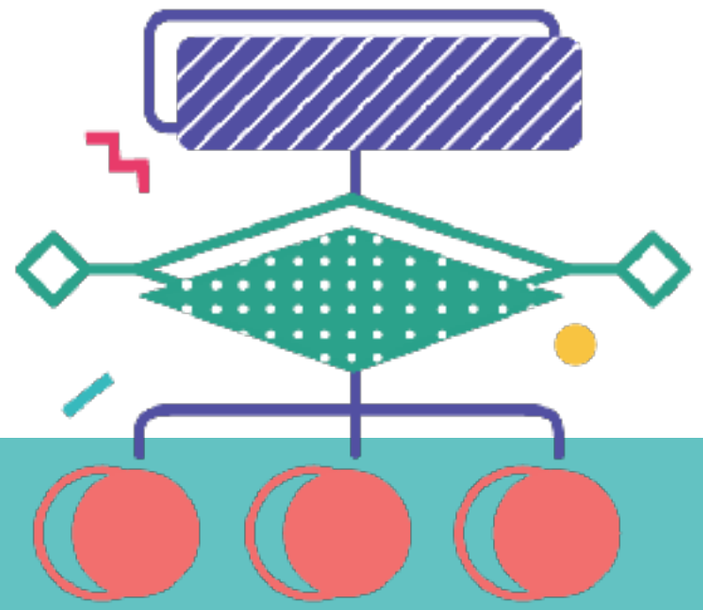
/* 알고리즘 II */

신현규 강사, 일 14:00

8월 20일 ~ 9월 16일



/* elice */

목차

01 지난 시간 요약

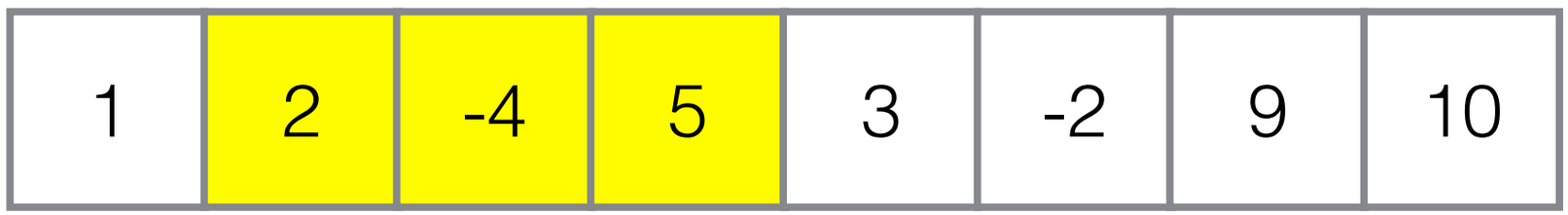
[예제 3] 연속 부분 최대합

연속된 부분을 선택하였을 때, 그 최대 합을 출력

1	2	-4	5	3	-2	9	10

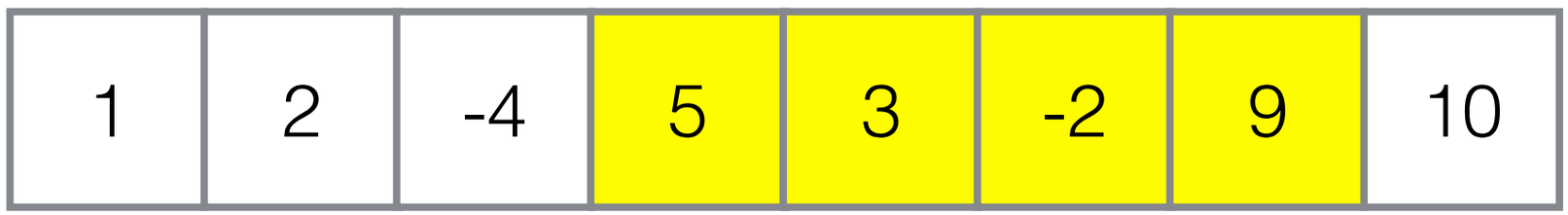
[예제 3] 연속 부분 최대합

연속된 부분을 선택하였을 때, 그 최대 합을 출력



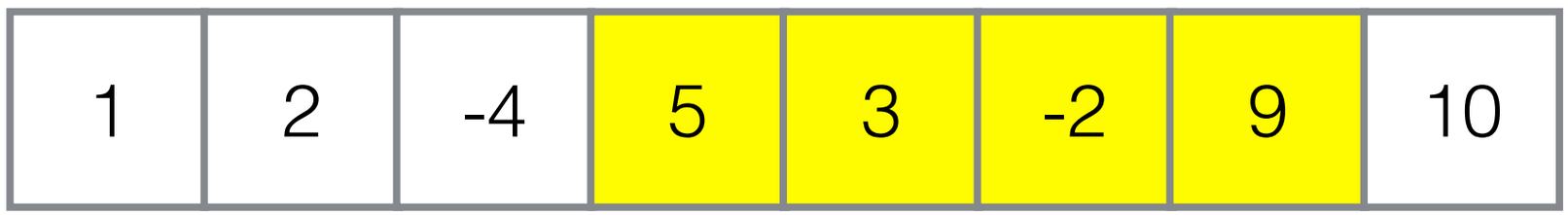
[예제 3] 연속 부분 최대합

연속된 부분을 선택하였을 때, 그 최대 합을 출력



[예제 3] 연속 부분 최대합

연속된 부분을 선택하였을 때, 그 최대 합을 출력



완전탐색법	**$O(n^3)$**
분할정복법	**O(n log n)**

[예제 3] 연속 부분 최대합

연속된 부분을 선택하였을 때, 그 최대 합을 출력

1	2	-4	5	3	-2	9	10

완전탐색법 **$O(n^3)$**

분할정복법 **O(n log n)**

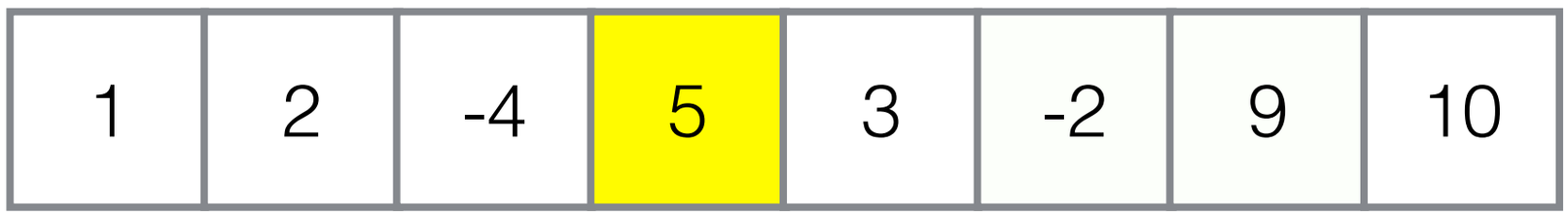
[예제 3] 연속 부분 최대합 (DP)

연속된 부분을 선택하였을 때, 그 최대 합을 출력

1	2	-4	5	3	-2	9	10

[예제 3] 연속 부분 최대합 (DP)

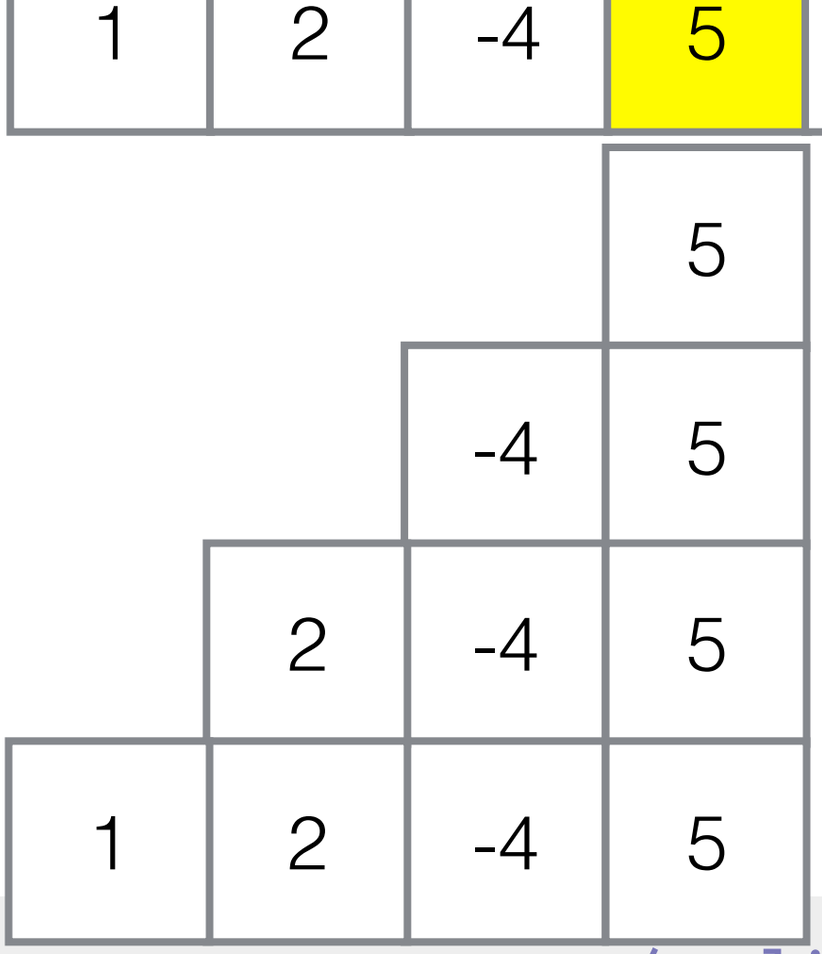
연속된 부분을 선택하였을 때, 그 최대 합을 출력



[예제 3] 연속 부분 최대합 (DP)

연속된 부분을 선택하였을 때, 그 최대 합을 출력

1	2	-4	5	3	-2	9	10



[예제 3] 연속 부분 최대합 (DP)

연속된 부분을 선택하였을 때, 그 최대 합을 출력

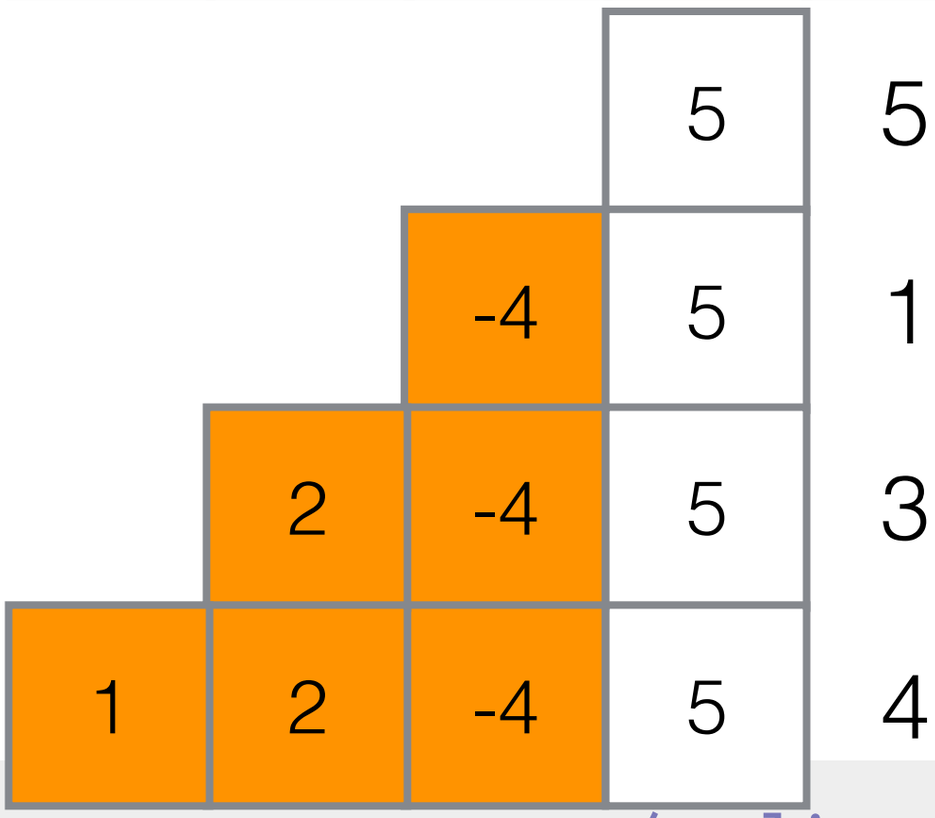
1	2	-4	5	3	-2	9	10

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

[예제 3] 연속 부분 최대합 (DP)

연속된 부분을 선택하였을 때, 그 최대 합을 출력

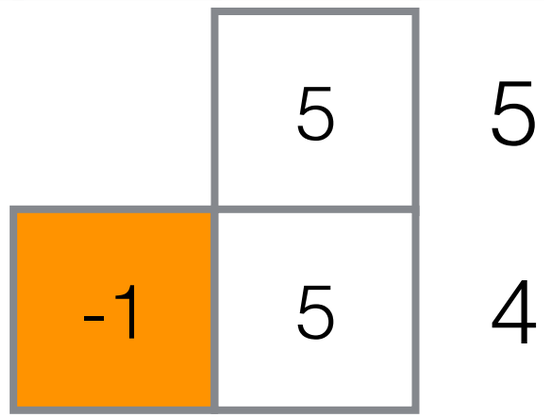
1	2	-4	5	3	-2	9	10



[예제 3] 연속 부분 최대합 (DP)

연속된 부분을 선택하였을 때, 그 최대 합을 출력

1	2	-4	5	3	-2	9	10



동적 계획법의 문제 풀이 순서

1. 부분 문제를 정의한다.
   무슨 값을 구할지를 정의한다

2. 점화식을 구한다
   그 값을 어떻게 구할지에 대한 식을 세운다

3. 문제를 해결한다
   값을 직접 구하는 코드를 작성한다

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

1. 부분 문제를 정의한다.
   무슨 값을 구할지를 정의한다

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

1. 부분 문제를 정의한다.
   무슨 값을 구할지를 정의한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

2. 점화식을 구한다
   그 값을 어떻게 구할지에 대한 식을 세운다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

2. 점화식을 구한다
   그 값을 어떻게 구할지에 대한 식을 세운다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

동적 계획법의 문제 풀이 순서

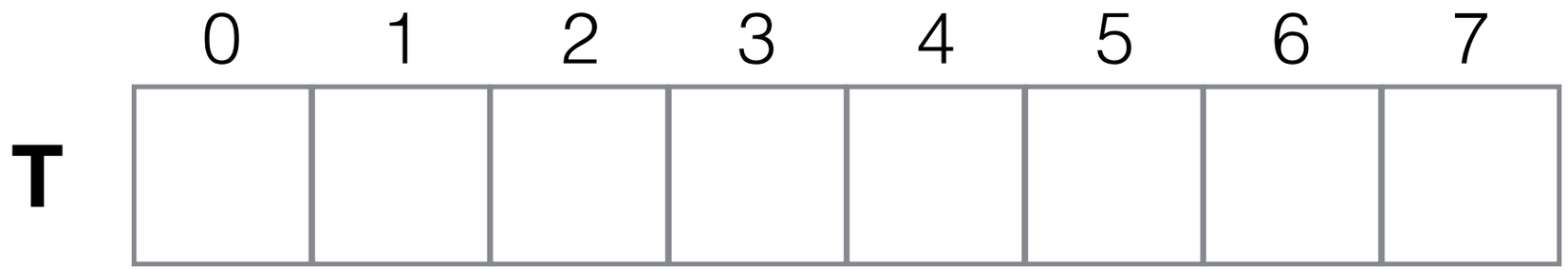
1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))



동적 계획법의 문제 풀이 순서

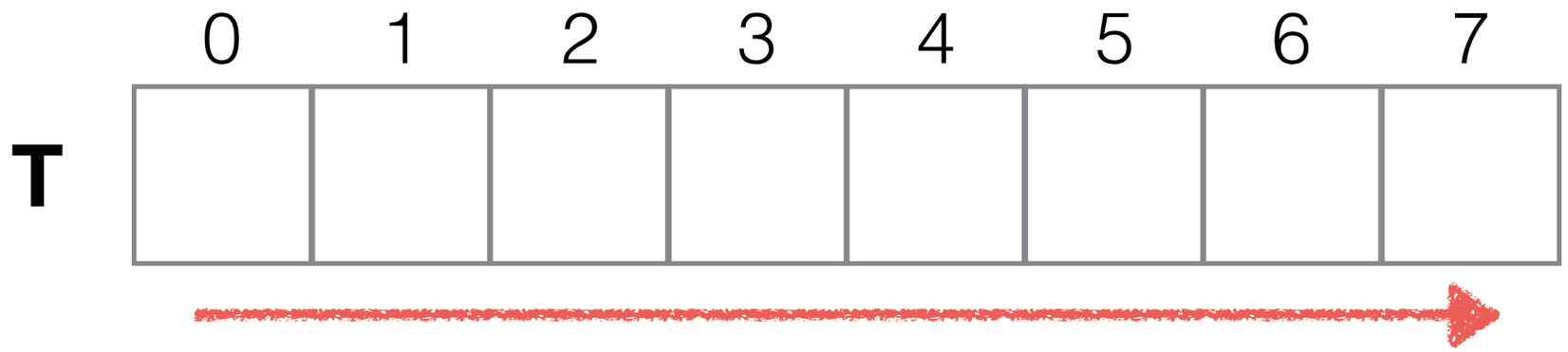
1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))



동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1							

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3						

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

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

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

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

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3	1	6	9			

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3	1	6	9	7		

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3	1	6	9	7	16	

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

3. 문제를 해결한다

   값을 직접 구하는 코드를 작성한다

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3	1	6	9	7	16	26

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

그래서 답은 어디있나 ?

