

DIJKSTRA'S ALGORITHM

SCV Junior - 그래프

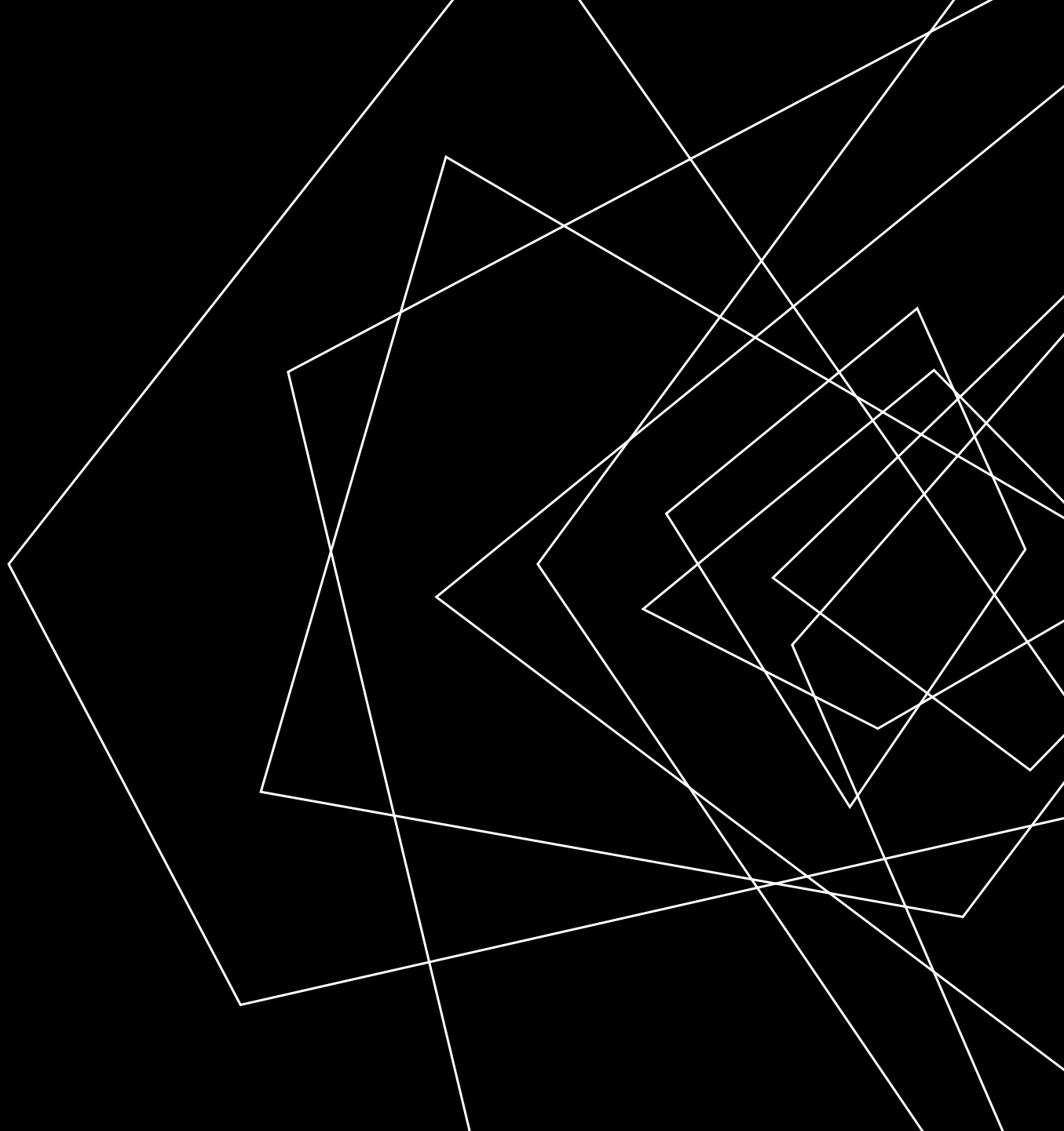
CONTENTS

