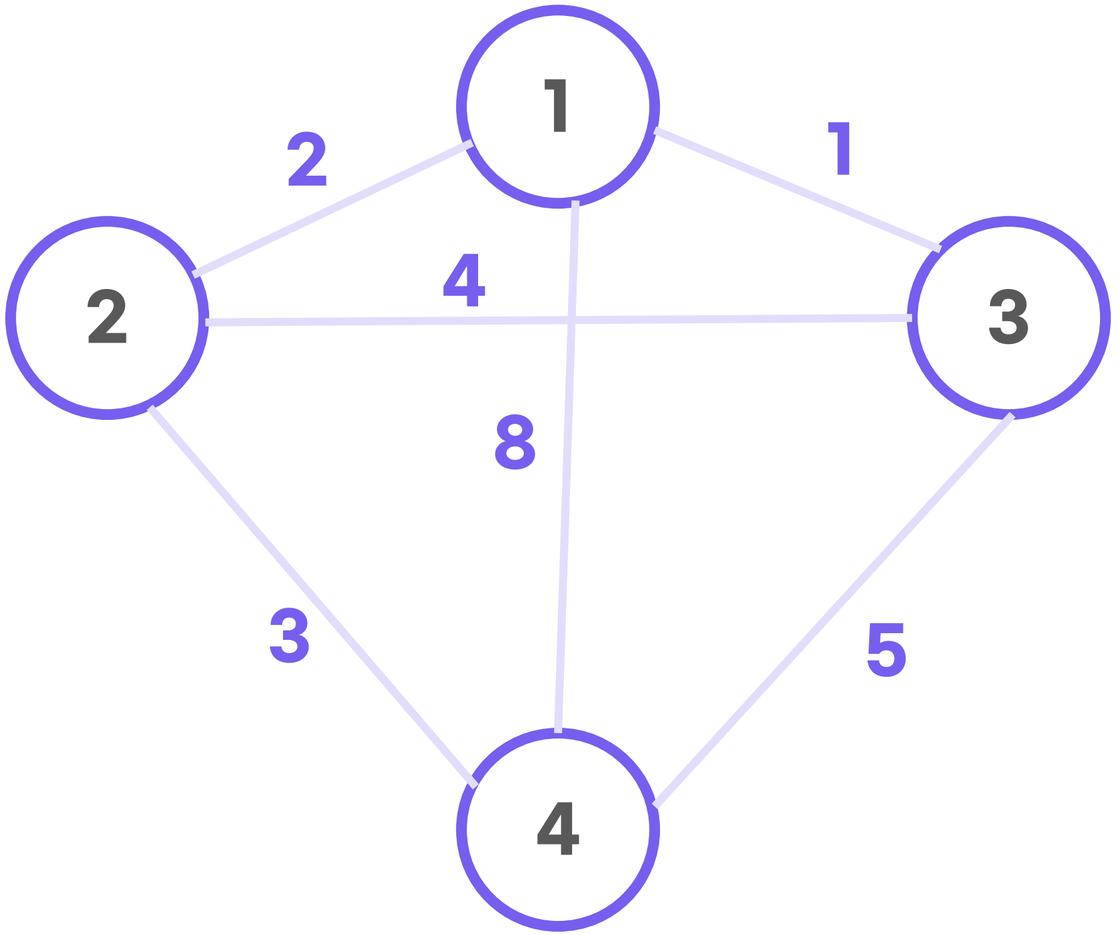
## 최적의 해에 근사한 값을 빠르게 구할 수 있다.