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3	1	6	9	7	16	26

동적 계획법의 문제 풀이 순서

1	2	-4	5	3	-2	9	10

그래서 답은 어디있나 ?

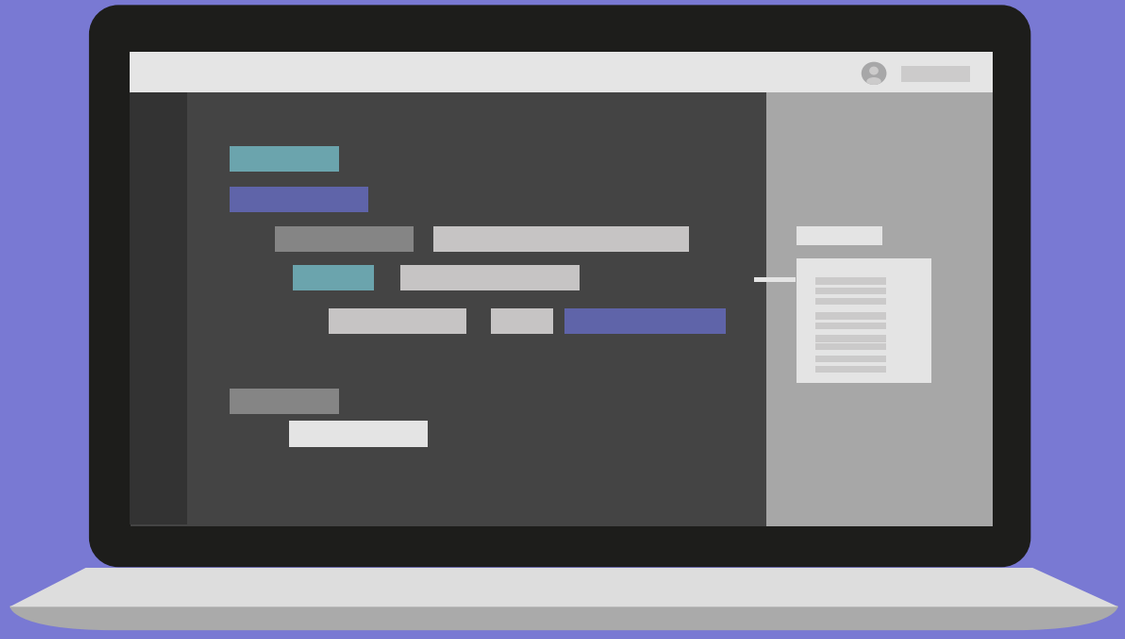
T의 최댓값

T(i) = i번째 숫자를 오른쪽 끝으로 하는 연속 부분 최대합

T(i) = max(T(i-1) + Data(i), Data(i))

	0	1	2	3	4	5	6	7
T	1	3	1	6	9	7	16	26

[예제 3] 연속 부분 최대합



[예제 1] 최장 증가 부분 수열

최장 증가 부분 수열의 길이를 구하라

입력의 예

```
5
1 4 2 3 5
```

출력의 예

```
4
```

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

5를 포함하는 경우 :

5를 포함하지 않는 경우 :

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

5를 포함하는 경우 :

5를 포함하지 않는 경우 : (1 4 2 3 의 최장 증가 부분 수열)

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

5를 포함하는 경우 : **???**

5를 포함하지 않는 경우 : (1 4 2 3 의 최장 증가 부분 수열)

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

... 1 5 ... 4 5

5를 포함하는 경우 : ... 2 5 ... 3 5

5를 포함하지 않는 경우 : (1 4 2 3 의 최장 증가 부분 수열)

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

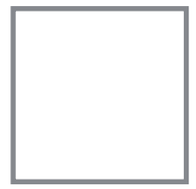
T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

i



[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

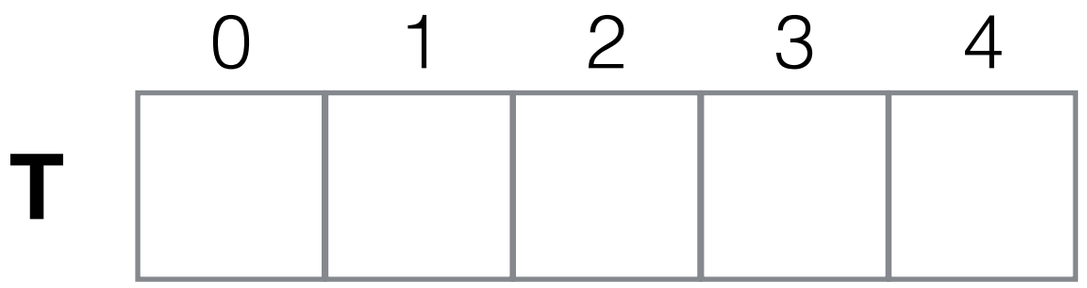
T(i) = max(T(j) + 1) if Data(j) < Data(i)

[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)

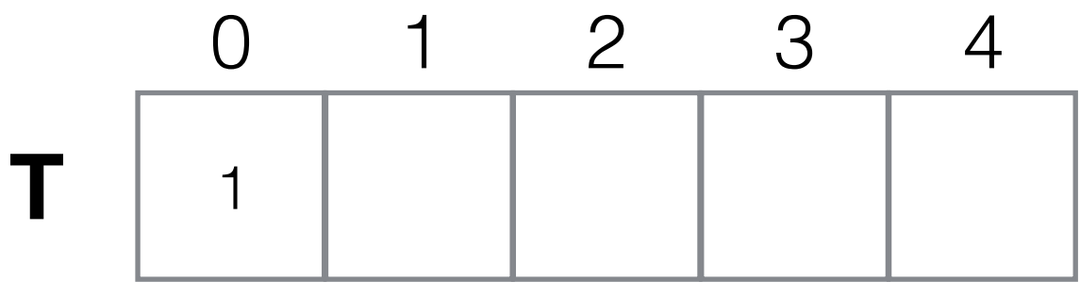


[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)

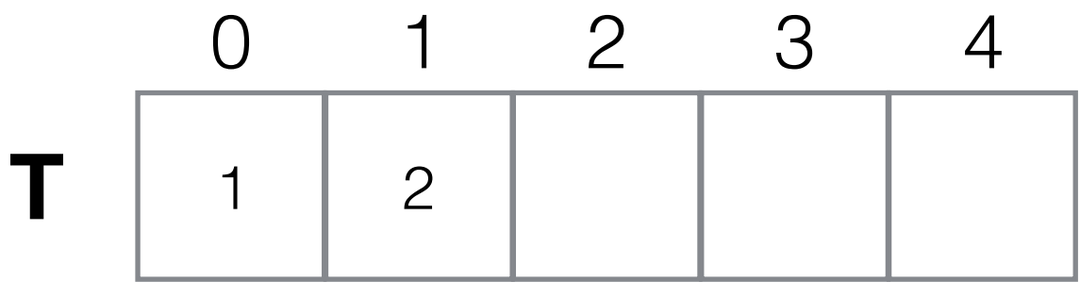


[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)

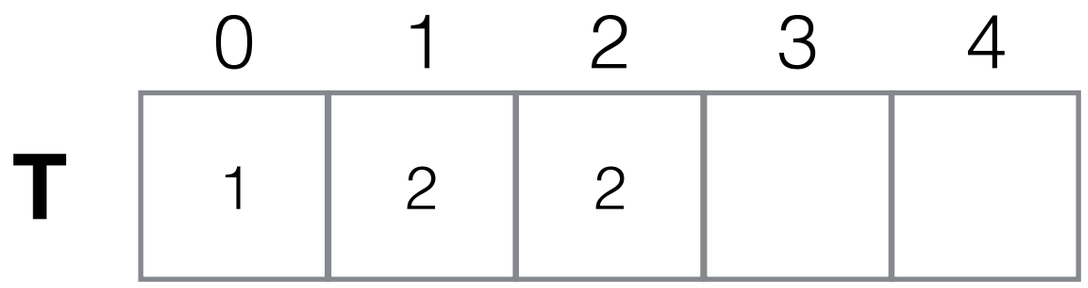


[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)

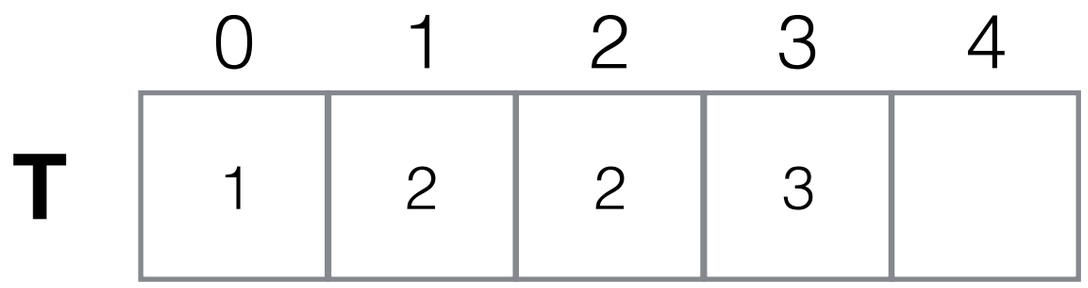


[예제 1] 최장 증가 부분 수열

1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)

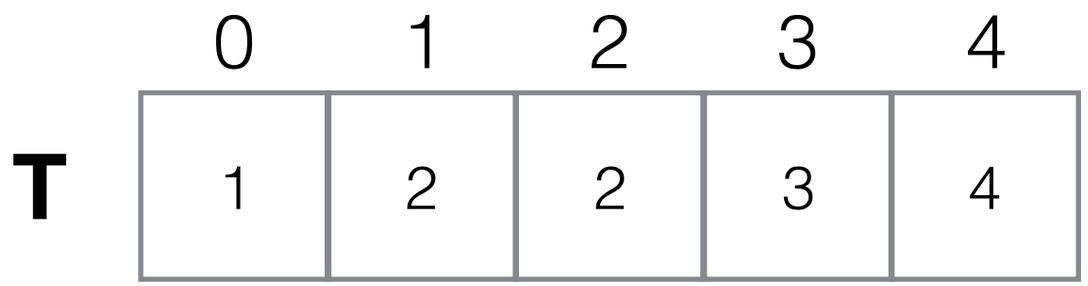


[예제 1] 최장 증가 부분 수열

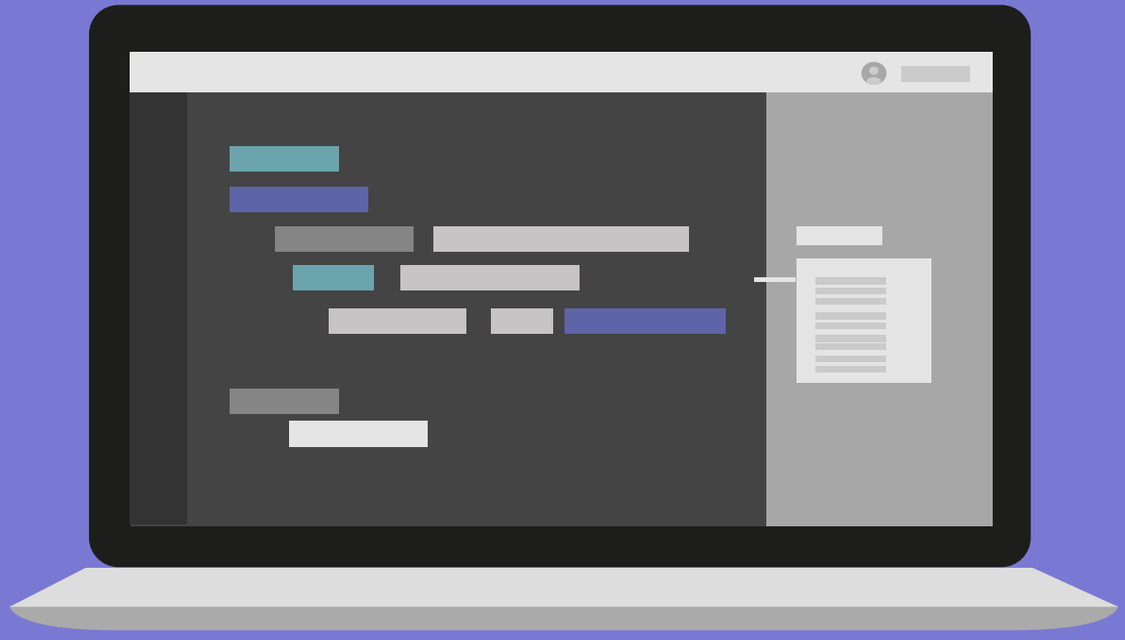
1 4 2 3 5

T(i) = i번째 숫자를 끝으로 하는 최장 증가 부분 수열의 길이

T(i) = max(T(j) + 1) if Data(j) < Data(i)