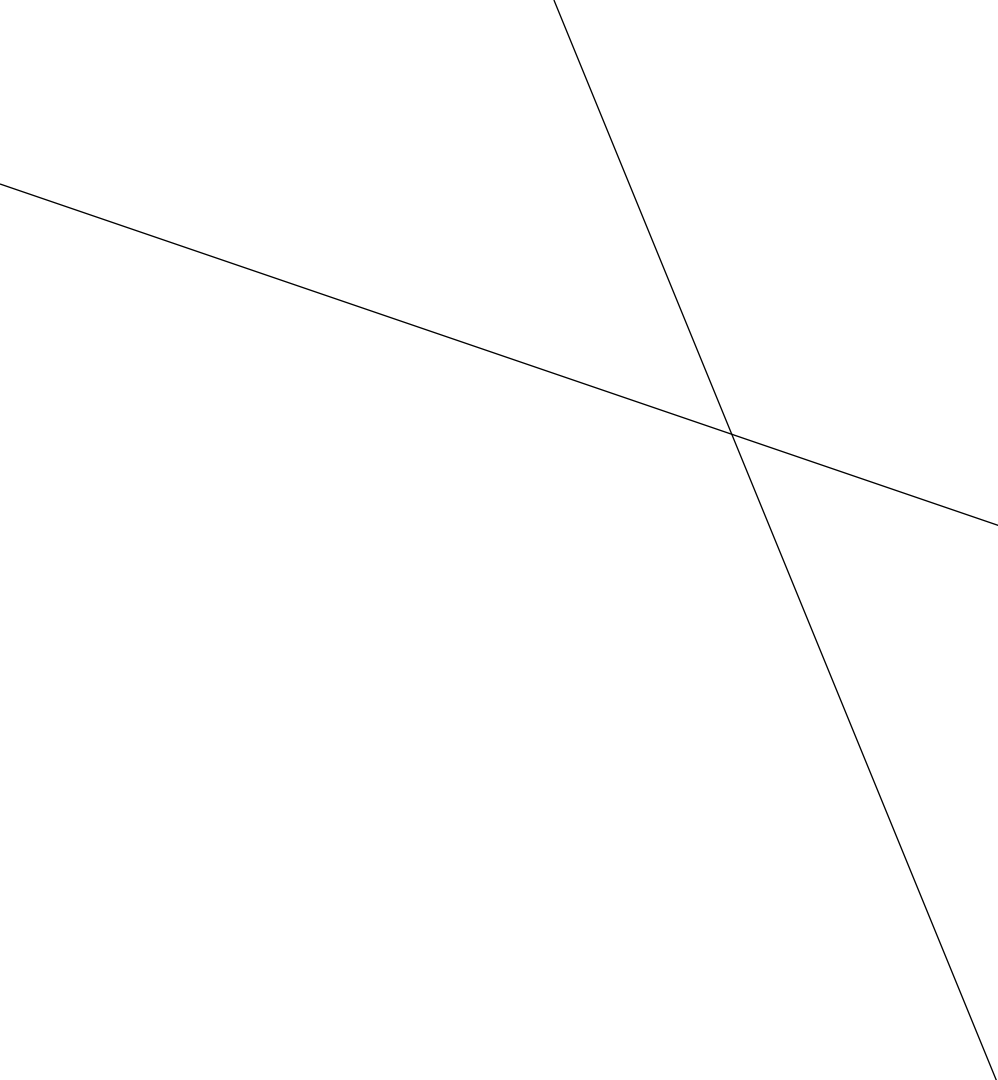
001 그래프와 탐색

- 그래프의 구성 요소
- BFS와 DFS

002 다익스트라

