

그래프 1

최백준 choi@startlink.io

그래프

그래프

Graph

정점 / 간선

$$G = (V, E)$$

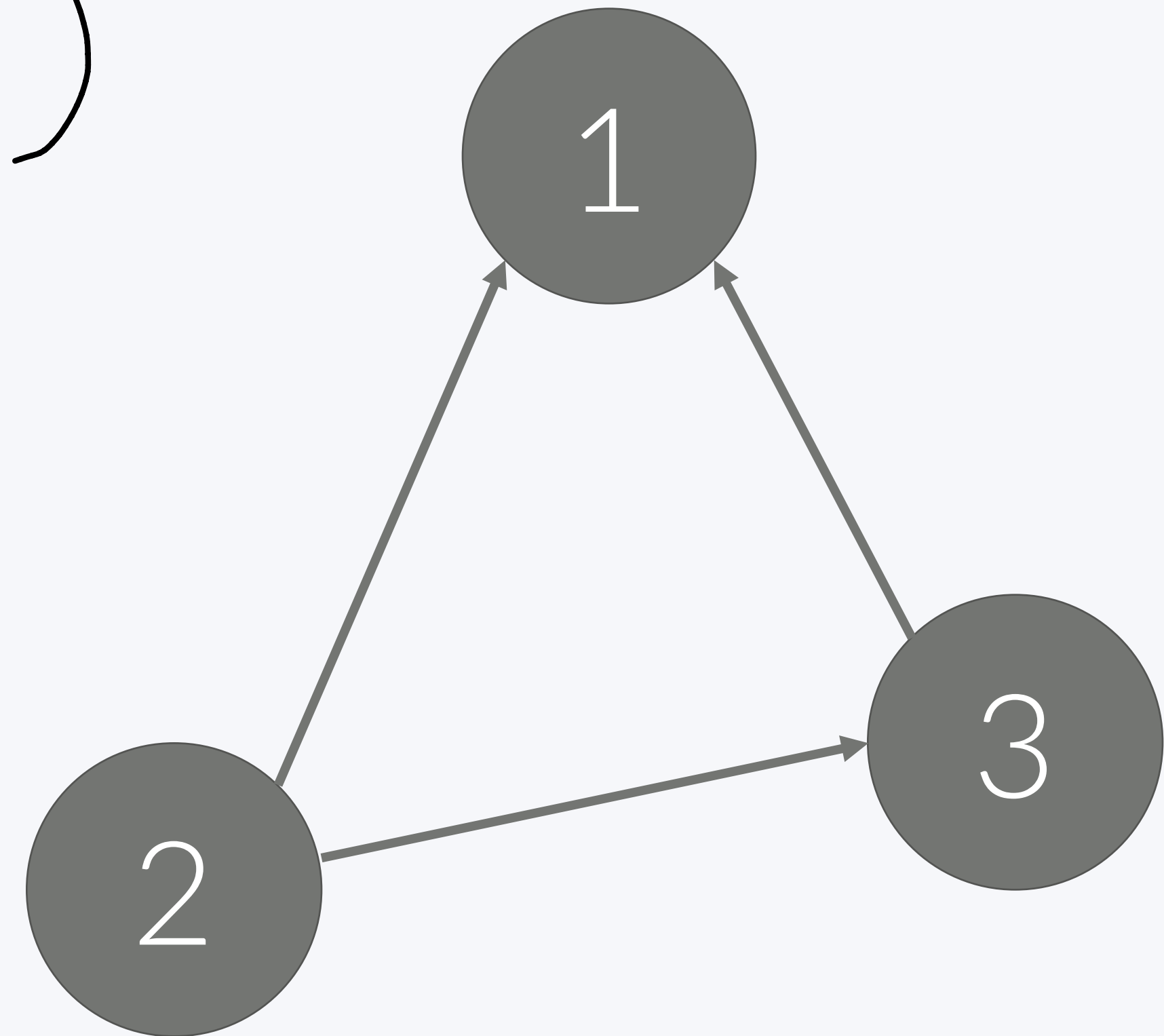
3

- 자료구조의 일종

- 정점(Node, Vertex)

- 간선(Edge): 정점간의 관계를 나타낸다.

- $G = (V, E)$ 로 나타낸다.



경로

Path

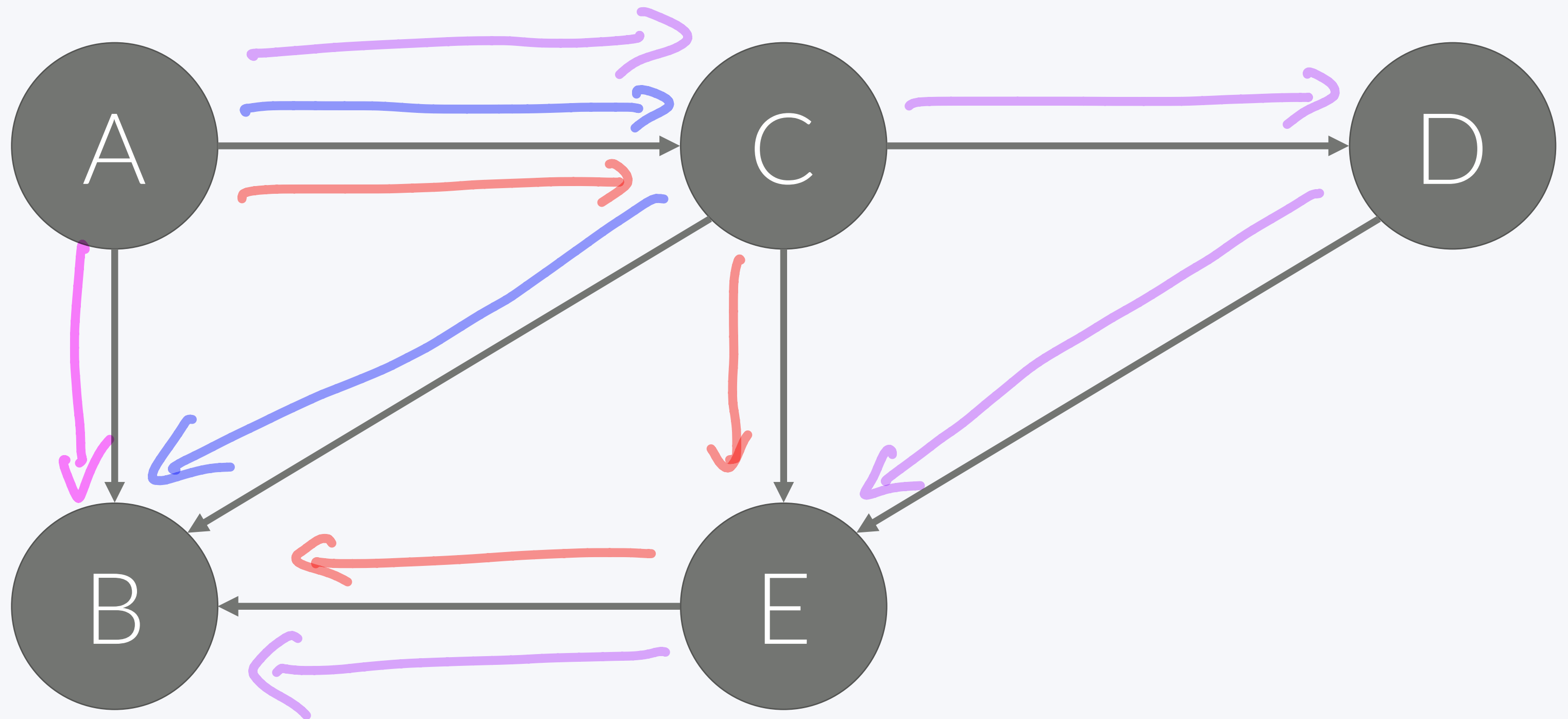
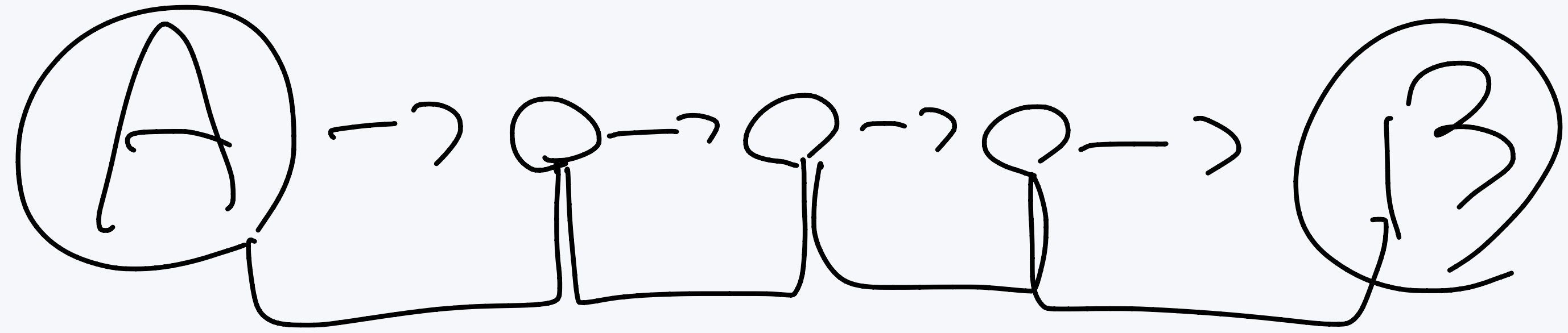
- 정점 A에서 B로 가는 경로

• $A \rightarrow C \rightarrow D \rightarrow E \rightarrow B$

• $A \rightarrow B$

• $A \rightarrow C \rightarrow B$

• $A \rightarrow C \rightarrow E \rightarrow B$



사이클

~~Path~~ Cycle

시작 = 도착

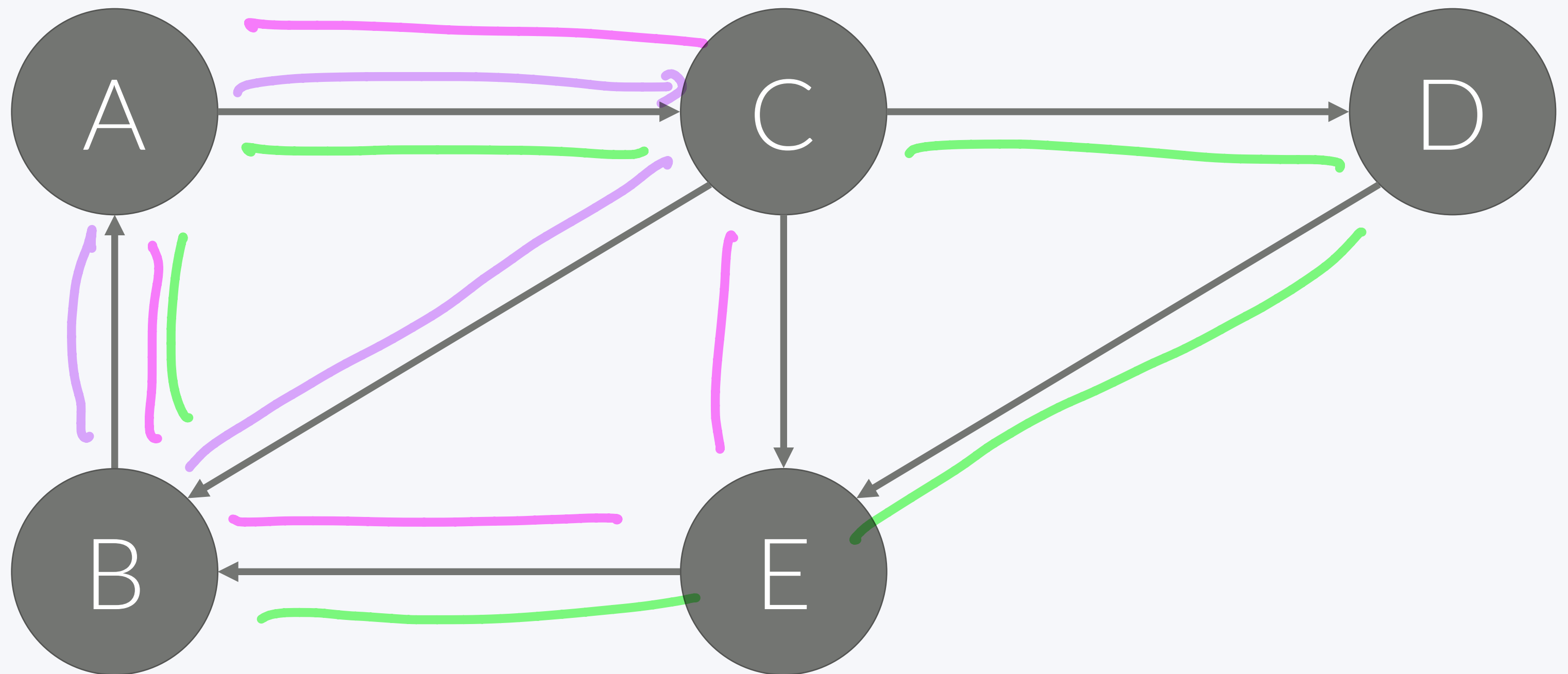
5

- 정점 A에서 다시 A로 돌아오는 경로

• $A \rightarrow C \rightarrow B \rightarrow A$

• $A \rightarrow C \rightarrow E \rightarrow B \rightarrow A$

• $A \rightarrow C \rightarrow D \rightarrow E \rightarrow B \rightarrow A$



단순 경로와 단순 사이클

Simple Path and Simple Cycle

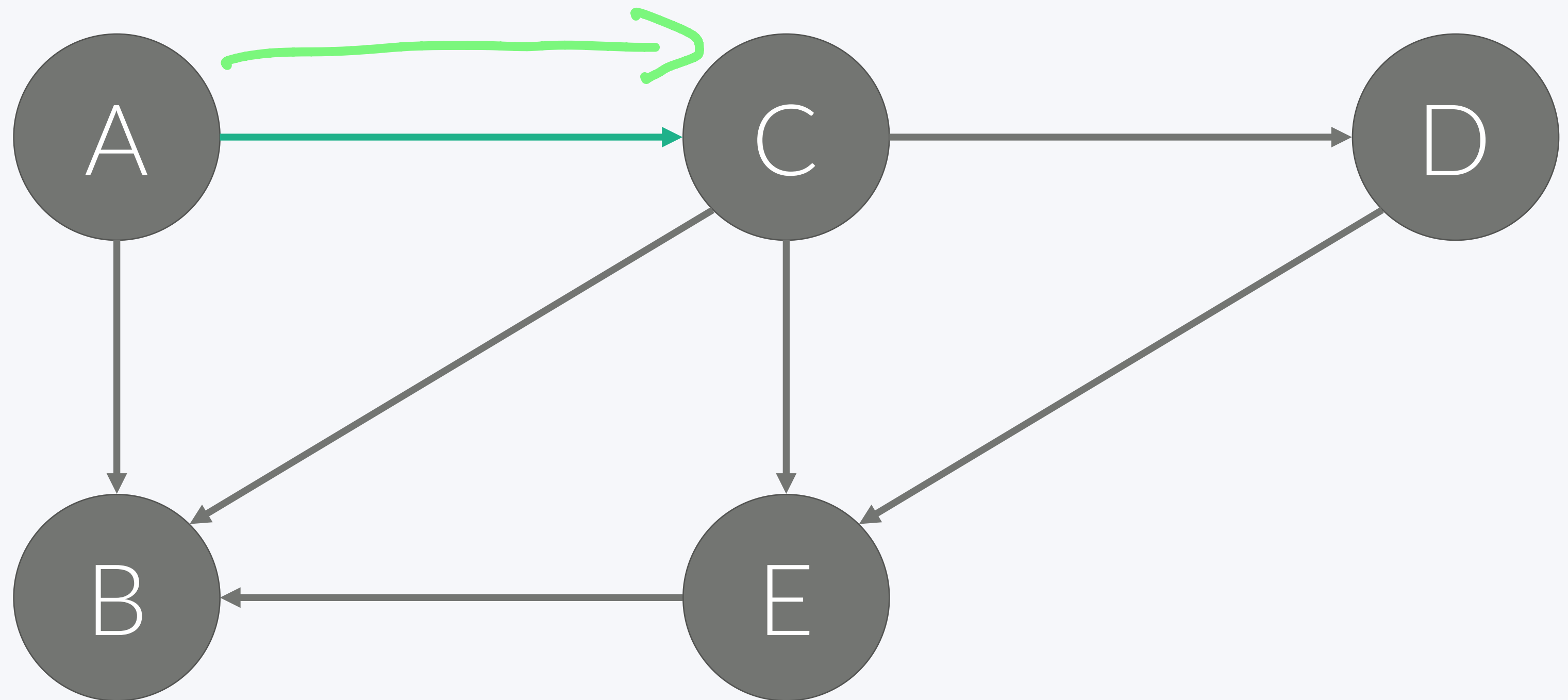
- 경로/사이클에서 같은 정점을 두 번 이상 방문하지 않는 경로/사이클
- 특별한 말이 없으면, 일반적으로 사용하는 경로와 사이클은 단순 경로/사이클을 말한다.

방향 있는 그래프

7

Directed Graph

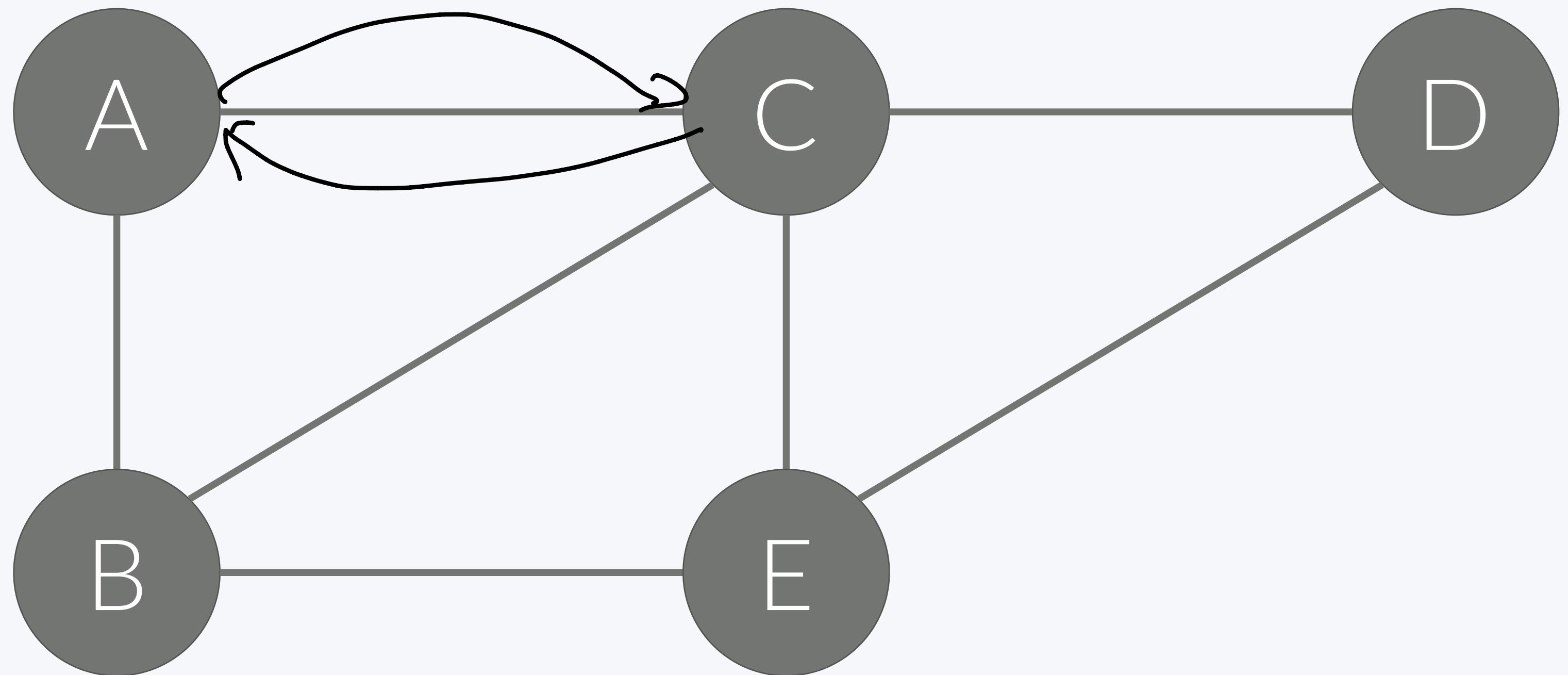
- $A \rightarrow C$ 와 같이 간선에 방향이 있다.
- $A \rightarrow C$ 는 있지만, $C \rightarrow A$ 는 없다.



방향 없는 그래프

Undirected Graph

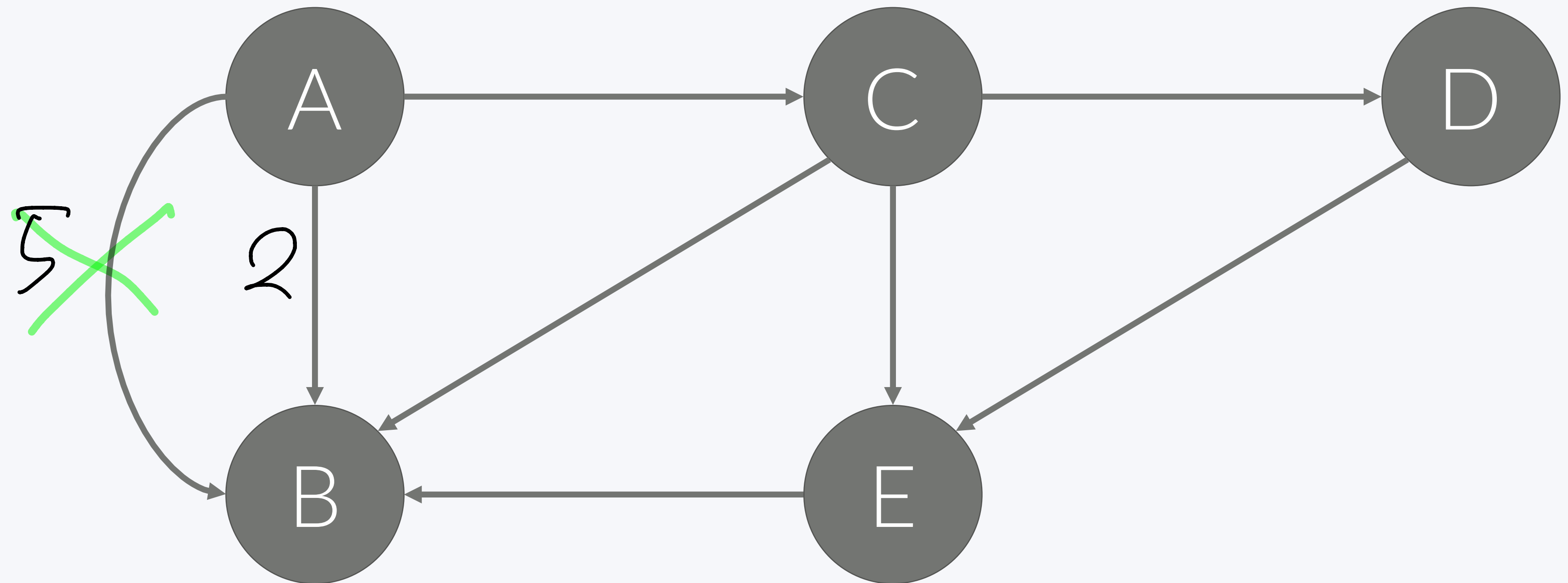
- A – C 와 같이 간선에 방향이 없다.
- A – C는 $A \rightarrow C$ 와 $C \rightarrow A$ 를 나타낸다.
- 양방향 그래프 (Bidirection Graph) 라고도 한다.



간선 여러개

Multiple Edge

- 두 정점 사이에 간선이 여러 개일 수도 있다.
- 아래 그림의 A-B는 연결하는 간선이 2개이다.
- 두 간선은 서로 다른 간선이다

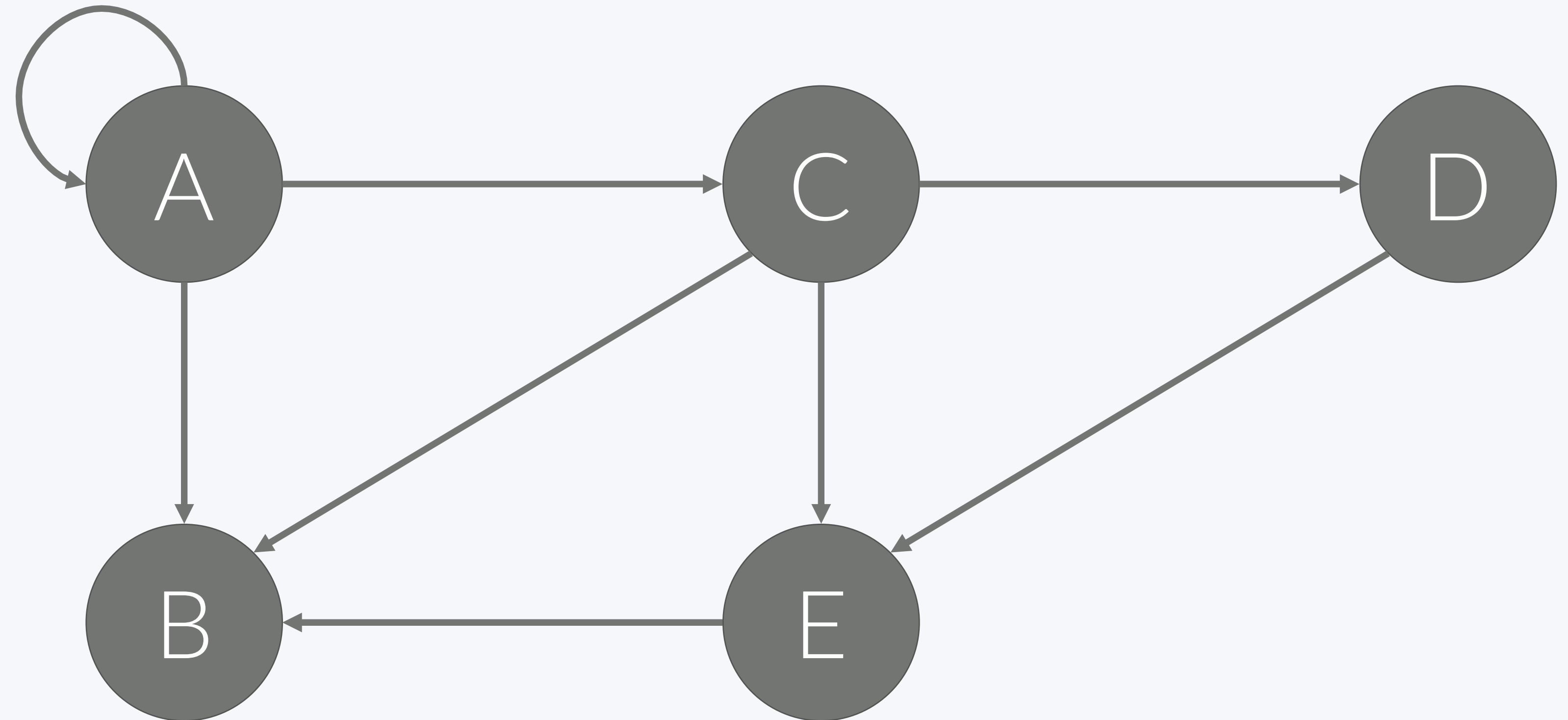


루프

Loop

10

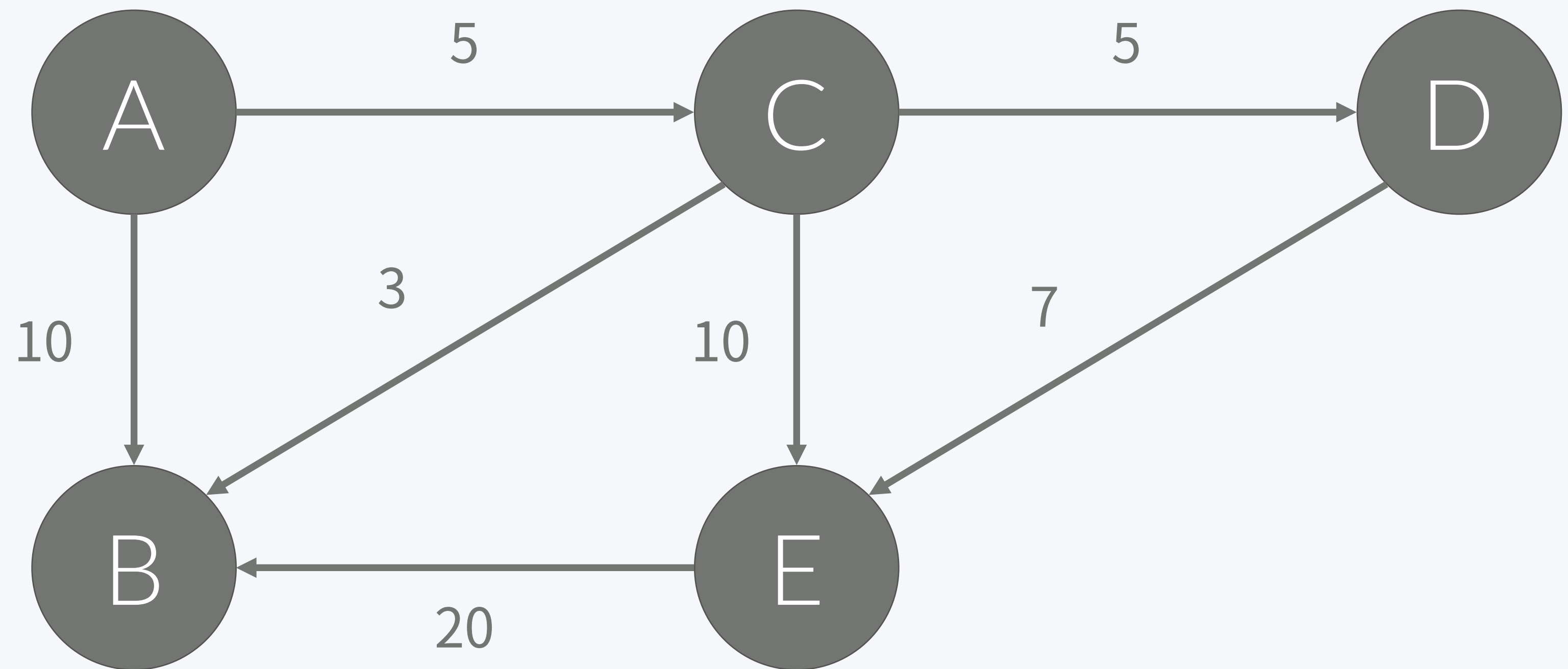
- 간선의 양 끝 점이 같은 경우가 있다.
- $A \rightarrow A$



가중치

Weight

- 간선에 가중치가 있는 경우에는
- A에서 B로 이동하는 거리, 이동하는데 필요한 시간, 이동하는데 필요한 비용 등등등...