[예제 1] 최장 증가 부분 수열



[예제 2] 최대 공통 부분 수열

최대 공통 부분 수열의 길이를 구하라

입력의 예

```
Telephone
Television
```

출력의 예

```
6
```

[예제 2] 최대 공통 부분 수열

Telephone

Television

[예제 2] 최대 공통 부분 수열

T e l e p h o n **e**

T e l e v i s i o **n**

[예제 2] 최대 공통 부분 수열

Telephon**e**

Televisio**n**

MAX(Telephon**e** , Televisio**n** **)**

[예제 2] 최대 공통 부분 수열

Telephon**e**

Televisio**n**

$T(i, j) = s_1[1..i], s_2[1..j]$ 의 최대 공통 부분 수열의 길이

[예제 2] 최대 공통 부분 수열

T e l e p h o n **e**

T e l e v i s i o **n**

T(i, j) = s_1[1..i], s_2[1..j] 의 최대 공통 부분 수열의 길이

[예제 2] 최대 공통 부분 수열

Telephone

Television

T(i, j) = $s_1[1..i]$, $s_2[1..j]$ 의 최대 공통 부분 수열의 길이

[예제 2] 최대 공통 부분 수열

T e l e p h o n **e**

T e l e v i s i o **n**

T(i, j) = s_1[1..i], s_2[1..j] 의 최대 공통 부분 수열의 길이

T(i, j) = max(T(i, j-1), T(i, j-1))

[예제 2] 최대 공통 부분 수열

Telephon**e**

Televisio**n**

T(i, j) = s_1[1..i], s_2[1..j] 의 최대 공통 부분 수열의 길이

[예제 2] 최대 공통 부분 수열

T e l e p h o n **e**

T e l e v i s i o **n**

T(i, j) = s_1[1..i], s_2[1..j] 의 최대 공통 부분 수열의 길이

T(i, j) = T(i-1, j-1) + 1

[예제 2] 최대 공통 부분 수열

T e l e p h o n **e**

T e l e v i s i o **n**

T(i, j) = s_1[1..i], s_2[1..j] 의 최대 공통 부분 수열의 길이

$$T(i, j) = \begin{cases} \max(\ T(i-1, j), T(I, j-1)\ ) & \text{if } s_1[i] \neq s_2[j] \\ T(i-1, j-1) + 1 & \text{if } s_1[i] == s_2[j] \end{cases}$$

[예제 2] 최대 공통 부분 수열

		T	e	l	e	v	i	s	i	o	n
T											
e											
l											
e											
p											
h											
o											
n											
e											

[예제 2] 최대 공통 부분 수열

		T	e	l	e	v	i	s	i	o	n
	T	1									
	e										
	l										
	e										
	p										
	h										
	o										
	n										
	e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1								
e										
l										
e										
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e										
l										
e										
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

		T	e	l	e	v	i	s	i	o	n
		1	1	1	1	1	1	1	1	1	1
e	1										
l											
e											
p											
h											
o											
n											
e											

[예제 2] 최대 공통 부분 수열

		T	e	l	e	v	i	s	i	o	n
	T	1	1	1	1	1	1	1	1	1	1
	e	1	2								
	l										
	e										
	p										
	h										
	o										
	n										
	e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l										
e										
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1									
e										
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

		T	e	l	e	v	i	s	i	o	n
		1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2	2
l	1	2									
e											
p											
h											
o											
n											
e											

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3							
e										
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e										
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1									
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2								
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3							
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4						
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p										
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1									
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1	2	3	4	4	4	4	4	4	4
h										
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1	2	3	4	4	4	4	4	4	4
h	1	2	3	4	4	4	4	4	4	4
o										
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1	2	3	4	4	4	4	4	4	4
h	1	2	3	4	4	4	4	4	4	4
o	1	2	3	4	4	4	4	4	5	5
n										
e										

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1	2	3	4	4	4	4	4	4	4
h	1	2	3	4	4	4	4	4	4	4
o	1	2	3	4	4	4	4	4	5	5
n	1	2	3	4	4	4	4	4	5	6
e										

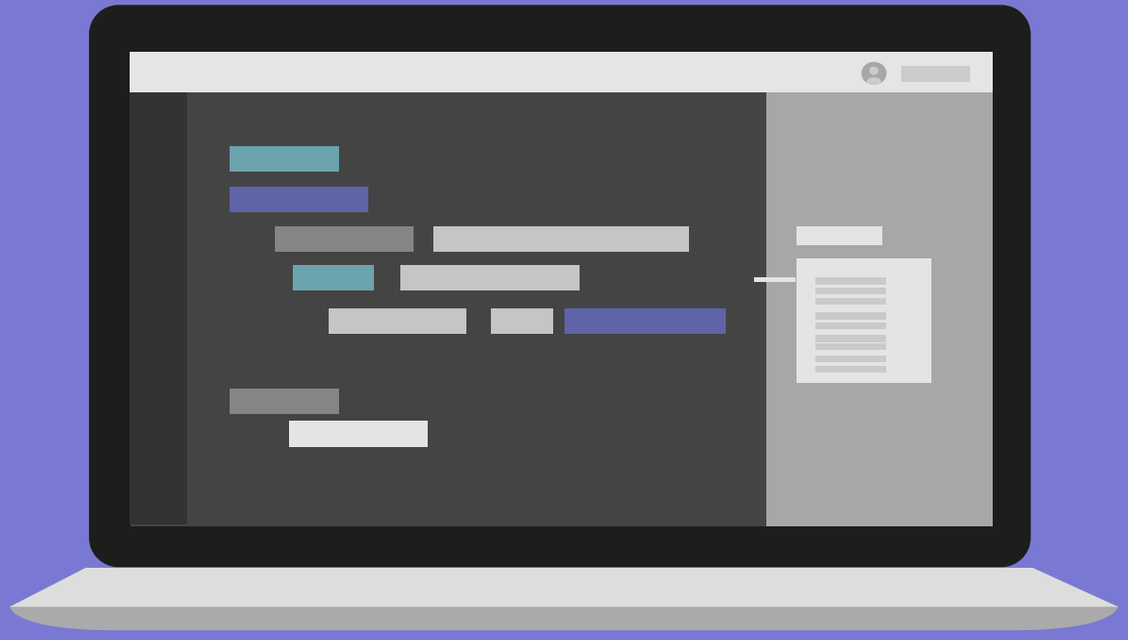
[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1	2	3	4	4	4	4	4	4	4
h	1	2	3	4	4	4	4	4	4	4
o	1	2	3	4	4	4	4	4	5	5
n	1	2	3	4	4	4	4	4	5	6
e	1	2	3	4	4	4	4	4	5	6

[예제 2] 최대 공통 부분 수열

	T	e	l	e	v	i	s	i	o	n
T	1	1	1	1	1	1	1	1	1	1
e	1	2	2	2	2	2	2	2	2	2
l	1	2	3	3	3	3	3	3	3	3
e	1	2	3	4	4	4	4	4	4	4
p	1	2	3	4	4	4	4	4	4	4
h	1	2	3	4	4	4	4	4	4	4
o	1	2	3	4	4	4	4	4	5	5
n	1	2	3	4	4	4	4	4	5	6
e	1	2	3	4	4	4	4	4	5	**6**

[예제 2] 최대 공통 부분 수열



[문제 1] 숫자 삼각형

가장 위에서 가장 아래로 내려갈 때,
최대 경로에 존재하는 숫자의 합은 ?

입력의 예

```
4
3 5
1 3 8
2 5 3 4
3 6 6 8 2
```

출력의 예

```
29
```

[문제 1] 숫자 삼각형

T(i, j) = (i, j) 에 도착하였을 때, 얻을 수 있는 합의 최댓값

[문제 1] 숫자 삼각형

T(i, j) = (i, j) 에 도착하였을 때, 얻을 수 있는 합의 최댓값

T(i, j) = max(T(i-1, j-1), T(i-1, j)) + Data(i, j)

[문제 2] 두 문자열 사이의 거리

두 문자열 사이의 거리를 구하여라

단, 거리란 삽입/삭제의 연산의 횟수를 말한다

입력의 예

```
abc
bdcf
```

출력의 예

```
3
```

[문제 2] 두 문자열 사이의 거리

a b c

b d c f

[문제 2] 두 문자열 사이의 거리

a b c

b d c f

((첫번째 문자열의 길이) - LCS) + ((두번째 문자열의 길이) - LCS)

[문제 3] 팰린드롬 만들기

팰린드롬을 만들기 위해서

추가해야 하는 문자의 최소 개수는 ?

입력의 예

```
abcfba
```

출력의 예

```
1
```

[문제 3] 팰린드롬 만들기

a b c d e d c b a

[문제 3] 팰린드롬 만들기

a b c d e d c b a

[문제 3] 팰린드롬 만들기

a b e d e c c p a

[문제 3] 팰린드롬 만들기

a b e d e c c p a

[문제 3] 팰린드롬 만들기

a b e d e c c p b

[문제 3] 팰린드롬 만들기

a b e d e c c p b

[문제 3] 팰린드롬 만들기

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

$$T(i, j) = \begin{cases} T(i+1, j-1) & \text{if } s_1(i) == s_2(j) \end{cases}$$

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

$$T(i, j) = \begin{cases} T(i+1, j-1) & \text{if } s_1(i) == s_2(j) \\ \min(T(i+1, j), T(i, j-1)) + 1 & \text{otherwise} \end{cases}$$

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b						
f						
d						

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b						
f						
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b						
f					0	
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b						
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b				0		
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b				0	1	
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c						
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c			0			
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c			0	1		
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c			0	1	2	
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b						
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b		0				
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b		0	1			
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b		0	1	**0**		
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b		0	1	0	1	
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a						
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a	0					
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a	0	1				
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a	0	1	2			
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a	0	1	2	1		
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

	a	b	c	b	f	d
a	0	1	2	1	2	
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

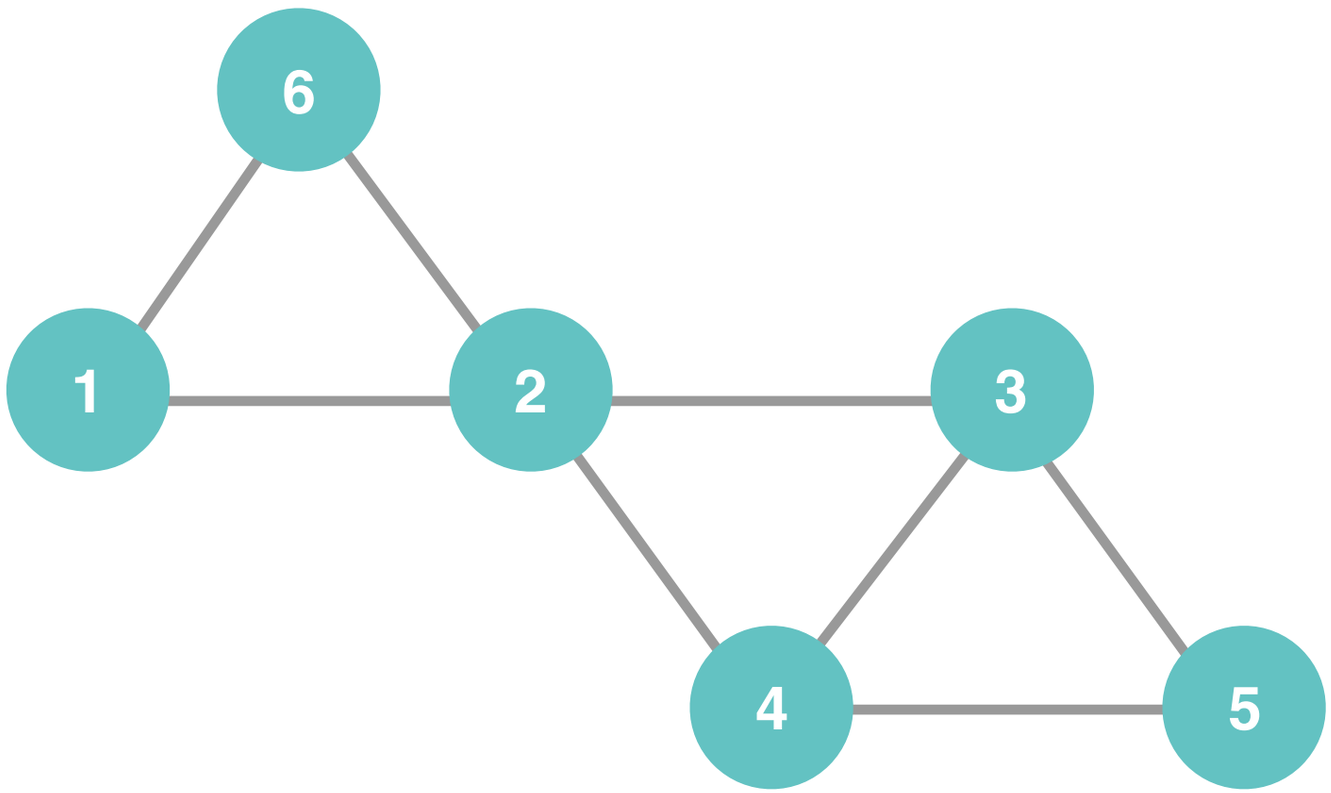
[문제 3] 팰린드롬 만들기

T(i, j) = i번째 문자부터 j번째 문자까지를 팰린드롬으로 만들기 위해 추가해야 하는 문자의 최소 개수

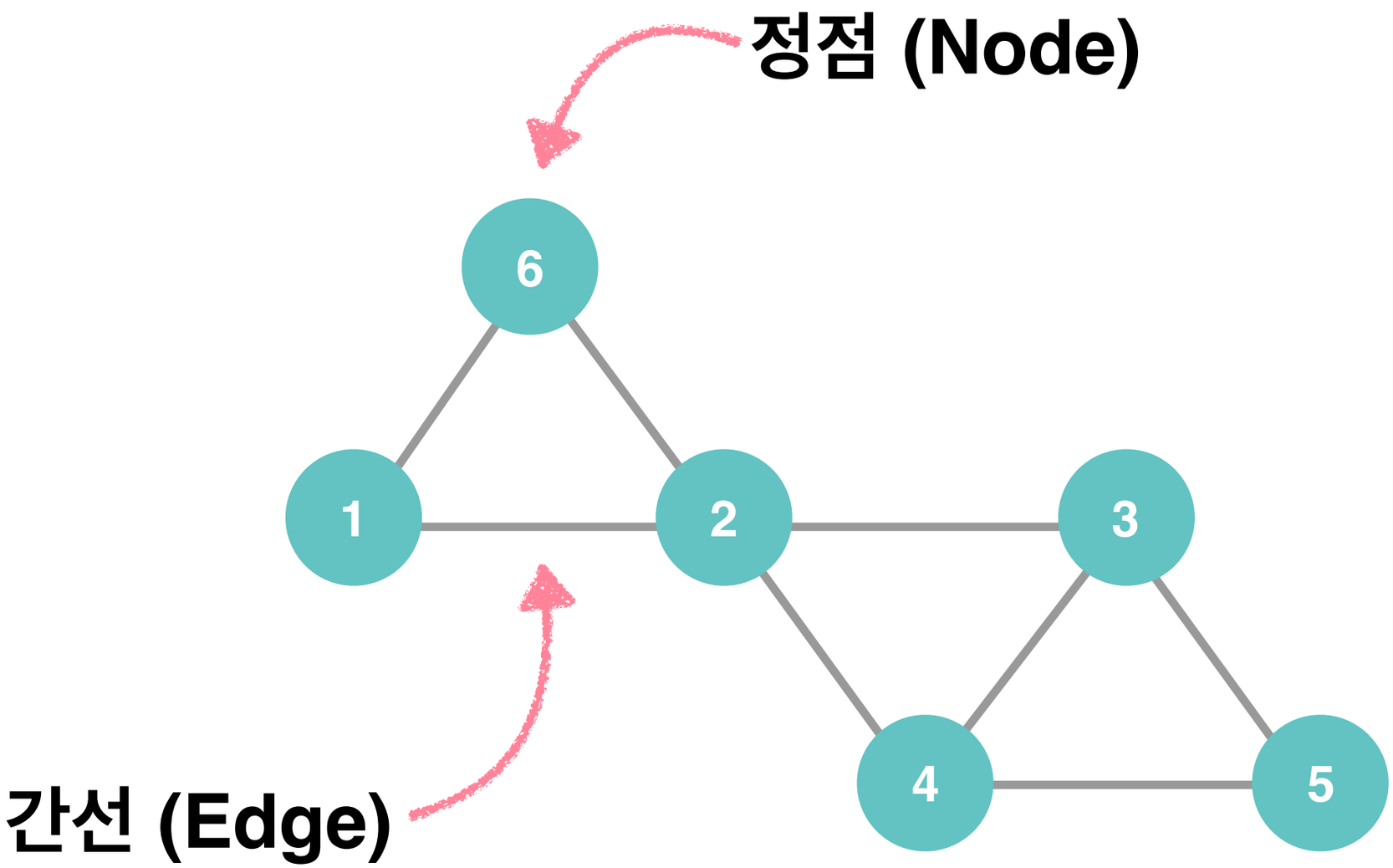
	a	b	c	b	f	d
a	0	1	2	1	2	3
b		0	1	0	1	2
c			0	1	2	3
b				0	1	2
f					0	1
d						0