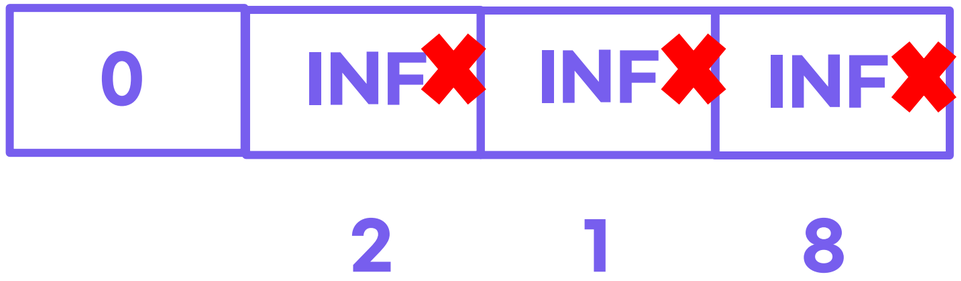
## < 다익스트라 알고리즘 >

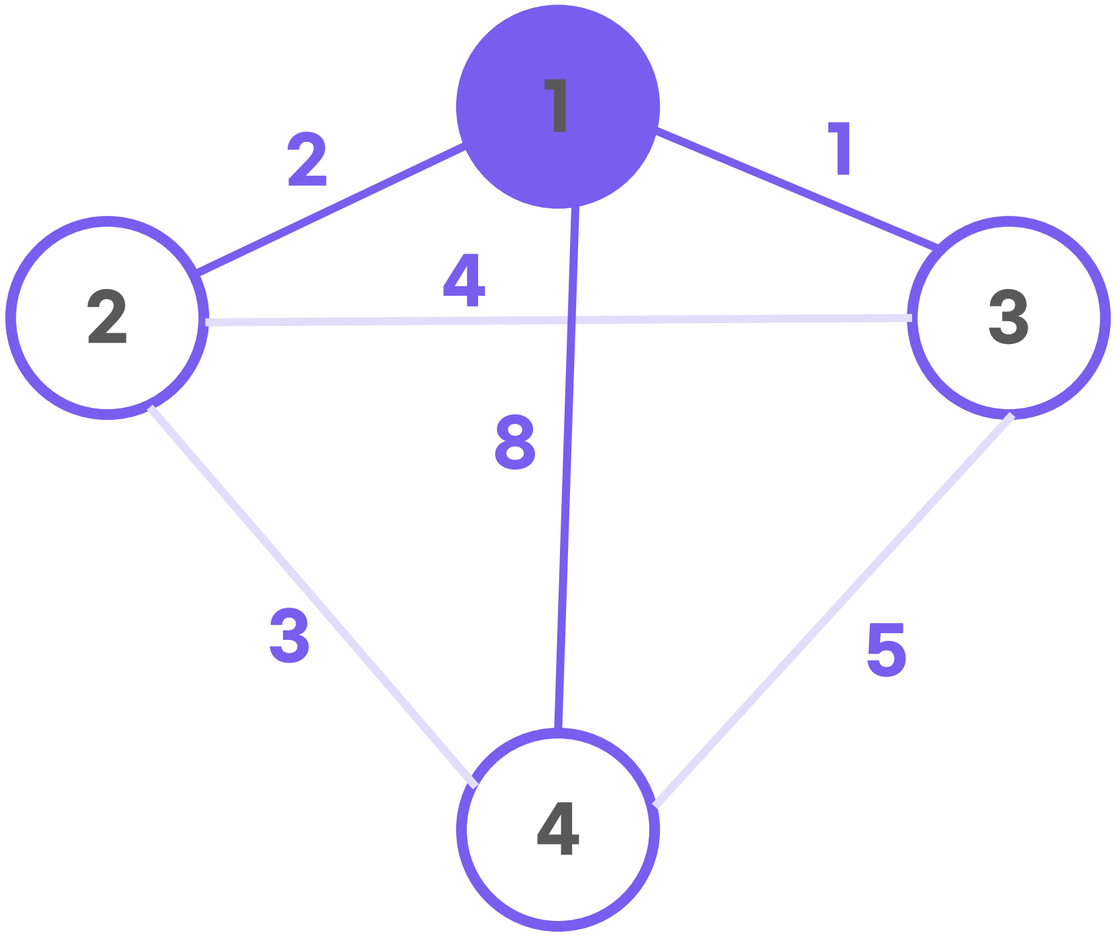
특정 하나의 정점에서