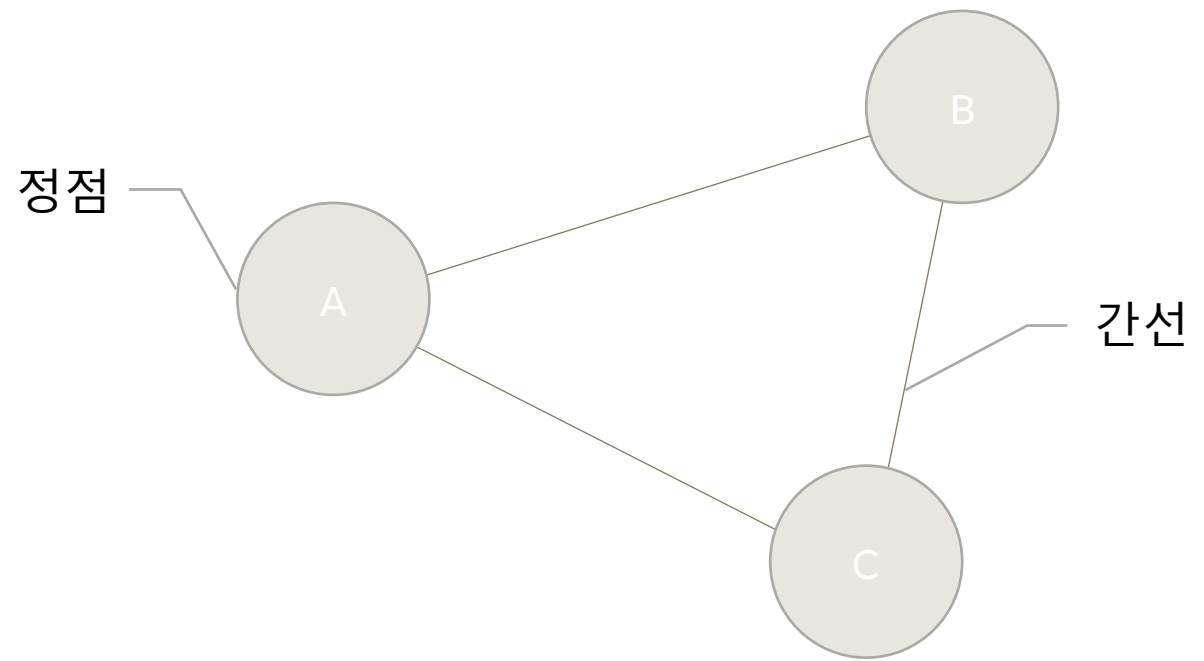
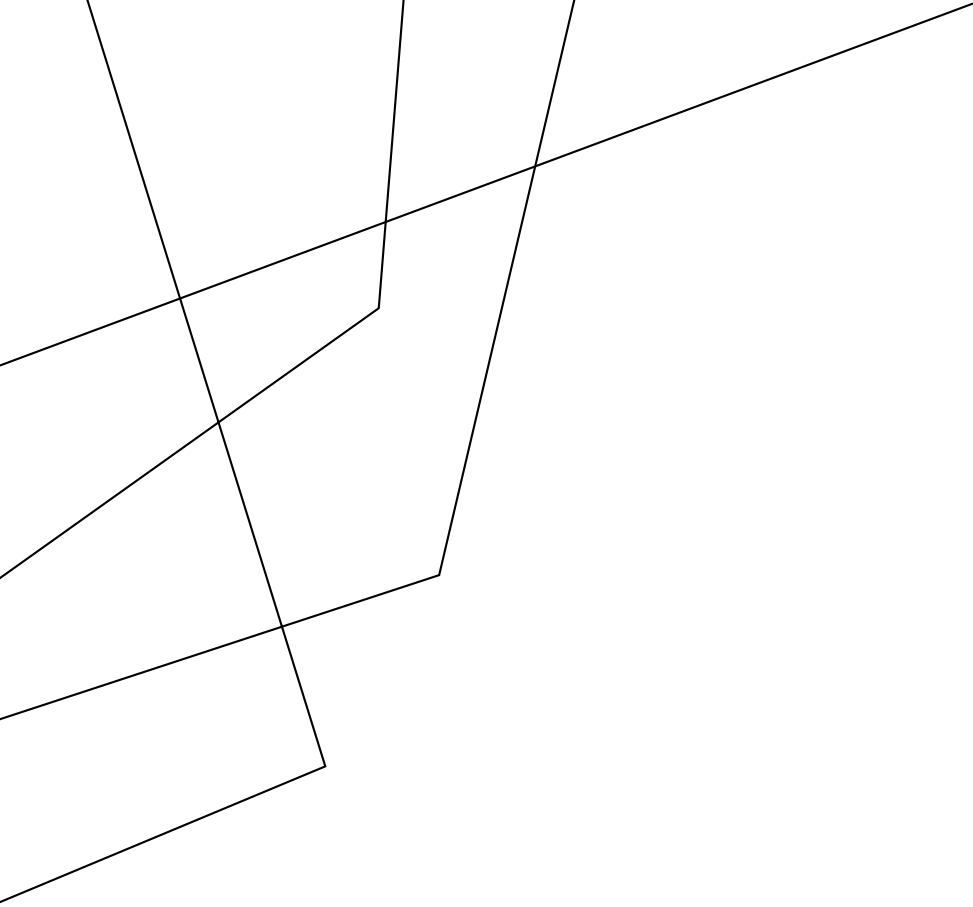
- 다익스트라의 개념
- 다익스트라의 동작
- 다익스트라의 구현