그래프

정점과 간선으로 이루어진 자료구조



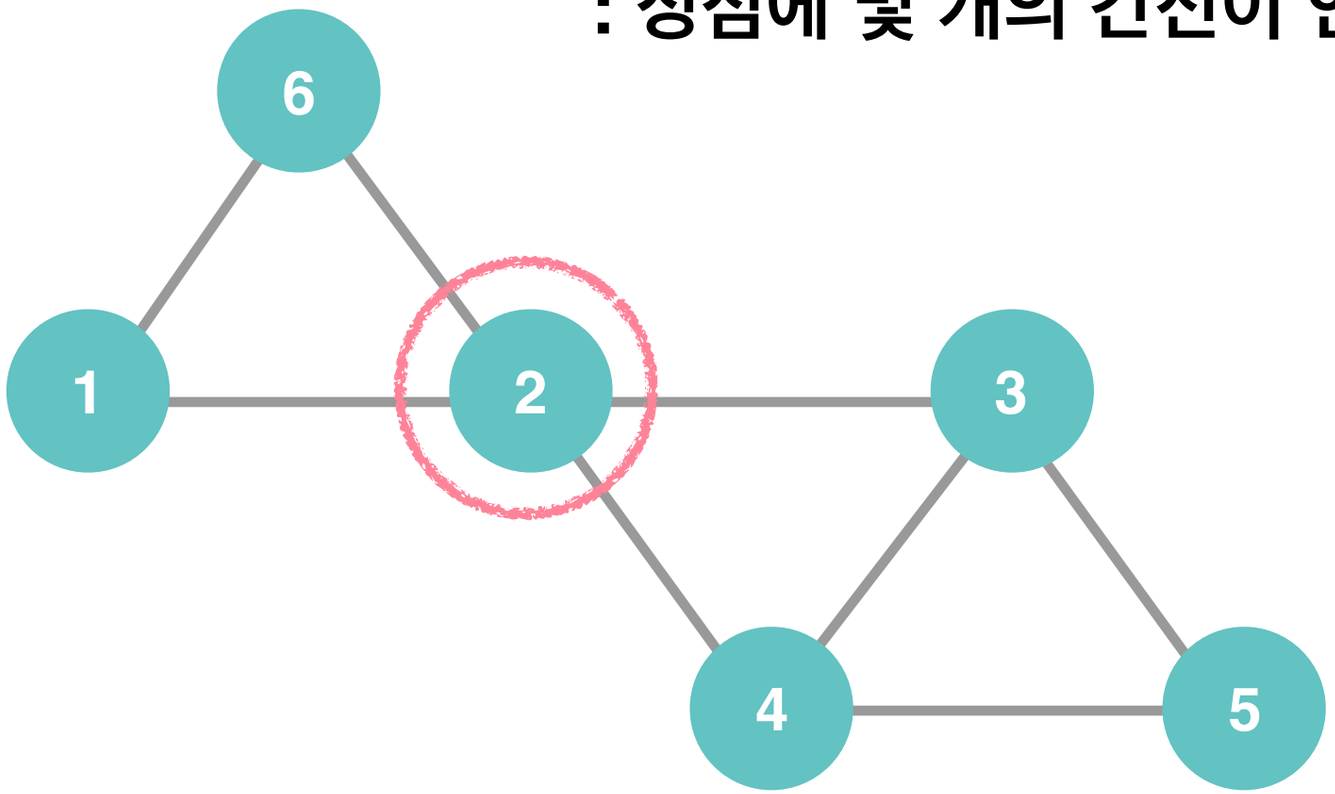
용어



용어

차수 (Degree)

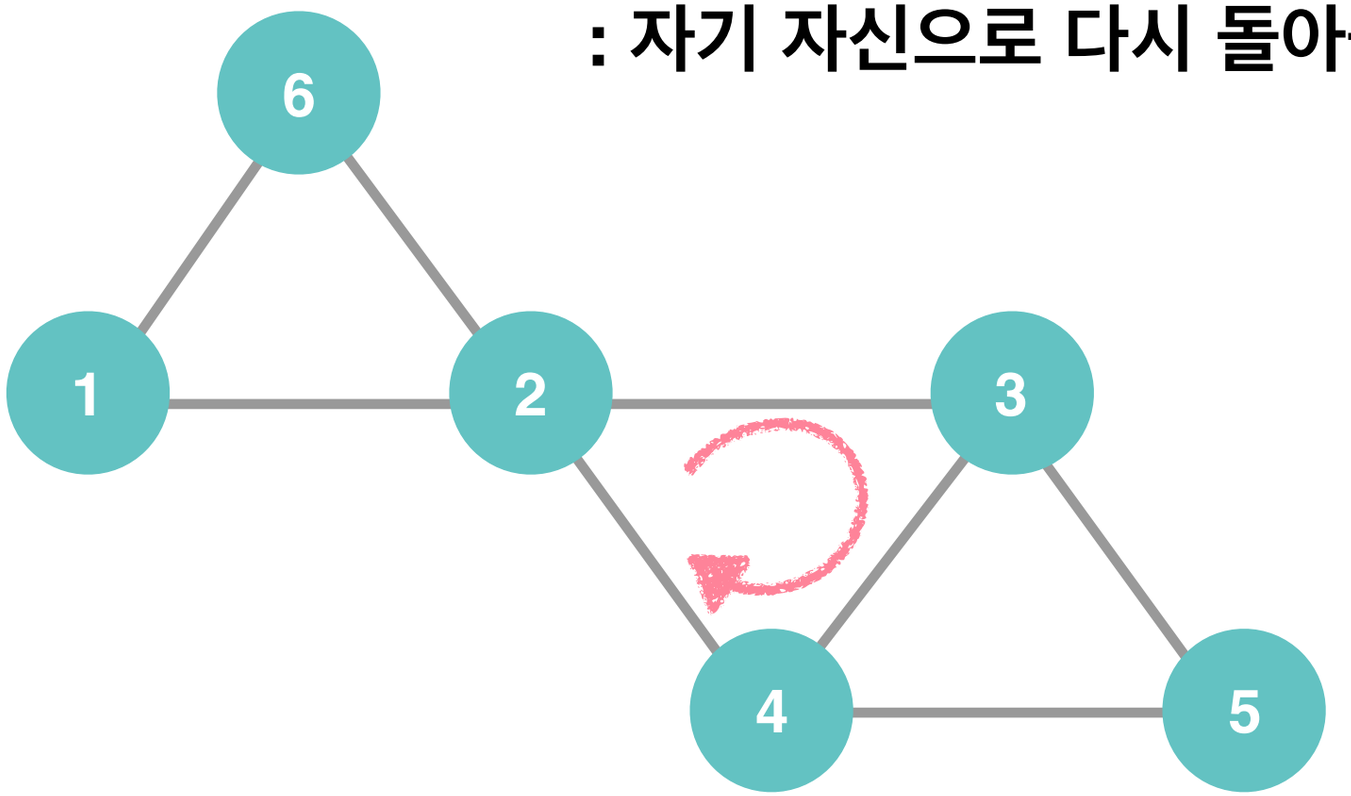
: 정점에 몇 개의 간선이 연결되어 있는가 ?



용어

사이클 (Cycle)

: 자기 자신으로 다시 돌아올 수 있는 경로



그래프는 왜 중요한가 ?

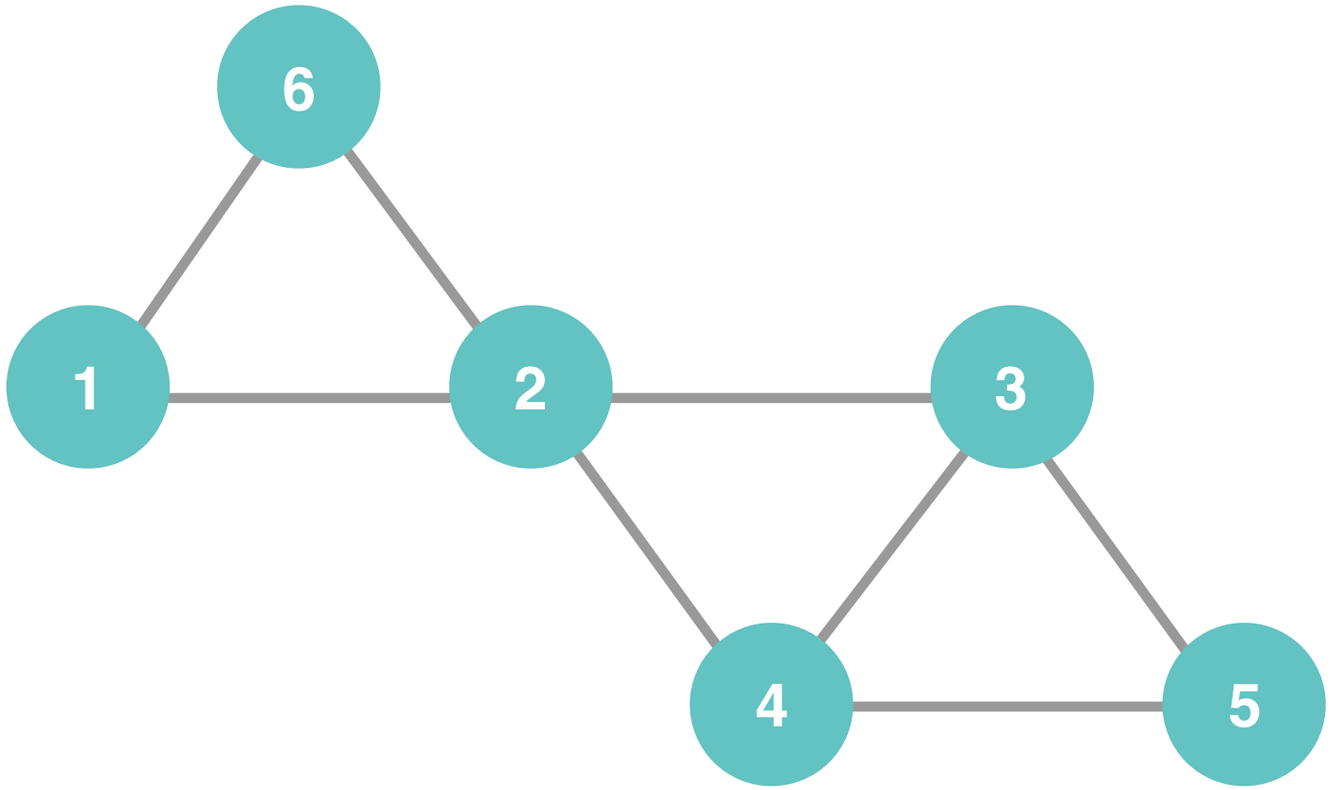
현실 세계의 많은 것들을 그래프로 나타낼 수 있다
즉, 그래프와 관련된 문제가 매우 많다

그래프와 관련된 수학적 정리가 매우 많다
그래프 이론이라는 분야가 따로 있다 (Graph theory)

어렵다(...)
그래프와 관련된 이론도 어렵고 구현도 어렵고...ㅠㅠ

그래프 순회

그래프라는 자료구조에 어떠한 값이 저장되어 있는가?



그래프 순회

깊이우선탐색 (Depth First Search)

너비우선탐색 (Breadth First Search)

그래프 순회

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

너비우선탐색 (Breadth First Search)

큐를 이용하여 그래프를 순회하는 방법

깊이우선탐색 (DFS)

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

깊이우선탐색 (DFS)

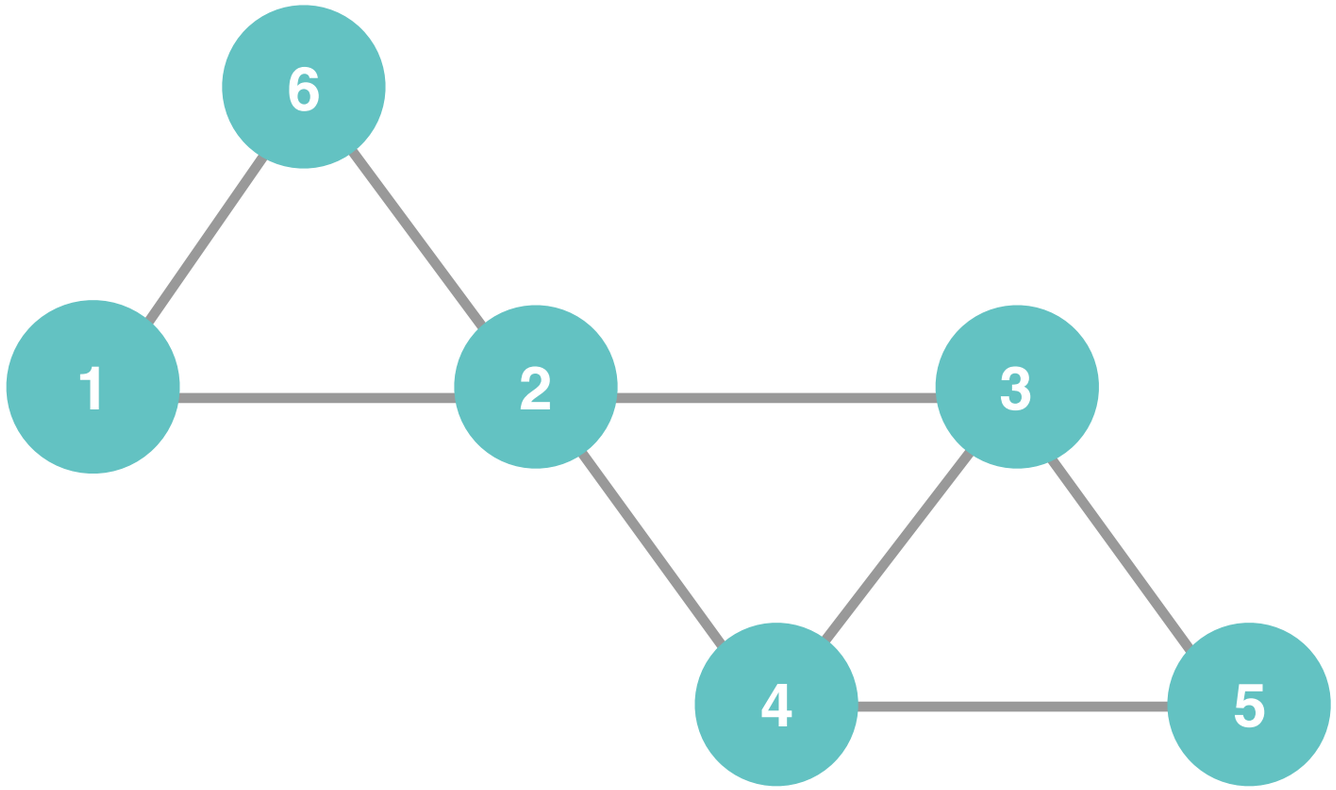
깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