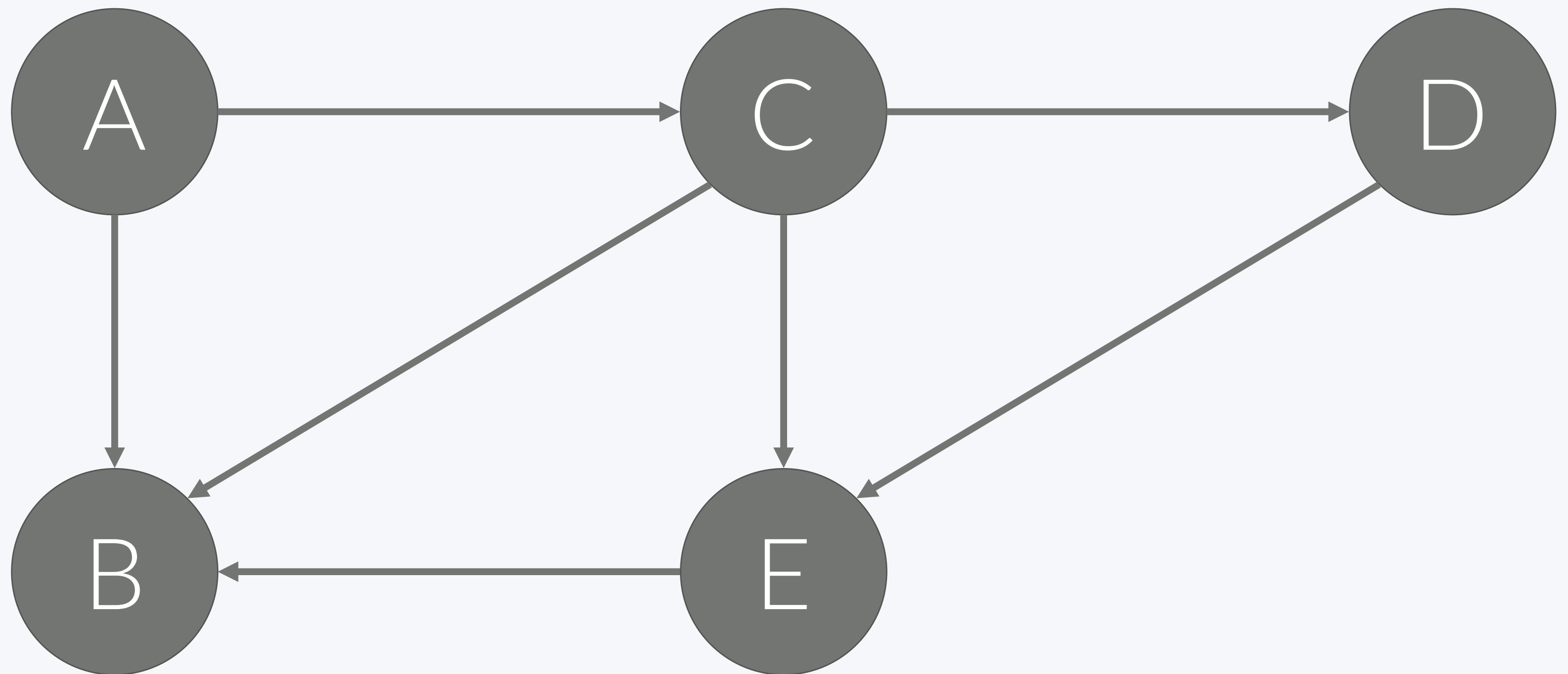


가중치

Weight

12

- 가중치가 없는 경우에는 1이라고 생각하면 된다

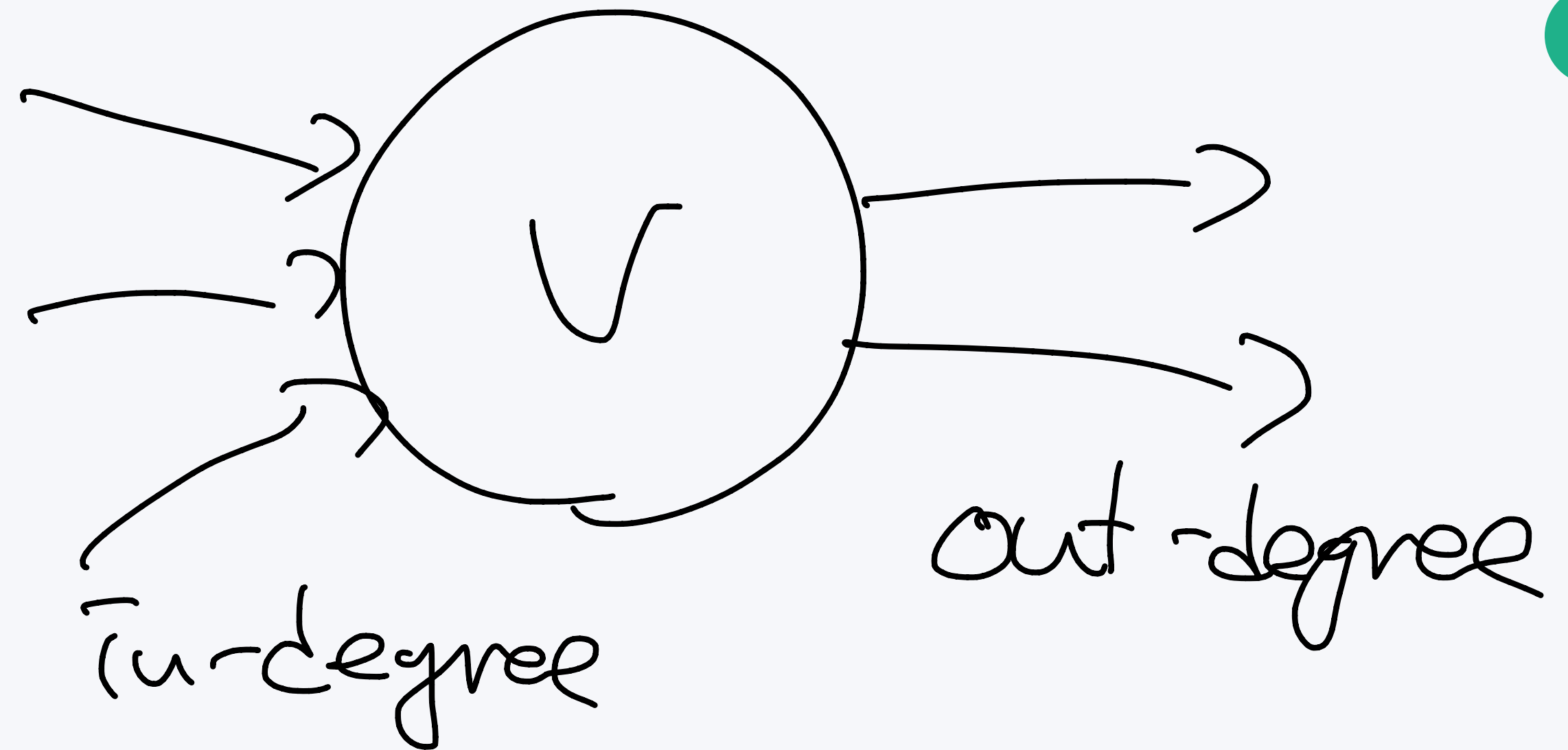


차수

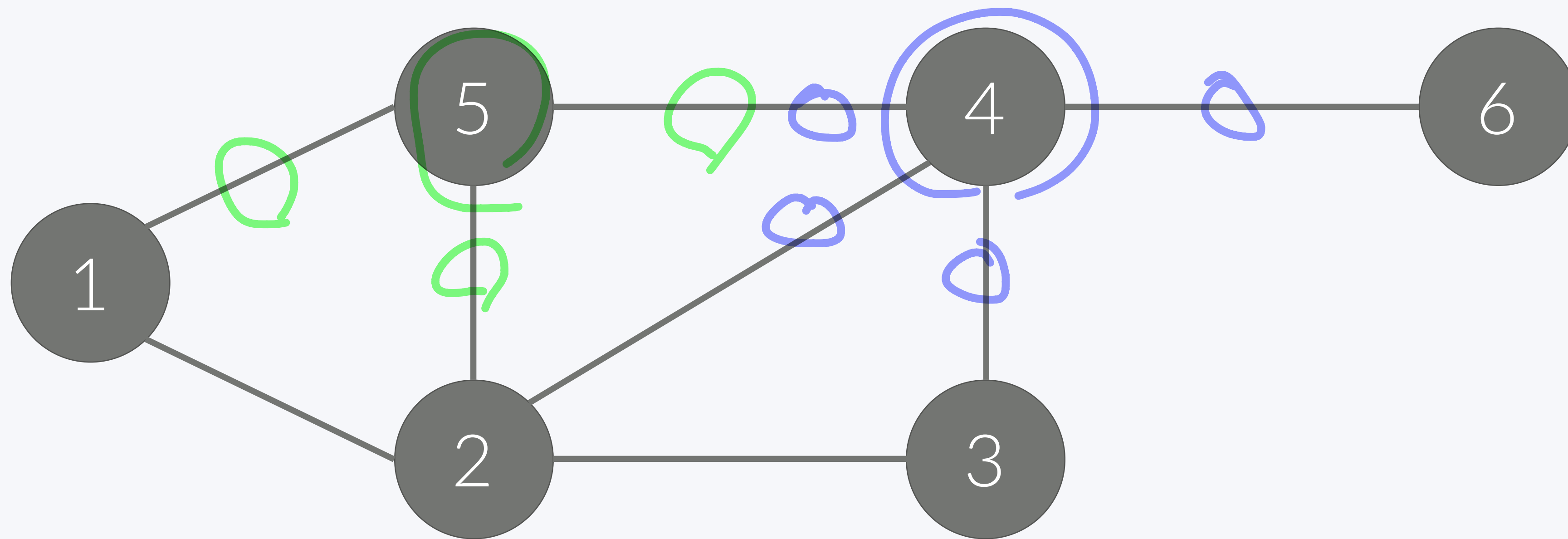
Degree

- 정점과 연결되어 있는 간선의 개수
- 5의 차수: 3
- 4의 차수: 4

방향성



13



간선

그래프의 표현

그래프의 표현

Representation of Graph

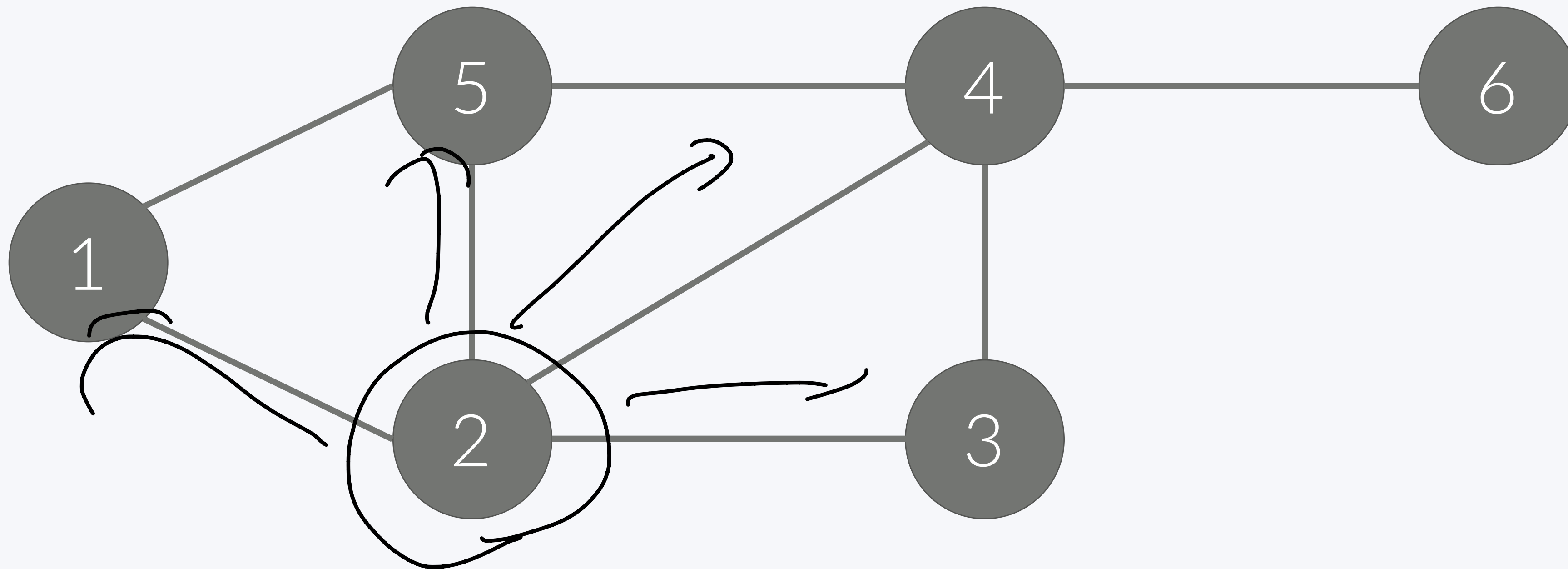
15

정점: V 개, 간선: E 개
 N M

- 아래와 같은 그래프는 정점이 6개, 간선이 8개 있다.
- 간선에 방향이 없기 때문에, 방향이 없는 그래프이다.
- 정점: $\{1, 2, 3, 4, 5, 6\}$
- 간선: $\{(1, 2), (1, 5), (2, 5), (2, 3), (3, 4), (2, 4), (4, 5), (4, 6)\}$

$1 \sim 5$
 $0 \sim \sqrt{-1}$

정점: 6개
간선: 8개

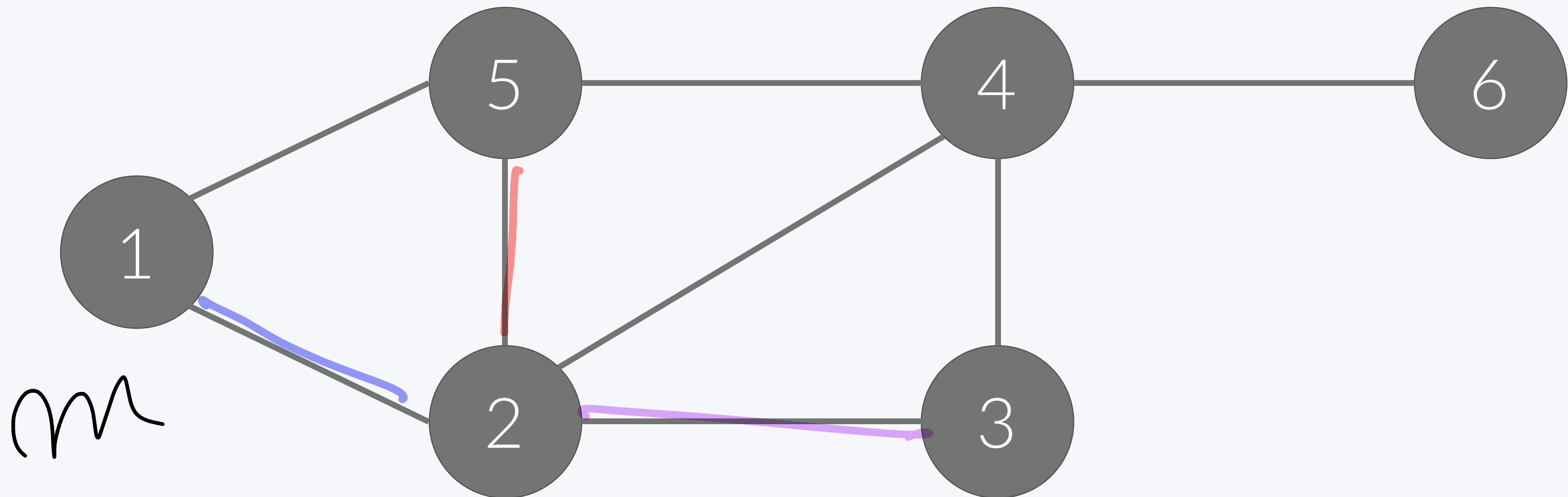


그래프의 표현

Representation of Graph

6 8 # n m (정점의 개수, 간선의 개수)

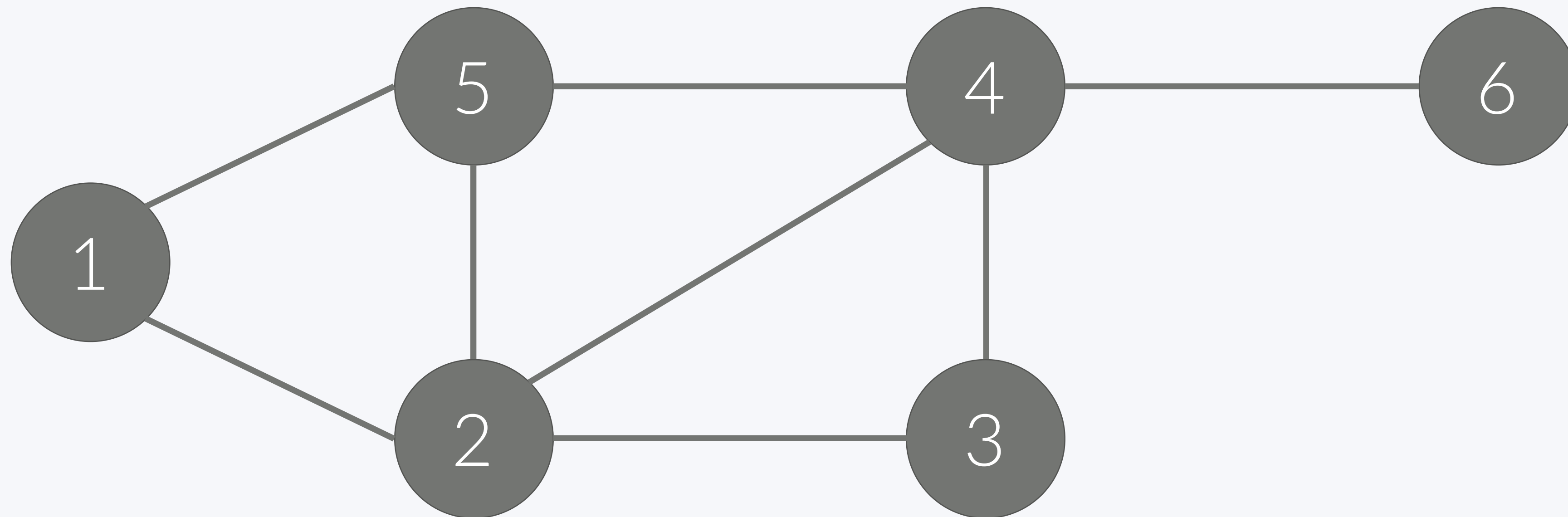
- 1 2
- 1 5
- 2 3
- 2 4
- 2 5
- 5 4
- 4 3
- 4 6



그래프의 표현

Representation of Graph

- 주로 앞 페이지와 같은 입력 형식으로 주어진다.
- 첫째 줄에는 정점의 개수와 간선의 개수
- 둘째 줄부터 간선의 정보

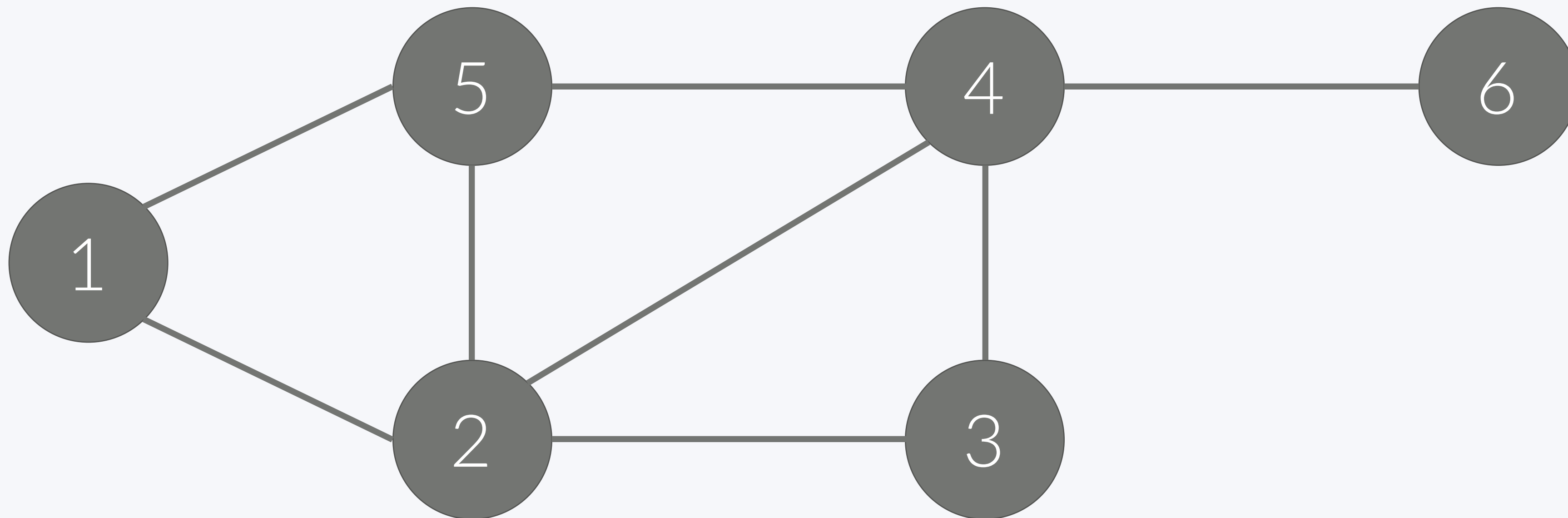


인접 행렬

Adjacency-matrix

- 정점의 개수를 V 이라고 했을 때
- $V \times V$ 크기의 이차원 배열을 이용한다
- $A[i][j] = 1$ ($i \rightarrow j$ 간선이 있을 때), 0 (없을 때)

$A[i][j]$: $i \rightarrow j$
 $i \nrightarrow j$
 $i \rightarrow i$



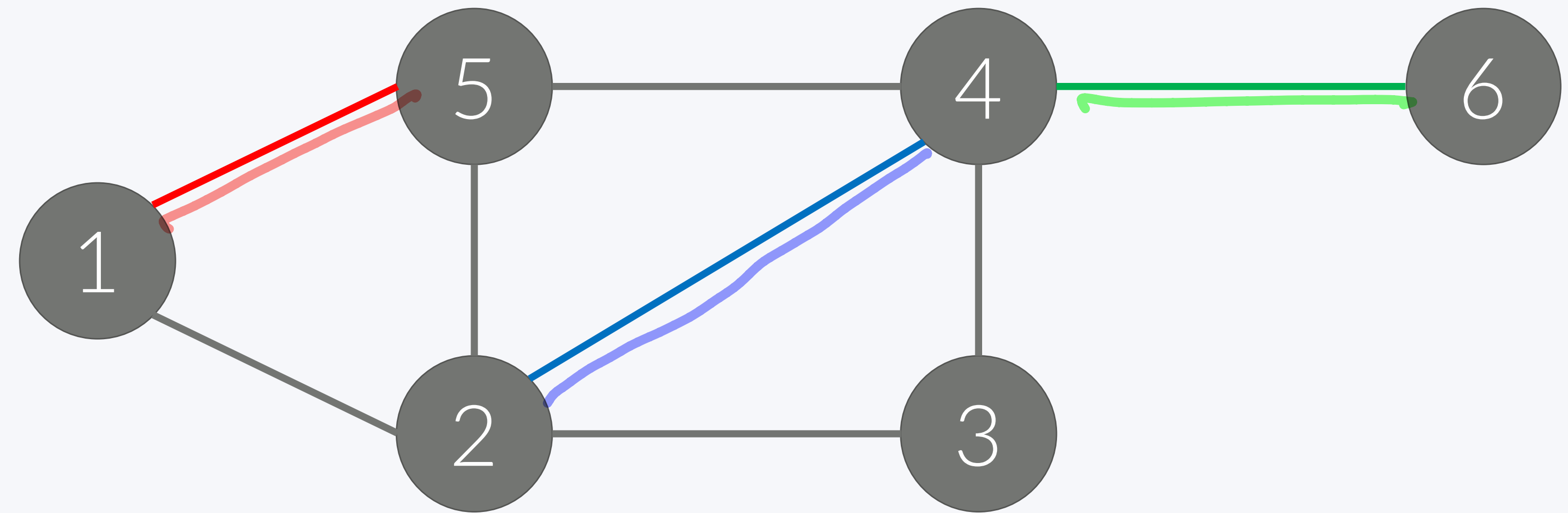
인접 행렬

Adjacency-matrix

	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	1	1	0
3	0	1	0	1	0	0
4	0	1	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

양방향

$A[i][j], A[j][i]$



$$V^2 \geq E$$

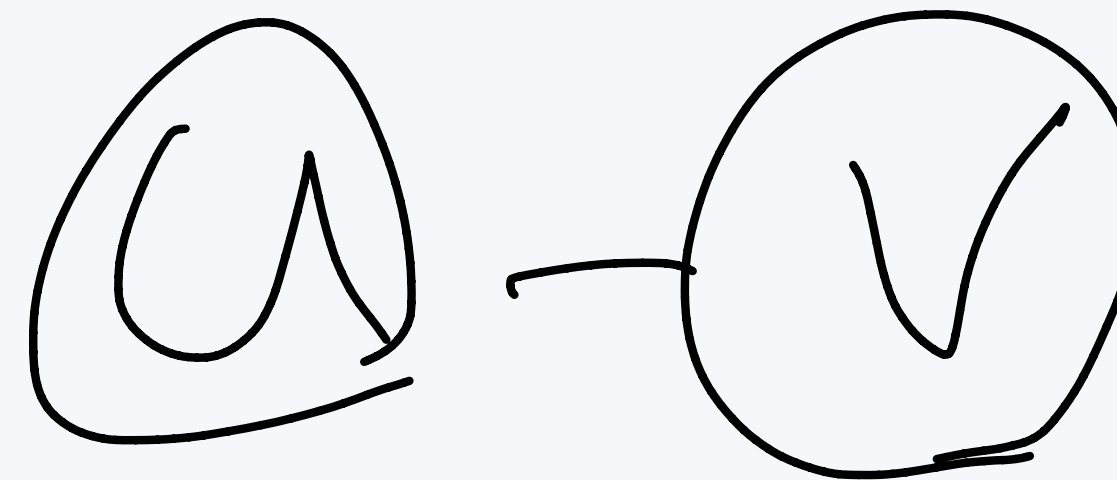
인접 행렬

Adjacency-matrix

20



```
#include <cstdio>
#include <vector>
int a[10][10];
int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    for (int i=0; i<m; i++) {
        int u, v;
        scanf("%d %d", &u, &v);
        a[u][v] = a[v][u] = 1;
    }
}
```

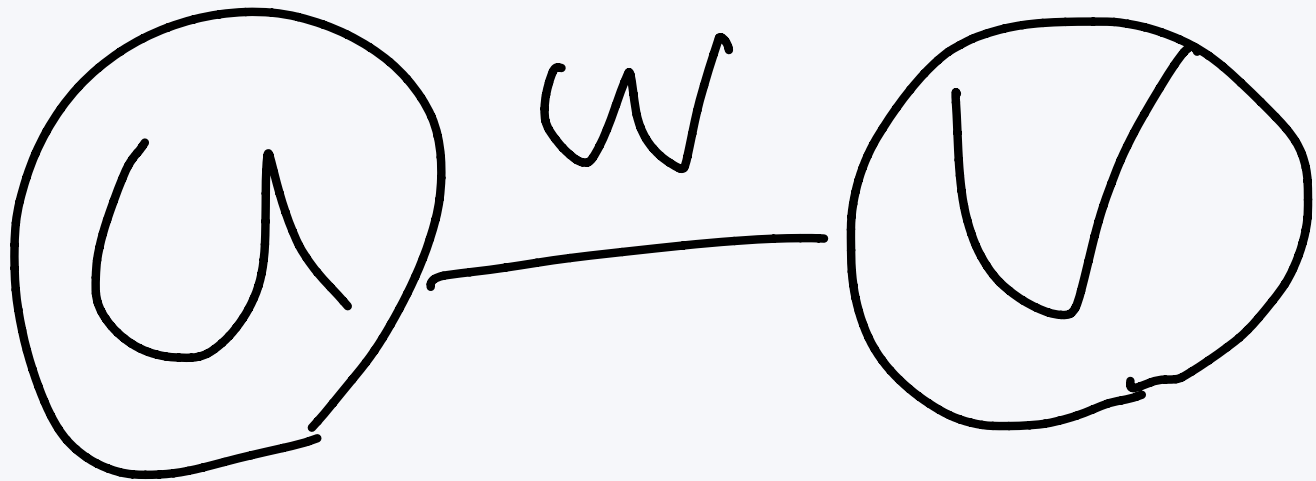


↪ 방향성 그래프에선 $A_{2,2}$

}

그래프의 표현

Representation of Graph



6 8 # n m (정점의 개수, 간선의 개수)

1 2 2

1 5 7

2 3 2

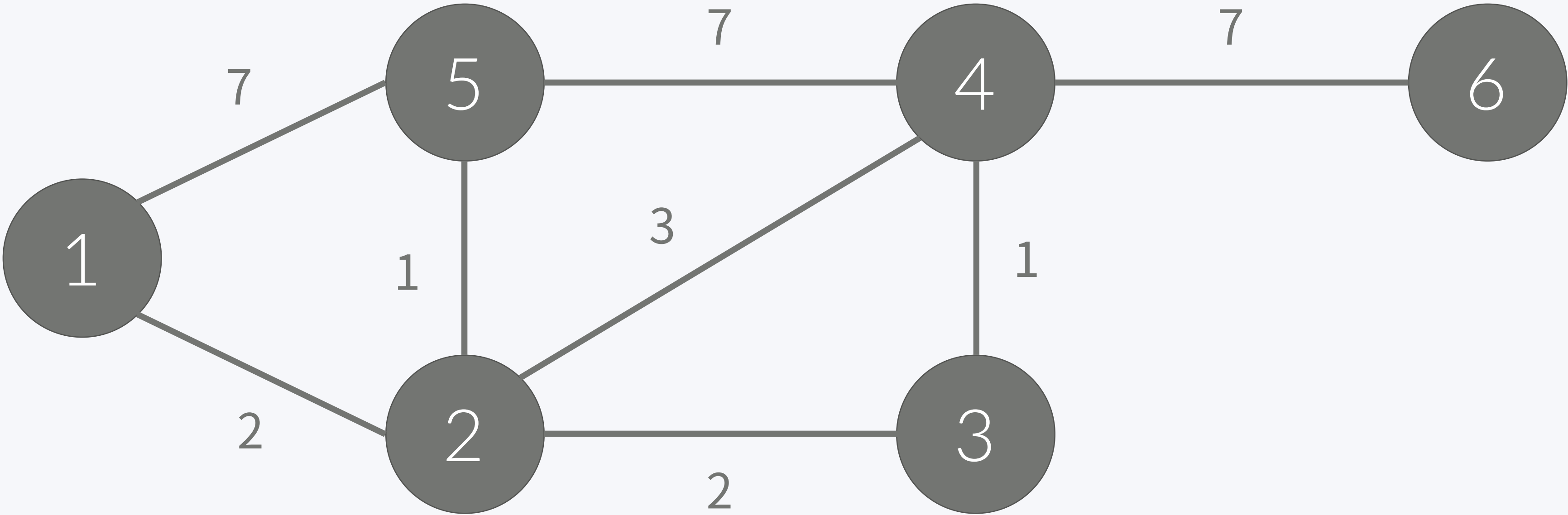
2 4 3

2 5 1

5 4 7

4 3 1

4 6 7



인접 행렬

Adjacency-matrix

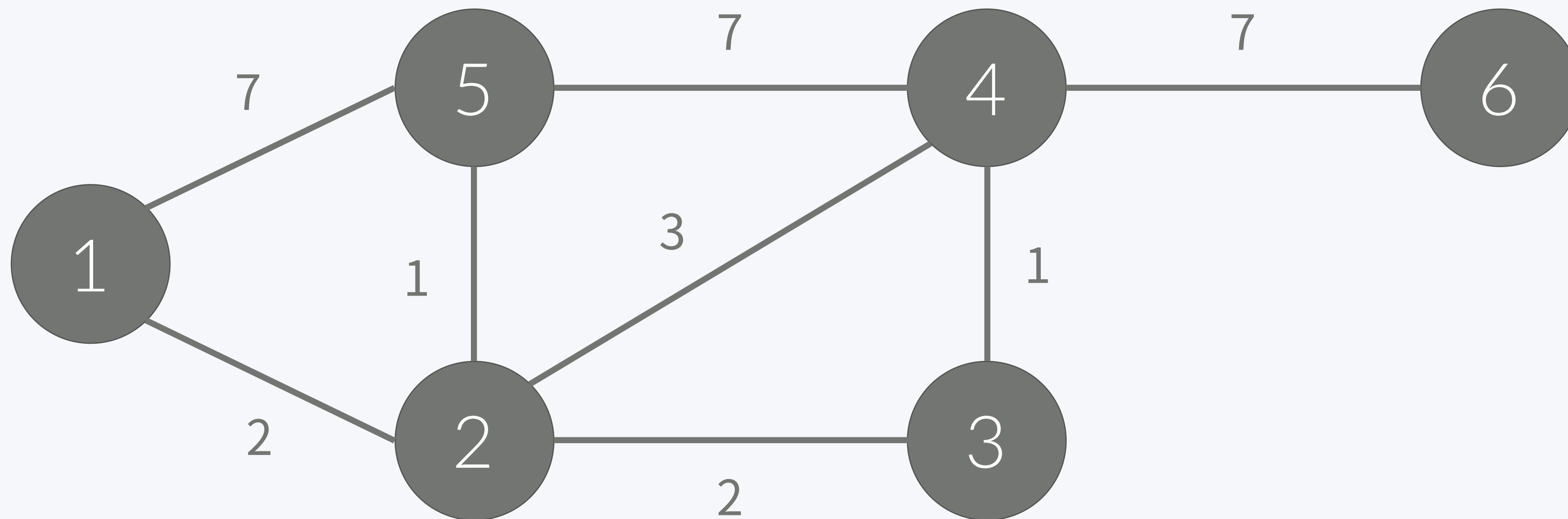
22

- 정점의 개수를 N 이라고 했을 때
- $N \times N$ 크기의 이차원 배열을 이용한다
- $A[i][j] = w$ ($i \rightarrow j$ 간선이 있을 때, 그 가중치), 0 (없을 때)

$$A[i][j] = w$$

$0 \leq w$

-1

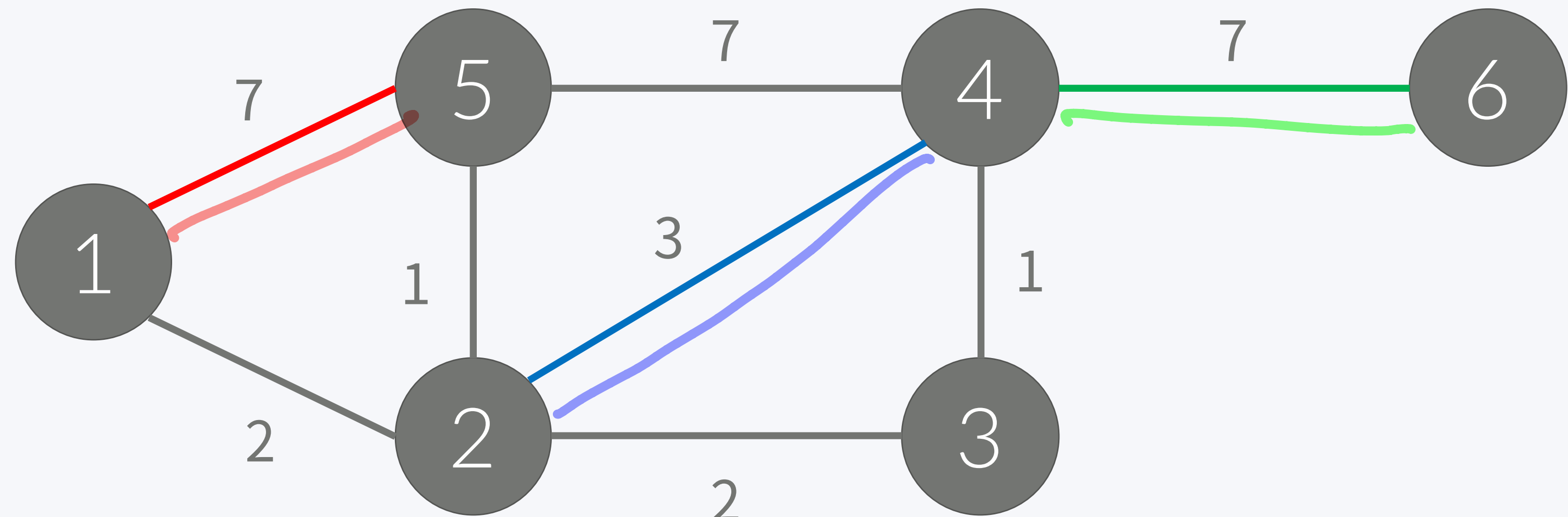


인접 행렬

Adjacency-matrix

23

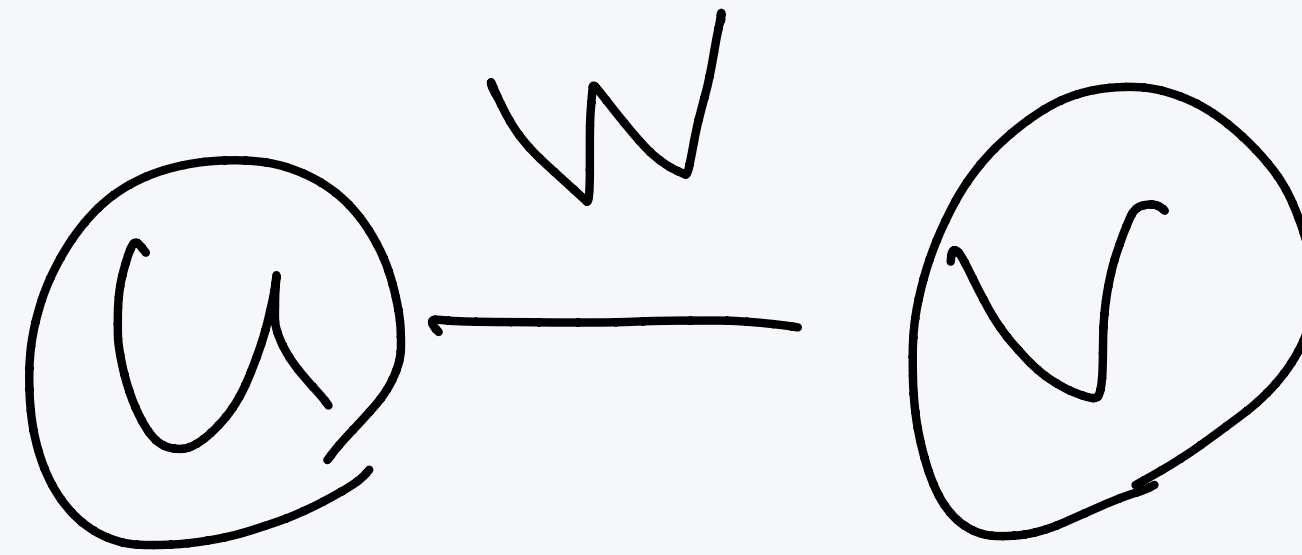
	1	2	3	4	5	6
1	0	2	0	0	7	0
2	2	0	2	3	1	0
3	0	2	0	1	0	0
4	0	3	1	0	7	7
5	7	1	0	7	0	0
6	0	0	0	7	0	0