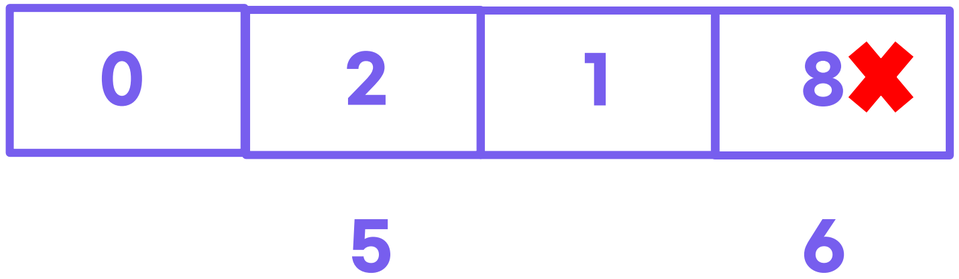
모든 정점으로 가는 최단 경로를 탐색하는 알고리즘

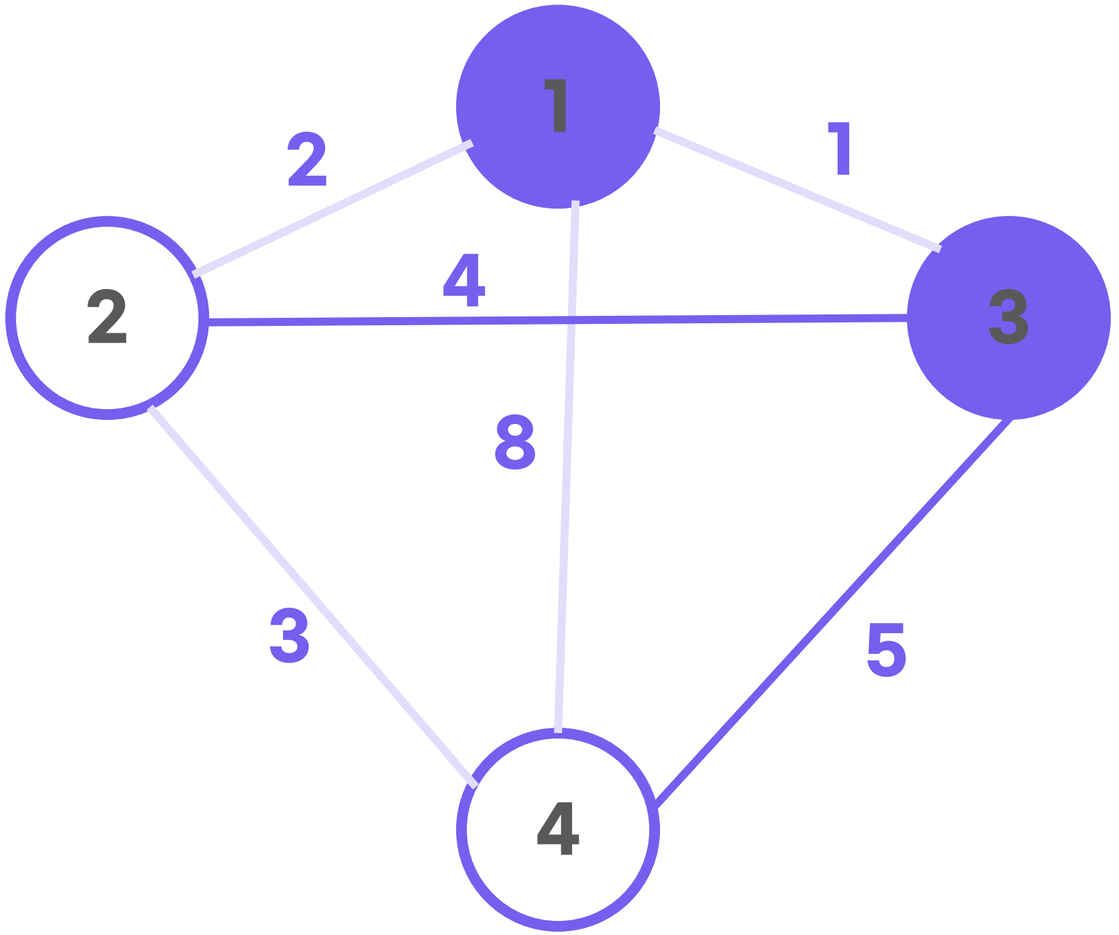
| 0 | INF | INF | INF |
|---|---|---|---|