나를 먼저 방문하고, 이웃한 정점을 방문한다

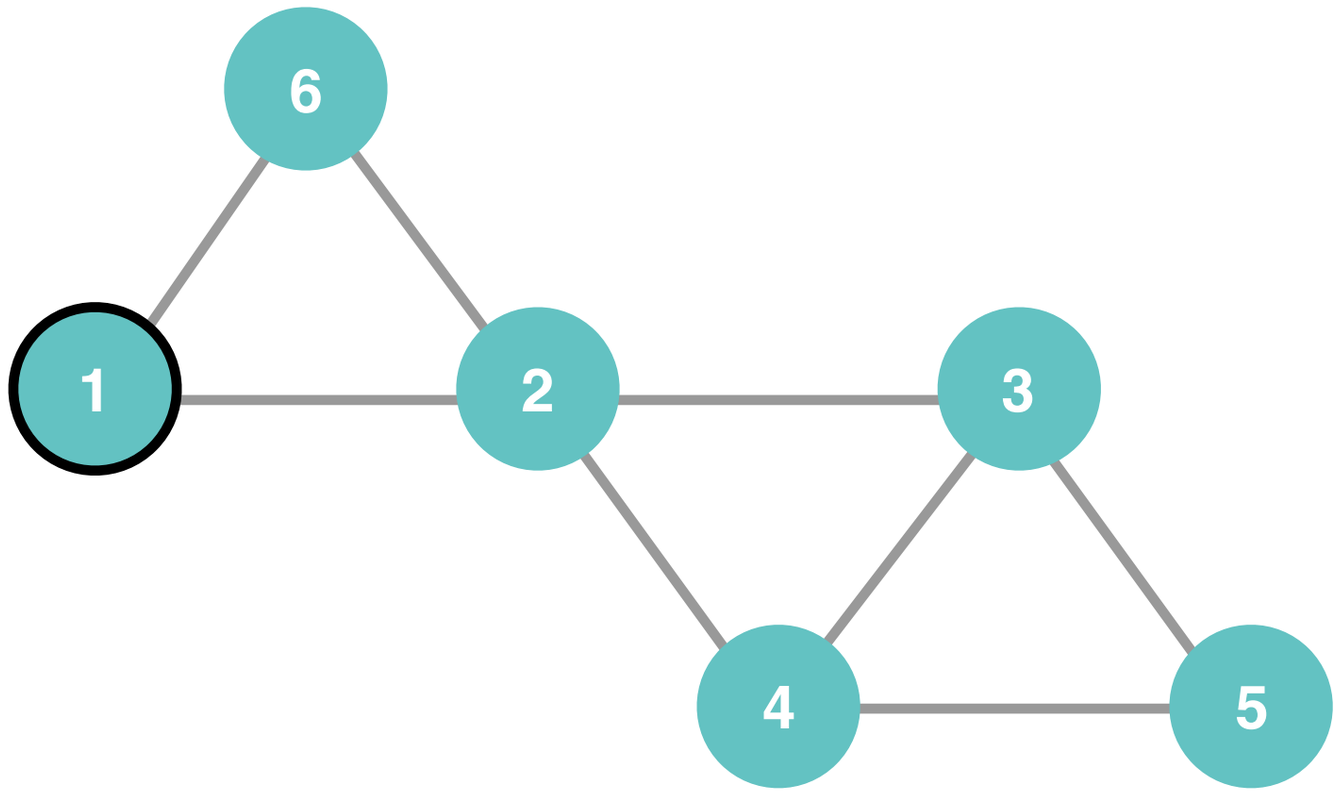
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



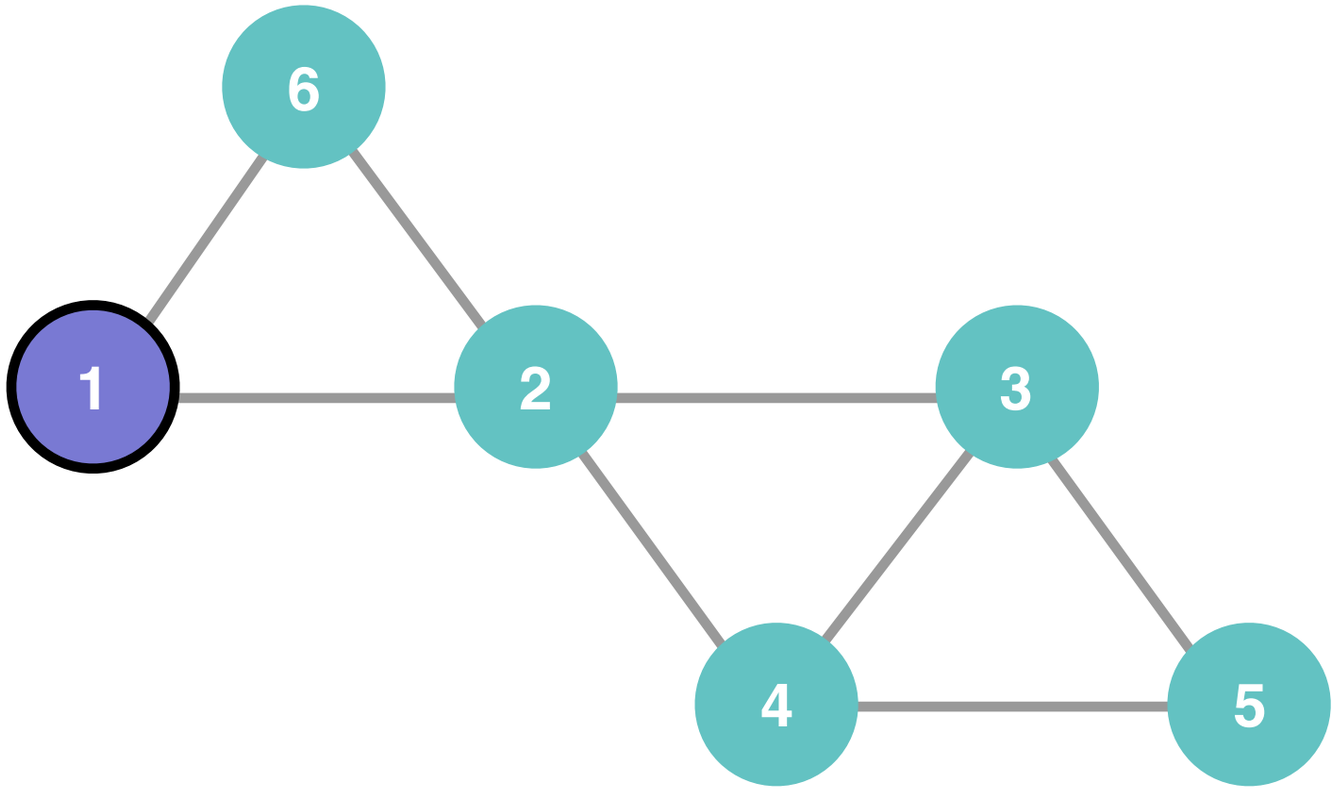
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



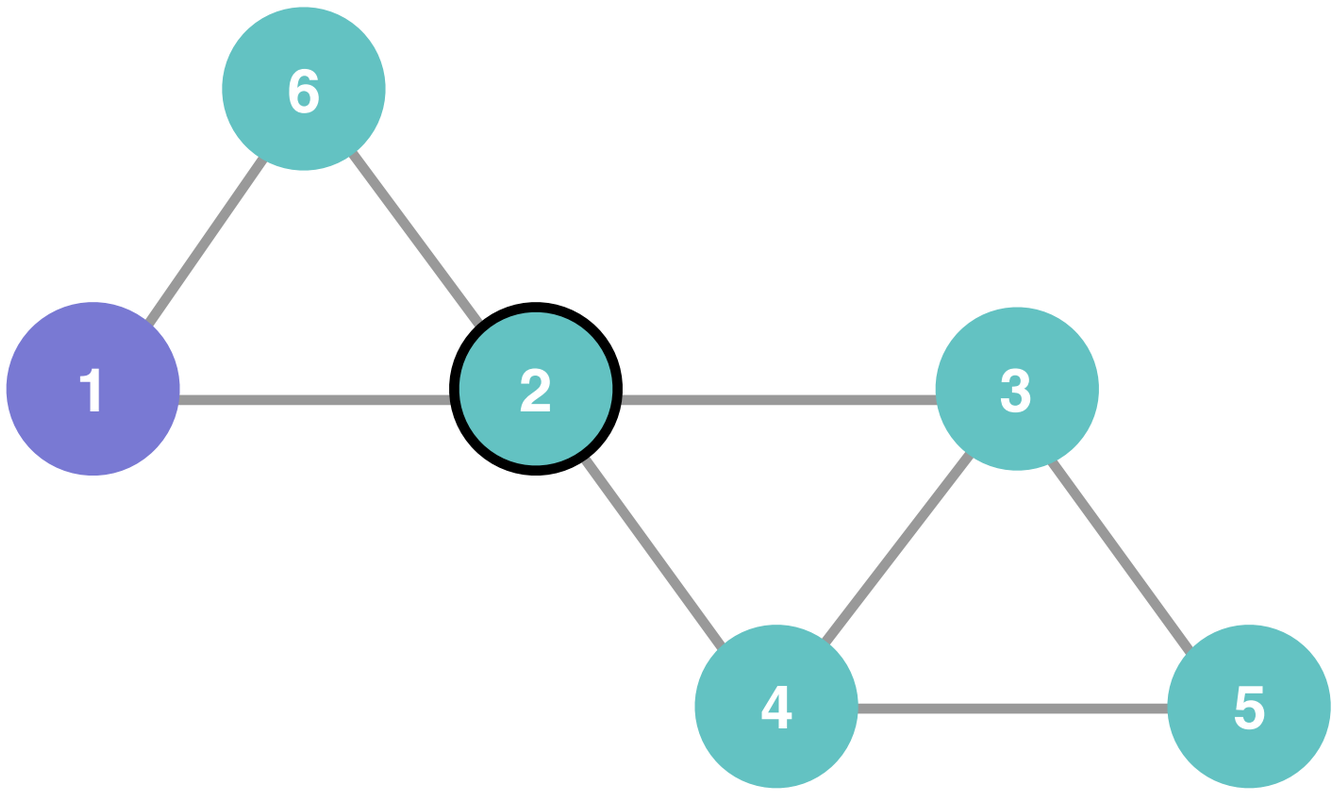
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



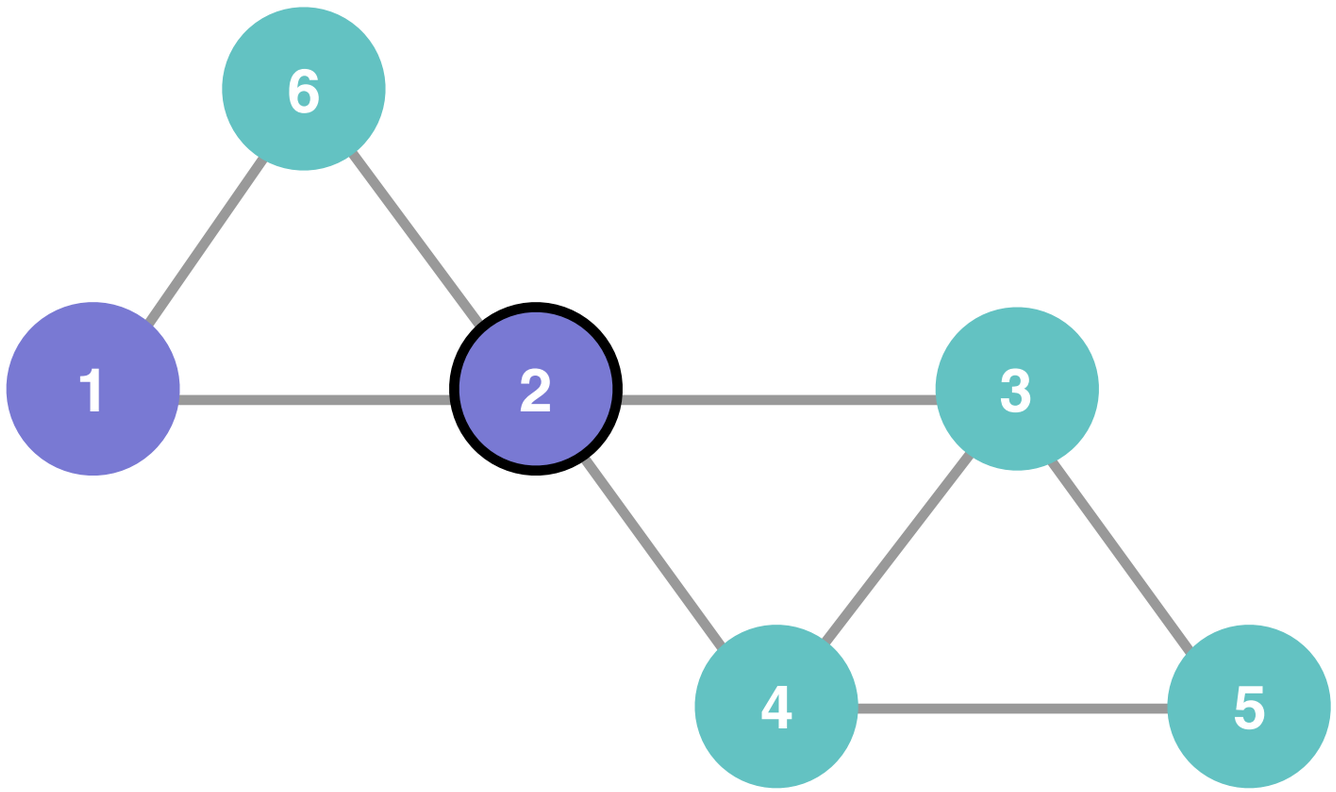
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