인접 행렬

Adjacency-matrix

```
#include <cstdio>
#include <vector>
int a[10][10];
int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    for (int i=0; i<m; i++) {
        int u, v, w;
        scanf("%d %d %d", &u, &v, &w);
        a[u][v] = a[v][u] = w;
    }
}
```



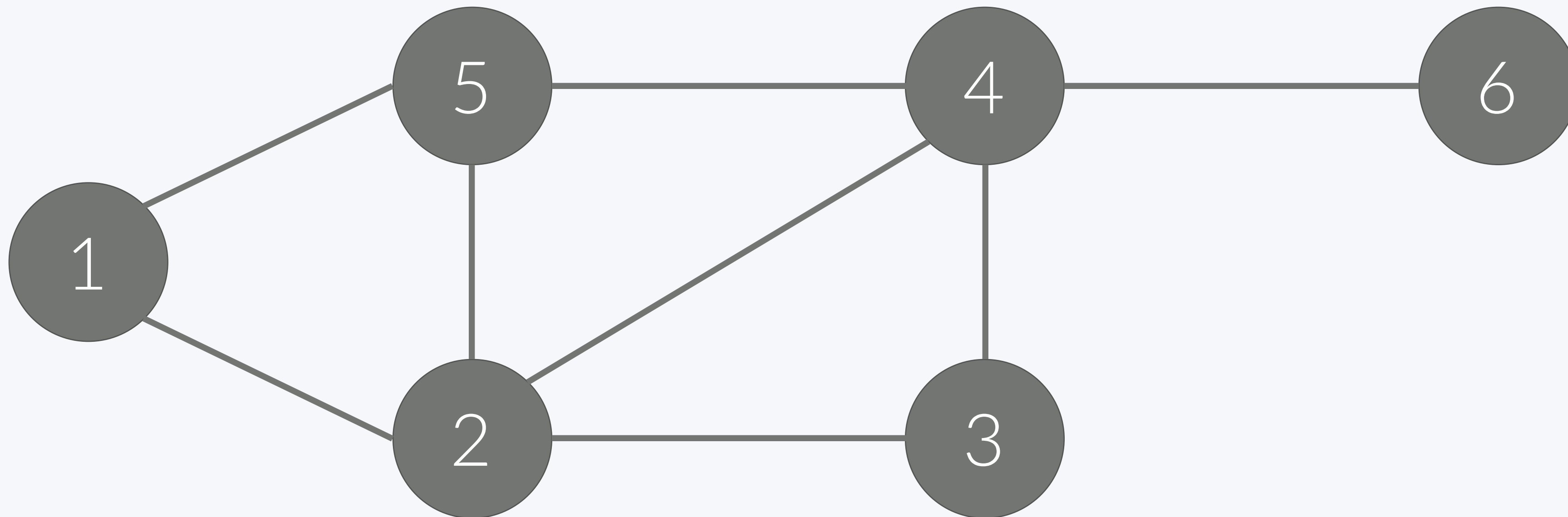
인접 리스트

Adjacency-list

$A[i] = i$ 와 연결된 정점
링크드 리스트

25

- 링크드 리스트를 이용해서 구현한다.
- $A[i] = i$ 와 연결된 정점을 링크드 리스트로 포함하고 있음



인접 리스트

Adjacency-list

A[1] 2 5

A[2] 1 3 4 5

A[3] 2 4

A[4] 3 5 2 6

A[5] 1 2 4

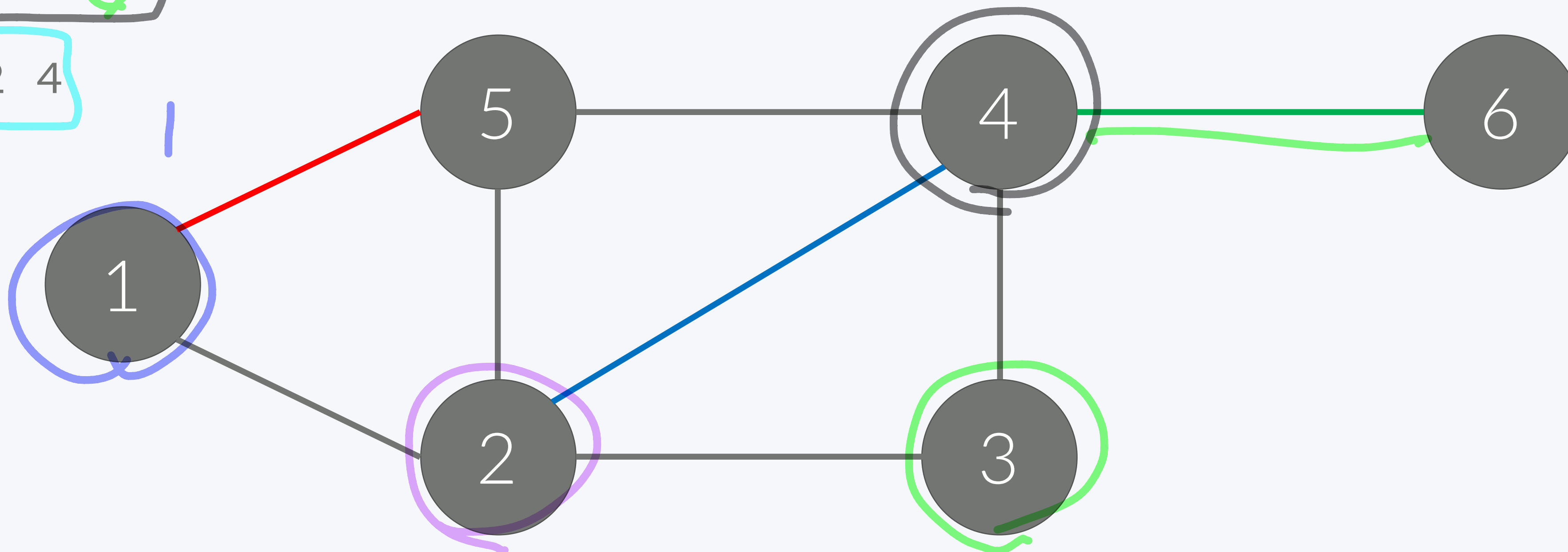
A[6] 4

정점 X

모든 간선 | (한번,

V^2

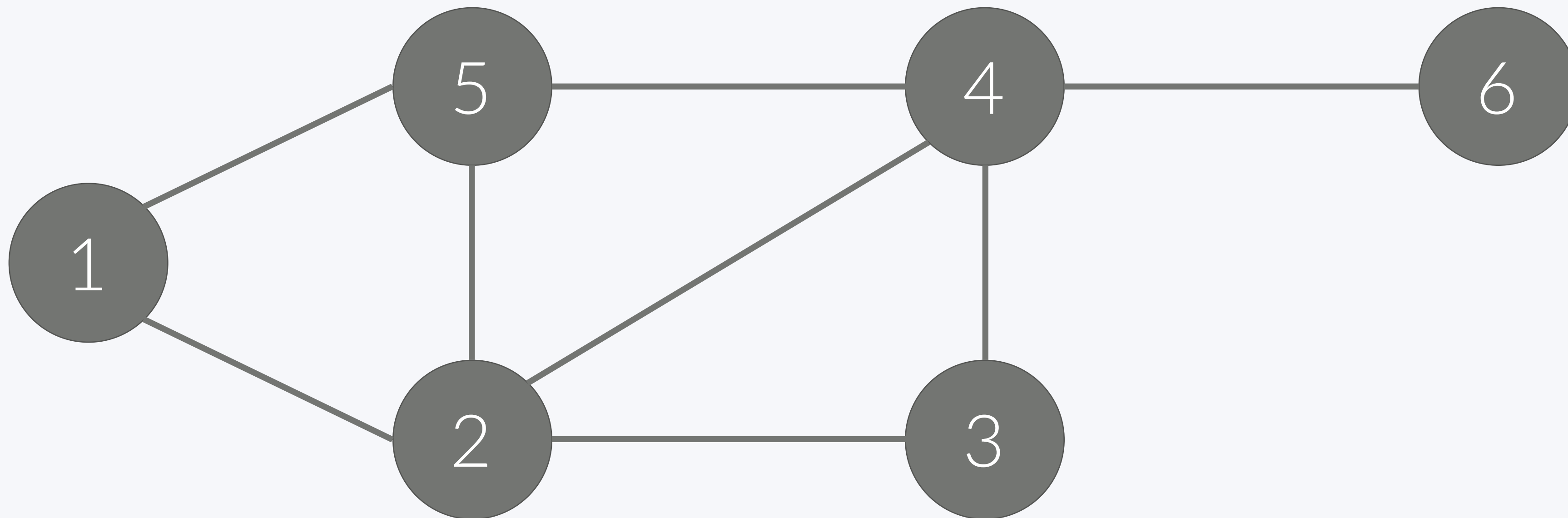
$O(E)$



인접 리스트

Adjacency-list

- 링크드 리스트는 구현하는데 시간이 오래걸리기 때문에, 주로 vector와 같이 길이를 변경할 수 있는 배열을 이용해서 구현한다.



인접 리스트

Adjacency-list

$\text{int}[10]$

→ vector<int> a[10]
 vector<int> a[10]

28

```
#include <stdio>
```

```
#include <vector>
```

```
using namespace std;
```

```
vector<int> a[10];
```

```
int main() {
```

```
    int n, m;
```

```
    scanf("%d %d", &n, &m);
```

```
    for (int i=0; i<m; i++) {
```

```
        int u, v;
```

```
        scanf("%d %d", &u, &v);
```

```
        a[u].push_back(v); a[v].push_back(u);
```

```
    }
```

```
}
```

인접 리스트

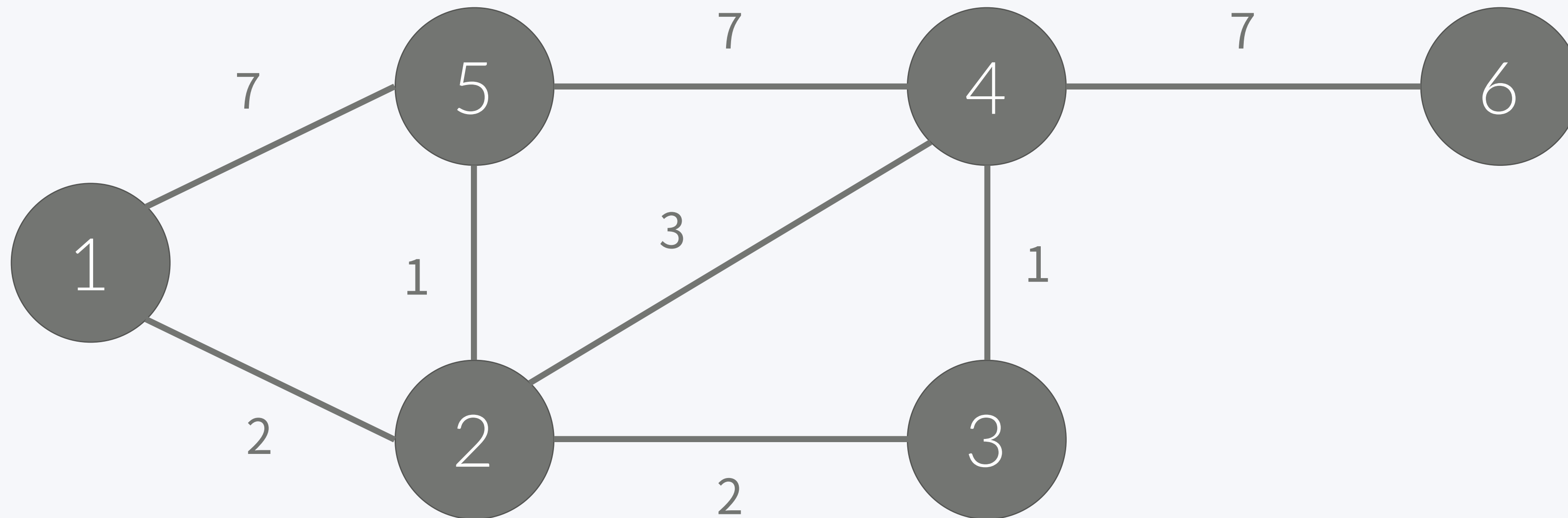
Adjacency-list

```
#include <cstdio>
#include <vector>
using namespace std;
int main() {
    int n, m;
    scanf("%d %d", &n, &m);
    vector<vector<int>> a(n+1);
    for (int i=0; i<m; i++) {
        int u, v;
        scanf("%d %d", &u, &v);
        a[u].push_back(v); a[v].push_back(u);
    }
}
```

인접 리스트

Adjacency-list

- 링크드 리스트를 이용해서 구현한다.
- $A[i] = i$ 와 연결된 정점과 그 간선의 가중치를 링크드 리스트로 포함하고 있음



인접 리스트

Adjacency-list

$O(E)$

31

A[1] (2,2) (5,7)

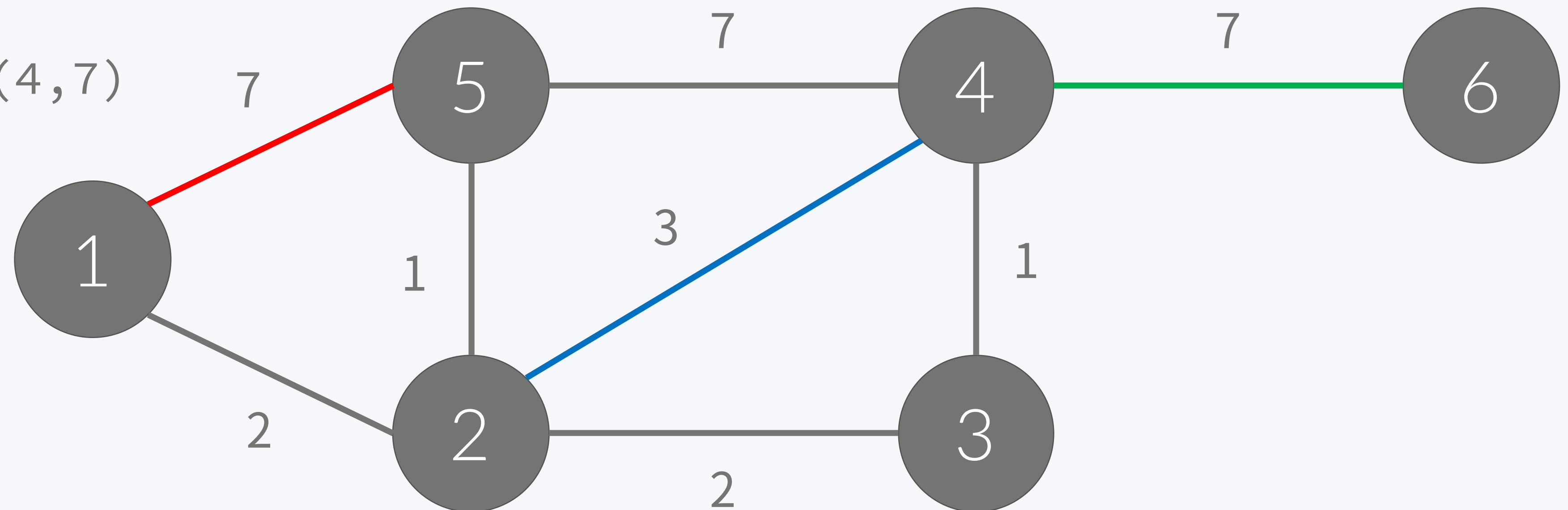
A[2] (1,2) (3,2) (4,3) (5,1)

A[3] (2,2) (4,1)

A[4] (3,1) (5,7) (2,3) (6,7)

A[5] (1,7) (2,1) (4,7)

A[6] (4,7)



인접 리스트

Adjacency-list

```
#include <cstdio>
#include <vector>
using namespace std;
vector<pair<int,int>> a[10];
int main() {
    int n,m;
    scanf("%d %d",&n,&m);
    for (int i=0; i<m; i++) {
        int u,v,w;
        scanf("%d %d %d",&u,&v,&w);
        a[u].push_back(make_pair(v,w)); a[v].push_back(make_pair(u,w));
    }
}
```



공간 복잡도

Space Complexity

- 인접 행렬: $O(V^2)$

- 인접 리스트: $O(E)$

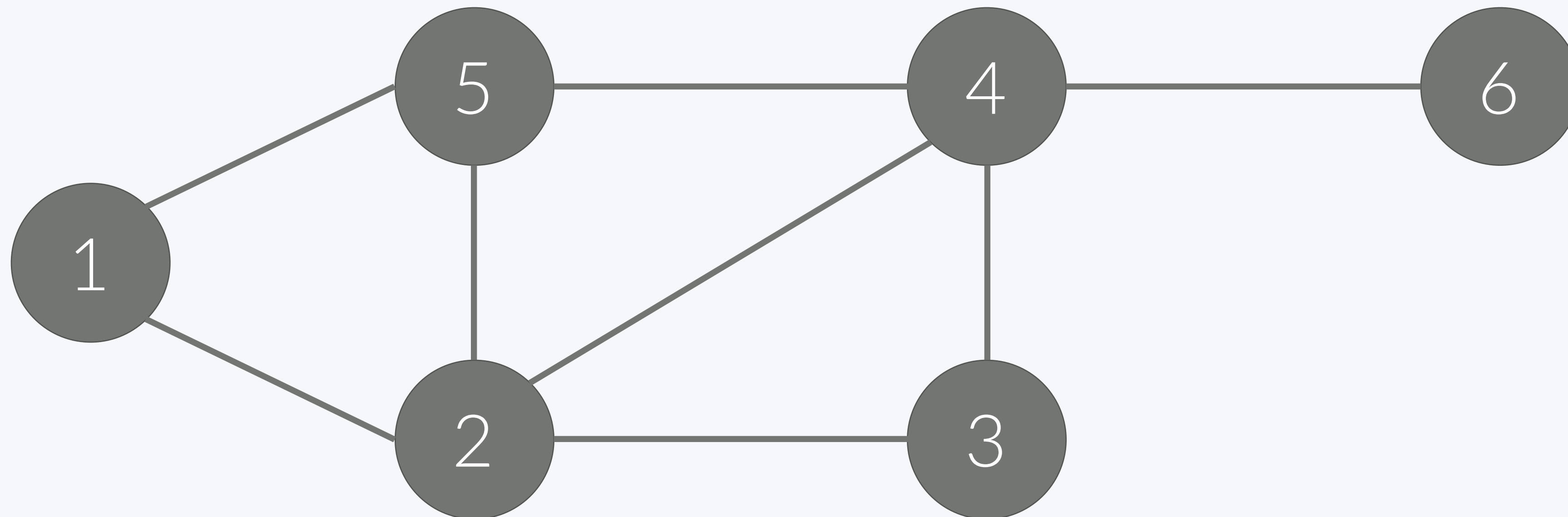
← 알고

간선 리스트

Edge-list

- 배열을 이용해서 구현한다.
- 간선을 모두 저장하고 있다.

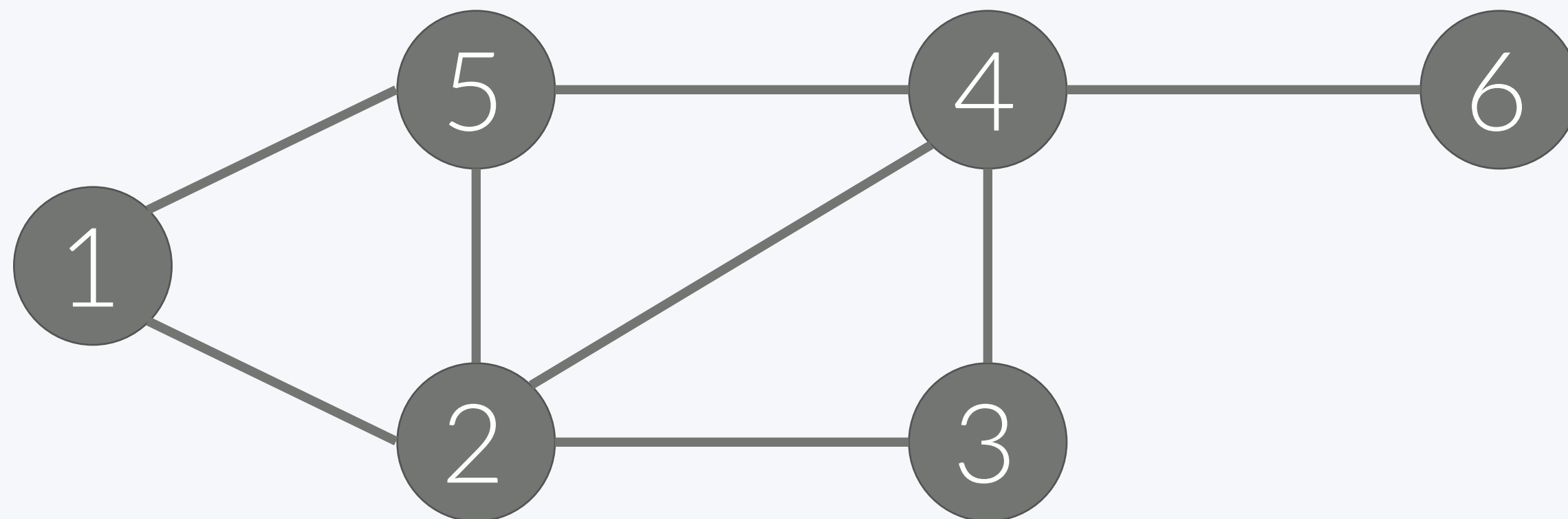
간선 리스트



간선 리스트

Edge List

- 배열을 이용해서 구현한다.
- 간선을 모두 저장하고 있다.
- E라는 배열에 간선을 모두 저장



$$\boxed{8} \times 2 = 16$$

Int E[16]

$$E[0] = 1 \ 2$$

$$E[1] = 1 \ 5$$

$$E[2] = 2 \ 3$$

$$E[3] = 2 \ 4$$

$$E[4] = 2 \ 5$$

$$E[5] = 5 \ 4$$

$$E[6] = 4 \ 3$$

$$E[7] = 4 \ 6$$

$$E[8] = 2 \ 1$$

$$E[9] = 5 \ 1$$

$$E[10] = 3 \ 2$$

$$E[11] = 4 \ 2$$

$$E[12] = 5 \ 2$$

$$E[13] = 4 \ 5$$

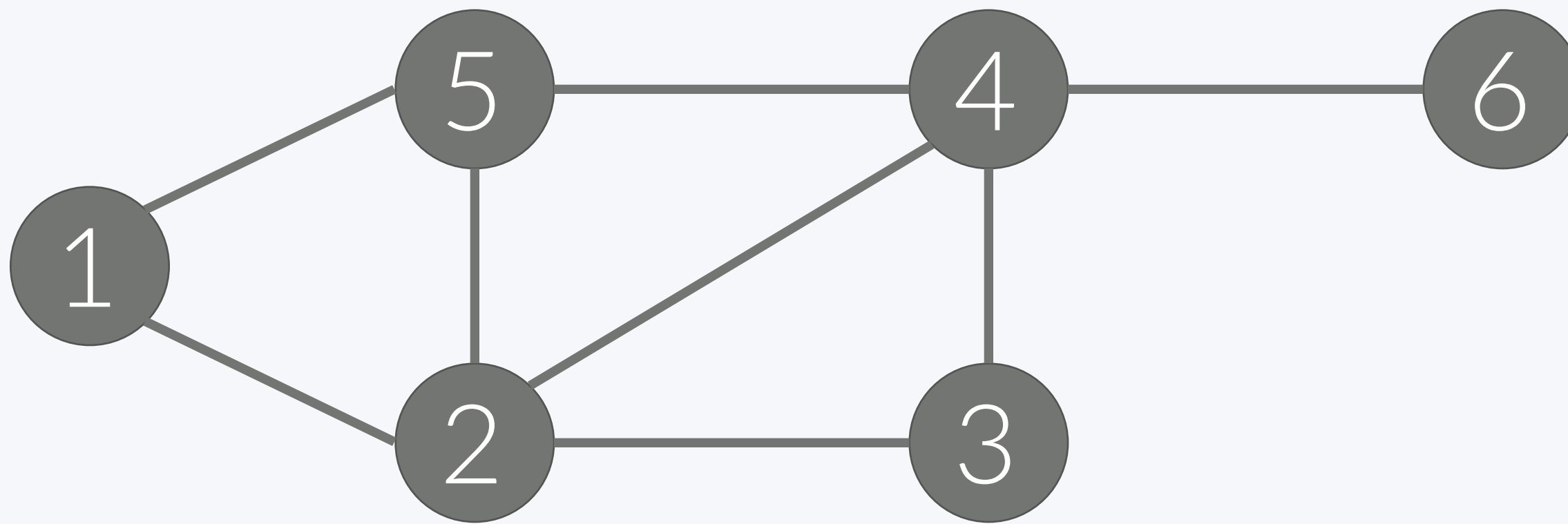
$$E[14] = 3 \ 4$$

$$E[15] = 6 \ 4$$

간선 리스트

Edge List

- 각 간선의 앞 정점을 기준으로 개수를 센다.



i	0	1	2	3	4	5	6
cnt[i]	0	2	4	2	4	3	1

$$\begin{aligned} E[0] &= 1 \ 2 \\ E[1] &= 1 \ 5 \end{aligned}$$

$$\begin{aligned} E[2] &= 2 \ 1 \\ E[3] &= 2 \ 3 \\ E[4] &= 2 \ 4 \\ E[5] &= 2 \ 5 \end{aligned}$$

$$\begin{aligned} E[6] &= 3 \ 2 \\ E[7] &= 3 \ 4 \end{aligned}$$

$$\begin{aligned} E[8] &= 4 \ 2 \\ E[9] &= 4 \ 3 \end{aligned}$$

$$\begin{aligned} E[10] &= 4 \ 5 \\ E[11] &= 4 \ 6 \end{aligned}$$

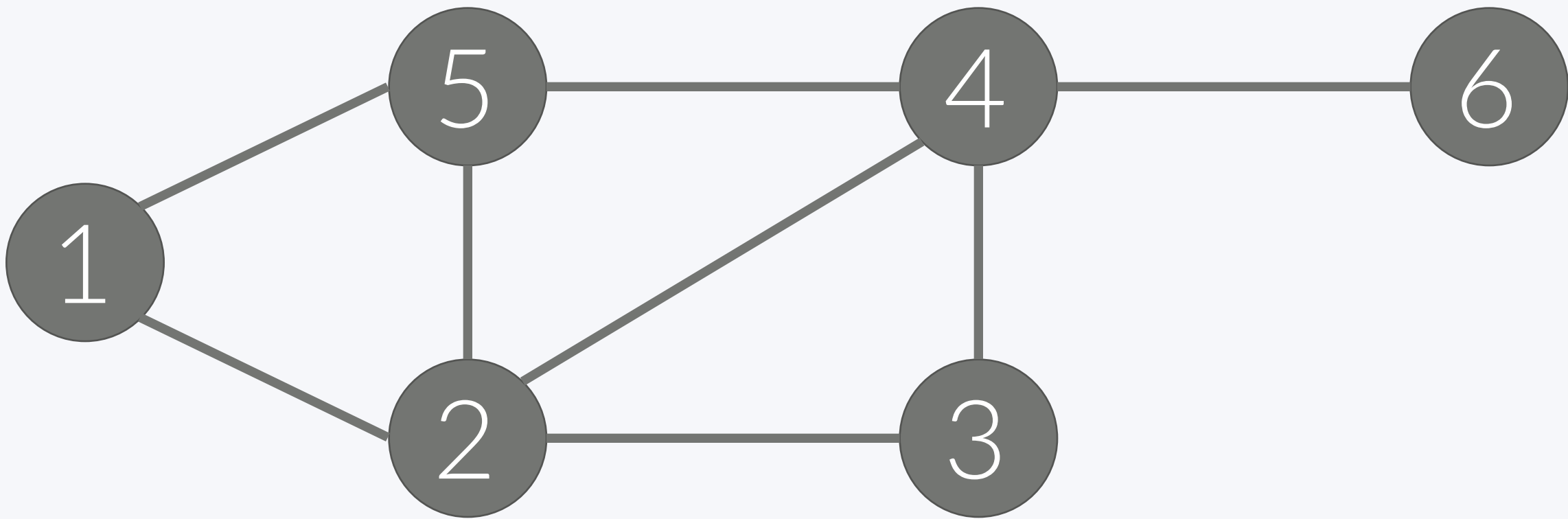
$$\begin{aligned} E[12] &= 5 \ 1 \\ E[13] &= 5 \ 2 \end{aligned}$$

$$\begin{aligned} E[14] &= 5 \ 4 \\ E[15] &= 6 \ 4 \end{aligned}$$

간선 리스트

Edge List

```
for (int i=0; i<m; i++) {  
    cnt[e[i][0]] += 1;  
}
```



i	0	1	2	3	4	5	6
cnt[i]	0	2	4	2	4	3	1

- E[0] = 1 2

E[1] = 1 5

E[2] = 2 1

E[3] = 2 3

E[4] = 2 4

E[5] = 2 5

E[6] = 3 2

E[7] = 3 4
- E[8] = 4 2

E[9] = 4 3

E[10] = 4 5

E[11] = 4 6

E[12] = 5 1

E[13] = 5 2

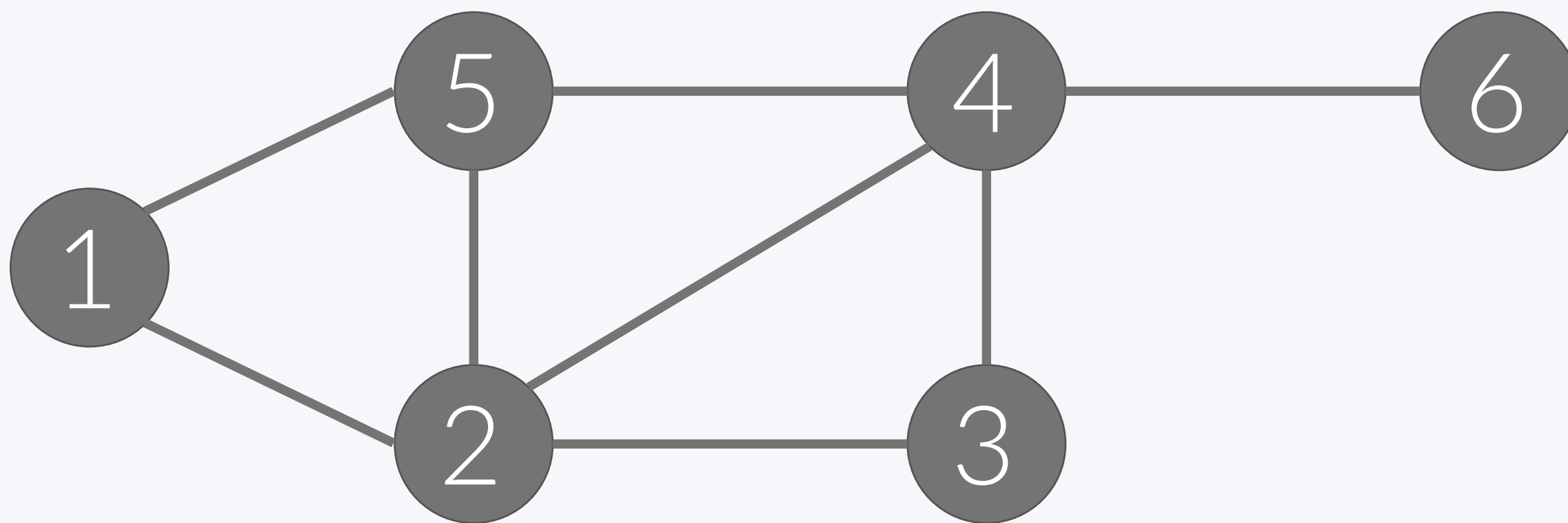
E[14] = 5 4

E[15] = 6 4

간선 리스트

Edge List

```
for (int i=1; i<=n; i++) {  
    cnt[i] = cnt[i-1] + cnt[i];  
}
```



i	0	1	2	3	4	5	6
cnt[i]	0	2	6	8	12	15	16

E[0] = 1 2

E[1] = 1 5

E[2] = 2 1

E[3] = 2 3

E[4] = 2 4

E[5] = 2 5

E[6] = 3 2

E[7] = 3 4

E[8] = 4 2

E[9] = 4 3

E[10] = 4 5

E[11] = 4 6

E[12] = 5 1

E[13] = 5 2

E[14] = 5 4

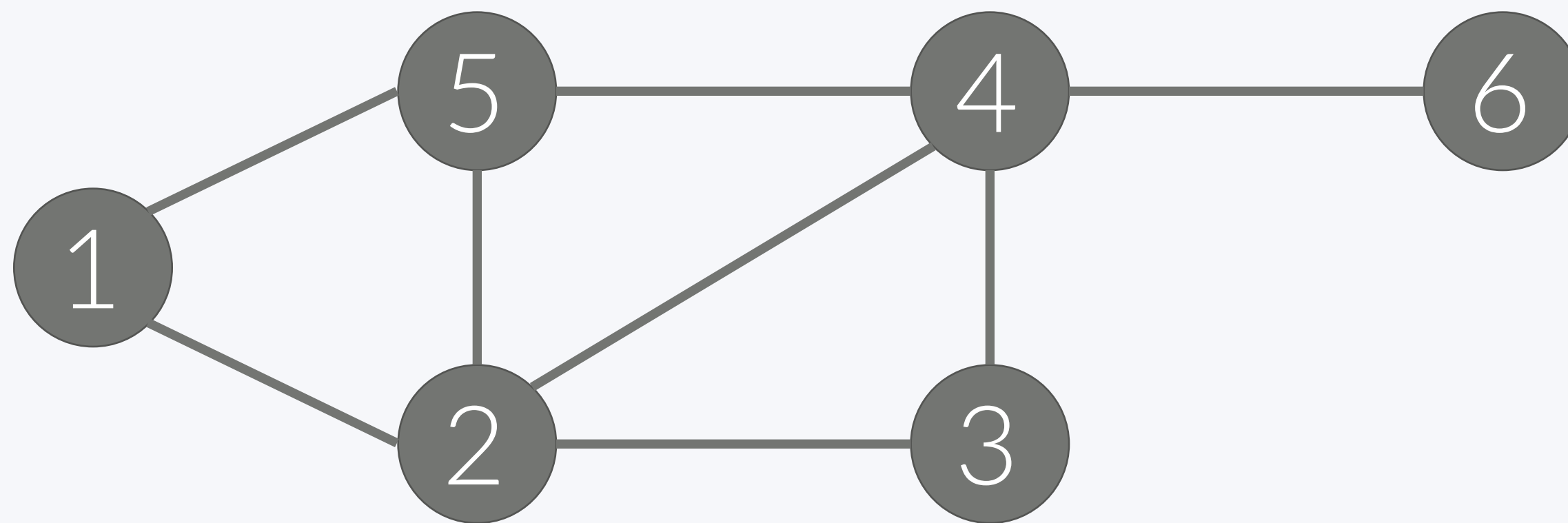
E[15] = 6 4

간선 리스트

Edge List

39

- i 번 정점과 연결된 간선은
- E배열에서 $\boxed{\text{cnt}[i-1]}$ 부터 $\boxed{\text{cnt}[i]-1}$ 까지이다.