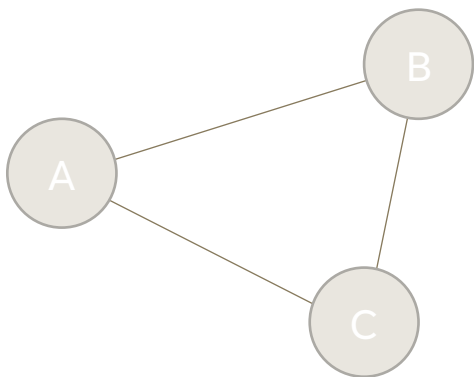


001. 그래프와 탐색

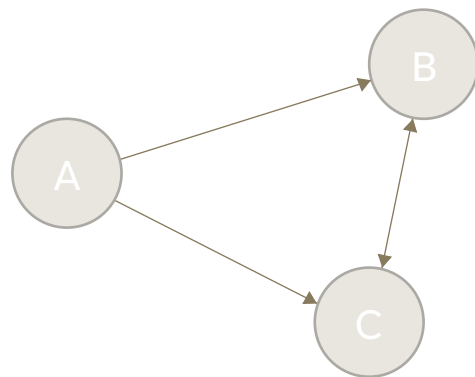
그래프를 구성하는 요소와
그래프를 탐색하는 방법 두 가지에 대해 학습합니다