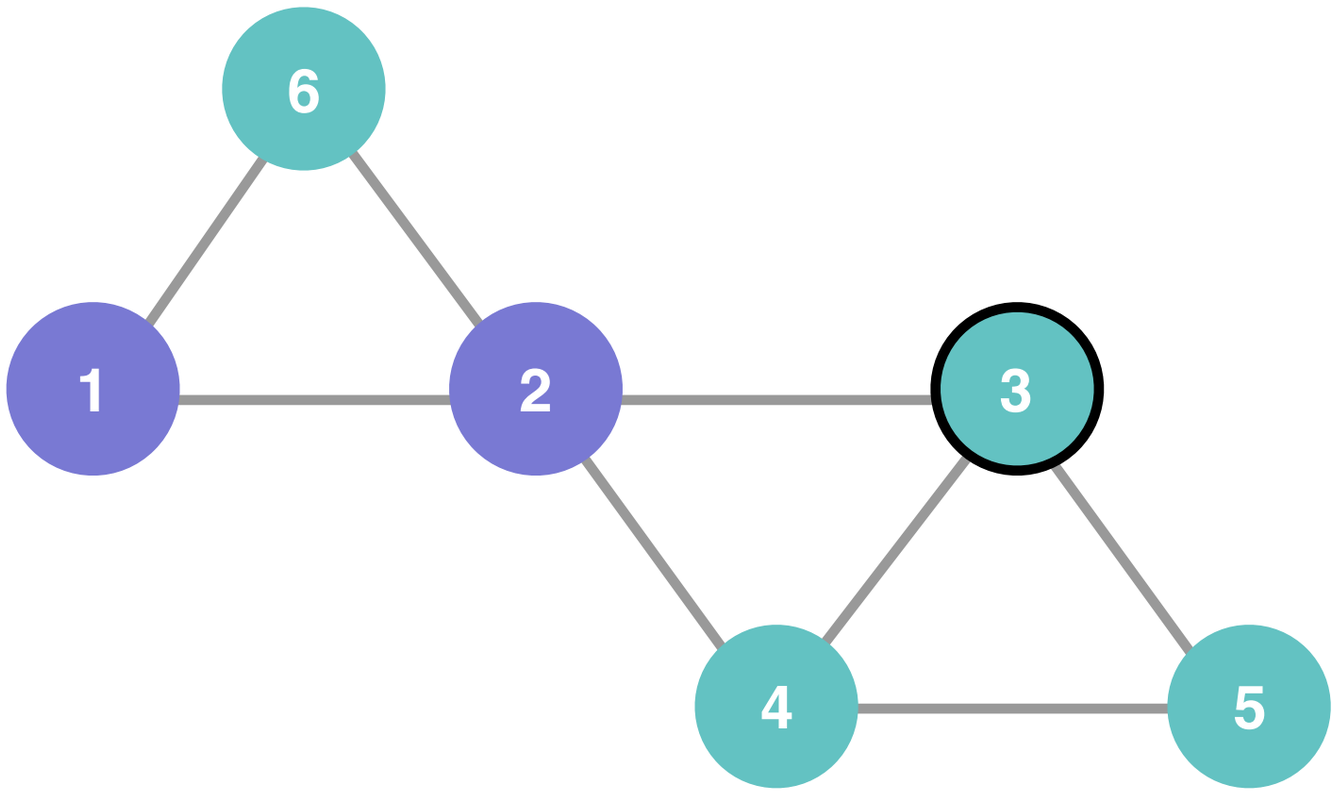
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



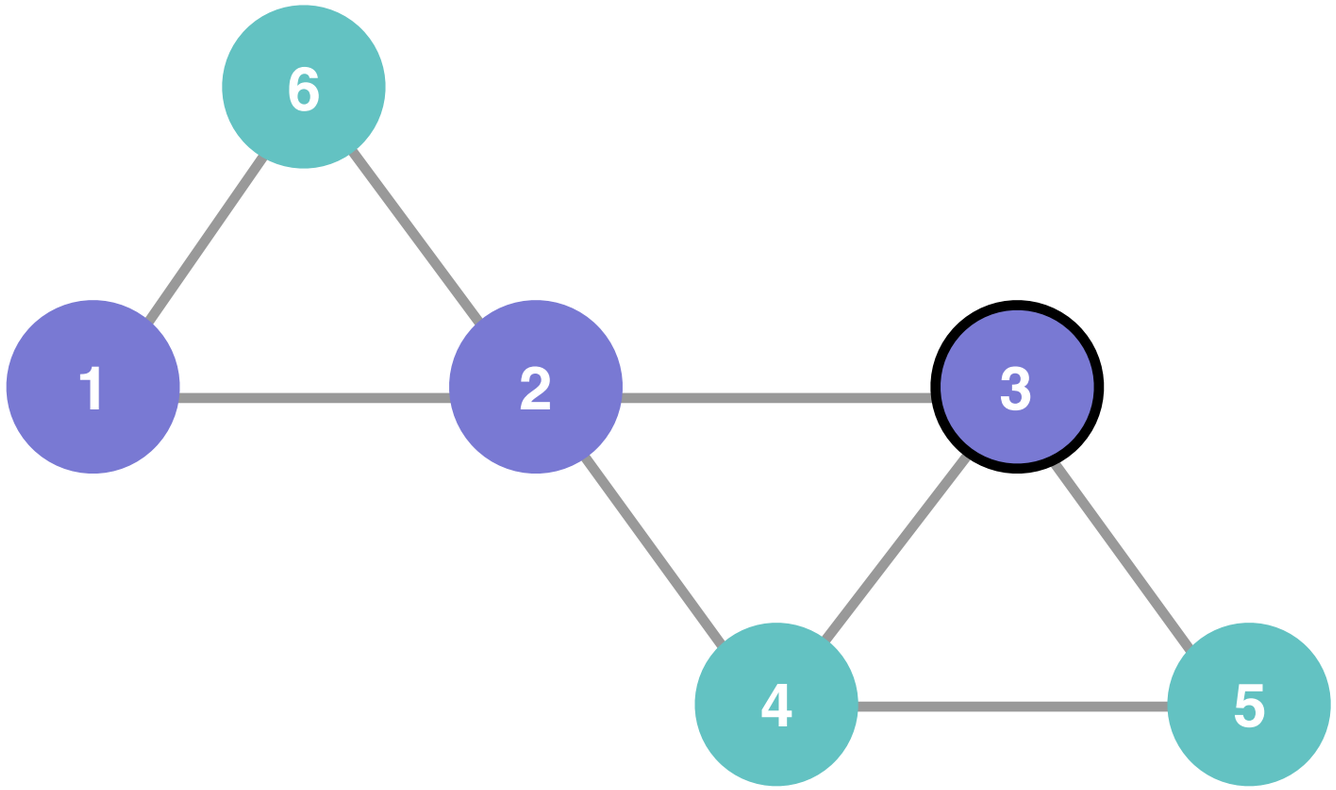
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



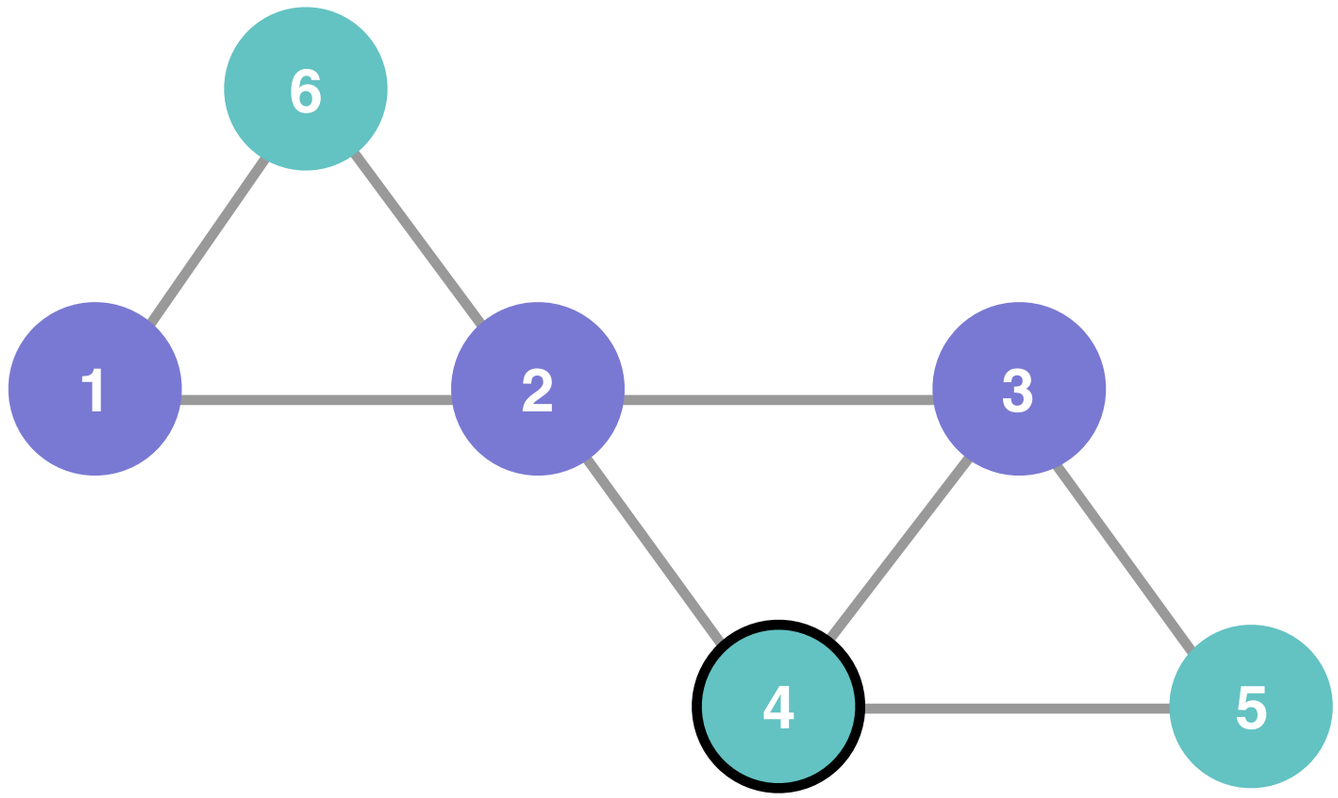
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



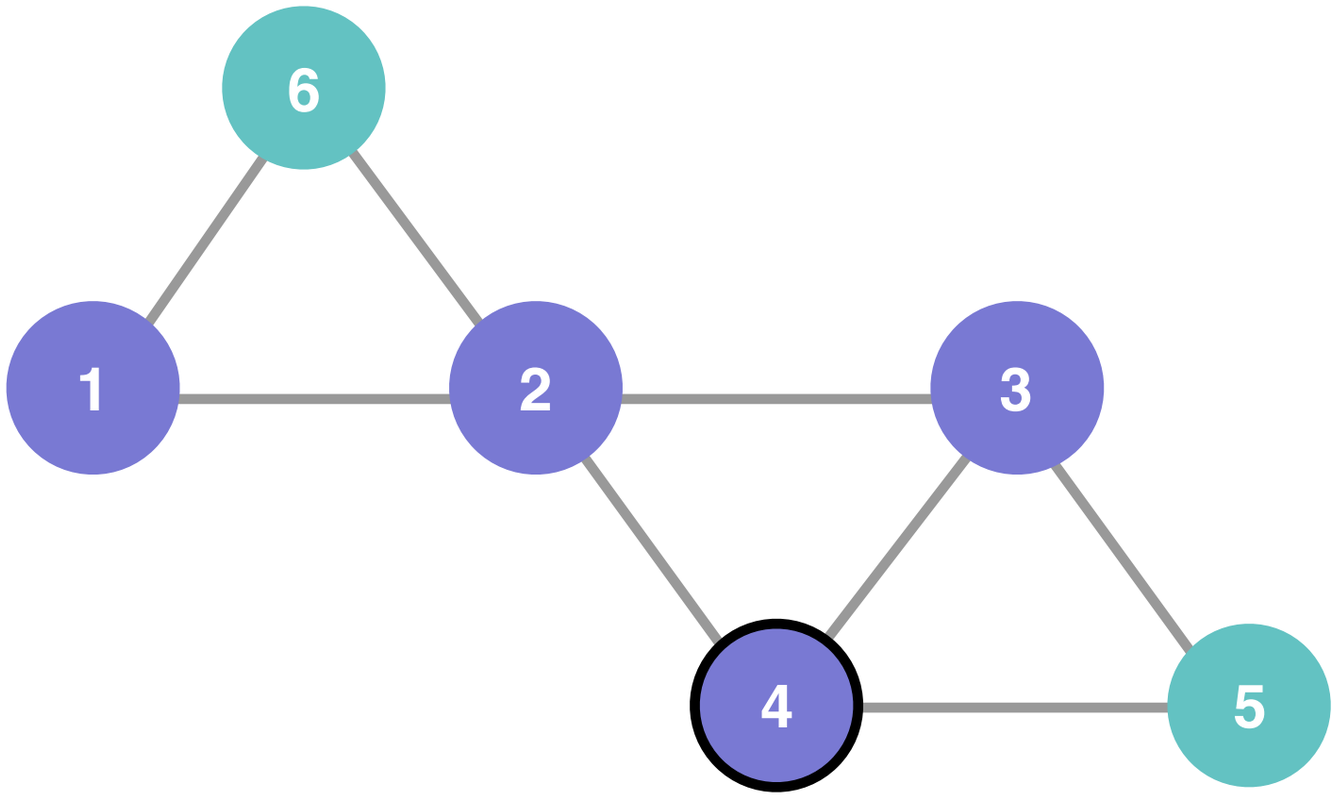
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



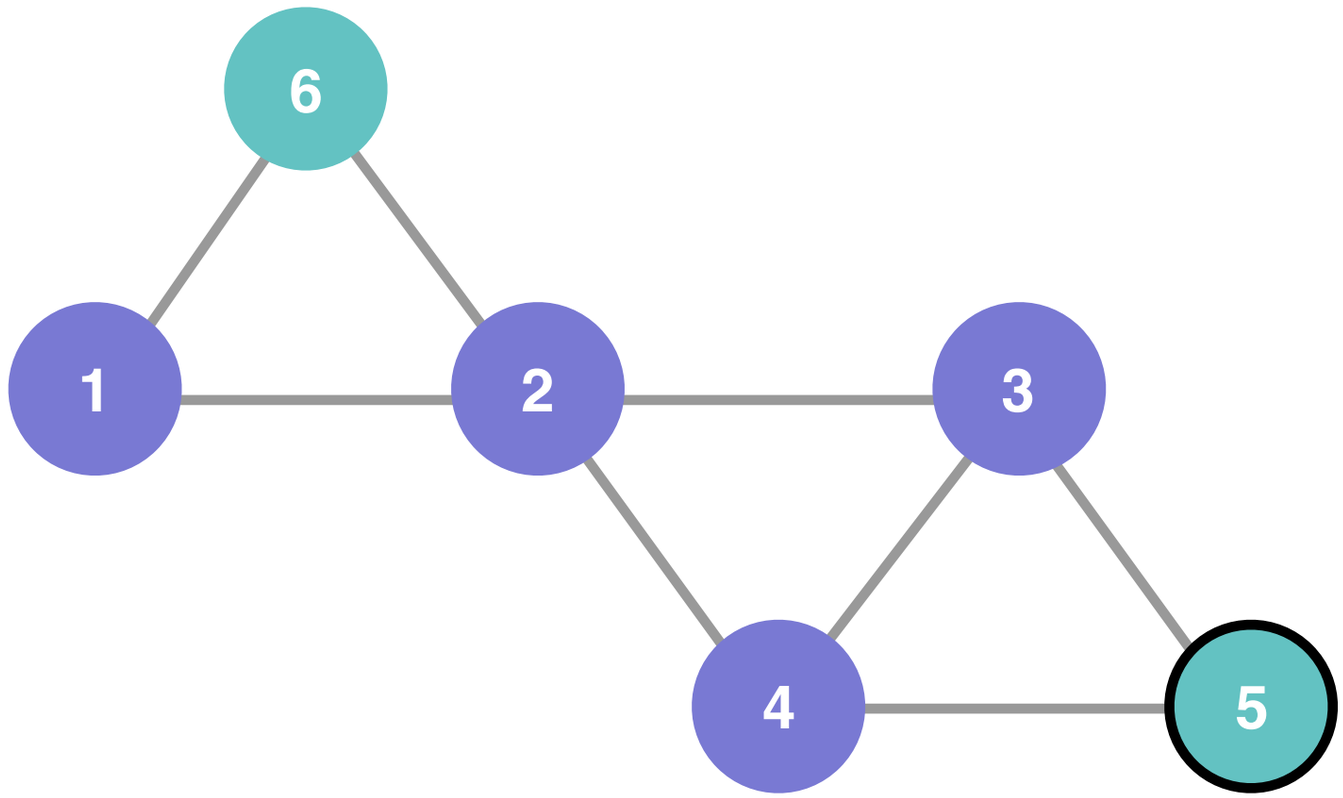
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