i	0	1	2	3	4	5	6
cnt[i]	0	2	6	8	12	15	16

$$E[0] = 1 \ 2$$

$$E[1] = 1 \ 5$$

$$E[2] = 2 \ 1$$

$$E[3] = 2 \ 3$$

$$E[4] = 2 \ 4$$

$$E[5] = 2 \ 5$$

$$E[6] = 3 \ 2$$

$$E[7] = 3 \ 4$$

$$E[8] = 4 \ 2$$

$$E[9] = 4 \ 3$$

$$E[10] = 4 \ 5$$

$$E[11] = 4 \ 6$$

$$E[12] = 5 \ 1$$

$$E[13] = 5 \ 2$$

$$E[14] = 5 \ 4$$

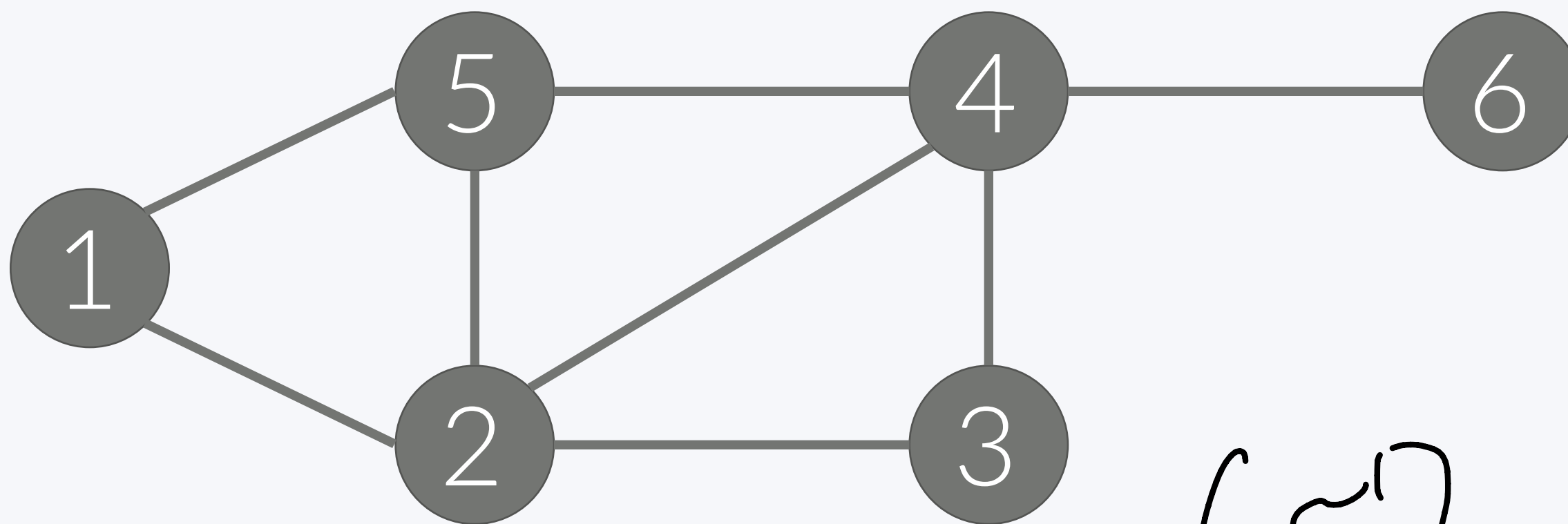
$$E[15] = 6 \ 4$$

간선 리스트

Edge List

40

- 3번 정점: $\text{cnt}[2] \sim \text{cnt}[3]-1$



i	0	1	2	3	4	5	6
cnt[i]	0	2	6	8	12	15	16

$$E[0] = 1 \ 2$$

$$E[1] = 1 \ 5$$

$$E[2] = 2 \ 1$$

$$E[3] = 2 \ 3$$

$$E[4] = 2 \ 4$$

$$E[5] = 2 \ 5$$

$$E[6] = 3 \ 2$$

$$E[7] = 3 \ 4$$

$$E[8] = 4 \ 2$$

$$E[9] = 4 \ 3$$

$$E[10] = 4 \ 5$$

$$E[11] = 4 \ 6$$

$$E[12] = 5 \ 1$$

$$E[13] = 5 \ 2$$

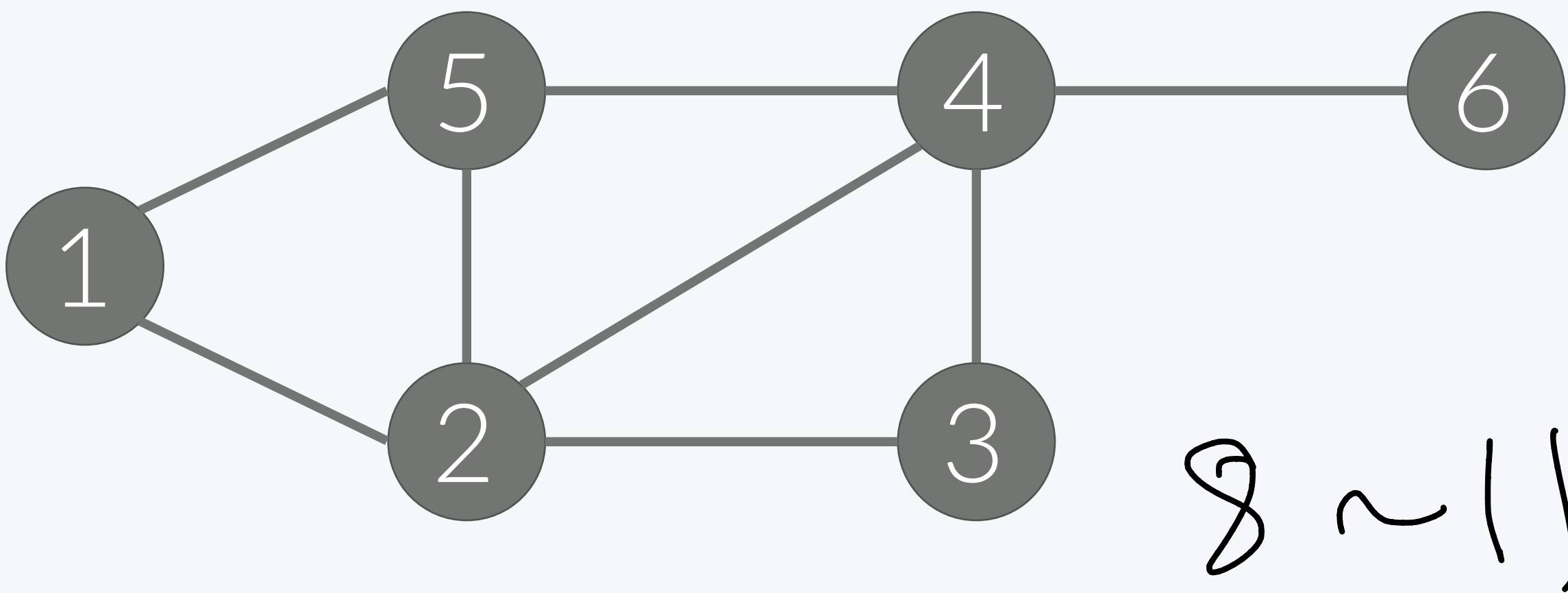
$$E[14] = 5 \ 4$$

$$E[15] = 6 \ 4$$

간선 리스트

Edge List

- 4번 정점: $\text{cnt}[3] \sim \text{cnt}[4]-1$



i	0	1	2	3	4	5	6
cnt[i]	0	2	6	8	12	15	16

~~STL~~

$O(E)$

$E[0] = 1\ 2$	$E[8] = 4\ 2$
$E[1] = 1\ 5$	$E[9] = 4\ 3$
$E[2] = 2\ 1$	$E[10] = 4\ 5$
$E[3] = 2\ 3$	$E[11] = 4\ 6$
$E[4] = 2\ 4$	$E[12] = 5\ 1$
$E[5] = 2\ 5$	$E[13] = 5\ 2$
$E[6] = 3\ 2$	$E[14] = 5\ 4$
$E[7] = 3\ 4$	$E[15] = 6\ 4$

그래프의 탐색

그래프의 탐색

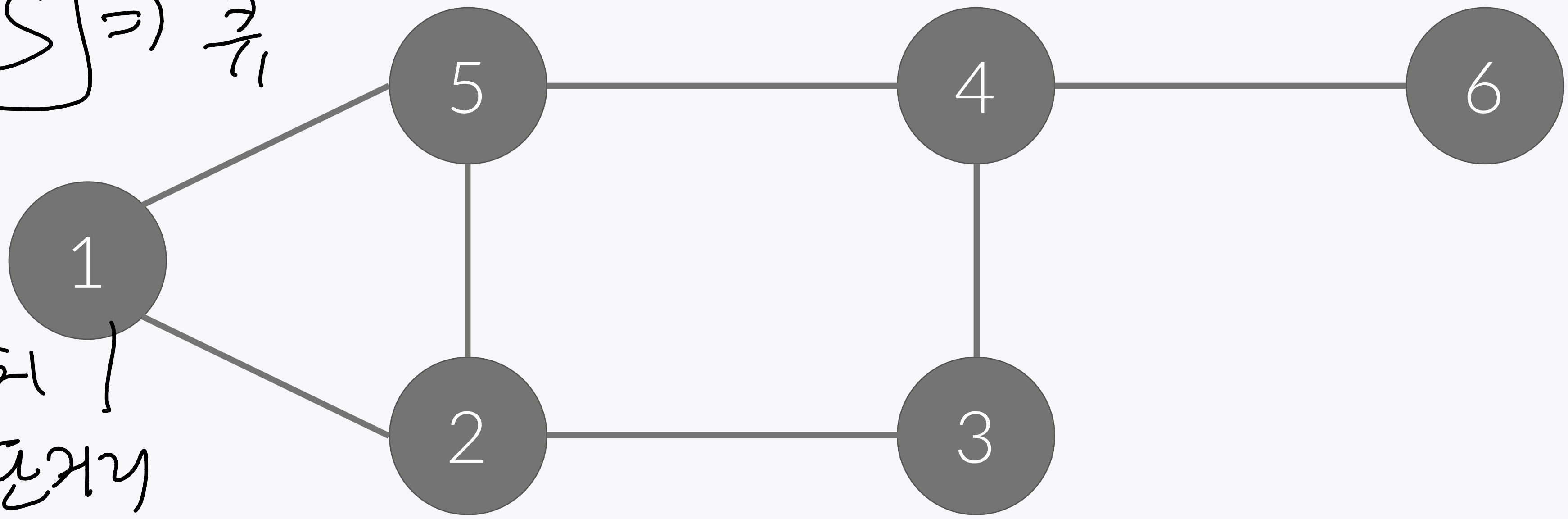
DFS, BFS

목적: 모든 정점을 1번씩 방문

- DFS: 깊이 우선 탐색 → 최대한 깊숙히 많이
- BFS: 너비 우선 탐색 → 최대한 넓게 가는데

DFS ⇒ 스택
BFS ⇒ 큐

↓
모든 가능한
최단거리

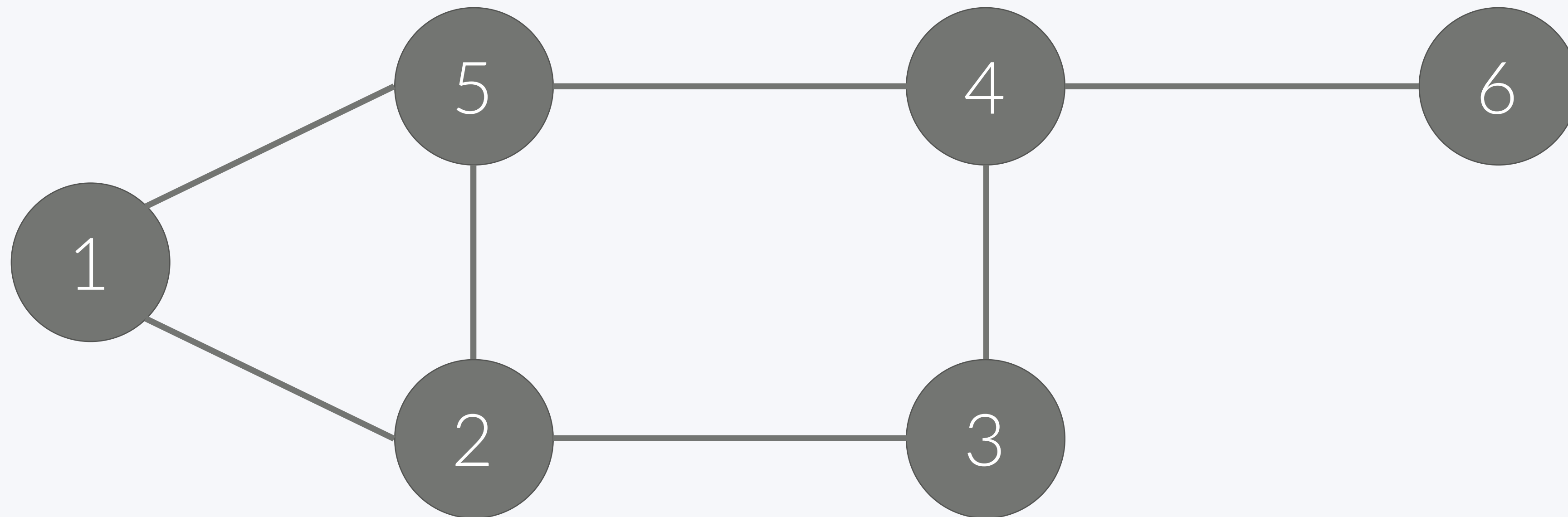


깊이 우선 탐색

Depth First Search

44

- 스택을 이용해서 갈 수 있는 만큼 최대한 많이 가고
- 갈 수 없으면 이전 정점으로 돌아간다.



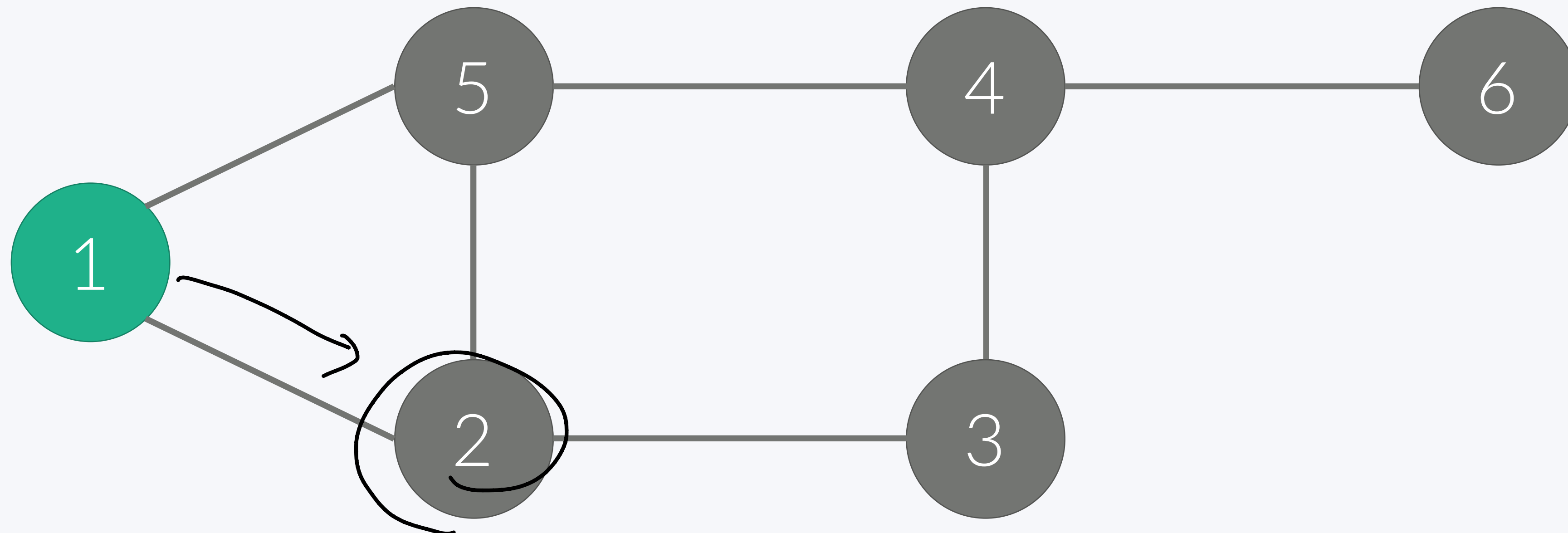
깊이 우선 탐색

Depth First Search

- 현재 정점: 1
- 순서: 1
- 스택: 1 2

check[i] : 0 또는 1
 ↑ ↑
 아직 방문 안함 방문함

i	1	2	3	4	5	6
check[i]	1	0	0	0	0	0



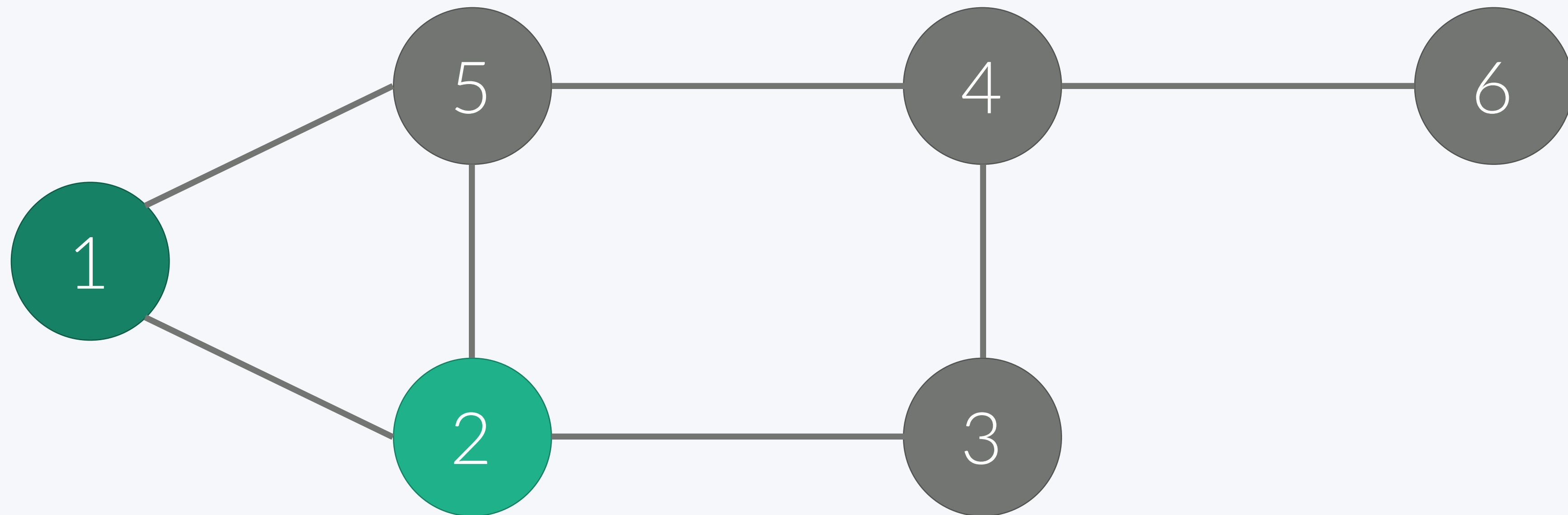
깊이 우선 탐색

46

Depth First Search

- 현재 정점: 2
- 순서: 1 2
- 스택: 1 2 3

i	1	2	3	4	5	6
check[i]	1	1	0	0	0	0



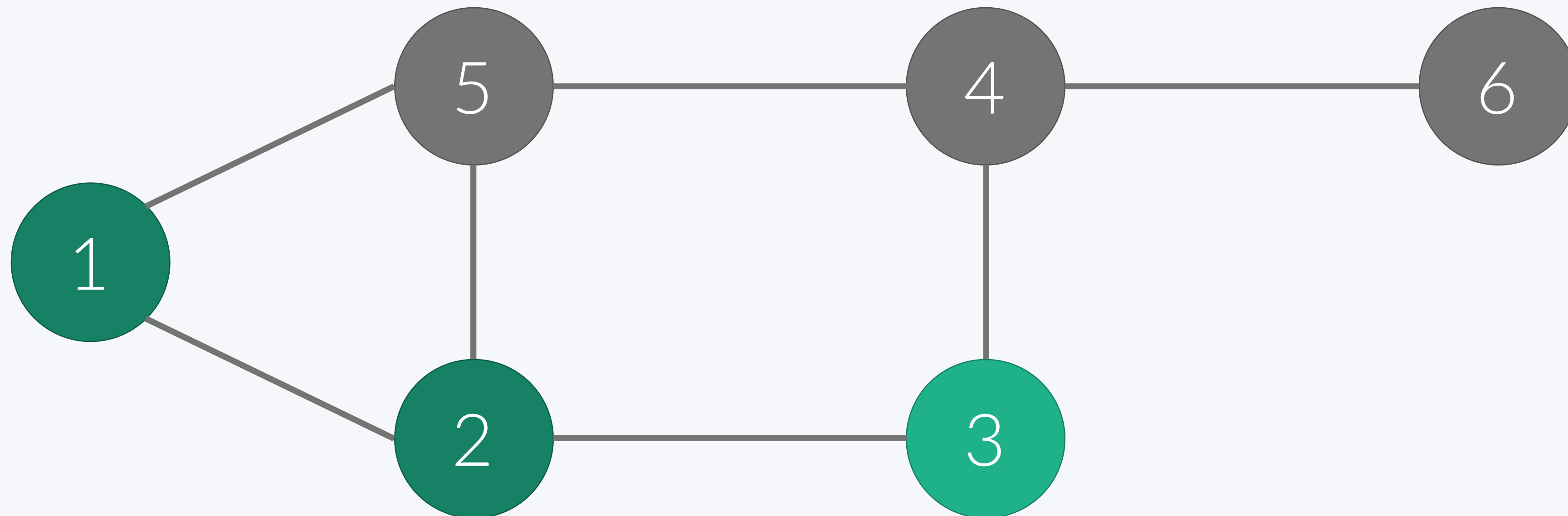
깊이 우선 탐색

47

Depth First Search

- 현재 정점: 3
- 순서: 1 2 3
- 스택: 1 2 3 4

i	1	2	3	4	5	6
check[i]	1	1	1	0	0	0



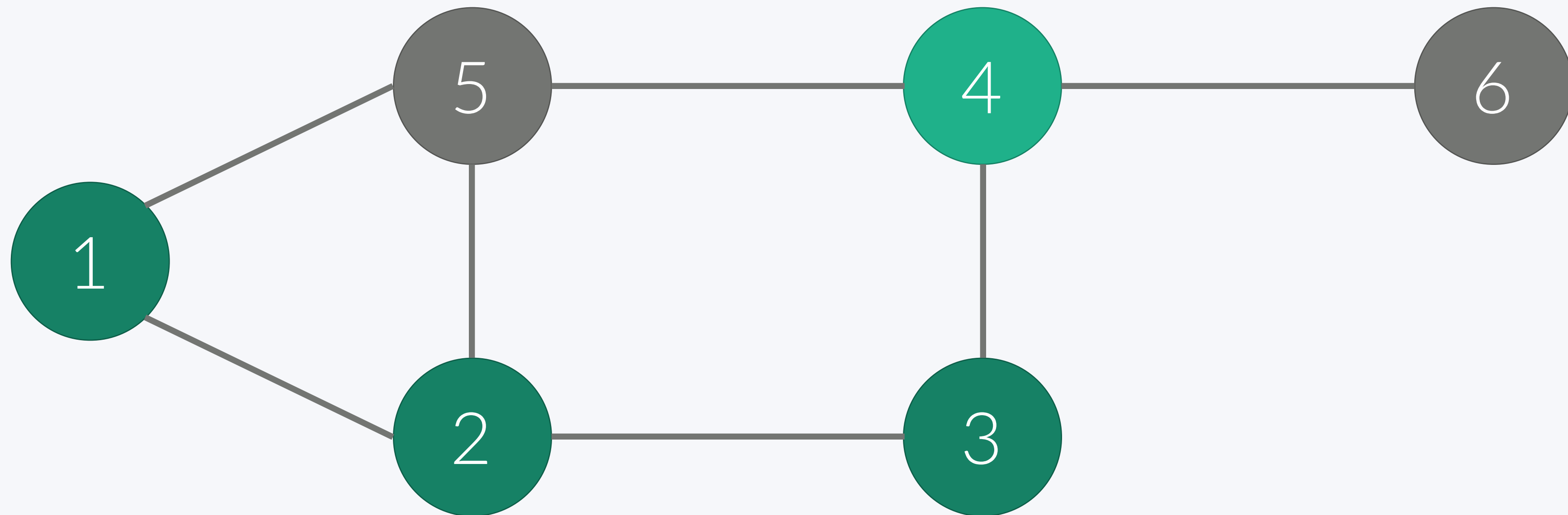
깊이 우선 탐색

48

Depth First Search

- 현재 정점: 4
- 순서: 1 2 3 4
- 스택: 1 2 3 4 5

i	1	2	3	4	5	6
check[i]	1	1	1	1	0	0



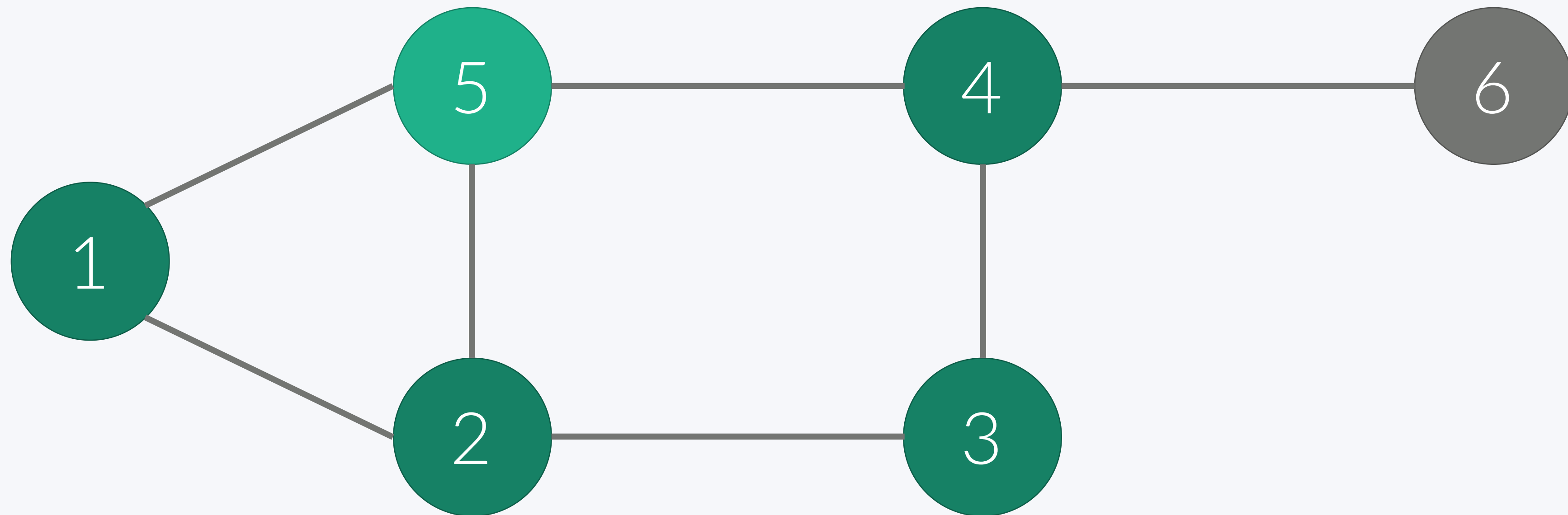
깊이 우선 탐색

49

Depth First Search

- 현재 정점: 5
- 순서: 1 2 3 4 5
- 스택: 1 2 3 4 5

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	0



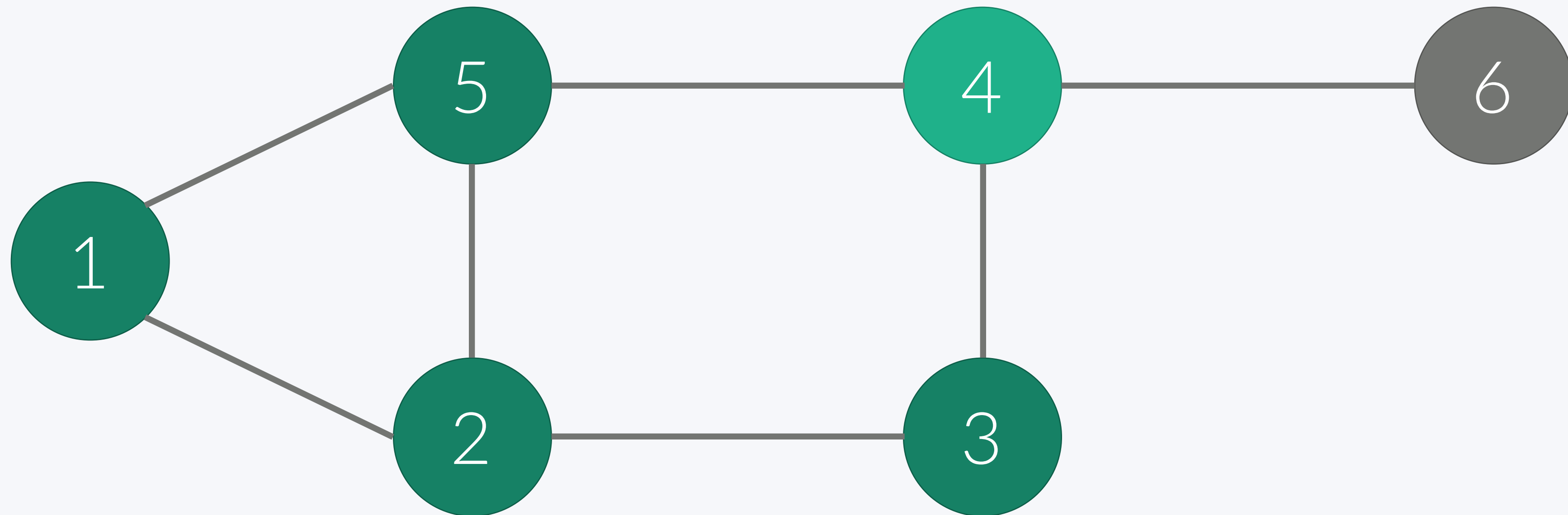
깊이 우선 탐색

50

Depth First Search

- 현재 정점: 4
- 순서: 1 2 3 4 5
- 스택: 1 2 3 4 6
- 5에서 더 갈 수 있는 것이 없기 때문에, 4로 돌아간다.

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	0



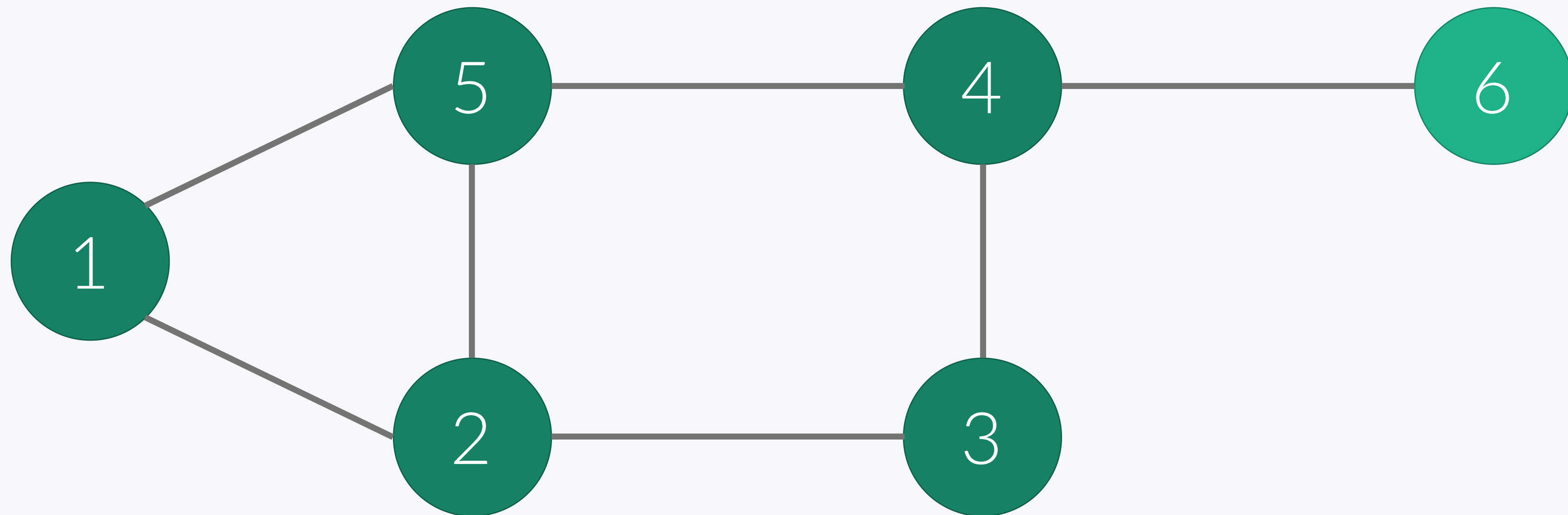
깊이 우선 탐색

51

Depth First Search

- 현재 정점: 6
- 순서: 1 2 3 4 5 6
- 스택: 1 2 3 4 6

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



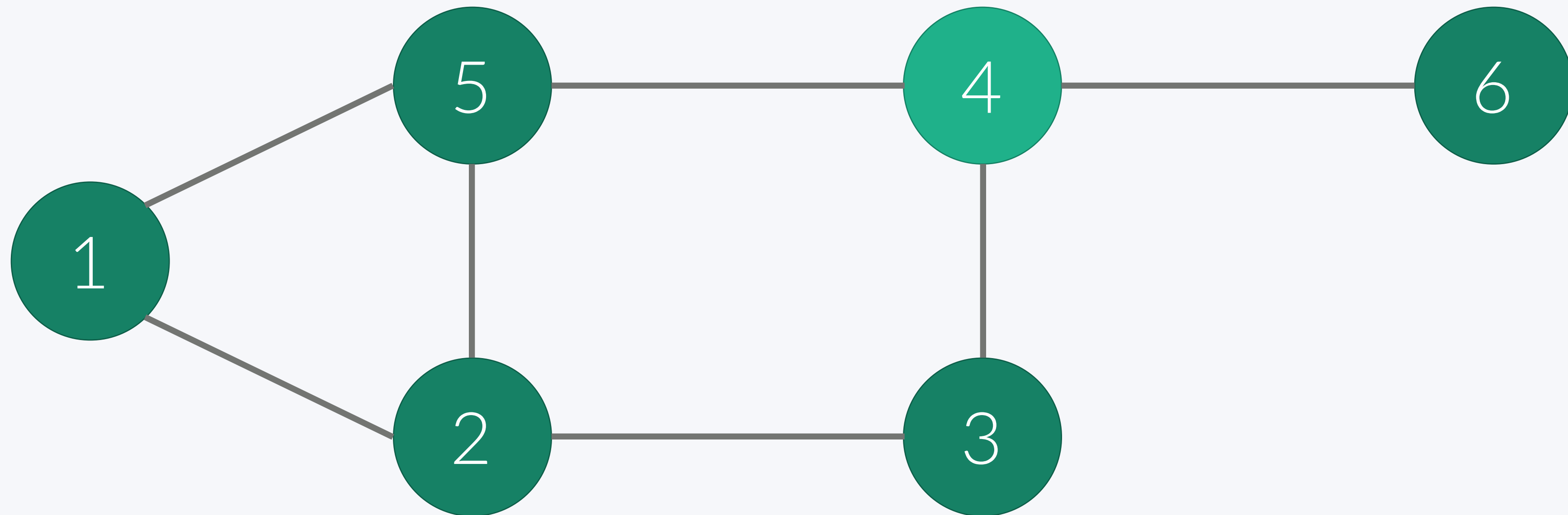
깊이 우선 탐색

52

Depth First Search

- 현재 정점: 4
- 순서: 1 2 3 4 5 6
- 스택: 1 2 3 4
- 6에서 갈 수 있는 것이 없기 때문에 4로 돌아간다.