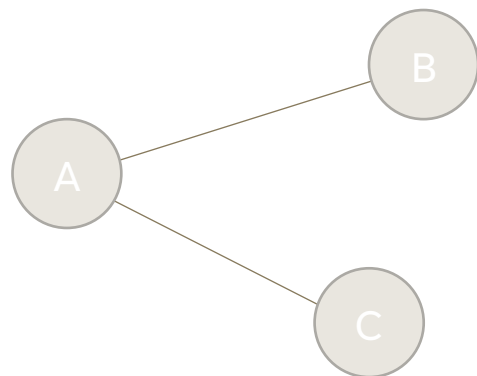
그래프의 구성 요소



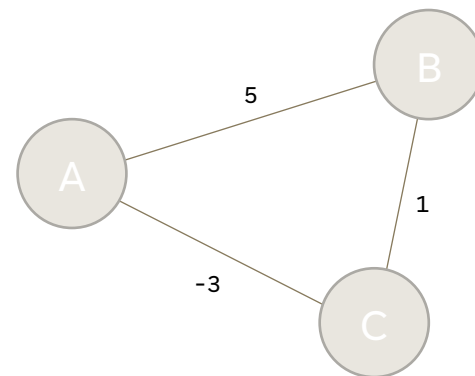
연결그래프



방향그래프



트리



가중치 그래프

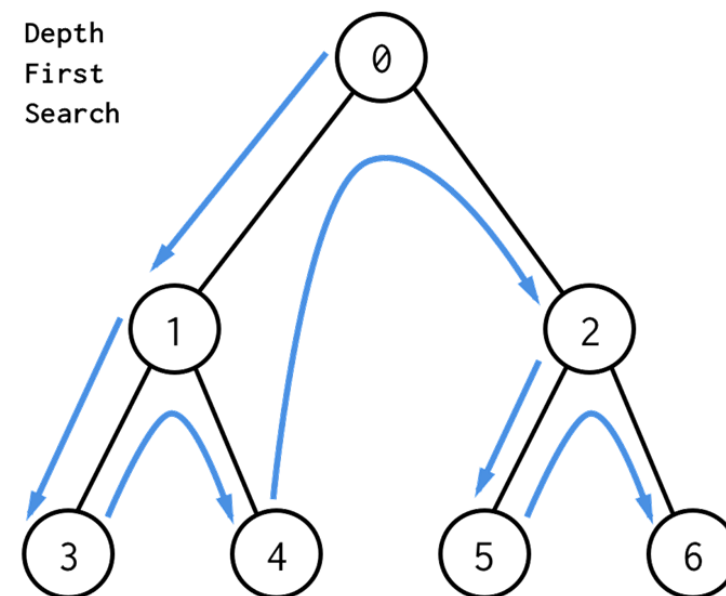
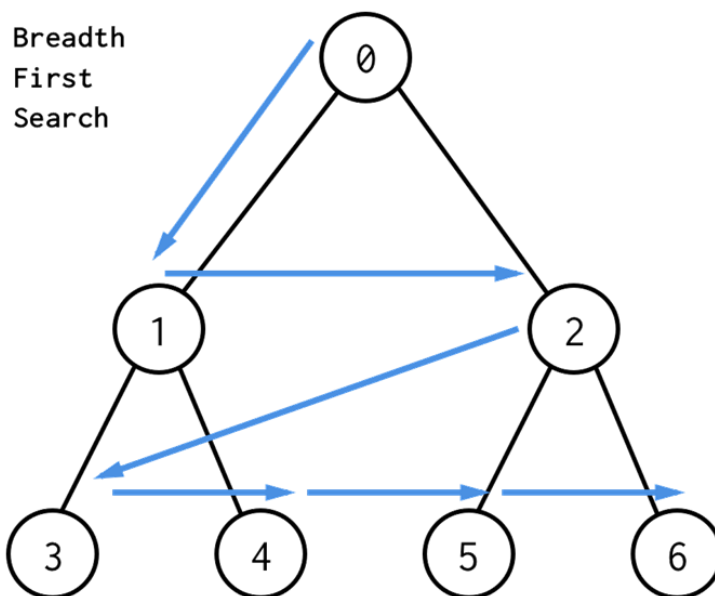
그래프 탐색