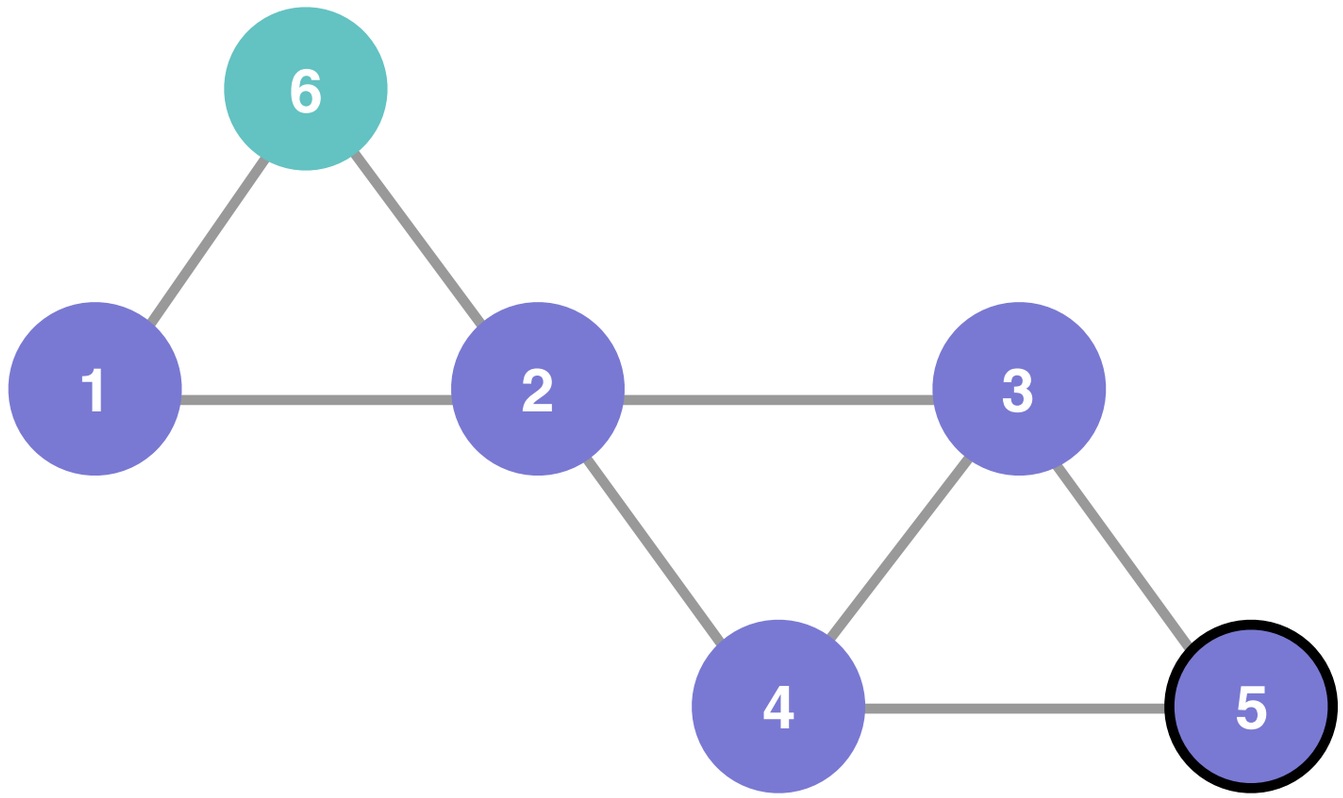
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



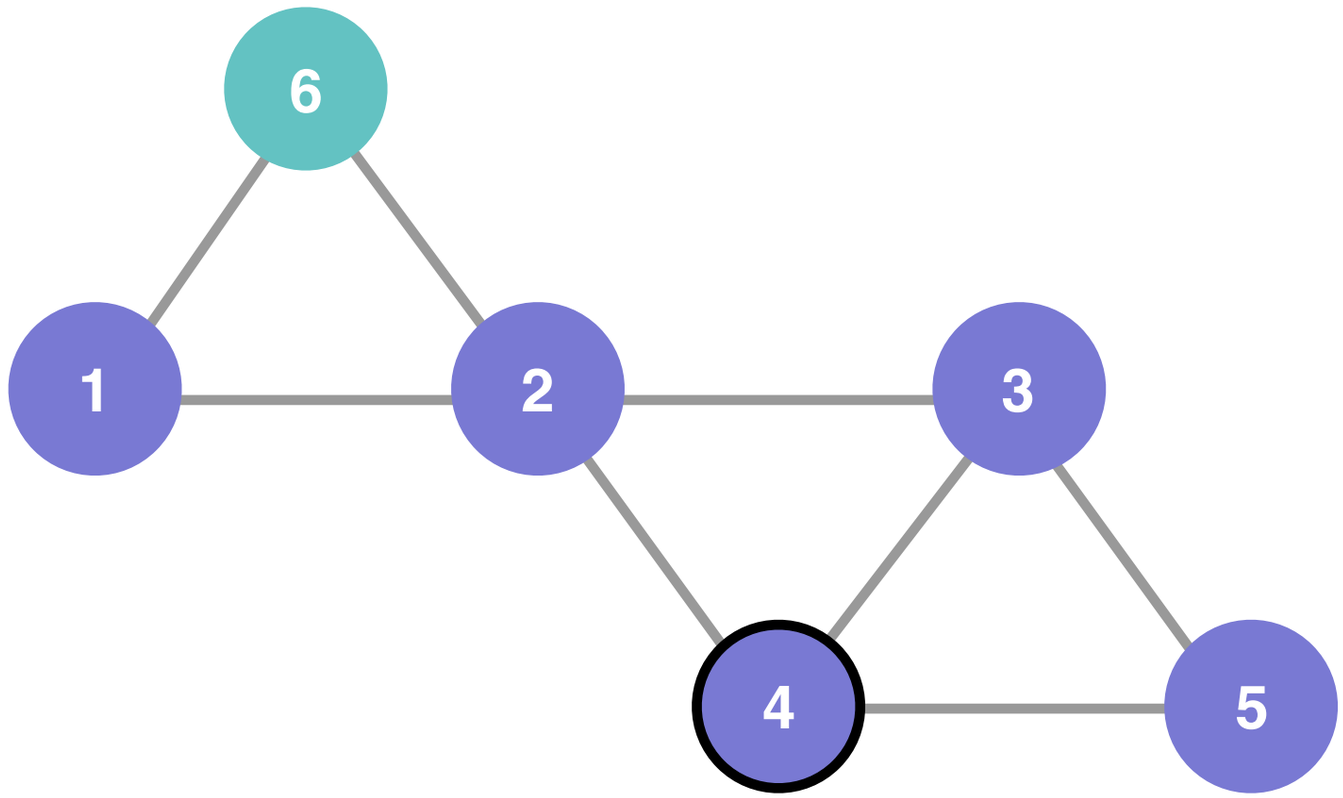
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



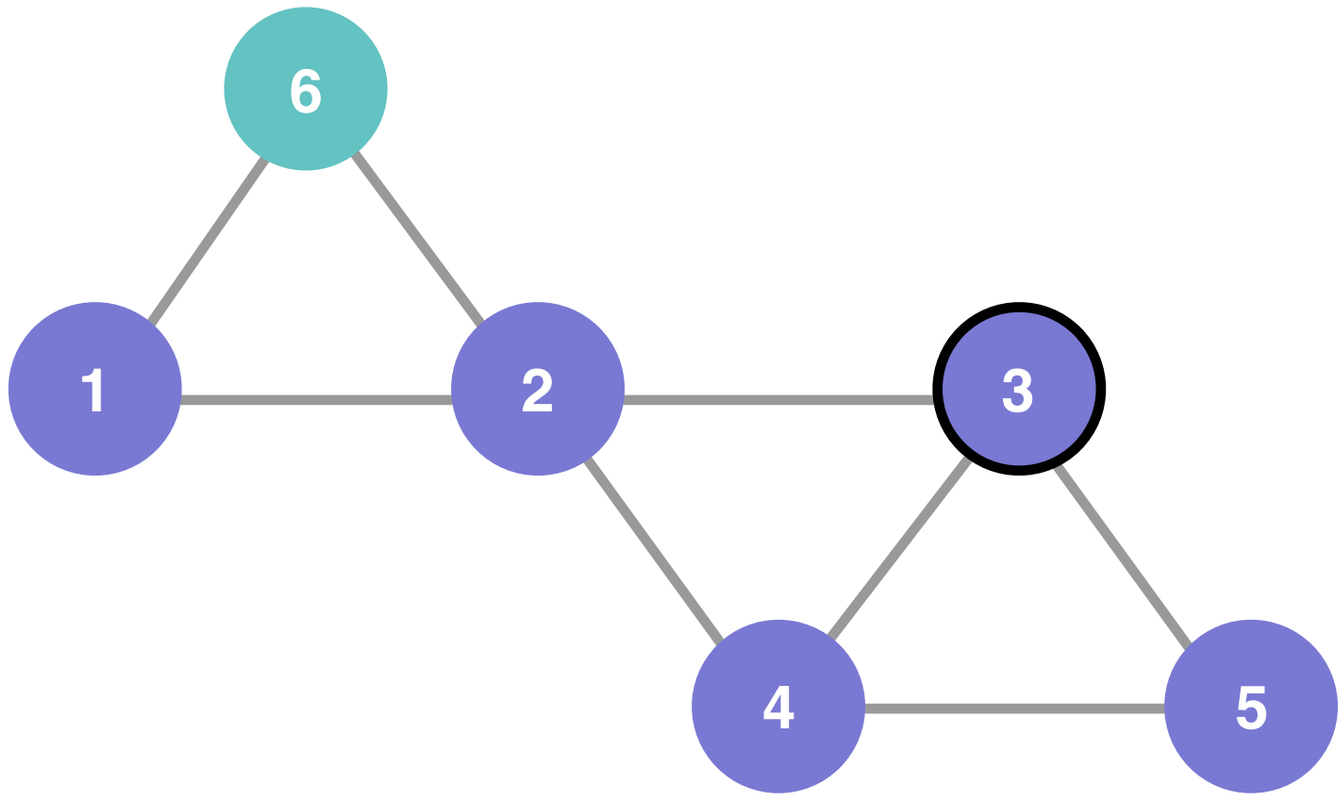
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



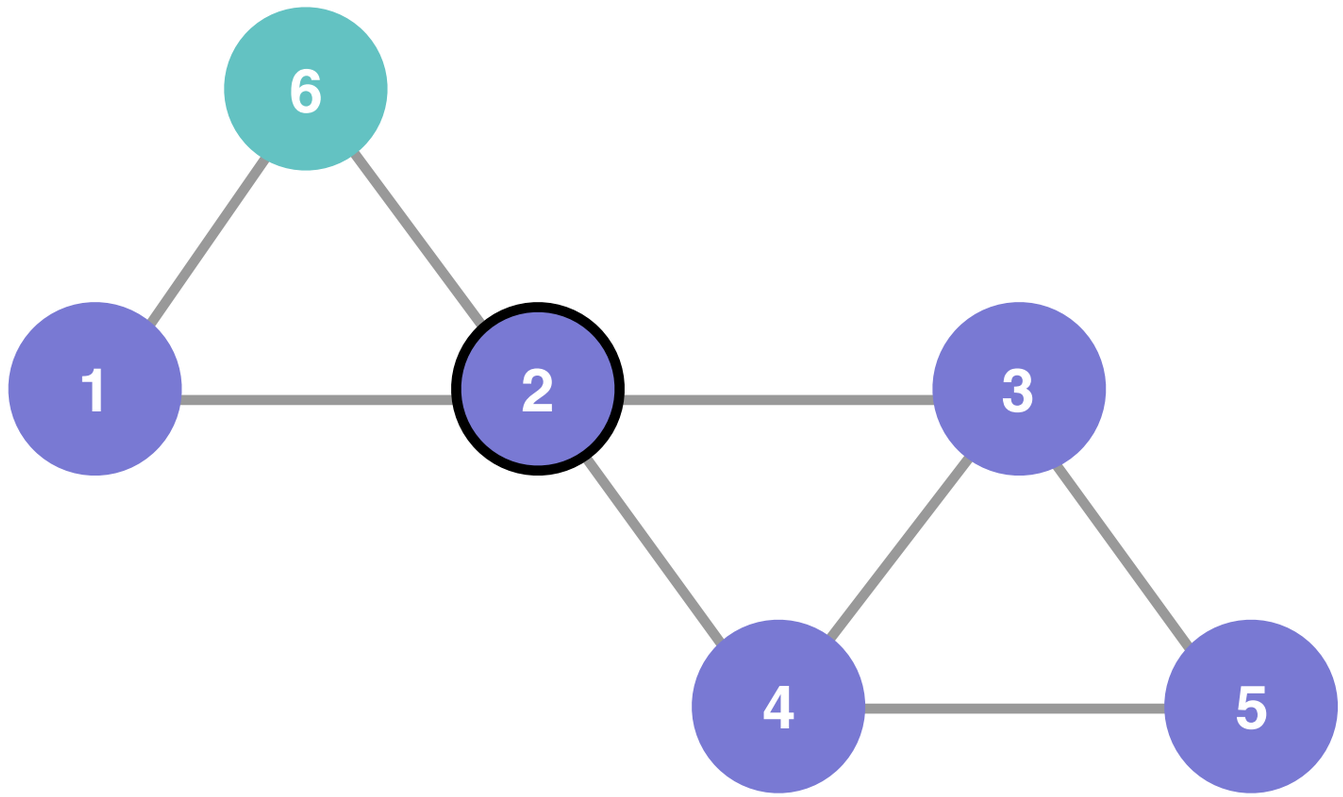
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