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



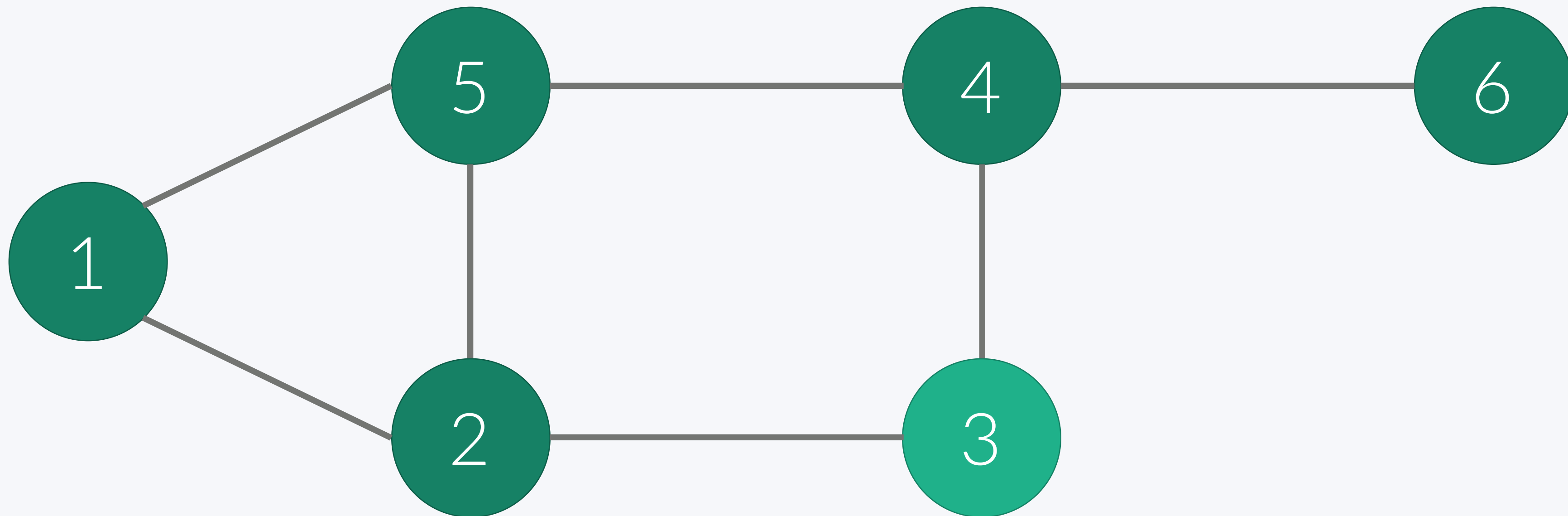
깊이 우선 탐색

53

Depth First Search

- 현재 정점: 3
- 순서: 1 2 3 4 5 6
- 스택: 1 2 3
- 4에서 갈 수 있는 것이 없기 때문에 3으로 돌아간다.

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



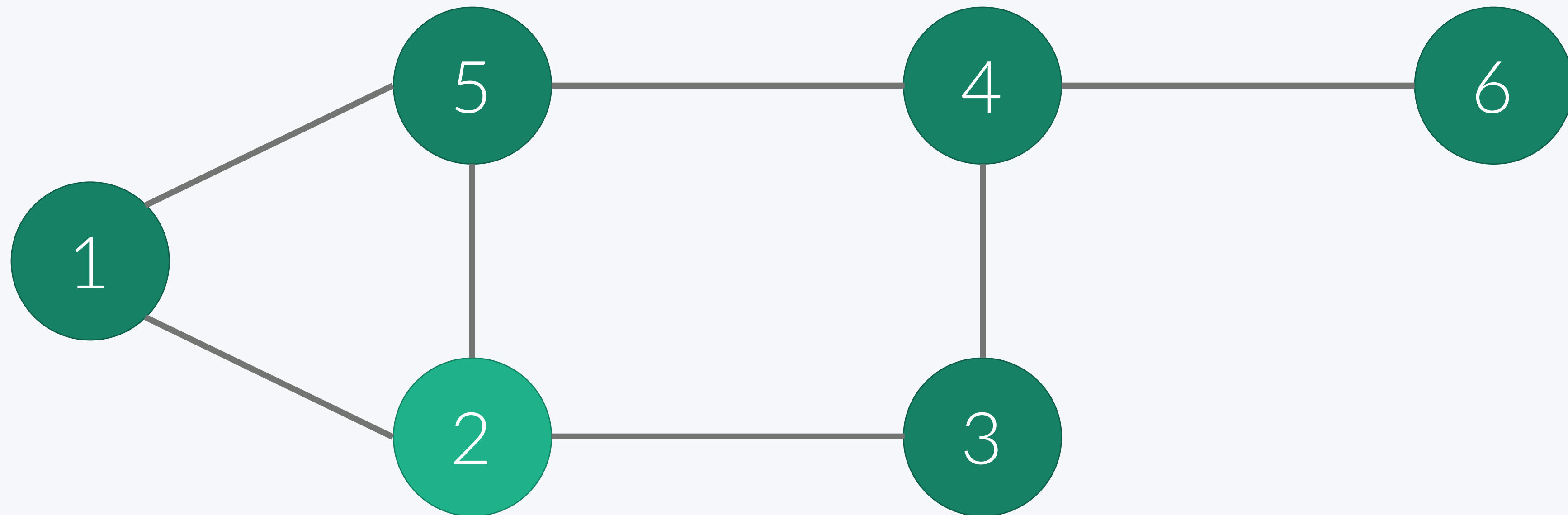
깊이 우선 탐색

54

Depth First Search

- 현재 정점: 2
- 순서: 1 2 3 4 5 6
- 스택: 1 2
- 3에서 갈 수 있는 것이 없기 때문에 2으로 돌아간다.

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



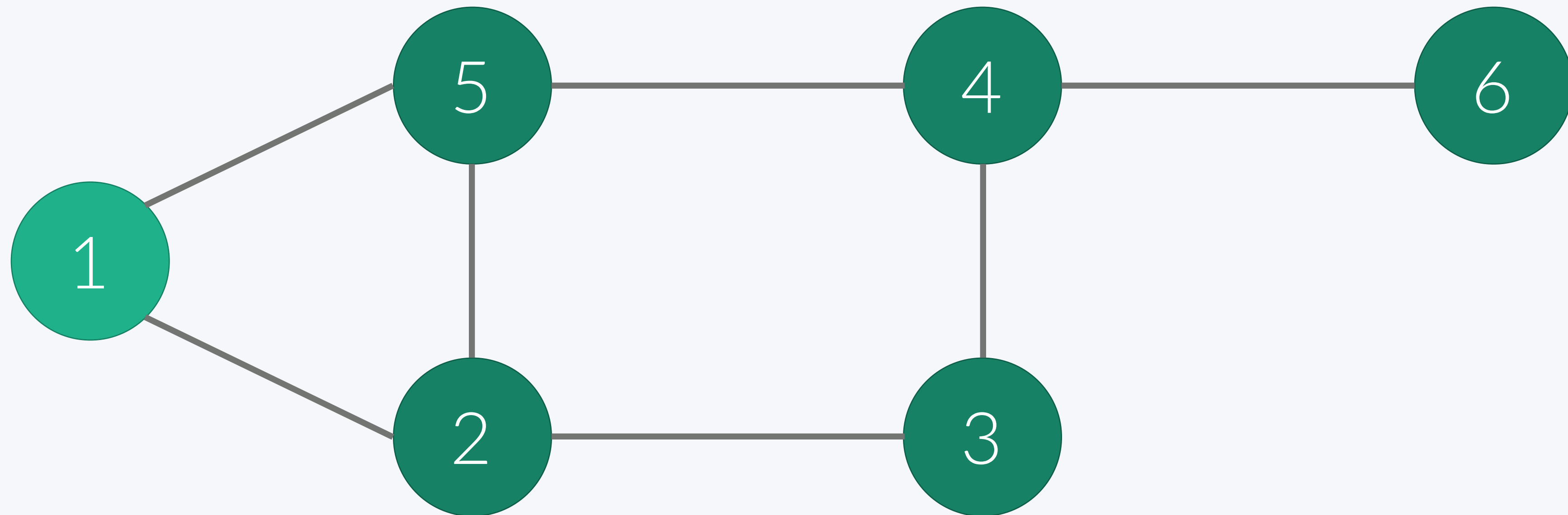
깊이 우선 탐색

55

Depth First Search

- 현재 정점: 1
- 순서: 1 2 3 4 5 6
- 스택: 1
- 2에서 갈 수 있는 것이 없기 때문에 1으로 돌아간다.

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



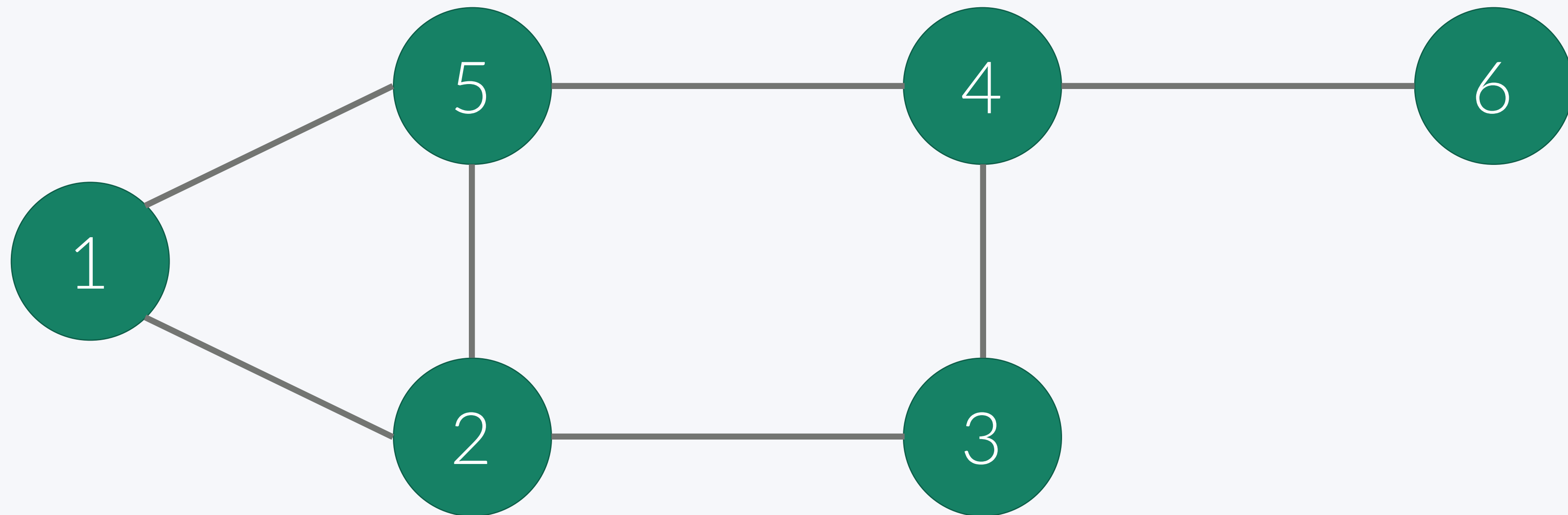
깊이 우선 탐색

56

Depth First Search

- 현재 정점:
- 순서: 1 2 3 4 5 6
- 스택:
- 탐색 종료

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



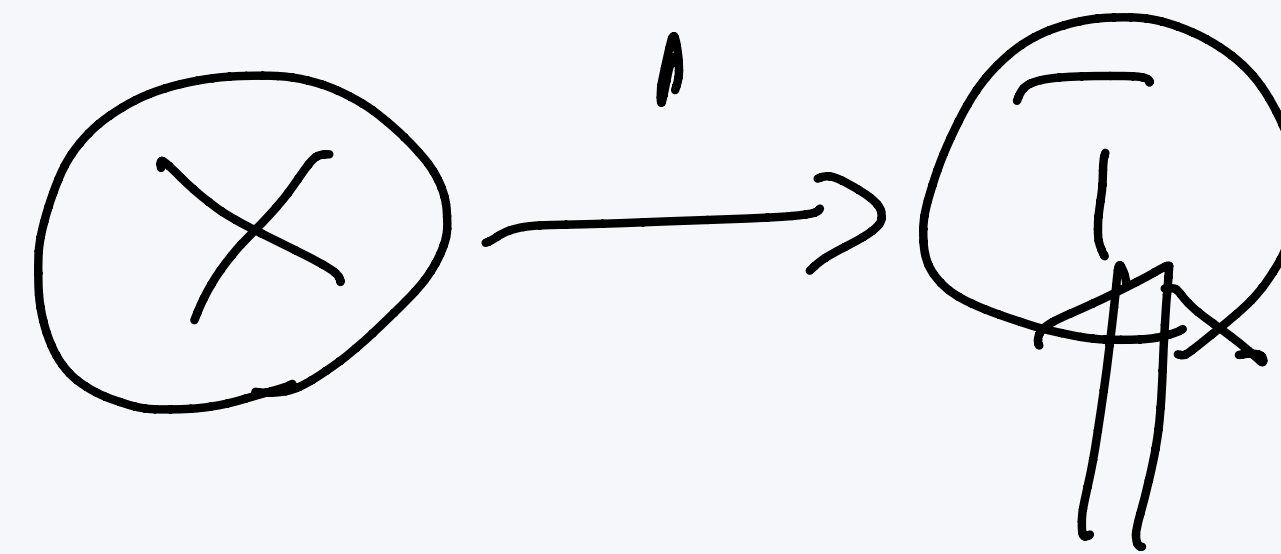
깊이 우선 탐색

Depth First Search

dfs(x) \times^2 방향

57

- 재귀 호출을 이용해서 구현할 수 있다.



```
void dfs(int x) {  
    check[x] = true;  
    printf("%d ", x);
```

```
    for (int i=1; i<=n; i++) {  
        if (a[x][i] == 1 && check[i] == false) {  
            dfs(i);  
        }  
    }  
}
```

다음 정점

dfs $O(V)$

- 인접 행렬을 이용한 구현

$$V \cdot O(V) = O(V^2)$$

깊이 우선 탐색

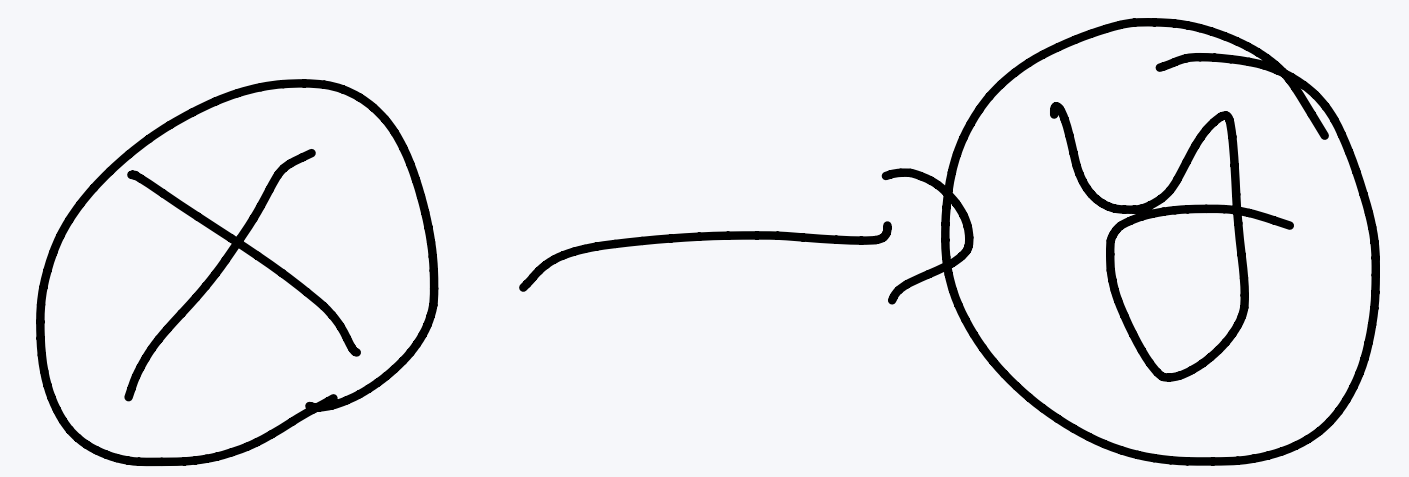
Depth First Search

- 재귀 호출을 이용해서 구현할 수 있다.

```
void dfs(int x) {  
    check[x] = true;  
    printf("%d ", x);  
    for (int i=0; i<a[x].size(); i++) {  
        int y = a[x][i];  
        if (check[y] == false) {  
            dfs(y);  
        }  
    }  
}
```

- 인접 리스트를 이용한 구현

adj: x와 연결된
정점



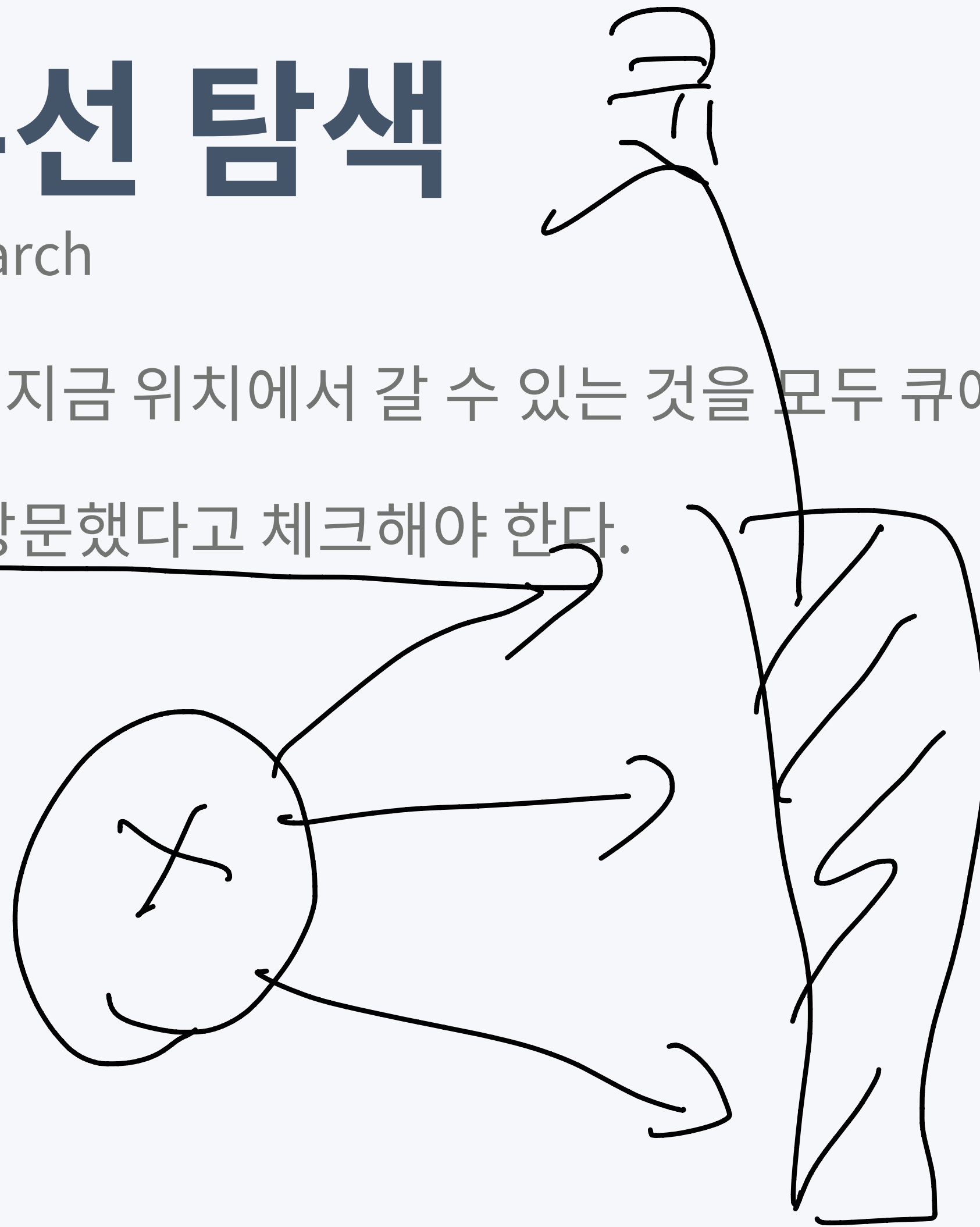
$$V + E$$
$$O(V + E)$$

너비 우선 탐색

Breadth First Search

59

- 큐를 이용해서 지금 위치에서 갈 수 있는 것을 모두 큐에 넣는 방식
- 큐에 넣을 때 방문했다고 체크해야 한다.



너비 우선 탐색

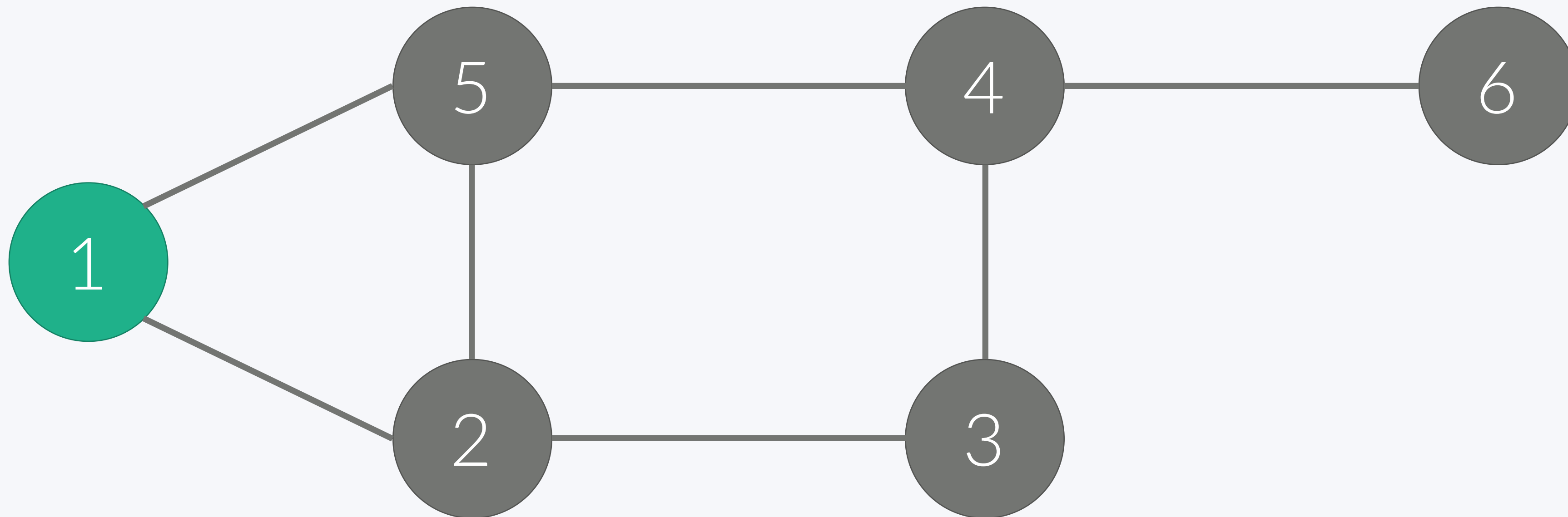
BFS

60

Breadth First Search

- 현재 정점: 1
- 순서: 1
- 큐: 1

i	1	2	3	4	5	6
check[i]	1	0	0	0	0	0



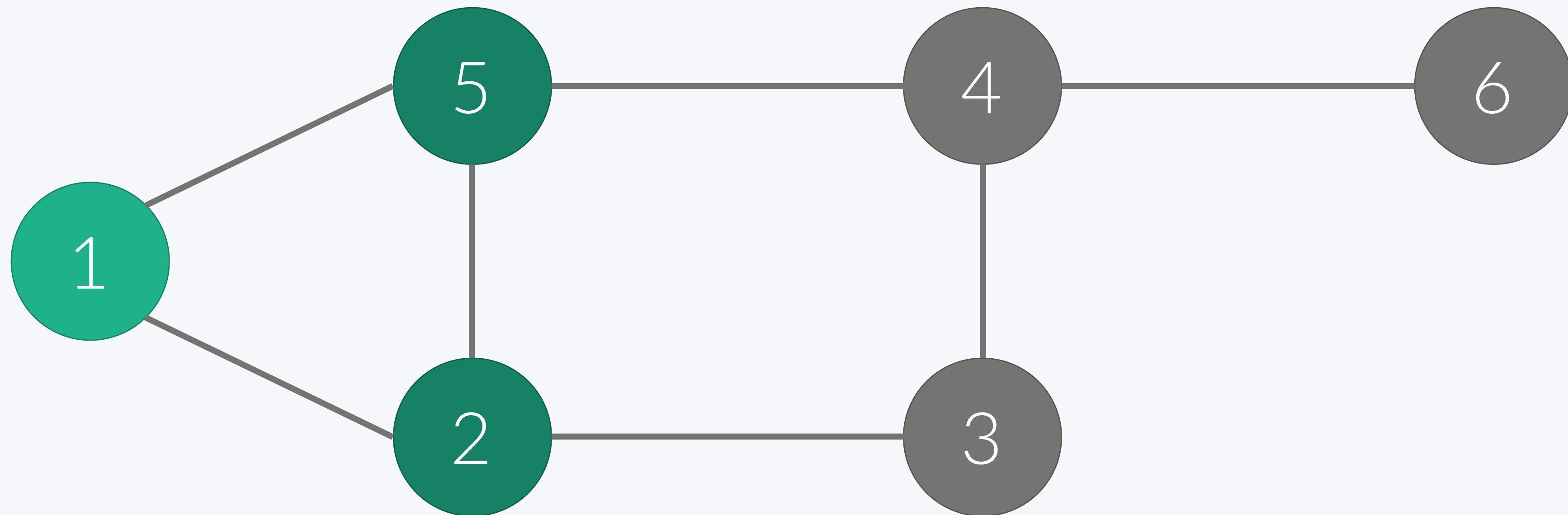
너비 우선 탐색

61

Breadth First Search

- 현재 정점: 1
- 순서: 1 2 5
- 큐: ~~1~~ 2 5
 ↑

i	1	2	3	4	5	6
check[i]	1	1	0	0	1	0



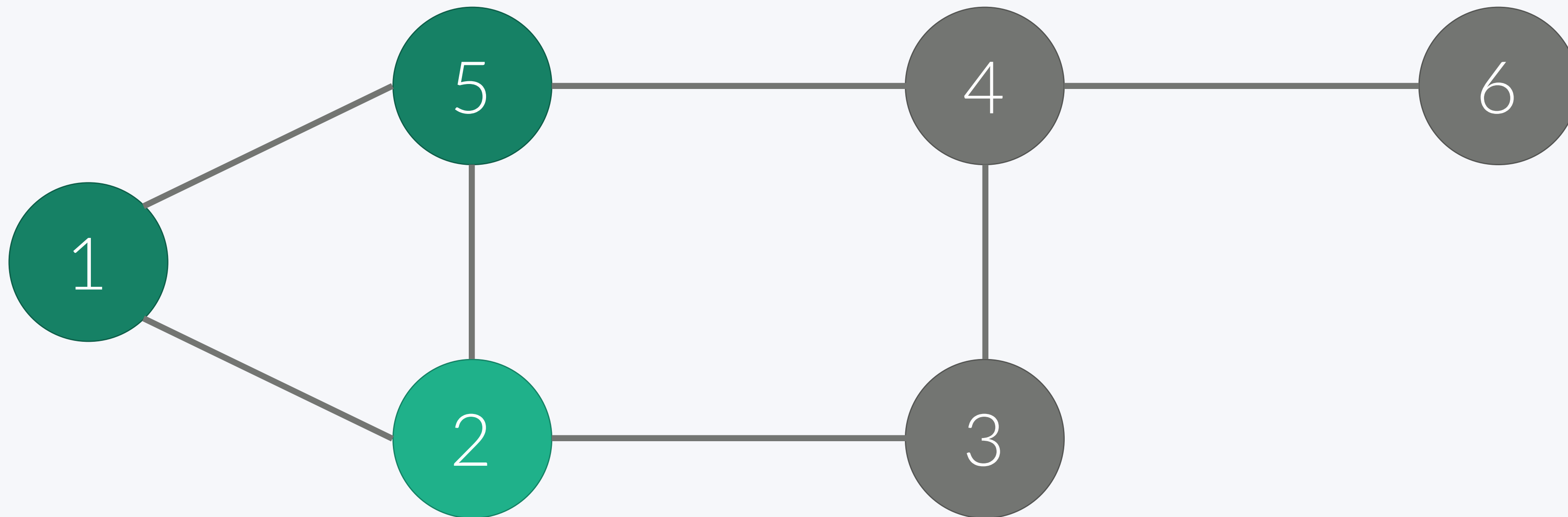
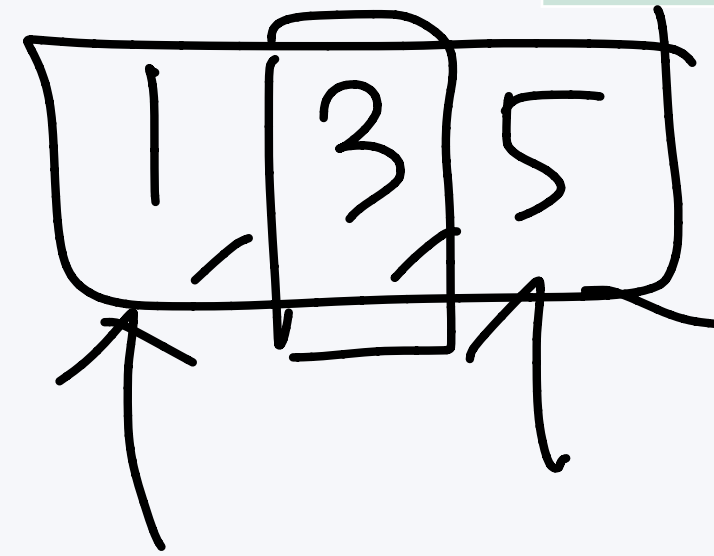
너비 우선 탐색