BFS

|현재 레벨의 모든 노드를 우선 탐색

DFS

|다음 레벨을 우선 탐색





002. 다익스트라

다익스트라 알고리즘의 개념과
프로그래밍적 구현 방법에 대해 살펴봅니다

다익스트라 알고리즘

최단 거리를 찾는 알고리즘 중 하나

다익스트라 특징

특정 노드에서
모든 노드로

어느 노드에서
다른 모든 노드까지 이르는
최단 거리를 찾는 알고리즘

음수 가중치
허용 안함

음수 가중치가 없는
그래프에서 사용할 수 있다
물론, 음수 가중치가 있는
그래프에서의 최단거리를 찾는
알고리즘도 존재한다

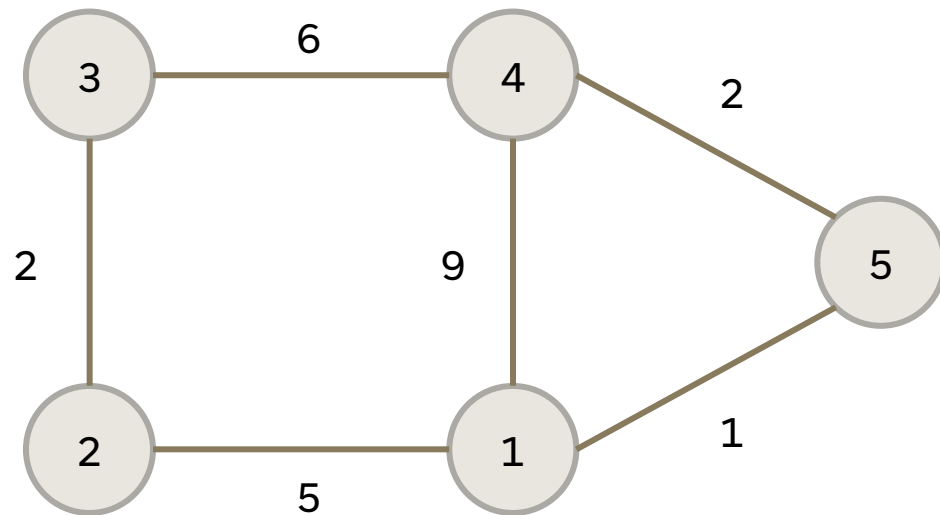
모든 간선을
단 한번 처리

음수 가중치를 허용하지 않기에,
간선을 단 한 번 처리하여
최단 거리를 보장한다

다익스트라의 작동 방식

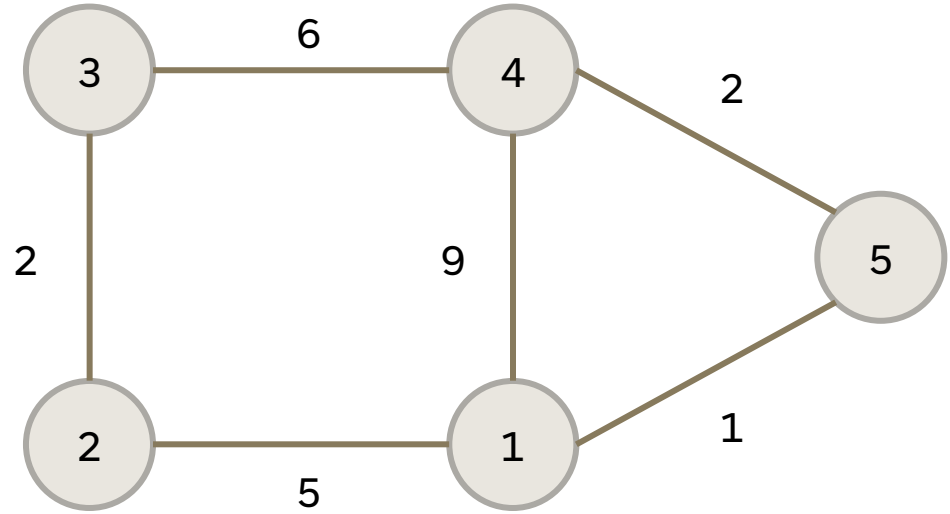
시작 노드 : 1

초기에는 모든 노드까지의 거리가 무한대이며,
아직 어느 노드에도 방문하지 않았다



	1	2	3	4	5
distance	INF	INF	INF	INF	INF
visited	FALSE	FALSE	FALSE	FALSE	FALSE

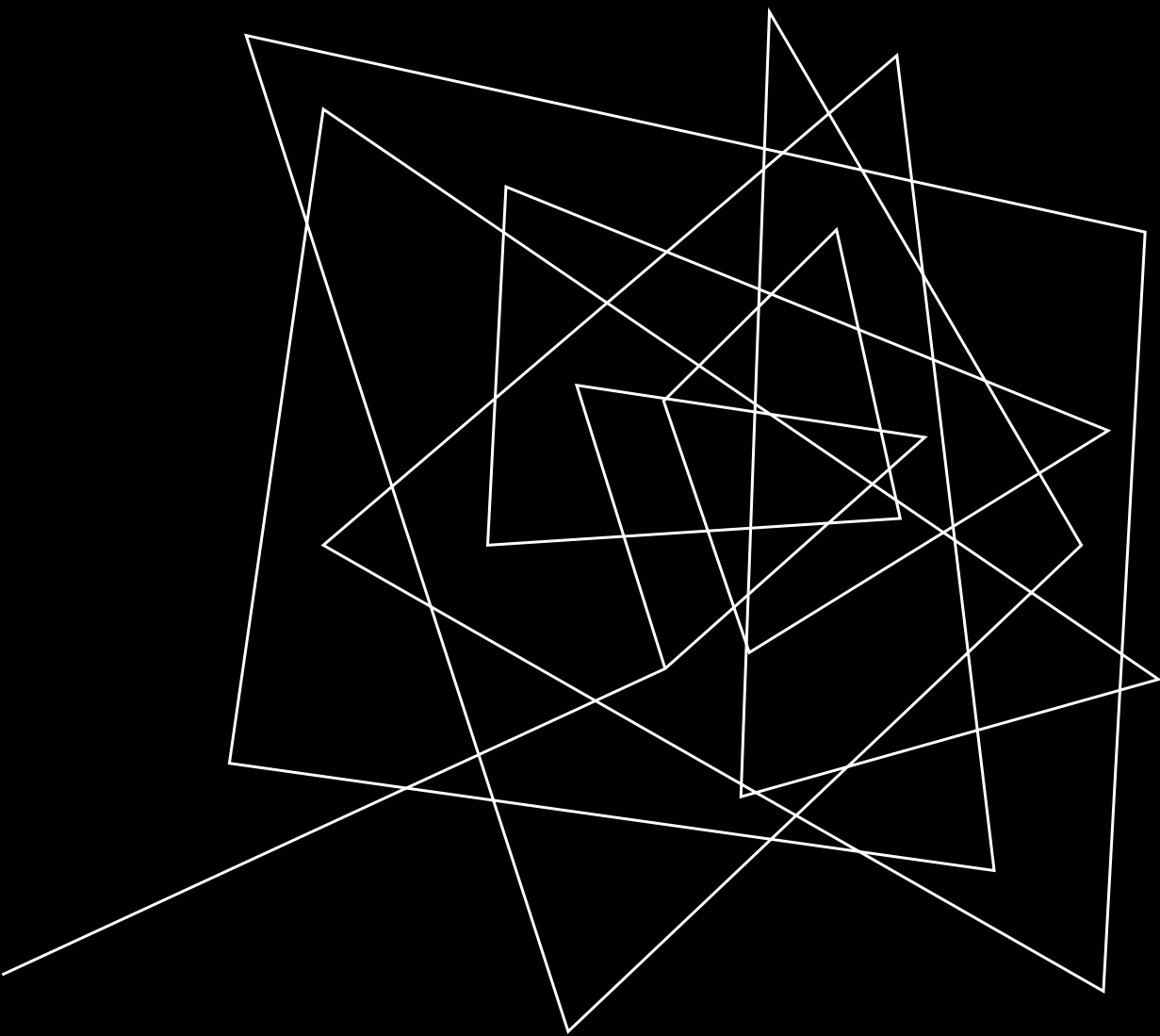
다익스트라의 작동 방식



	1	2	3	4	5
distance					
visited					

다익스트라 작동 방식

1. 방문하지 않은 정점 중 가장 거리 값이 작은 정점을 방문한다. (처음엔 시작 정점 방문)
2. 해당 정점을 거쳐서 갈 수 있는 정점의 거리가, 이전에 기록한 값보다 작다면 그 거리를 갱신한다.



다익스트라 구현

C++, Python 구현

C++ 구현

아직 방문하지 않은 노드 중
가장 거리가 짧은 노드 얻기

인접리스트로부터 이웃한
노드와 간선 가중치 가져오기

거리 업데이트

*우선순위큐
값이 더 작은 요소가 앞서서 나오는 자료구조.
내부적으로 힙으로 구현되어 있다

```
#include <queue>

void dijkstra_algorithm(int start) {

    priority_queue<pair<int, int>> q;

    distance[start] = 0;
    q.push({0, start}); // distance value, node

    while (!q.empty()) {

        int node = q.top().second; q.pop();

        if (visited[node]) continue;

        visited[node] = true;