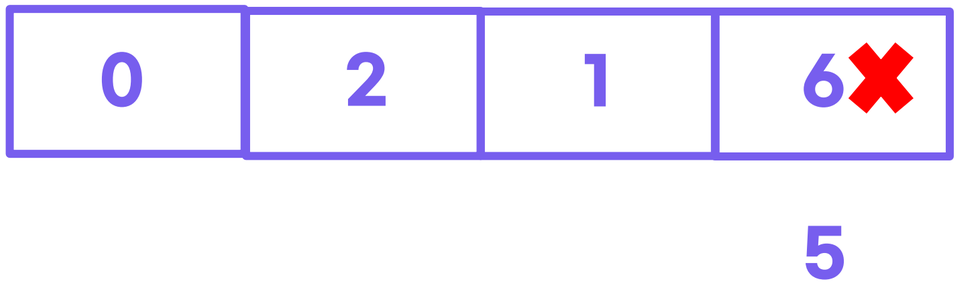


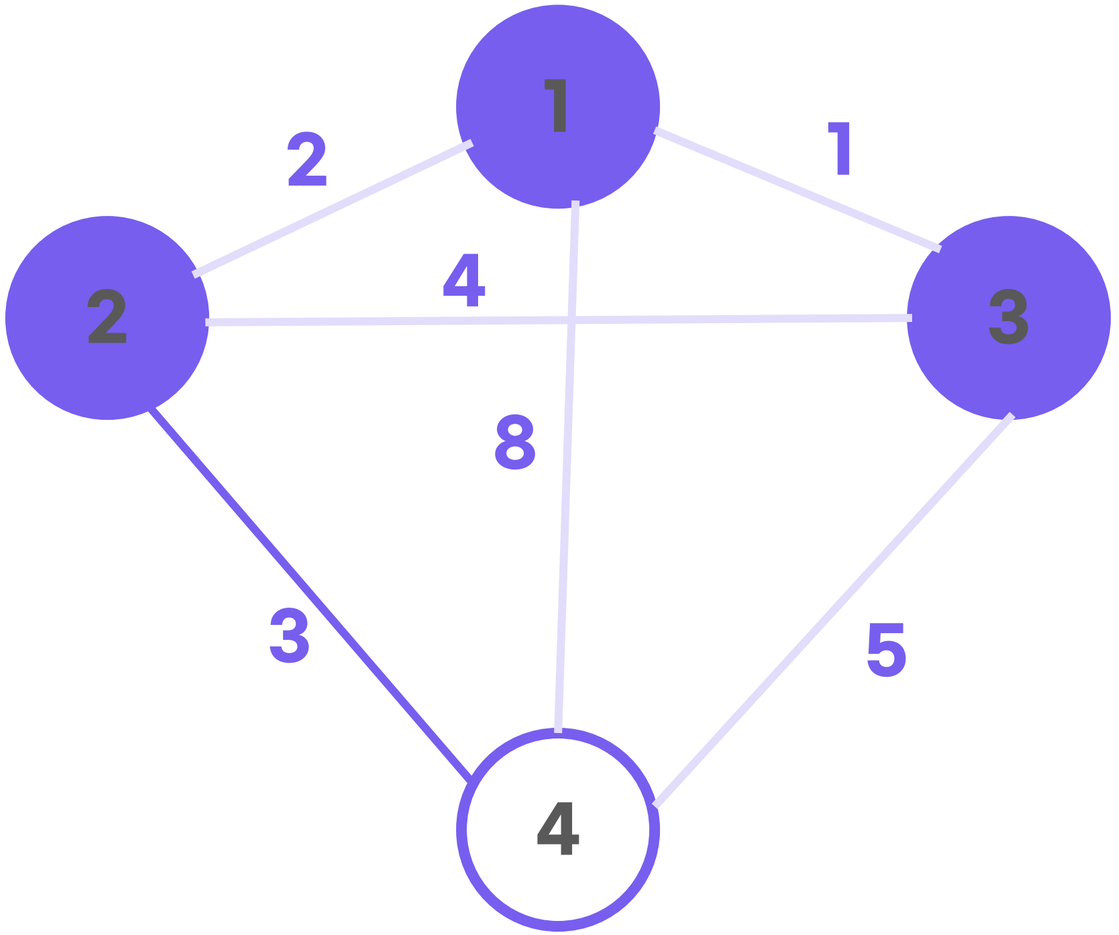

0
INF
INF
INF
2
1
8

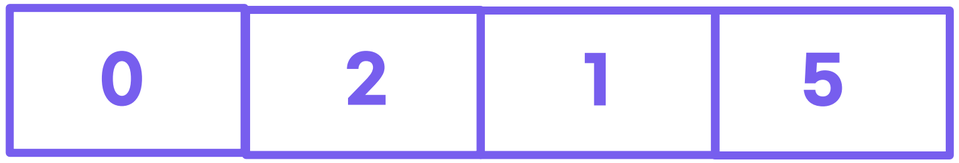

1
2
3
4
2
1
4
8
3
5

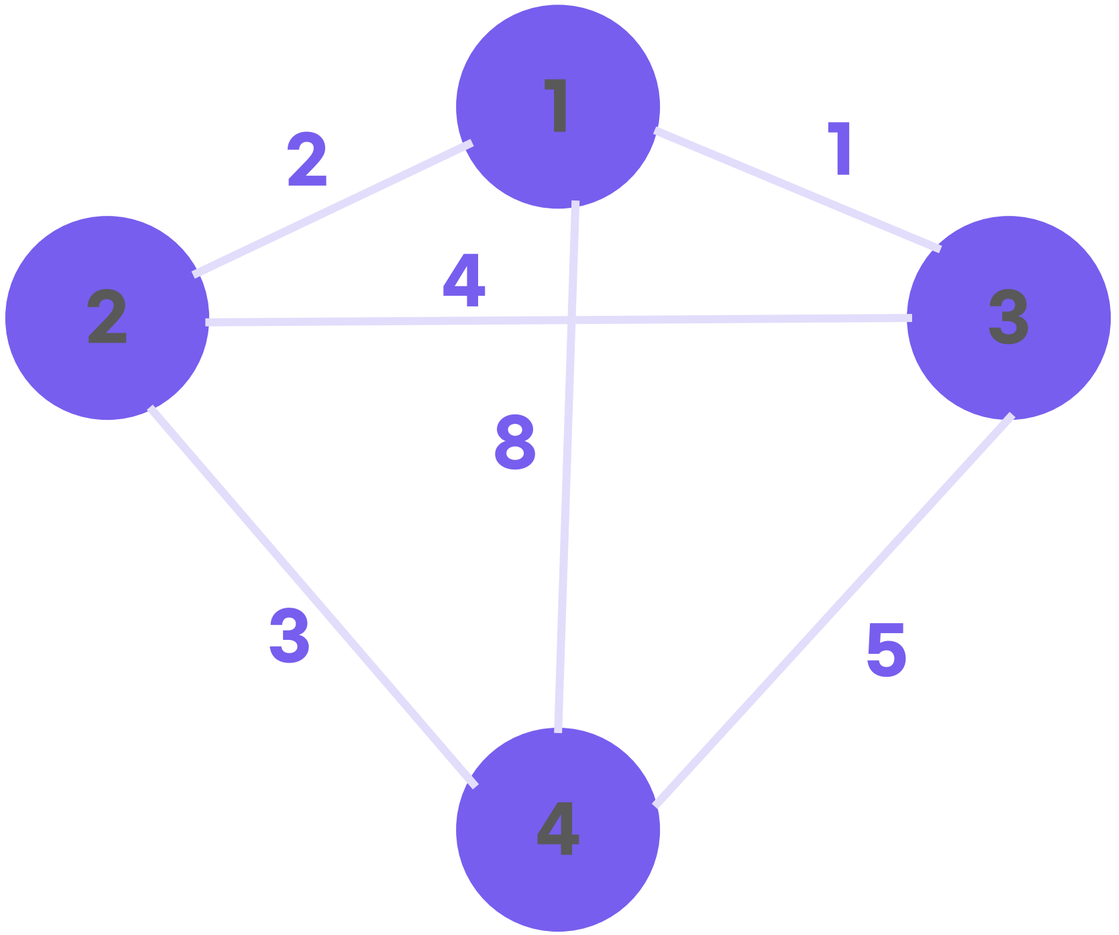

0
2
1
8
5
6


1
2
3
4
2
1
4
8
3
5


0
2
1
6
5


1
2
1
2
4
3
8
3
5
4


0
2
1
5


1
2
1
2
4
3
8
3
5
4

# 1. 의사 결정 트리 학습방법

- 새로운 데이터가 들어오면 체크리스트를 바탕으로 하나씩 질문하고 이 질문에 대한 대답을 바탕으로 분류해준다.
- 높은 정보 획득량을 제공하는 속성을 우선적으로 선택한다.

# 2. 최소 신장 트리

- 도로 건설, 전기 회로, 통신, 배관, 네트워크 등에 이용
- 주어진 가중치 그래프에서 모든 점들을 연결시킨 트리들 중 가중치 합이 최소인 트리
- 크루스칼 알고리즘, 프림 알고리즘

## 3. 다익스트라 알고리즘

## 4. 허프만 코드

- 데이터를 효율적으로 압축하는데 사용하는 알고리즘
- 빈도수가 작은 문자 두개를 골라 이를 합쳐 트리를 만든다.