62

Breadth First Search

- 현재 정점: 2
- 순서: 1 2 5 3
- 큐: 2 5 3

i	1	2	3	4	5	6
check[i]	1	1	0	0	1	0



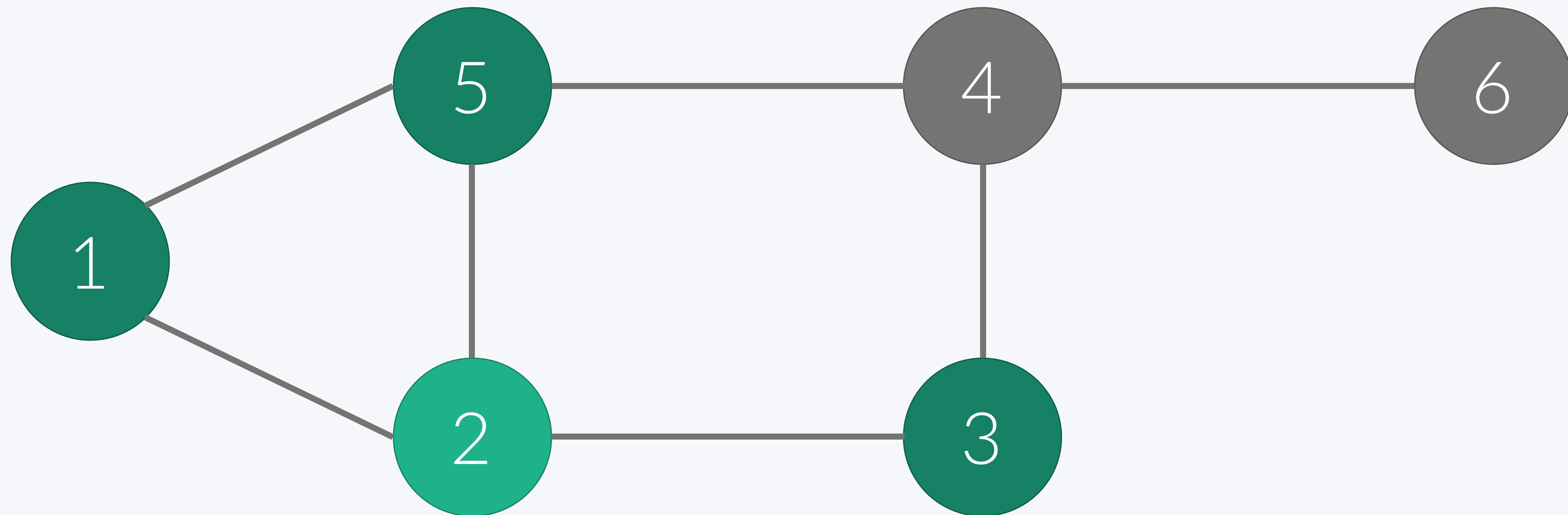
너비 우선 탐색

63

Breadth First Search

- 현재 정점: 2
- 순서: 1 2 5 3
- 큐: 2 5 3

i	1	2	3	4	5	6
check[i]	1	1	1	0	1	0



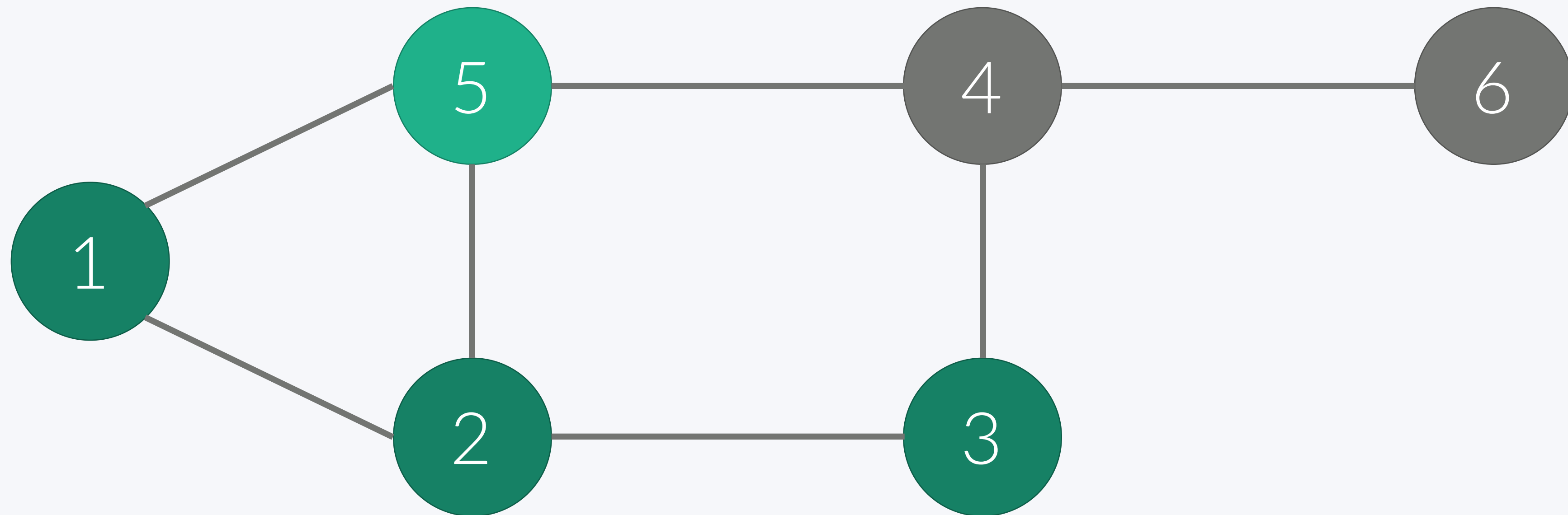
너비 우선 탐색

64

Breadth First Search

- 현재 정점: 5
- 순서: 1 2 5 3
- 큐: 5 3

i	1	2	3	4	5	6
check[i]	1	1	1	0	1	0



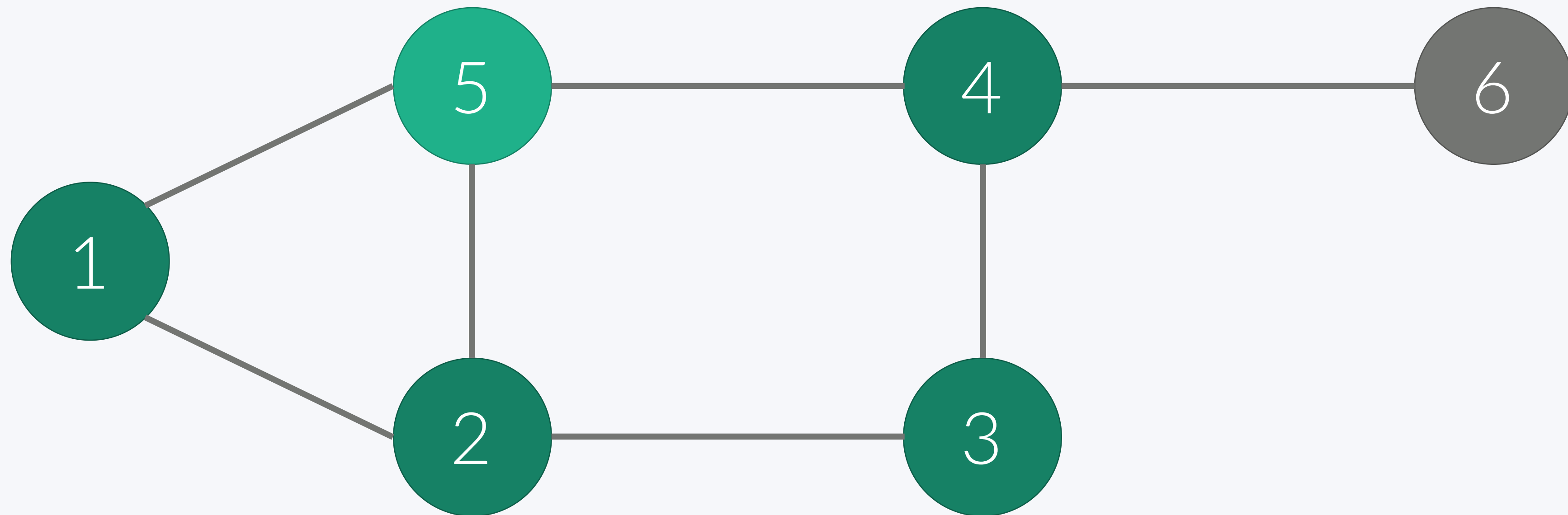
너비 우선 탐색

65

Breadth First Search

- 현재 정점: 5
- 순서: 1 2 5 3 4
- 큐: 5 3 4

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	0



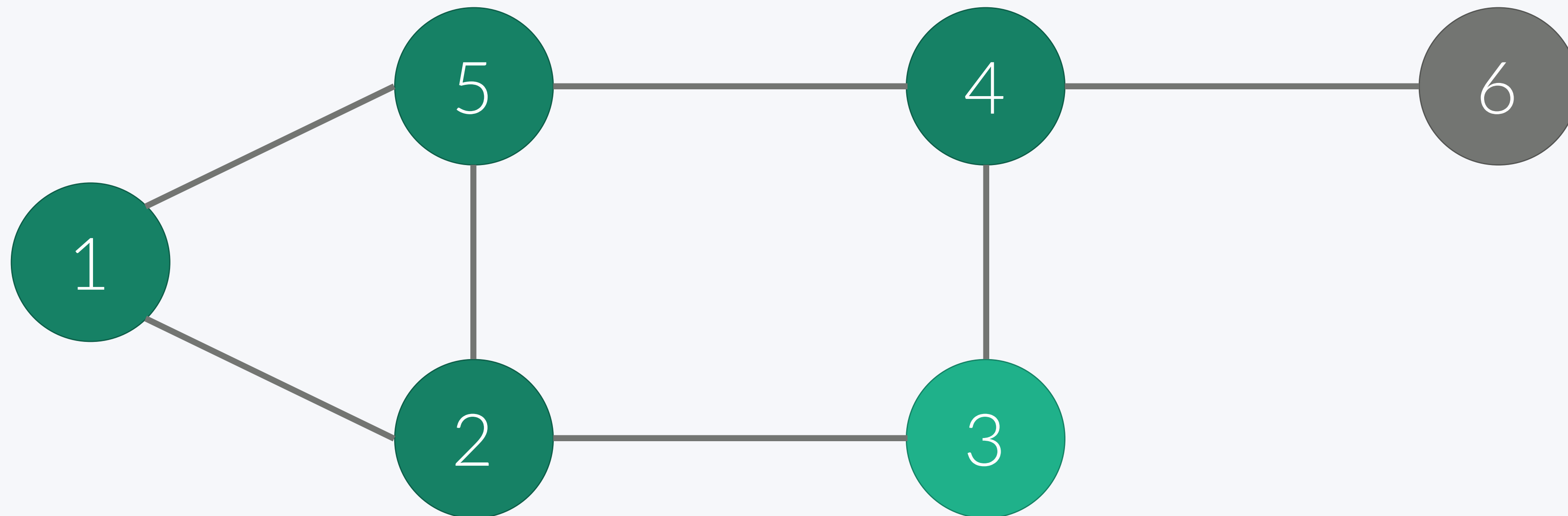
너비 우선 탐색

66

Breadth First Search

- 현재 정점: 3
- 순서: 1 2 5 3 4
- 큐: 3 4

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	0



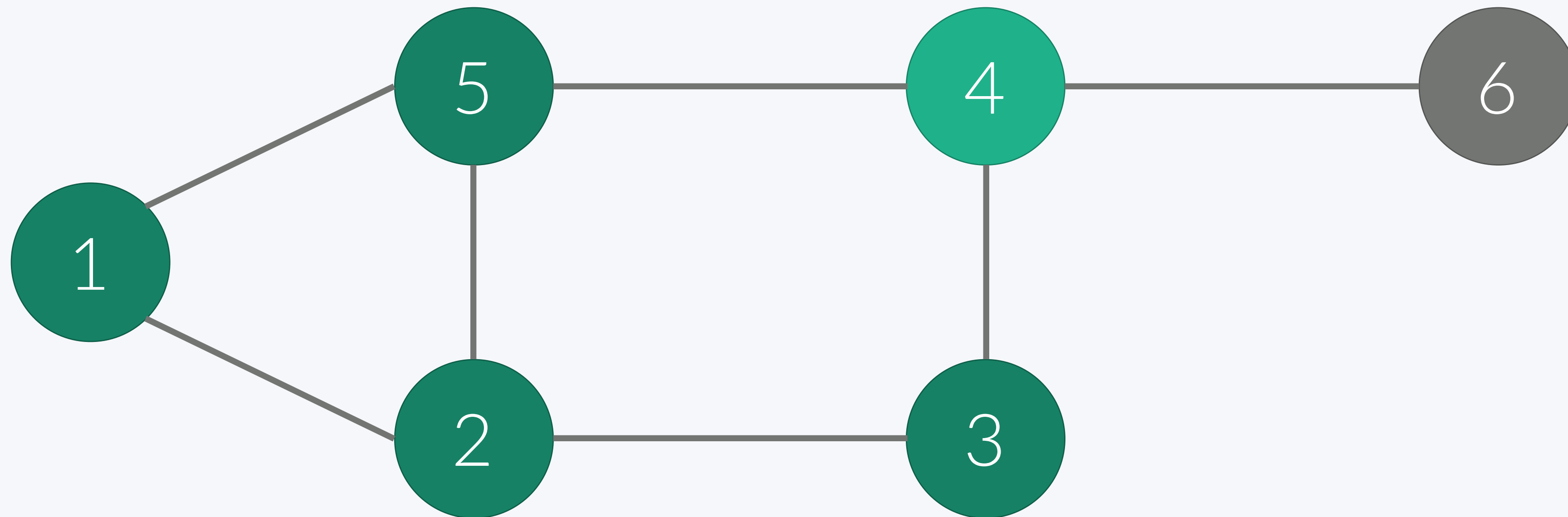
너비 우선 탐색

67

Breadth First Search

- 현재 정점: 4
- 순서: 1 2 5 3 4
- 큐: 4

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	0



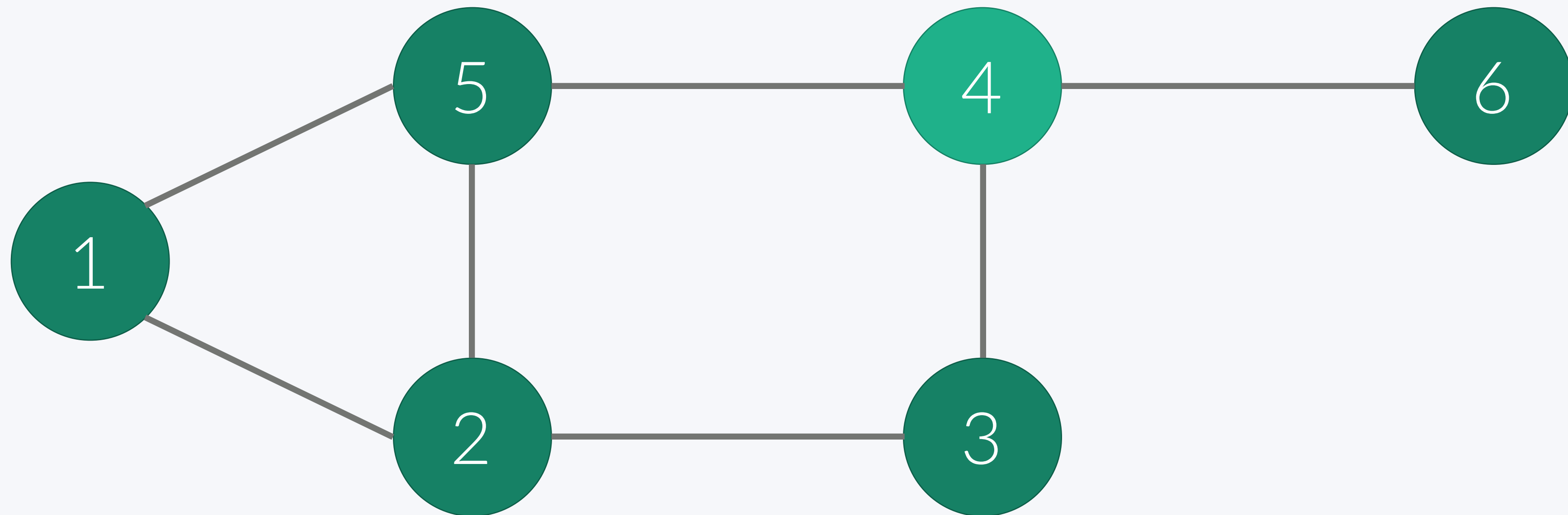
너비 우선 탐색

68

Breadth First Search

- 현재 정점: 4
- 순서: 1 2 5 3 4 6
- 큐: 4 6

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



너비 우선 탐색

Breadth First Search

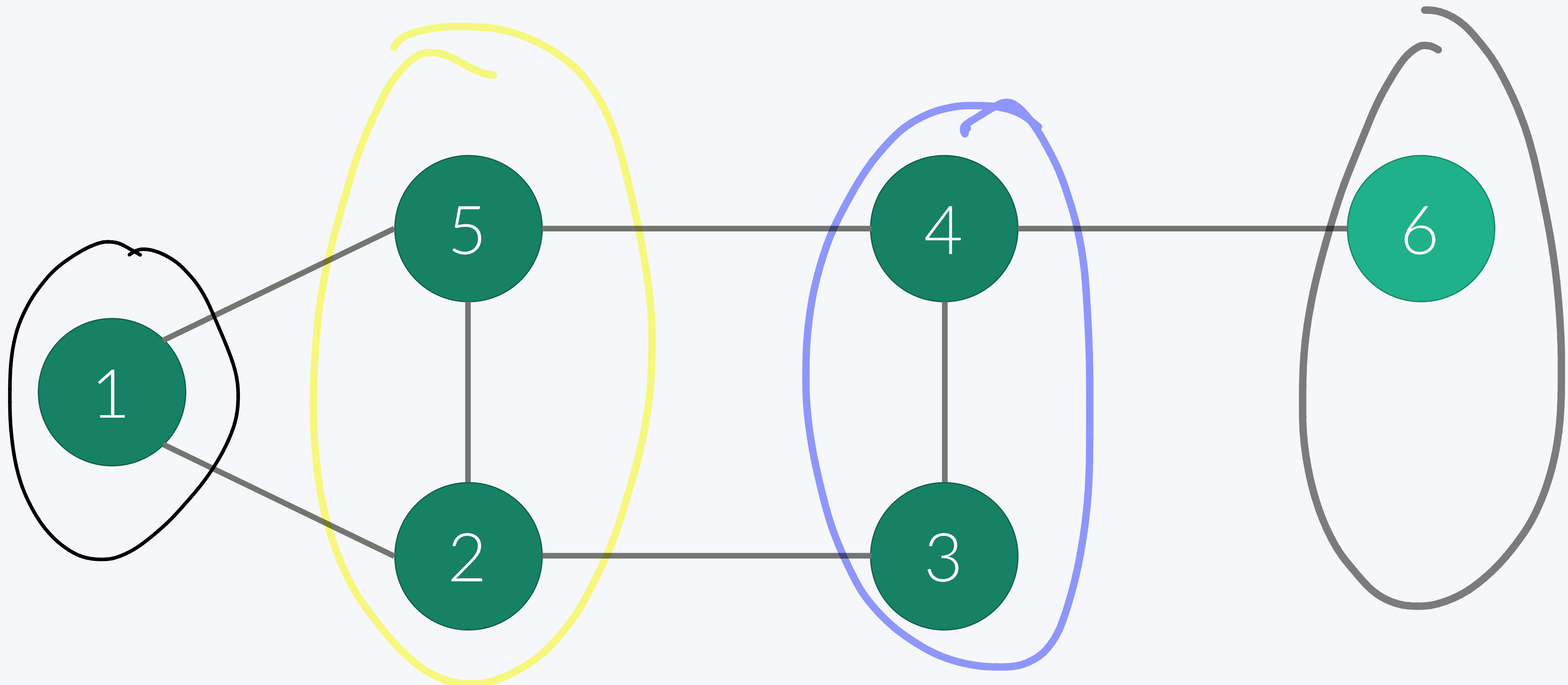
모든 가-중치 1

=> 최단거리
BFS

69

- 현재 정점: 6
- 순서: 1 2 5 3 4 6
시 1 2 3
- 큐: 6

i	1	2	3	4	5	6
check[i]	1	1	1	1	1	1



너비 우선 탐색

Queue

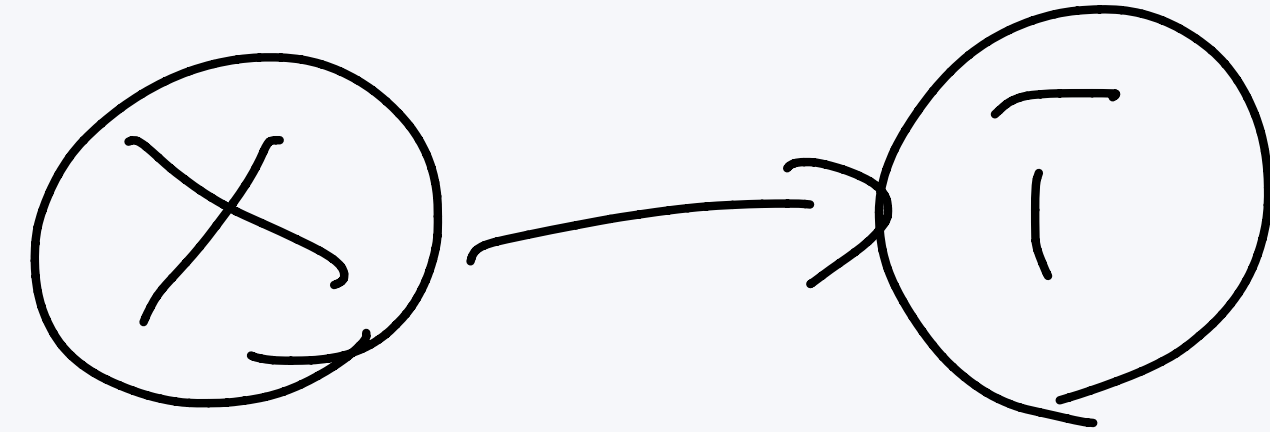
70

Breadth First Search

- BFS의 구현은 Queue를 이용해서 할 수 있다. (인접 행렬)

```
queue<int> q;  
check[1] = true; q.push(1);  
while (!q.empty()) {
```

시작



```
    int x = q.front(); q.pop();  
    printf("%d ", x);
```

```
    for (int i=1; i<=n; i++) {  
        if (a[x][i] == 1 && check[i] == false) {  
            check[i] = true;  
            q.push(i);  
        }  
    }
```

다음
0, 1, 2
3, 4, 5, 6

```
}
```

너비 우선 탐색

Breadth First Search

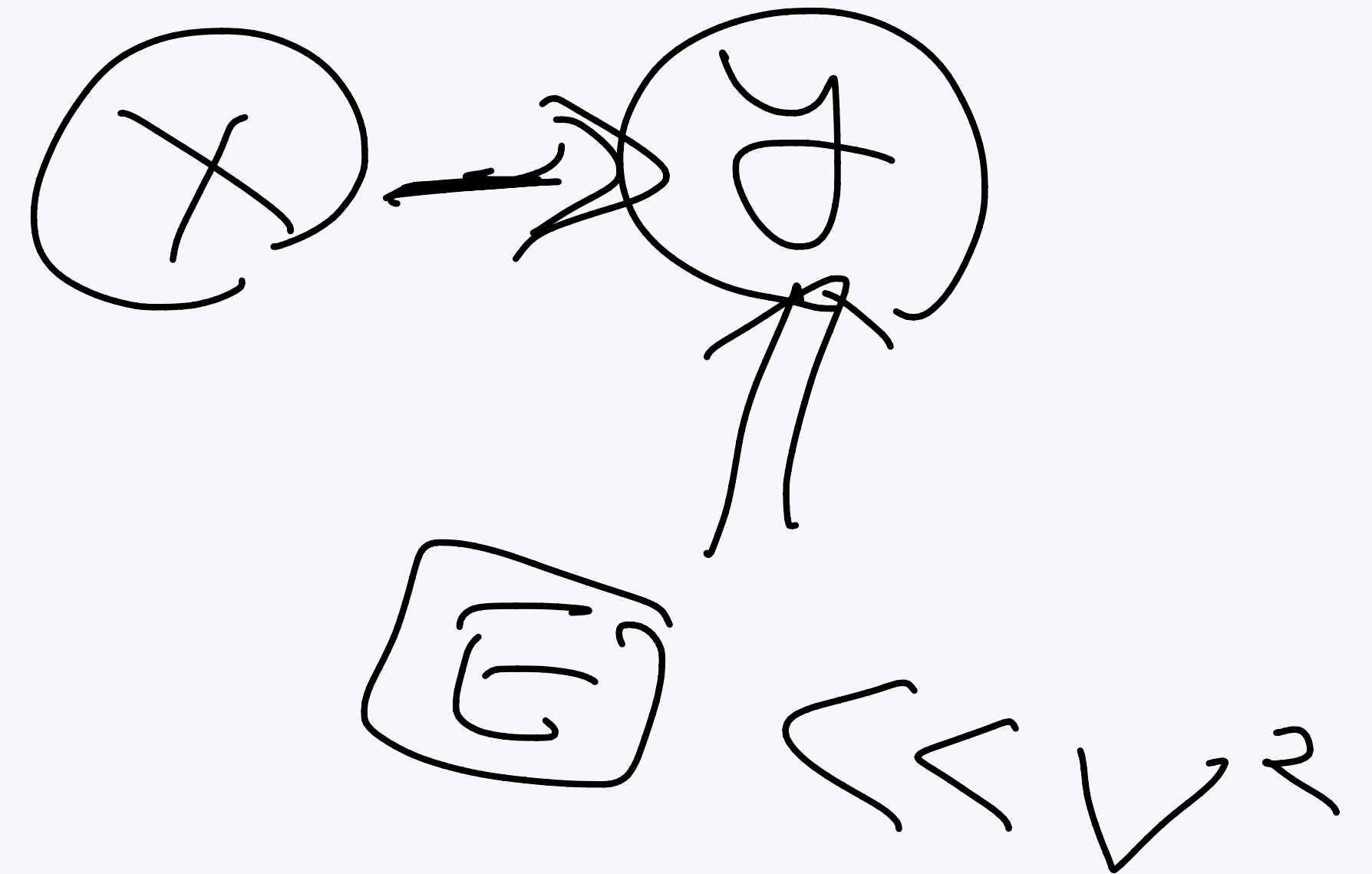
DFS
BFS

인접행렬 $O(V^2)$
인접리스트 $O(V+E)$

71

- BFS의 구현은 Queue를 이용해서 할 수 있다. (인접 리스트)

```
queue<int> q;  
check[1] = true; q.push(1);  
while (!q.empty()) {  
    int x = q.front(); q.pop();  
    printf("%d ", x);  
    for (int i=0; i<a[x].size(); i++) {  
        int y = a[x][i];  
        if (check[y] == false) {  
            check[y] = true; q.push(y);  
        }  
    }  
}
```



시간 복잡도

Time Complexity

- 인접 행렬: $O(V^2)$
- 인접 리스트: $O(V+E)$

DFS와 BFS

<https://www.acmicpc.net/problem/1260>

- 그래프를 DFS로 탐색한 결과와 BFS로 탐색한 결과를 출력하는 문제

DFS와 BFS

<https://www.acmicpc.net/problem/1260>

- C/C++
 - 인접 리스트 사용
 - <https://gist.github.com/Baekjoon/1333250dbcbb72671cc7>
 - 간선 리스트 사용
 - <https://gist.github.com/Baekjoon/4c50590a0303e3037fef>
 - 비재귀 구현
 - <https://gist.github.com/Baekjoon/d2e726b5f85bd8c17200>
- Java
 - 인접 리스트 사용
 - <https://gist.github.com/Baekjoon/5e4fc178fa0d391a7ca9>

연결 요소

연결 요소

Connected Component

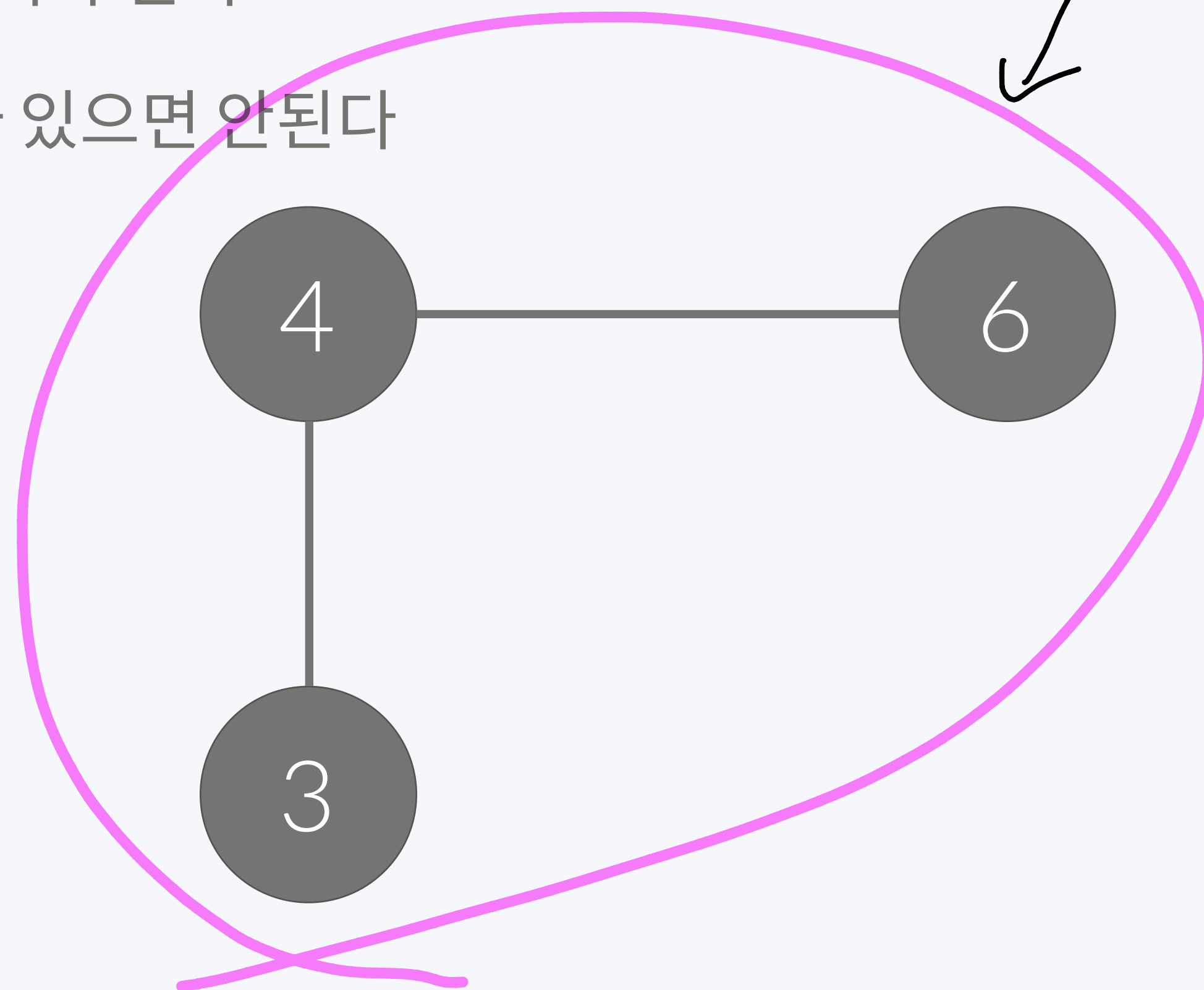
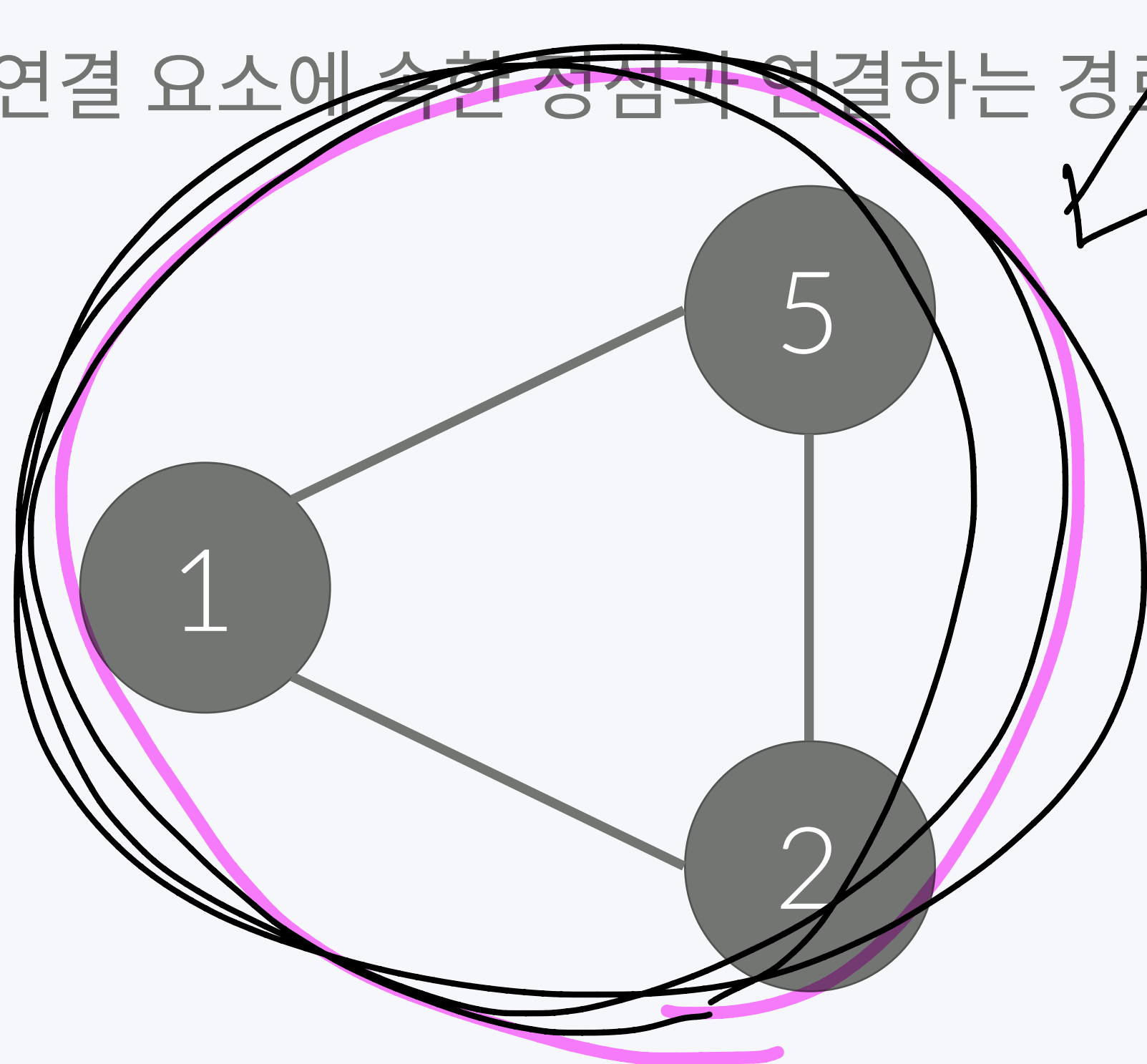
정답: 그래프 2개
1개

연결 요소

그래프 1개

연결 요소: 2개

- 그래프가 아래 그림과 같이 나누어져 있지 않은 경우가 있을 수도 있다
- 이렇게 나누어진 각각의 그래프를 연결 요소라고 한다.
- 연결 요소에 속한 모든 정점을 연결하는 경로가 있어야 한다
- 또, 다른 연결 요소에 속한 정점과 연결하는 경로가 있으면 안된다