        for(auto adj : adj_list[node]) {
            int adj_node = adj.first;
            int weight = adj.second;

            if (distance[node] + weight < distance[adj_node]) {
                distance[adj_node] = distance[node] + weight;
                q.push({-distance[adj_node], adj_node});
            }
        }
    }
}
```

PYTHON 구현

아직 방문하지 않은 노드 중
가장 거리가 짧은 노드 얻기

인접리스트로부터 이웃한
노드와 간선 가중치 가져오기

거리 업데이트

*우선순위큐
값이 더 작은 요소가 앞서서 나오는 자료구조.
내부적으로 힙으로 구현되어 있다

```
import heapq
```

```
def dijkstra_algorithm(start):
```

```
    q = []
```

```
    distance[start] = 0
```

```
    heapq.heappush(q, [0, start])
```

```
    while(q):
```

```
        _, node = heapq.heappop(q)
```

```
        if visited[node]:
```

```
            continue
```

```
        visited[node] = True
```

```
        for adj_node, weight = adj_list[node]:
```

```
            if distance[node] + weight < distance[adj_node]:
```

```
                distance[adj_node] = distance[node] + weight
```

```
                heapq.heappush(q, [distance[adj_node], adj_node])
```



문제 풀기

- BOJ 최소비용 구하기
- BOJ 서강그라운드

A series of white, thin, overlapping geometric lines on a black background, creating a complex, abstract pattern on the left side of the slide.

감사합니다.

다음 시간에는...