연결 요소

Connected Component

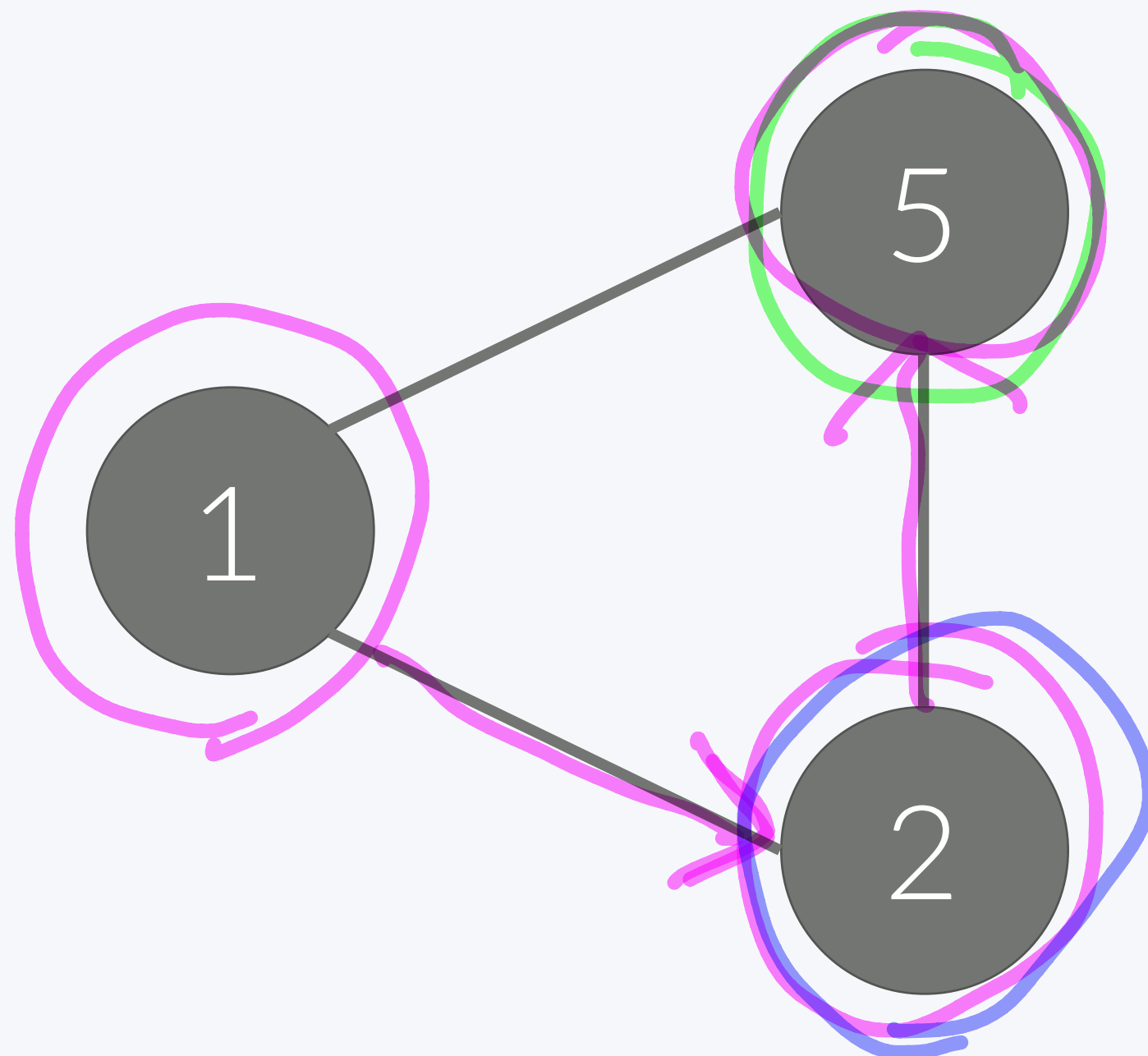
DFS/BFS 활용: 모든 정점 1번씩

77

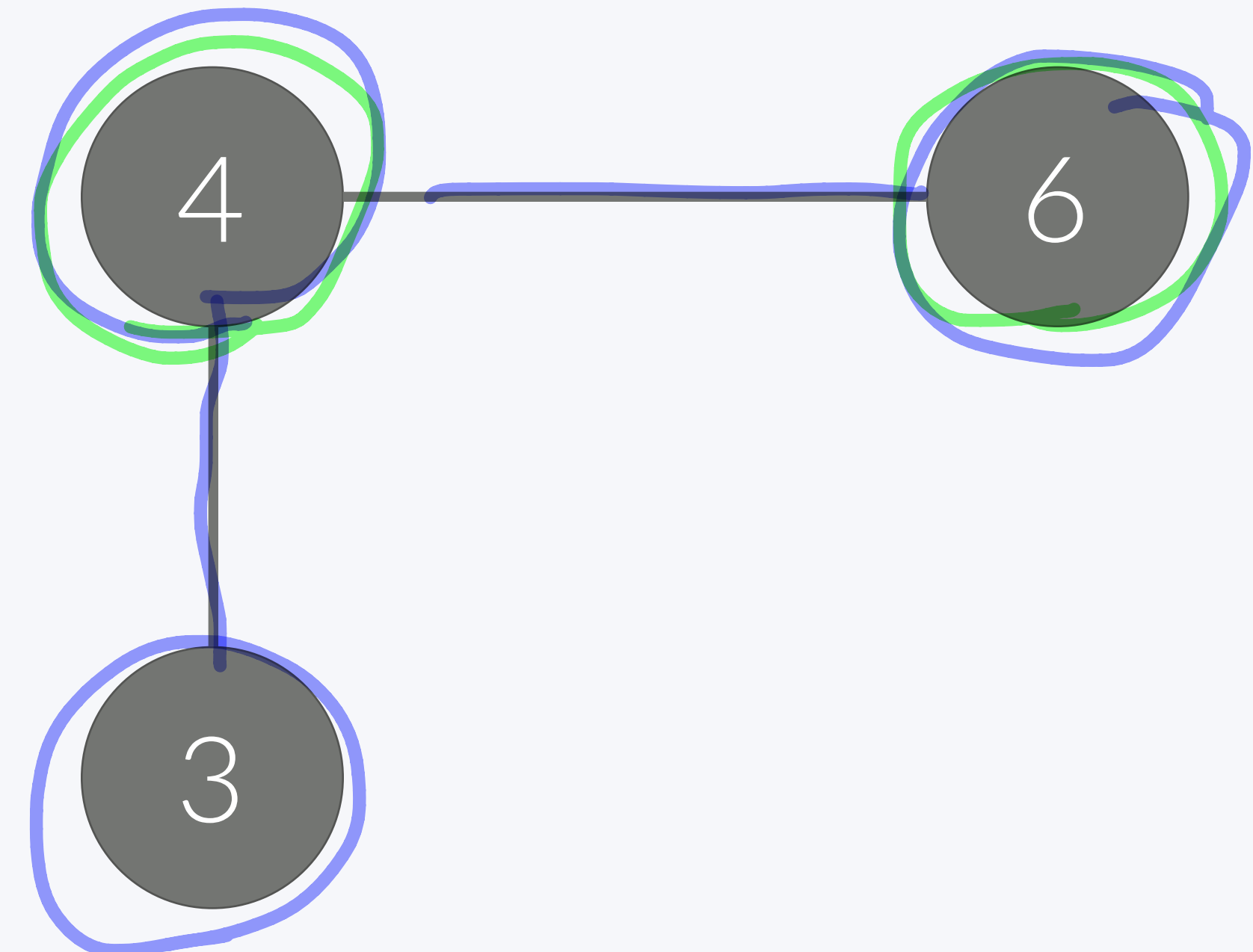
- 아래 그래프는 총 2개의 연결 요소로 이루어져 있다
- 연결 요소를 구하는 것은 DFS나 BFS 탐색을 이용해서 구할 수 있다.

2개

1개



1개



연결 요소

<https://www.acmicpc.net/problem/11724>

- 연결 요소의 개수를 구하는 문제

DFS/BFS

연결 요소

<https://www.acmicpc.net/problem/11724>

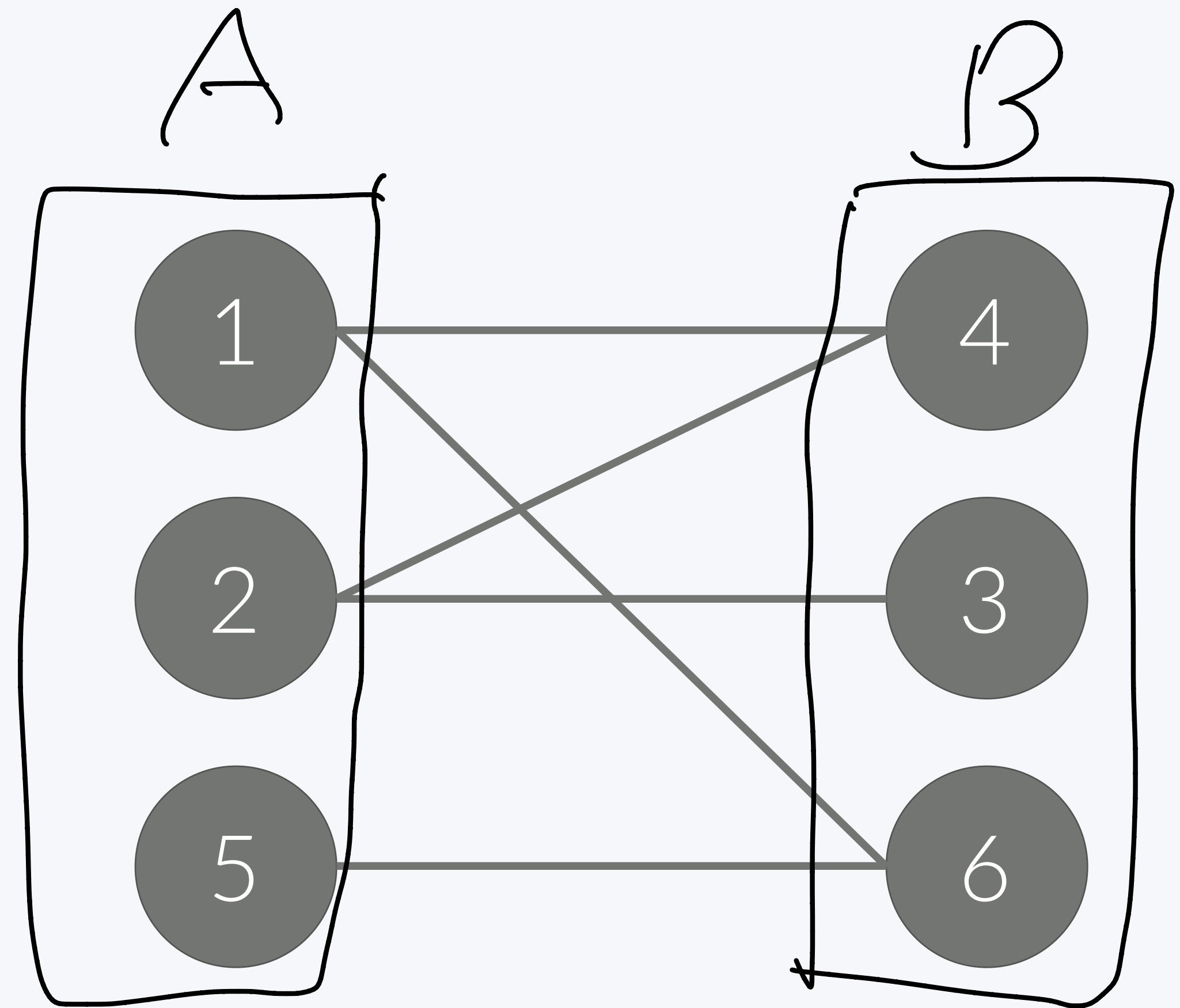
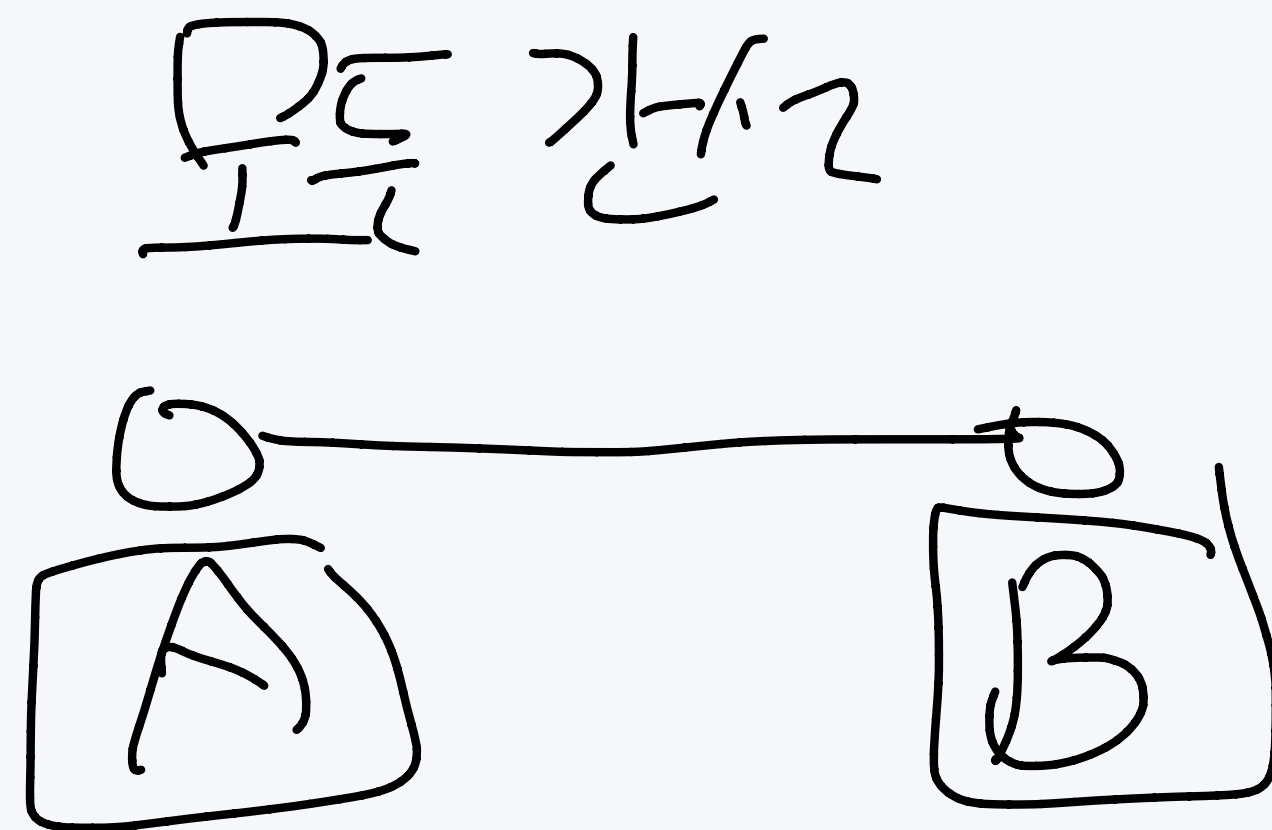
- C++
 - <https://gist.github.com/Baekjoon/d2546c70b63c112f18f2>
- Java
 - <https://gist.github.com/Baekjoon/4c879b1ec9802dc839989443bb8ad617>

이분 그래프

이분 그래프

Bipartite Graph

- 그래프를 다음과 같이 A와 B로 나눌 수 있으면 이분 그래프라고 한다.
- A에 포함되어 있는 정점끼리 연결된 간선이 없음
- B에 포함되어 있는 정점끼리 연결된 간선이 없음
- 모든 간선의 한 끝 점은 A에, 다른 끝 점은 B에



이분 그래프

Bipartite Graph


이분그래프

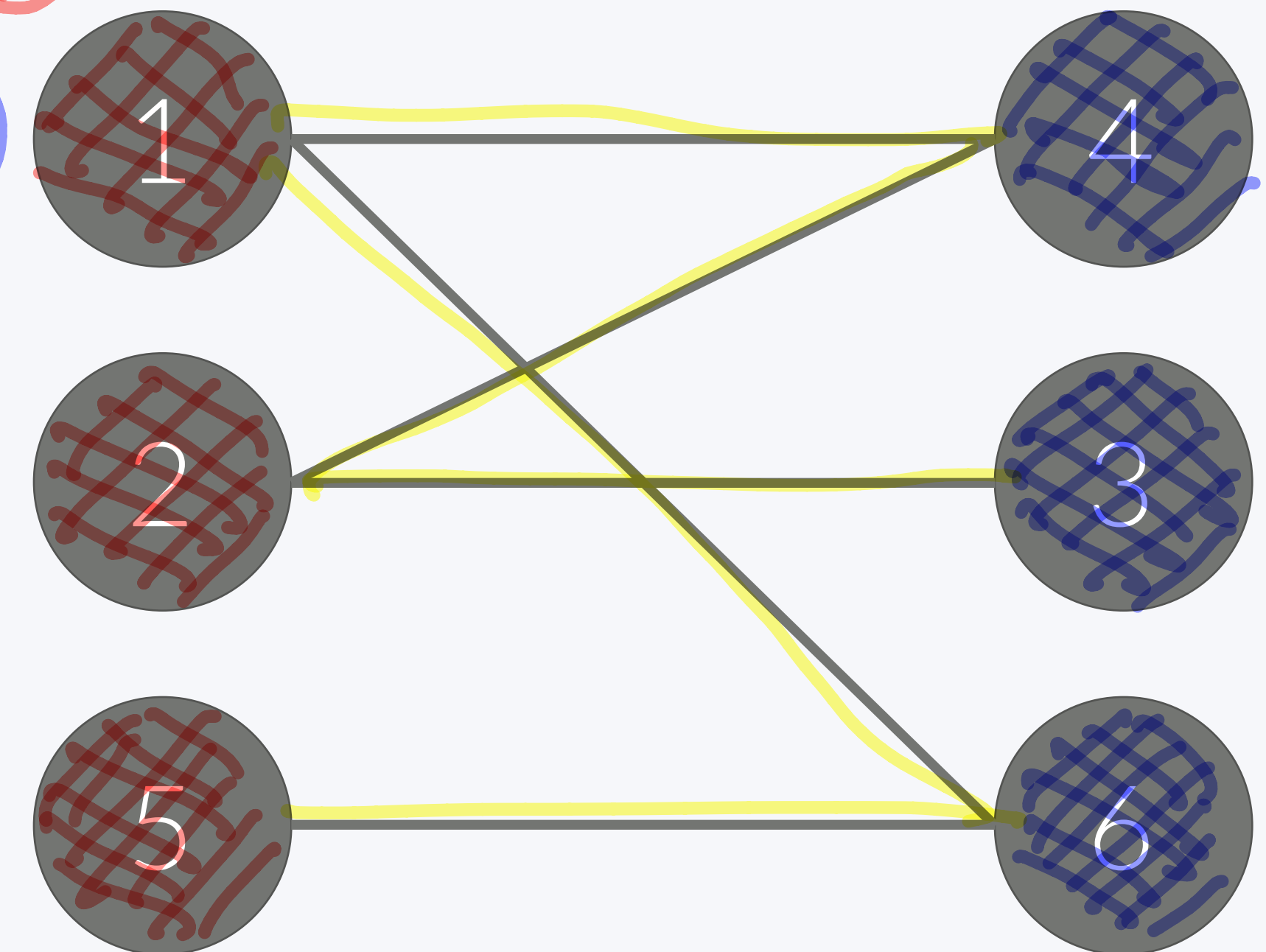
82

- 그래프를 DFS또는 BFS 탐색으로 이분 그래프인지 아닌지 알아낼 수 있다.

check[i] = 0 (아직 방문 X)

= 1 (방문 0), 

= 2 (방문 0), 



이분 그래프

<https://www.acmicpc.net/problem/1707>

- 그래프가 이분 그래프인지 아닌지 판별하는 문제

이분 그래프

<https://www.acmicpc.net/problem/1707>

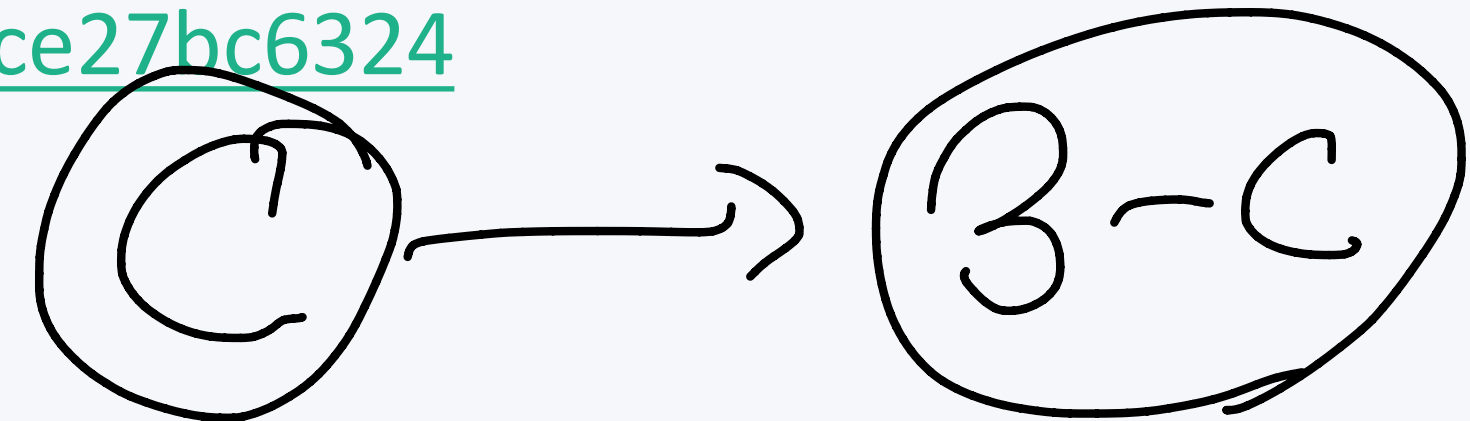
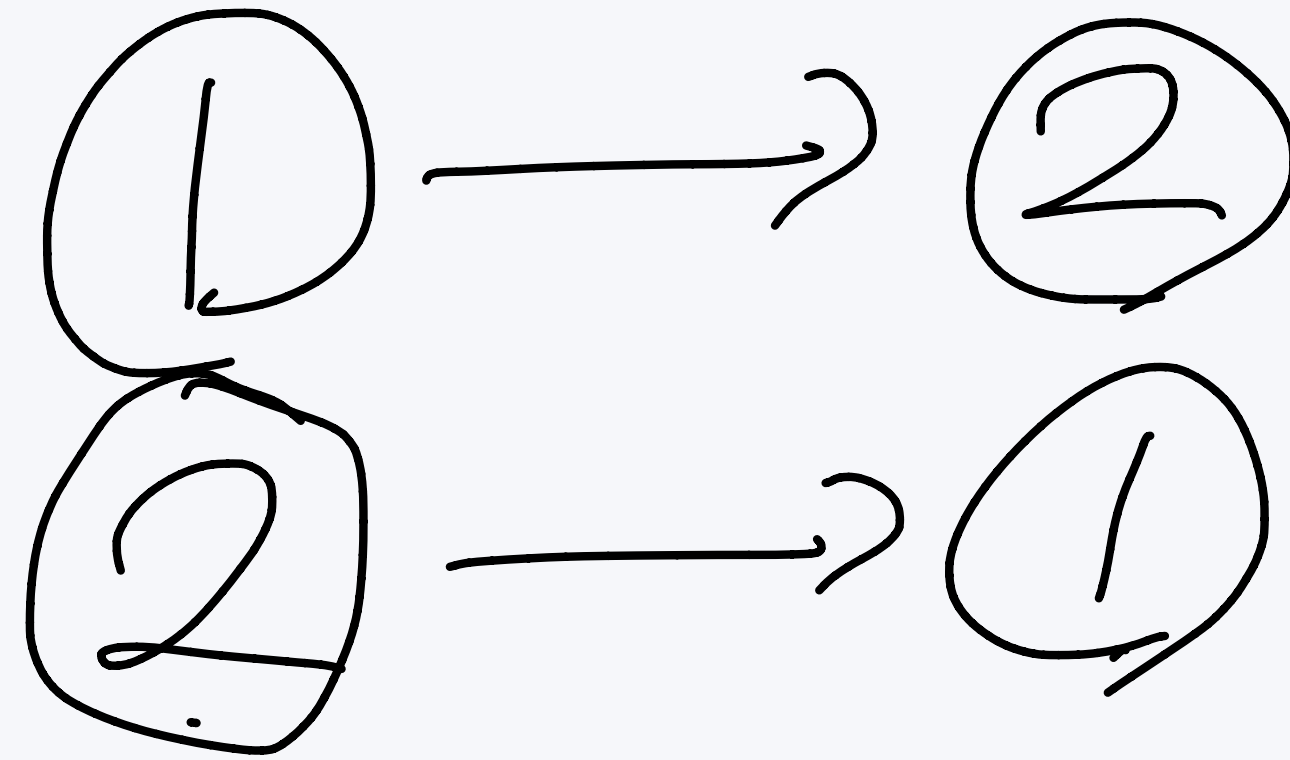
- C++

- <https://gist.github.com/Baekjoon/6a1e52d16b3b6a4ca701>

- <https://gist.github.com/Baekjoon/e32b56ebb4c53c7fb8a914ce27bc6324>

- Java

- <https://gist.github.com/Baekjoon/50e0fdd4ccdc424c4a6b564fa2e2cae7>



$$\underline{1} + \underline{2} = \textcircled{3}$$
$$C + (3 - C) = 3$$

사이클 찾기

순열 사이클

<https://www.acmicpc.net/problem/10451>

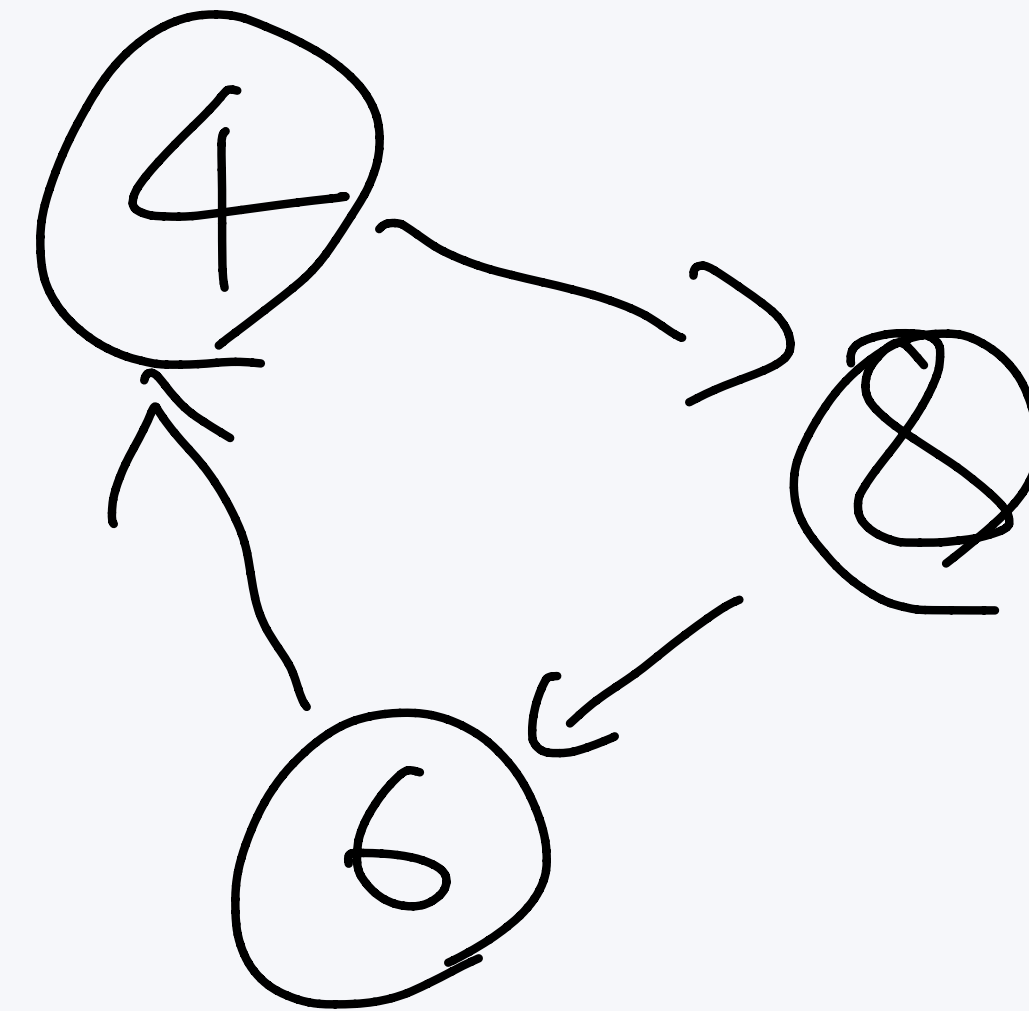
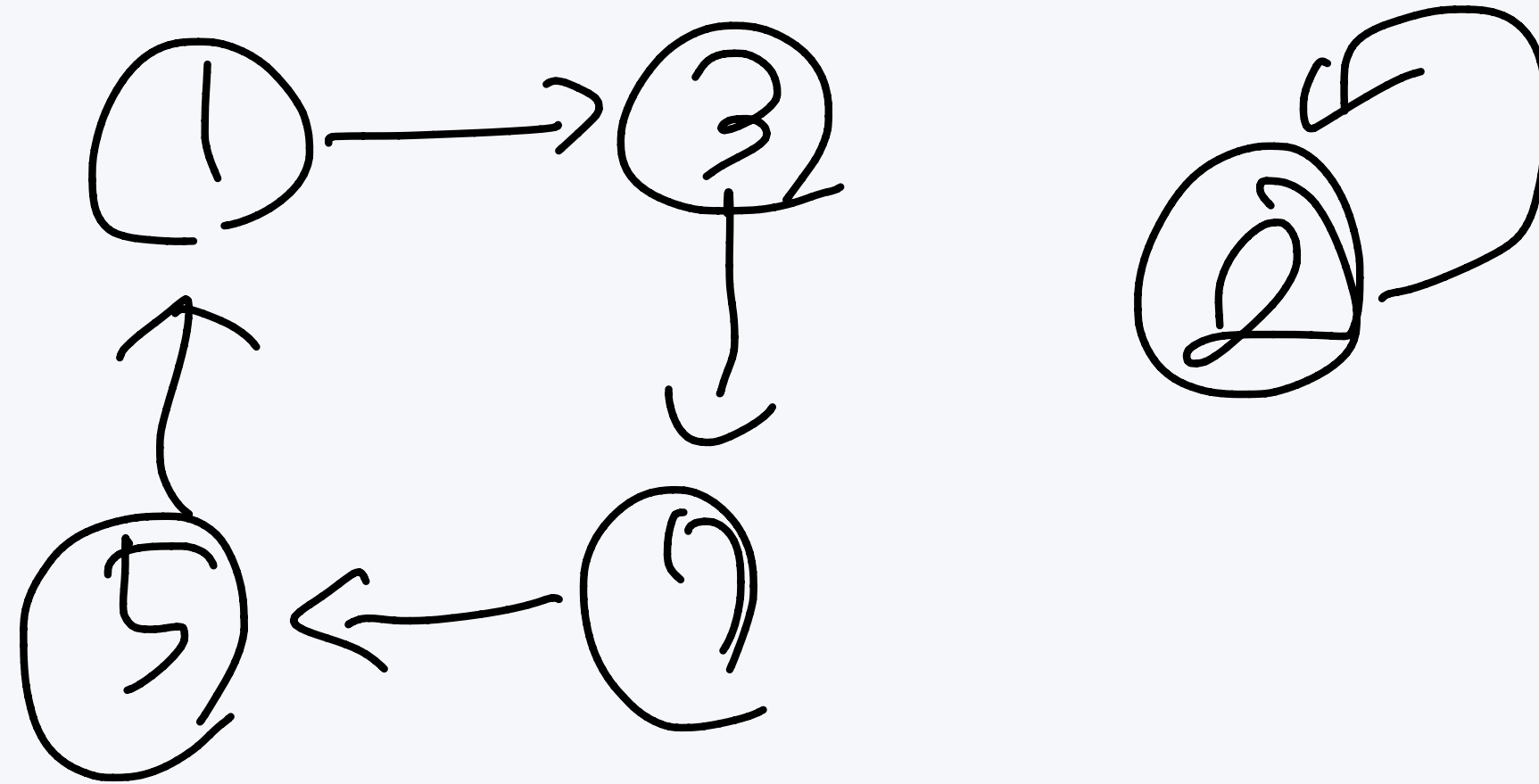
86

τ 1 2 3 4 5 6 7 8
 $p[\tau]$ 3 2 1 8 1 4 5 6

$\tau \rightarrow p[\tau]$

- 순열이 주어졌을 때, 순열 사이클의 개수를 찾는 문제
- DFS를 이용해서 이미 방문했던, 수를 방문하면 return 하는 방식으로 풀 수 있다.

DFS



$p[\tau]; \tau \rightarrow ?$

순열 사이클

<https://www.acmicpc.net/problem/10451>

87

```
void dfs(int x) {
```

```
    if (c[x]) return;
```

```
    c[x] = true;
```

```
    dfs(a[x]);
```

```
}
```

```
int ans = 0;
```

```
for (int i=1; i<=n; i++) {
```

```
    if (c[i] == false) {
```

```
        dfs(i);
```

```
        ans += 1;
```

```
    }
```

```
}
```

dfs(x) ↙

check

다음 방문

~~for~~

↓ 개

```
while (c[x] == false) {
```

```
    c[x] = true;
```

```
    x = a[x];
```

```
}
```

순열 사이클

<https://www.acmicpc.net/problem/10451>

- C/C++: <https://gist.github.com/Baekjoon/fa15f08ec6965973a3c9>
- Java: <https://gist.github.com/Baekjoon/1476b1890f3f3a20f25887ffd04fdee7>

반복수열

<https://www.acmicpc.net/problem/2331>

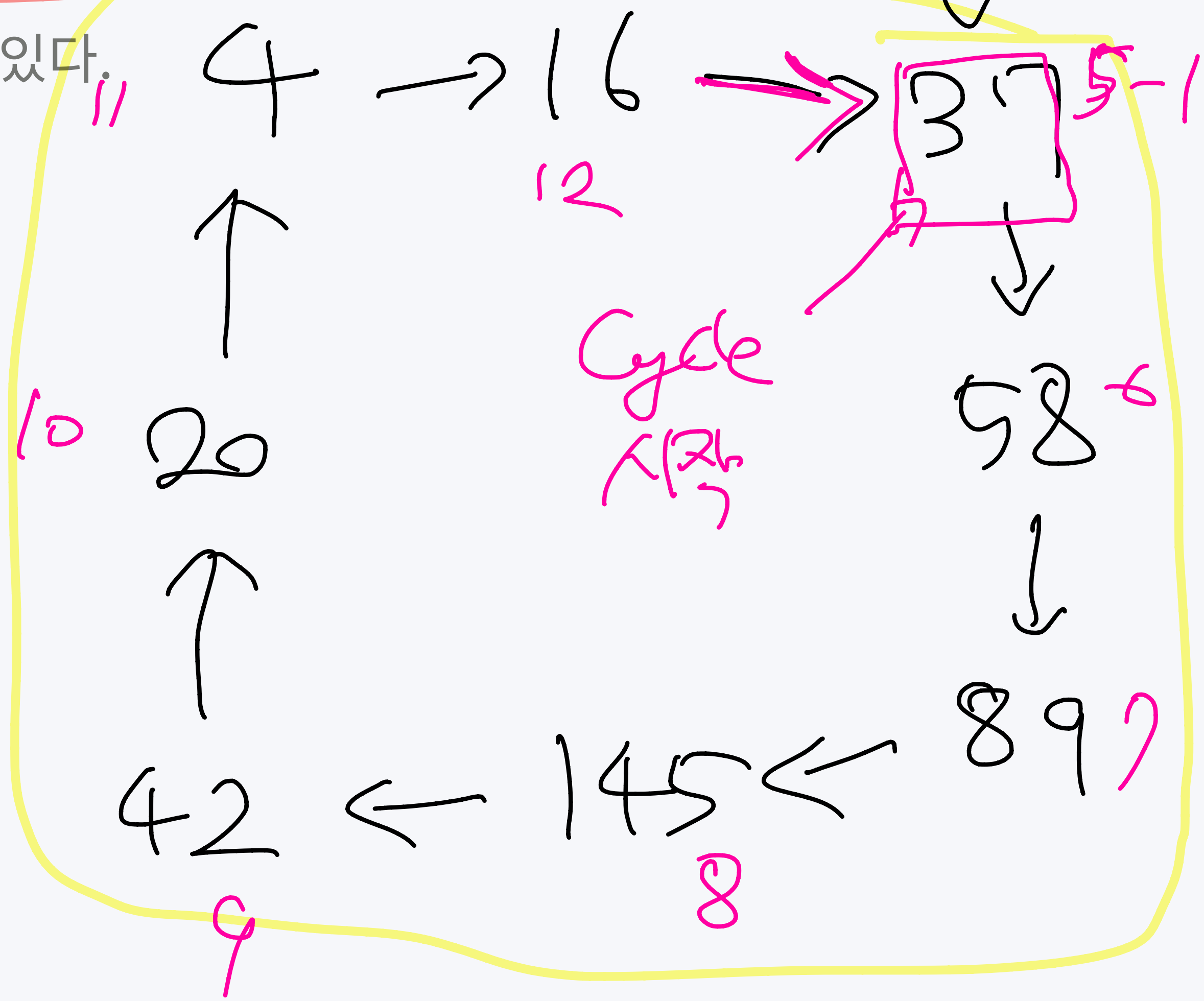
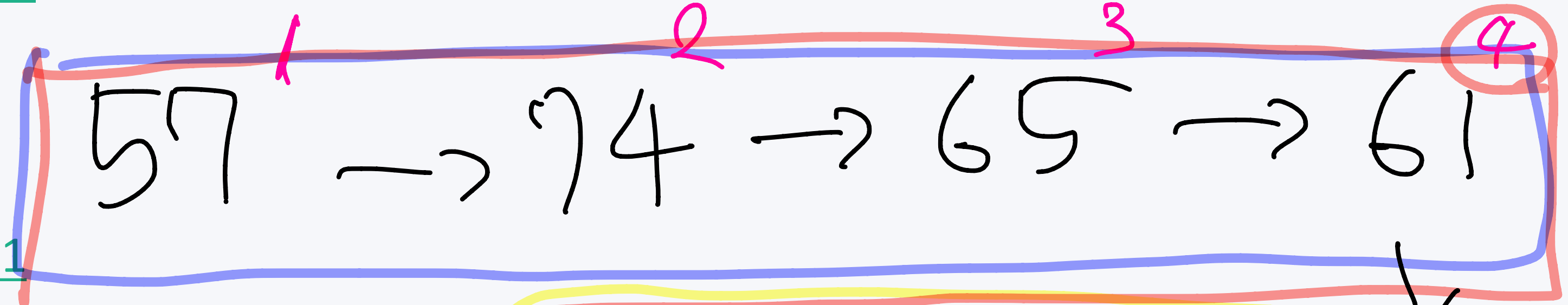
89

○ A와 P가 주어졌을 때, 다음과 같은 수열을 정할 수 있다.

- $D[1] = A$
- $D[n] = D[n-1]$ 의 각 자리 숫자를 P번 곱한 수의 합
- $A = 57, P = 2$ 인 경우
- $D[1] = 57$
- $D[2] = \boxed{5*5} + \boxed{7*7} = 74$
- $D[3] = \boxed{7*7} + \boxed{4*4} = 65$
- $D[4] = 6*6 + 5*5 = 61$

$$1 + 66 + 25 = 92$$

$$\begin{array}{r} 81 \\ 64 \\ \hline 145 \end{array} \quad \begin{array}{r} 16 \\ 74 \end{array}$$



반복수열

<https://www.acmicpc.net/problem/2331>

- 계속해서 수를 만들다가 이전에 만들었던 수를 만들면 그 수가 몇 번째로 만들었던 수인지 리턴하면 된다.

$check[i] = 0$ (이전 방문 X)
 $\neq 0$ (방문 O)
[1] = 1 (1, 2, 4)

반복수열

91

<https://www.acmicpc.net/problem/2331>

```
int length(int a, int p, int cnt) {  
    if (check[a] != 0) {  
        return check[a]-1;  
    }  
    check[a] = cnt;  
    int b = next(a, p);  
    return length(b, p, cnt+1);  
}
```

$$\frac{a}{cnt} \rightarrow \frac{b}{cnt+1}$$

반복수열

<https://www.acmicpc.net/problem/2331>

- C/C++: <https://gist.github.com/Baekjoon/87547f40baaf6867eb8f>
- Java: <https://gist.github.com/Baekjoon/a8f0c8b8e92994f9b68cbf526136593f>

Term Project

<https://www.acmicpc.net/problem/9466>

① 1 2 3 4 5 6 7
A[1] 3 1 3 7 3 4 2

93

• C/C++ 재귀: <https://gist.github.com/Baekjoon/a1c2451d685d8d184d17>

• C/C++ 비재귀: <https://gist.github.com/Baekjoon/ef95c25073ea6f6b057a>

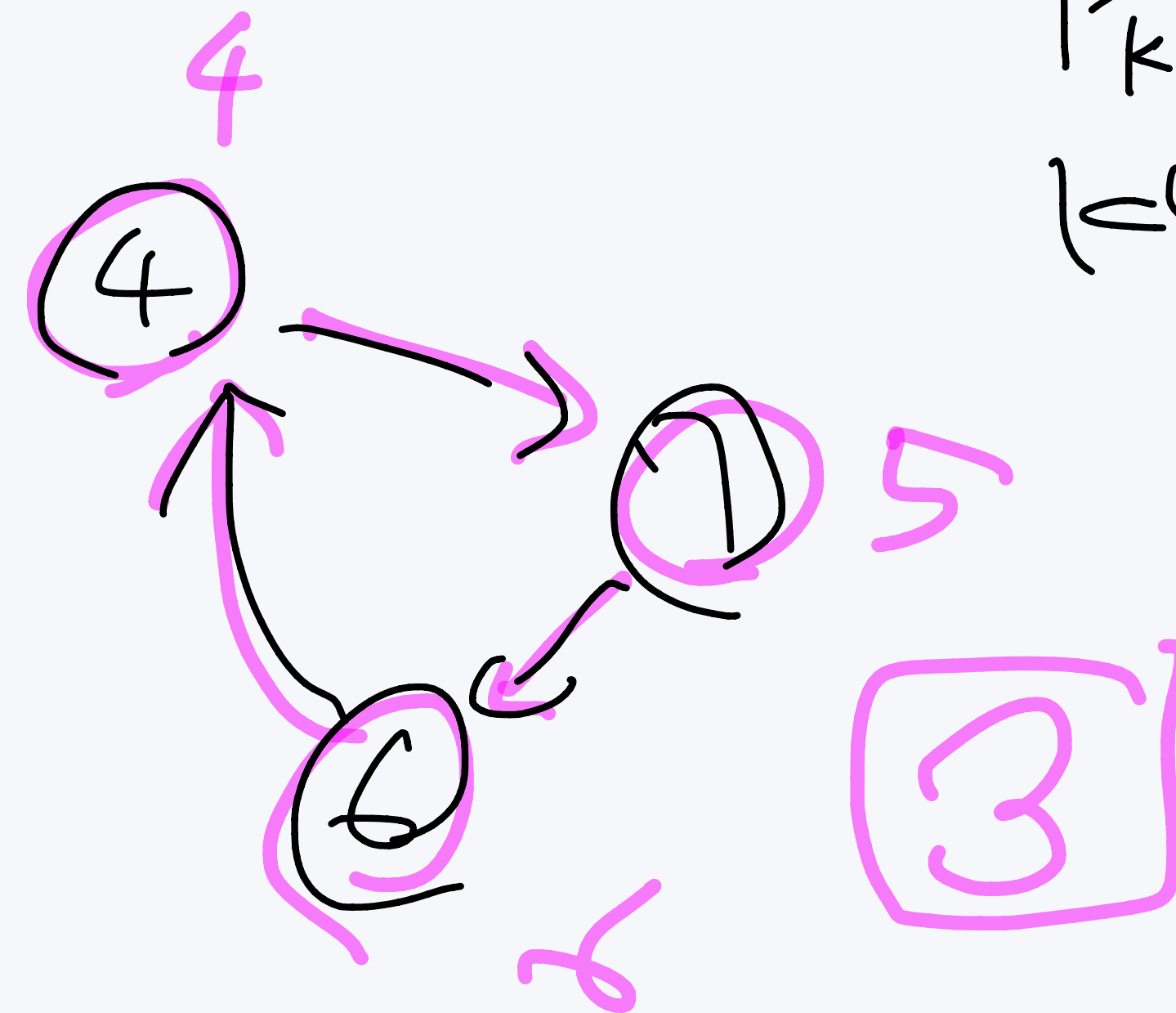
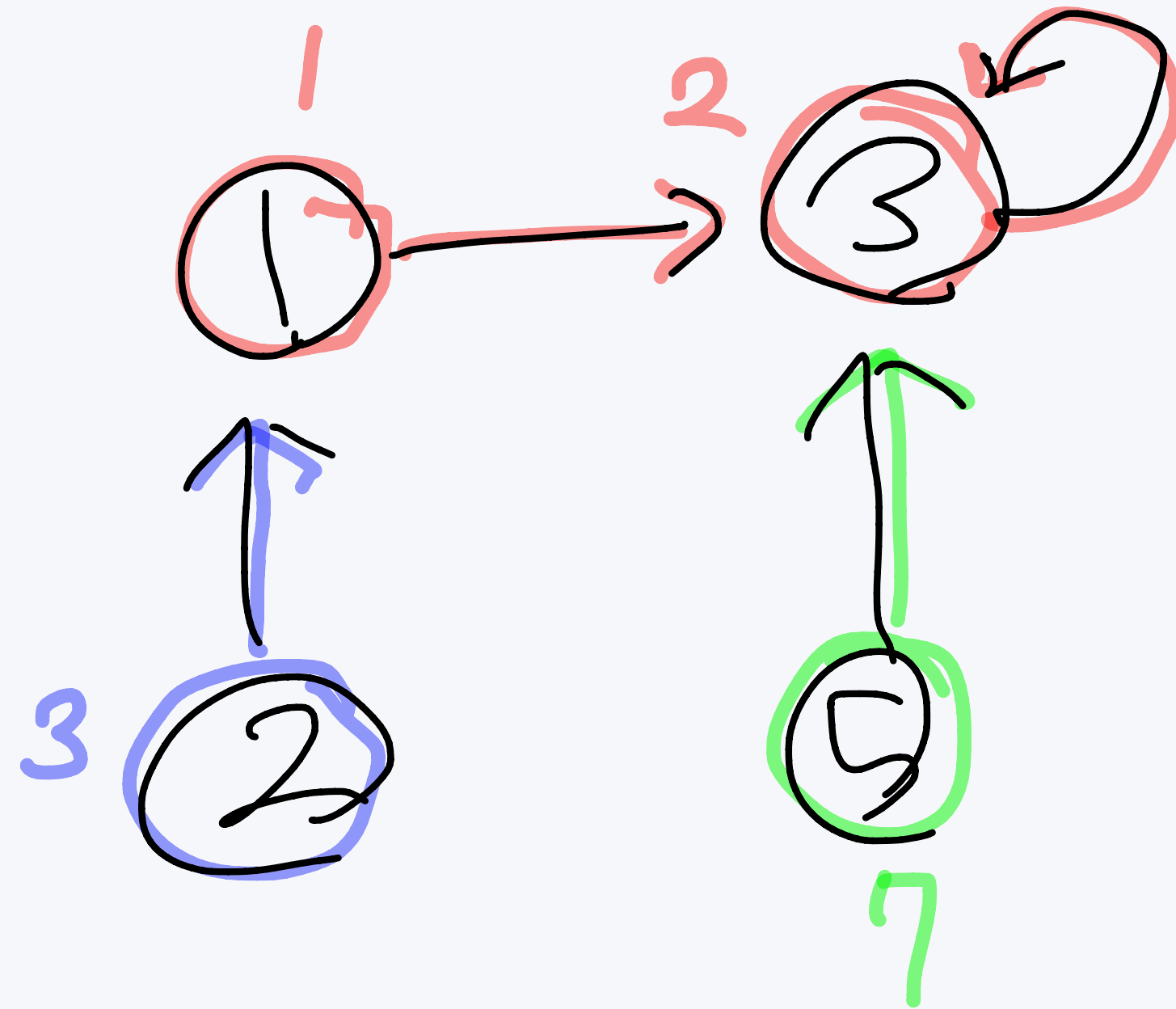
① 함수, $i == A[i]$

② $P_1 \Rightarrow P_2, P_2 \Rightarrow P_3, \dots$

$P_k \Rightarrow P_1$

1<178

이미 방문한 것
↓
사이클



플러드 필

플러드 필

Flood Fill

95

- 어떤 위치와 연결된 모든 위치를 찾는 알고리즘

단지번호붙이기

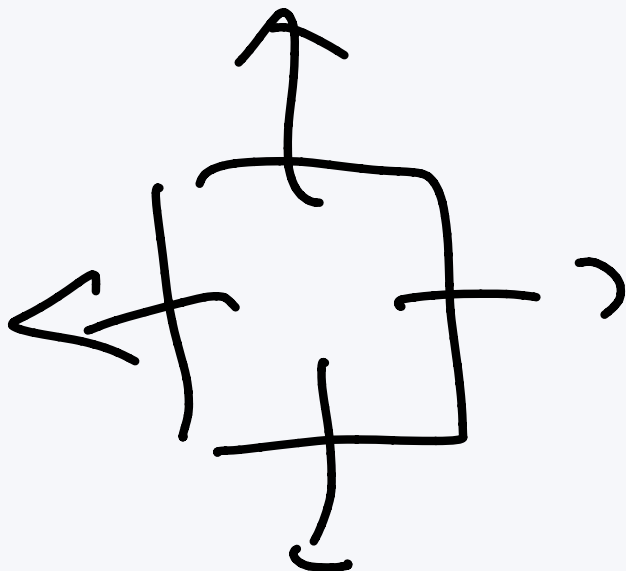
<https://www.acmicpc.net/problem/2667>

- 정사각형 모양의 지도가 있다
- 0은 집이 없는 곳, 1은 집이 있는 곳
- 지도를 가지고 연결된 집의 모임인 단지를 정의하고, 단지에 번호를 붙이려고 한다
- 연결: 좌우 아래위로 집이있는 경우

단지번호붙이기

<https://www.acmicpc.net/problem/2667>

(x, y)
↑ ↑
행 열



0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0

0	1	1	0	2	0	0
0	1	1	0	2	0	2
1	1	1	0	2	0	2
0	0	0	0	2	2	2
0	3	0	0	0	0	0
0	3	3	3	3	3	0
0	3	3	3	0	0	0

단지번호붙이기

<https://www.acmicpc.net/problem/2667>

- DFS나 BFS 알고리즘을 이용해서 어떻게 이어져있는지 확인할 수 있다.
- $d[i][j] = (i, j)$ 를 방문안했으면 0, 했으면 단지 번호

$d[i][j] =$

○

단지 번호

단지번호붙이기

<https://www.acmicpc.net/problem/2667>

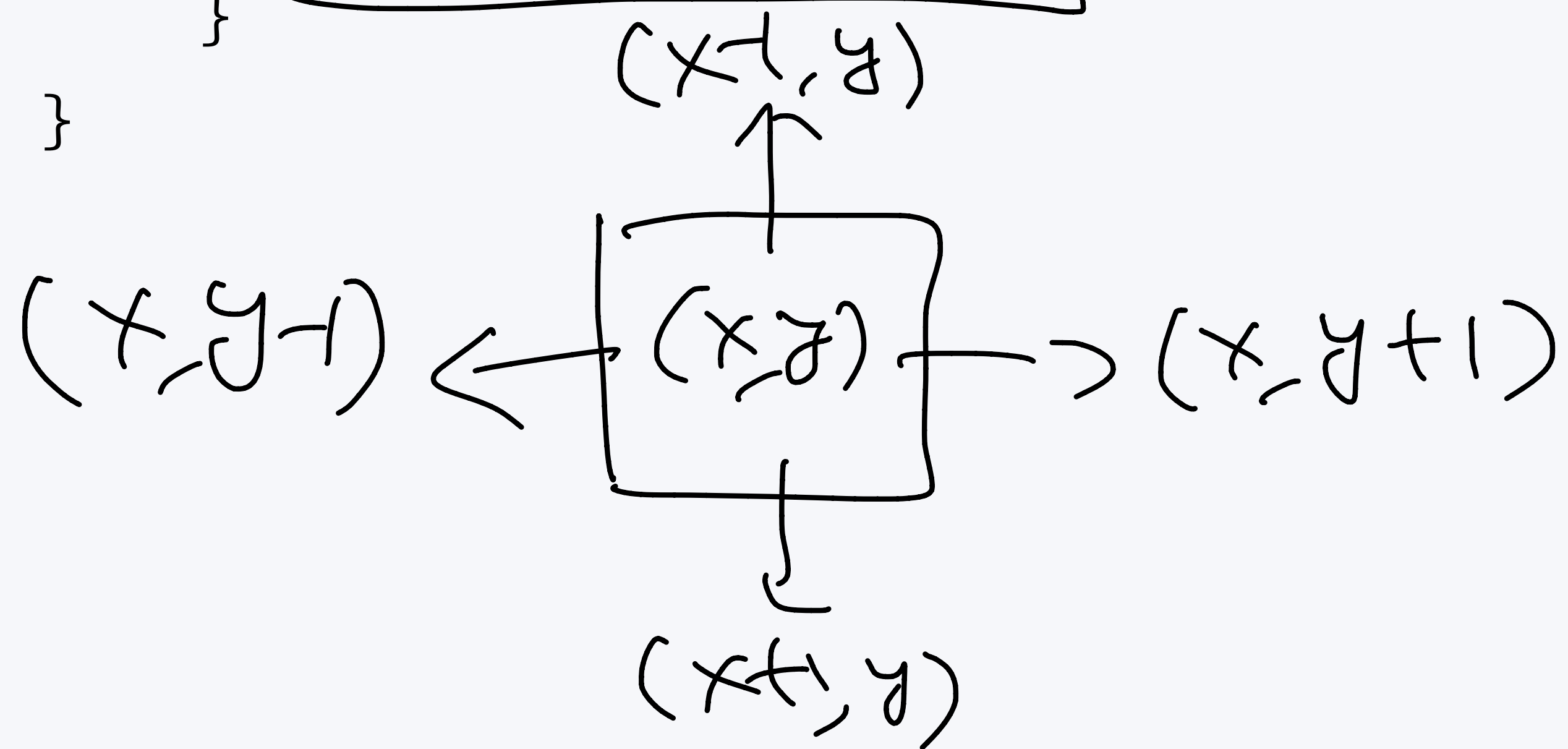
```
int cnt = 0;
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
        if (a[i][j] == 1 && d[i][j] == 0) {
            bfs(i, j, ++cnt);
        }
    }
}
```

정점

이런 방식은 X

BFS

- ① 시작점 $\Rightarrow Q$
- ② while (Q)
 - ① $x = front, pop$
 - ② 다음정점 $\Rightarrow Q$



단지번호붙이기

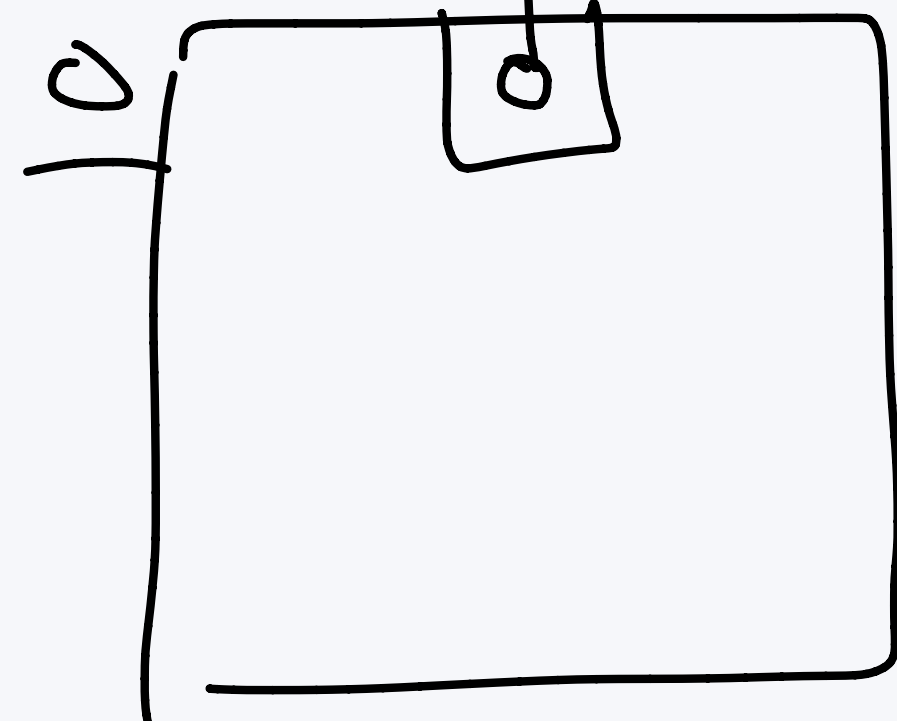
<https://www.acmicpc.net/problem/2667>

$\overline{xy} = \overline{yx}$
 $\Rightarrow \overline{xy} = \overline{yx}$

100

```
void bfs(int x, int y, int cnt) {  
    queue<pair<int, int>> q; q.push(make_pair(x, y)); d[x][y] = cnt;  
    while (!q.empty()) {  
        x = q.front().first; y = q.front().second; q.pop();  
        for (int k=0; k<4; k++) {  
            int nx = x+dx[k], ny = y+dy[k];  
            if (0 <= nx && nx < n && 0 <= ny && ny < n) {  
                if (a[nx][ny] == 1 && d[nx][ny] == 0) {  
                    q.push(make_pair(nx, ny)); d[nx][ny] = cnt;  
                }  
            }  
        }  
    }  
}
```

$dx[4] = \{0, 0, 1, -1\}$
 $dy[4] = \{1, -1, 0, 0\}$



단지번호붙이기

101

<https://www.acmicpc.net/problem/2667>

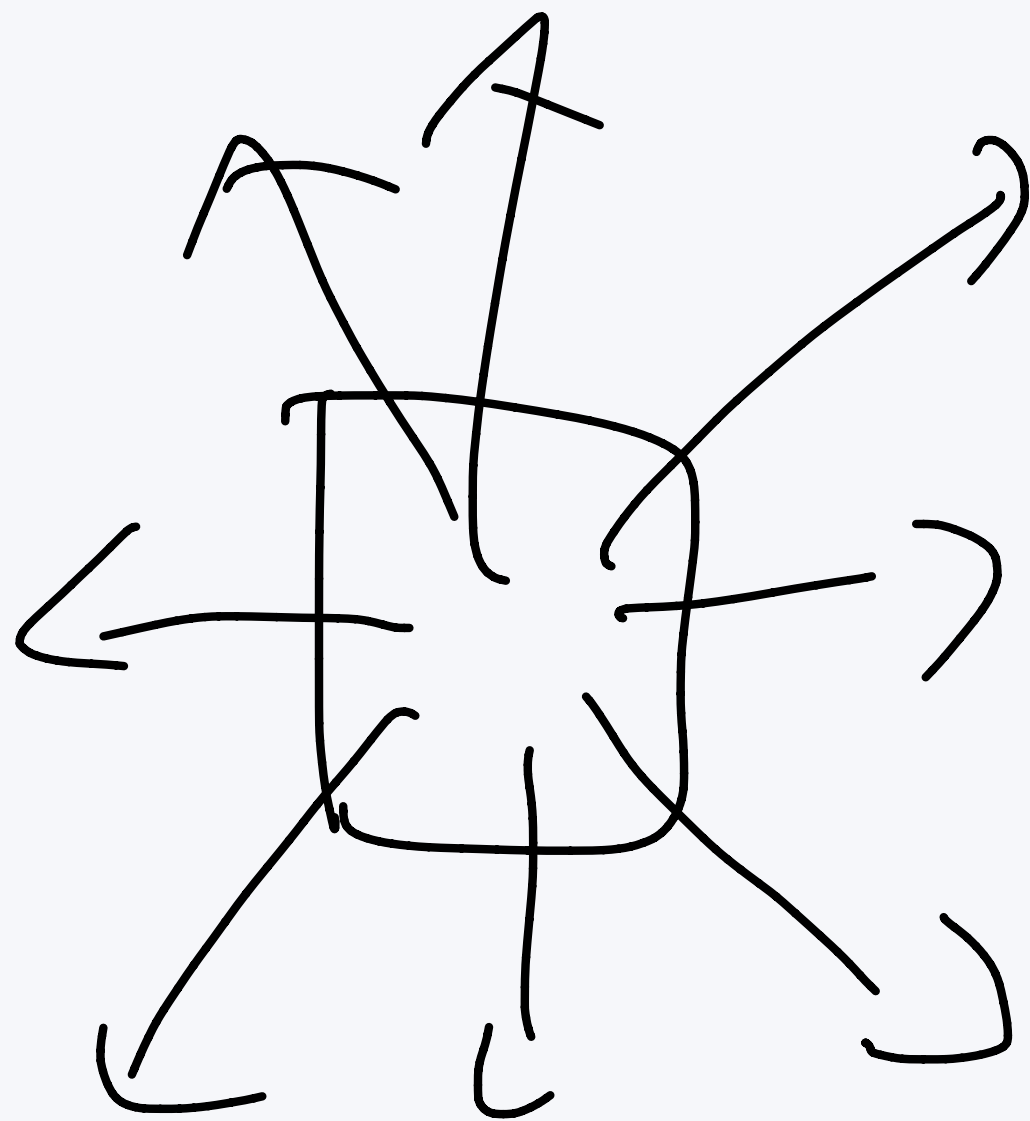
- C/C++: <https://gist.github.com/Baekjoon/c06384791ec922cd0802>
- C/C++ (DFS): <https://gist.github.com/Baekjoon/d99eae47da362bc7110ff1436fcf7e2>
- Java: <https://gist.github.com/Baekjoon/f5f2cd7f14b45ef634fe98b7c78ed23e>
- Java (DFS): <https://gist.github.com/Baekjoon/69358ae41392c18b7c22b70c976824ef>

섬의 개수

102

<https://www.acmicpc.net/problem/4963>

- C/C++: <https://gist.github.com/Baekjoon/95be3c1fb1769eaf2870>
- Java: <https://gist.github.com/Baekjoon/83f0eab9c0eba3e7e6796f04fa1d420b>



$$\begin{aligned} dx[8] &= \{1, -1, 0, 0, 1, 1, -1, -1\} \\ dy[8] &= \{0, 0, 1, -1, 1, -1, 1, -1\} \end{aligned}$$

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제
- DFS 탐색으로는 문제를 풀 수 없다.
- BFS 탐색을 사용해야 한다.
- BFS는 단계별로 진행된다는 사실을 이용

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1					

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2					

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3					

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	→ 1	1	1	1
1	→ 1	1	1	0	1

1	2				
2	3				
3	4	→			
4	→				

미로 탐색

108

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3	4	5			
4	5				

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3				
3	4	5	6		
4	5	6			

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2				
2	3		7		
3	4	5	6	7	
4	5	6	7		

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2		8		
2	3		7	8	
3	4	5	6	7	8
4	5	6	7		

미로 탐색

<https://www.acmicpc.net/problem/2178>

- (1, 1) 에서 (N, M)으로 가는 가장 빠른 길을 구하는 문제

1	1	0	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	1	0	1

1	2		8	9	
2	3		7	8	
3	4	5	6	7	8
4	5	6	7		9

미로 탐색

<https://www.acmicpc.net/problem/2178>

now → next

113

- C/C++: <https://gist.github.com/Baekjoon/a9b4e6fa9a984771172d>
- Java: <https://gist.github.com/Baekjoon/3a591ac8f928c56768a62ac8e072fdfb>

$dist[i] = 1 \Rightarrow i$ 거리

$dist[next] = dist[now] + 1$

토마토

<https://www.acmicpc.net/problem/7576>

- 하루가 지나면, 익은 토마토의 인접한 곳에 있는 익지 않은 토마토들이 익게 된다
- 인접한 곳: 앞, 뒤, 왼쪽, 오른쪽
- 토마토가 저절로 익는 경우는 없다
- 상자안의 익은 토마토와 익지 않은 토마토가 주어졌을 때, 며칠이 지나면 토마토가 모두 익는지 구하는 문제

토마토

115

<https://www.acmicpc.net/problem/7576>

- BFS 탐색을 하면서, 거리를 재는 방식으로 진행한다

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1

8	7	6	5	4	3
7	6	5	4	3	2
6	5	4	3	2	1
5	4	3	2	1	0

토마토

116

<https://www.acmicpc.net/problem/7576>

- C/C++: <https://gist.github.com/Baekjoon/0594e797e87ffd3f6963>
- Java: <https://gist.github.com/Baekjoon/90f0697d73bc04eda7dc3f12fab414ce>

다리 만들기

117

<https://www.acmicpc.net/problem/2146>

- 여러 섬으로 이루어진 나라에서
- 두 섬을 연결하는 가장 짧은 다리를 찾는 문제

다리 만들기

<https://www.acmicpc.net/problem/2146>

- 단지번호붙이기 + 토마토 문제
- 먼저, 섬을 그룹을 나눈다
- $g[i][j] = (i,j)$ 의 그룹 번호
- 그 다음 각각의 그룹에 대해서 다른 섬까지 거리를 계산한다
- 이 방법은 각각이 그룹에 대해서 BFS 알고리즘을 수행해야 하기 때문에 느리다

다리 만들기

<https://www.acmicpc.net/problem/2146>

- C/C++: <https://gist.github.com/Baekjoon/6003108df09bf82c0f84>
- Java: <https://gist.github.com/Baekjoon/b2abbb7d2a385eed36ccd755677f3d81>

- 더 빠른 알고리즘으로 땅을 확장하는 방식을 생각해 볼 수 있다.

[illegible]

다리 만들기

<https://www.acmicpc.net/problem/2146>

- 왼쪽: 그룹, 오른쪽: 거리

1	1	1					2	2	2
1	1	1	1					2	2
1		1	1					2	2
		1	1	1					2
			1						2
									2
				3	3				
				3	3	3			

가장 가까운
섬까지

0	0	0	-	-	-	-	0	0	0
0	0	0	0	-	-	-	-	0	0
0	-	0	0	-	-	-	-	0	0
-	-	0	0	0	-	-	-	-	0
-	-	-	0	-	-	-	-	-	0
-	-	-	-	-	-	-	-	-	0
-	-	-	-	-	-	-	-	-	-
-	-	-	-	0	0	-	-	-	-
-	-	-	-	0	0	0	-	-	-
-	-	-	-	-	-	-	-	-	-

다리 만들기

122

<https://www.acmicpc.net/problem/2146>

- 왼쪽: 그룹, 오른쪽: 거리

1	1	1	1			2	2	2	2
1	1	1	1	1			2	2	2
1	1	1	1	1			2	2	2
1	1	1	1	1	1			2	2
		1	1	1				2	2
			1					2	2
				3	3				2
			3	3	3	3			
			3	3	3	3	3		
				3	3	3			

0	0	0	1	-	-	1	0	0	0
0	0	0	0	1	-	-	1	0	0
0	1	0	0	1	-	-	1	0	0
1	1	0	0	0	1	-	-	1	0
-	-	1	0	1	-	-	-	1	0
-	-	-	1	-	-	-	-	1	0
-	-	-	-	1	1	-	-	-	1
-	-	-	1	0	0	1	-	-	-
-	-	-	1	0	0	0	1	-	-
-	-	-	-	1	1	1	-	-	-

다리 만들기

<https://www.acmicpc.net/problem/2146>

- 왼쪽: 그룹, 오른쪽: 거리

1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1		2	2	2
		1	1	1	3		2	2	2
			1	3	3	3		2	2
		3	3	3	3	3	3		2
		3	3	3	3	3	3	3	
			3	3	3	3	3		

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	-	2	1	0
-	-	2	1	2	2	-	2	1	0
-	-	-	2	1	1	2	-	2	1
-	-	2	1	0	0	1	2	-	2
-	-	2	1	0	0	0	1	2	-
-	-	-	2	1	1	1	2	-	-

다리 만들기

124

<https://www.acmicpc.net/problem/2146>

- 왼쪽: 그룹, 오른쪽: 거리

1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	3	2	2	2	2
		1	1	3	3	3	2	2	2
	3	3	3	3	3	3	3	2	2
	3	3	3	3	3	3	3	3	2
		3	3	3	3	3	3	3	

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	3	2	1	0
3	3	2	1	2	2	3	2	1	0
-	-	3	2	1	1	2	3	2	1
-	3	2	1	0	0	1	2	3	2
-	3	2	1	0	0	0	1	2	3
-	-	3	2	1	1	1	2	3	-

다리 만들기

125

<https://www.acmicpc.net/problem/2146>

- 왼쪽: 그룹, 오른쪽: 거리

1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	3	2	2	2	2
1	1	1	1	3	3	3	2	2	2
3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2
	3	3	3	3	3	3	3	3	2

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	3	2	1	0
3	3	2	1	2	2	3	2	1	0
4	4	3	2	1	1	2	3	2	1
4	3	2	1	0	0	1	2	3	2
4	3	2	1	0	0	0	1	2	3
-	4	3	2	1	1	1	2	3	4

다리 만들기

126

<https://www.acmicpc.net/problem/2146>

- 왼쪽: 그룹, 오른쪽: 거리

1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	3	2	2	2	2
1	1	1	1	3	3	3	2	2	2
3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2
3	3	3	3	3	3	3	3	3	2

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	3	2	1	0
3	3	2	1	2	2	3	2	1	0
4	4	3	2	1	1	2	3	2	1
4	3	2	1	0	0	1	2	3	2
4	3	2	1	0	0	0	1	2	3
5	4	3	2	1	1	1	2	3	4

다리 만들기

<https://www.acmicpc.net/problem/2146>

- 각 칸과 인접한 칸의 그룹 번호가 다르면 다리를 만들 수 있다

1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	3	2	2	2	2
1	1	1	1	3	3	3	2	2	2
3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2
3	3	3	3	3	3	3	3	3	2

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	3	2	1	0
3	3	2	1	2	2	3	2	1	0
4	4	3	2	1	1	2	3	2	1
4	3	2	1	0	0	1	2	3	2
4	3	2	1	0	0	0	1	2	3
5	4	3	2	1	1	1	2	3	4

다리 만들기

<https://www.acmicpc.net/problem/2146>

- 길이: 4



1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	3	3	2	2	2
3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2
3	3	3	3	3	3	3	3	3	2

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	3	2	1	0
3	3	2	1	2	2	3	2	1	0
4	4	3	2	1	1	2	3	2	1
4	3	2	1	0	0	1	2	3	2
4	3	2	1	0	0	0	1	2	3
5	4	3	2	1	1	1	2	3	4

다리 만들기

<https://www.acmicpc.net/problem/2146>

- 길이: $2+1=3$

1	1	1	1	1	2	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	2	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	1	1	2	2	2
1	1	1	1	1	3	2	2	2	2
1	1	1	1	3	3	3	2	2	2
3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2
3	3	3	3	3	3	3	3	3	2

0	0	0	1	2	2	1	0	0	0
0	0	0	0	1	2	2	1	0	0
0	1	0	0	1	2	2	1	0	0
1	1	0	0	0	1	2	2	1	0
2	2	1	0	1	2	3	2	1	0
3	3	2	1	2	2	3	2	1	0
4	4	3	2	1	1	2	3	2	1
4	3	2	1	0	0	1	2	3	2
4	3	2	1	0	0	0	1	2	3
5	4	3	2	1	1	1	2	3	4

다리 만들기

130

<https://www.acmicpc.net/problem/2146>

- C/C++: <https://gist.github.com/Baekjoon/8eaddae5eb16592a1c51>
- Java: <https://gist.github.com/Baekjoon/fdaea828f9374c3f5fc35ae446da704d>