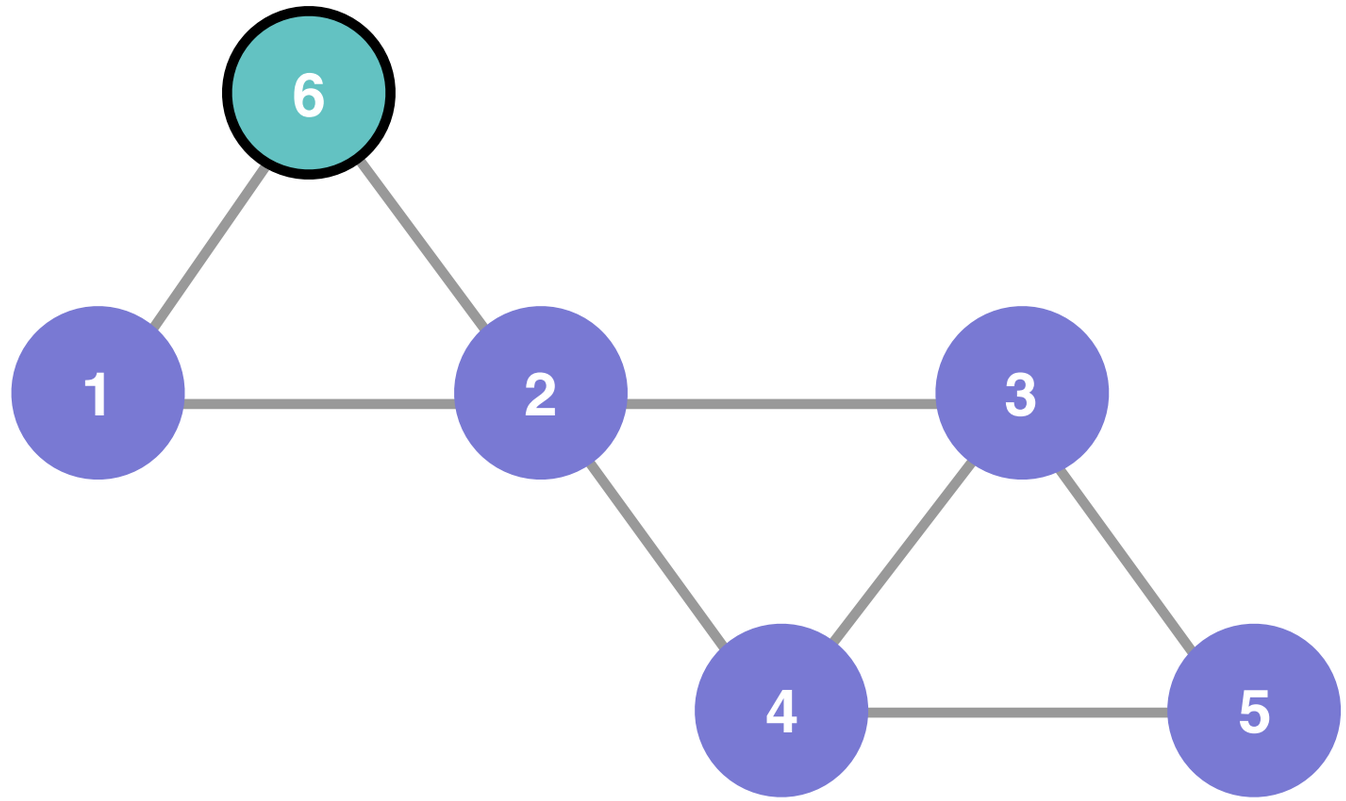
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



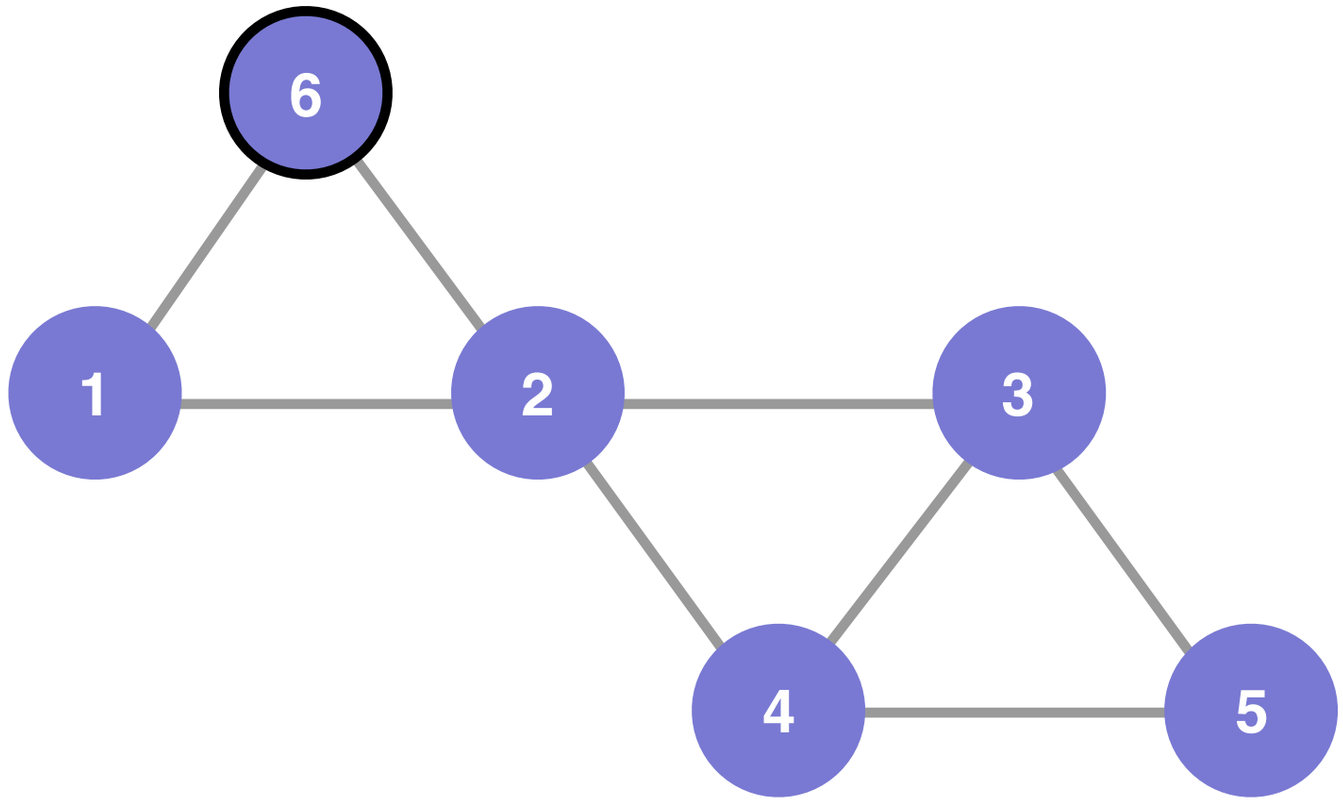
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



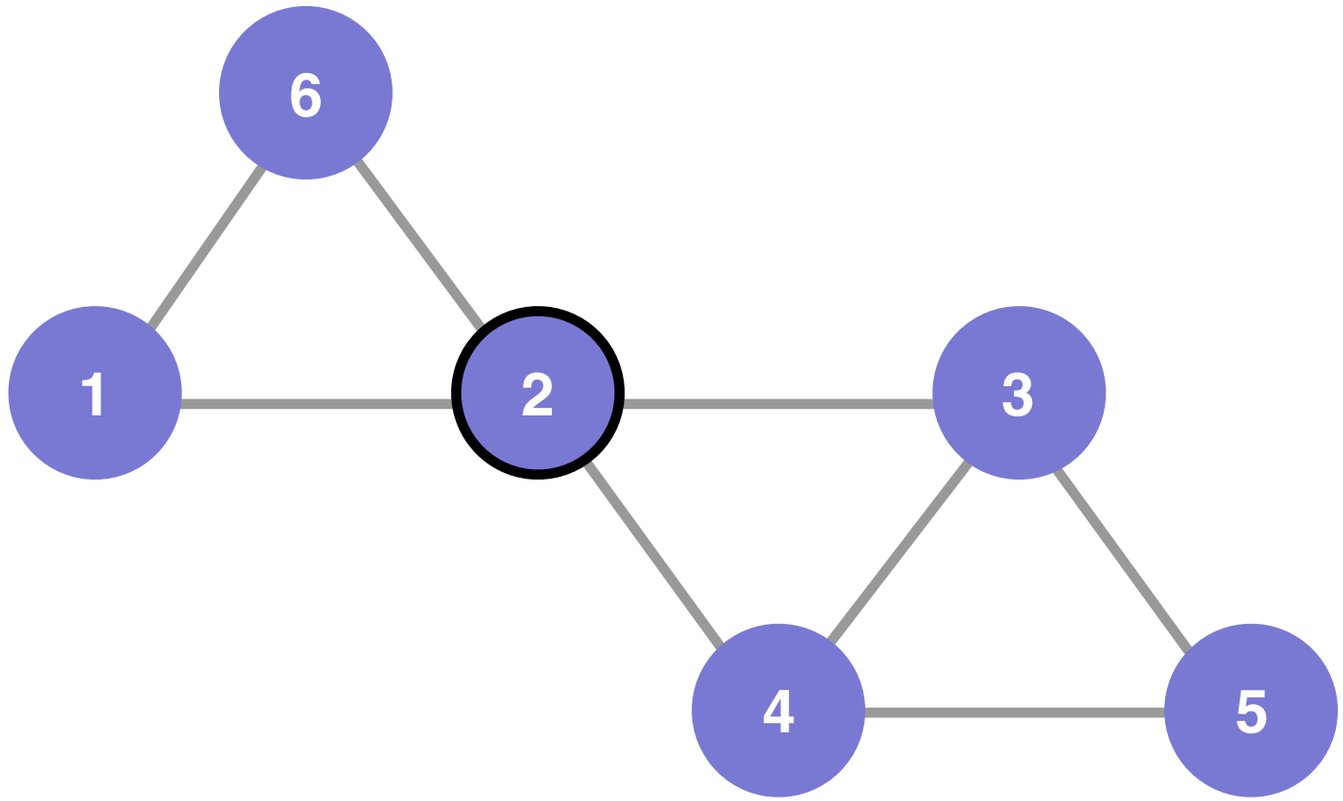
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



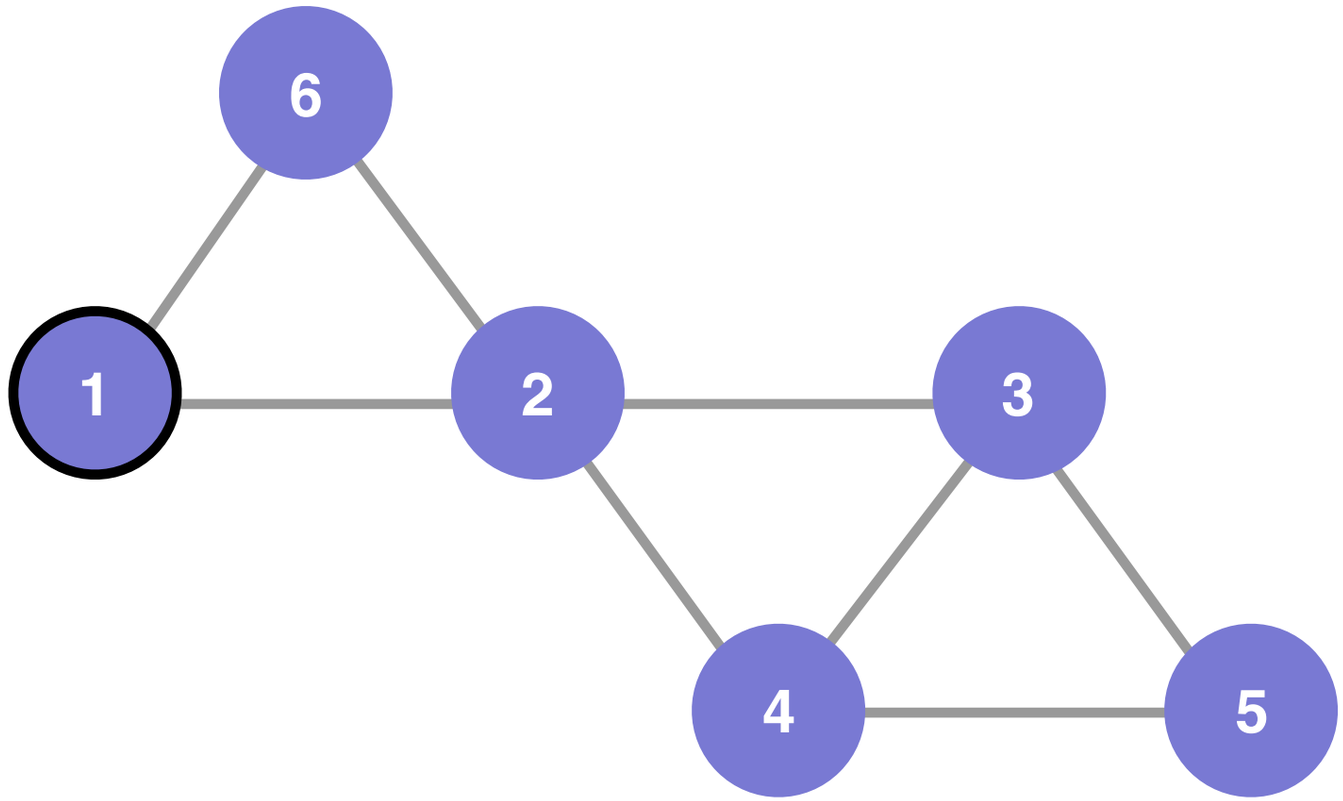
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



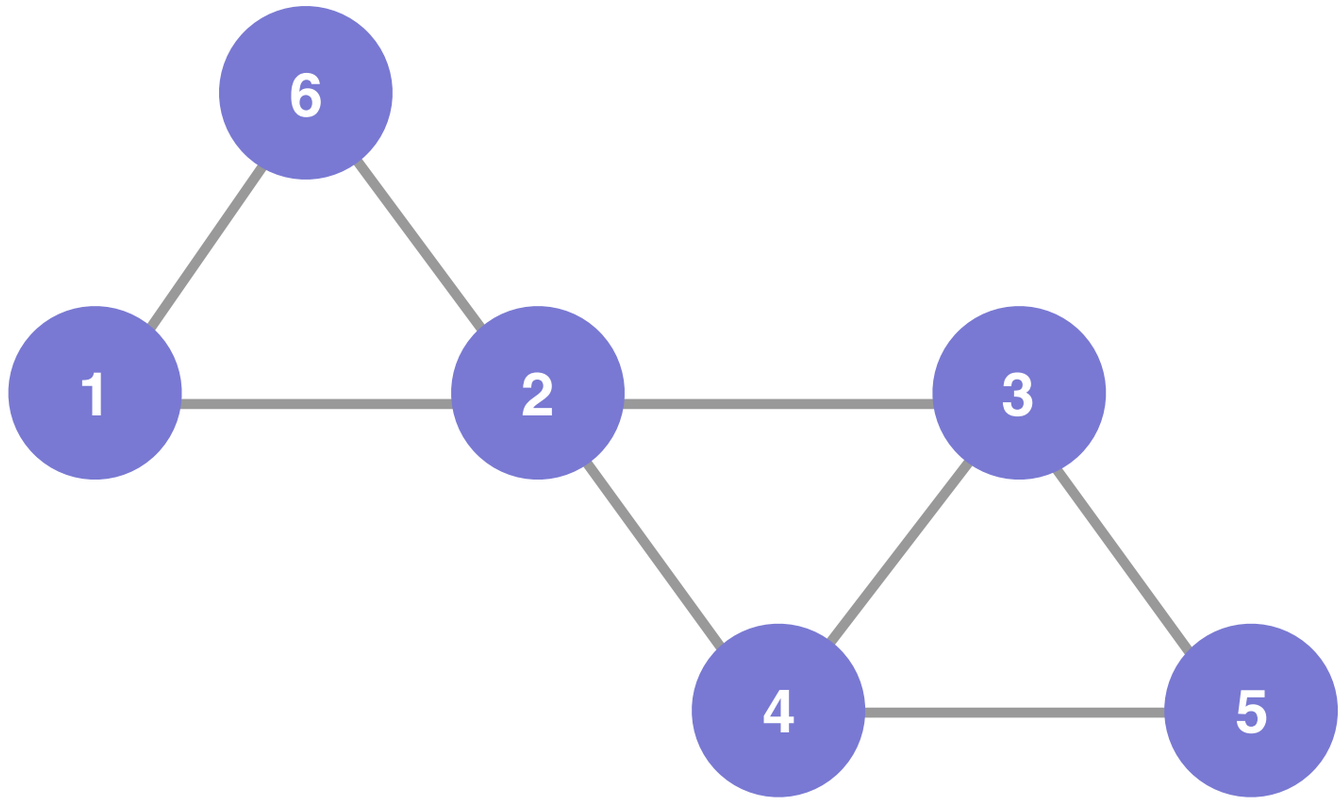
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



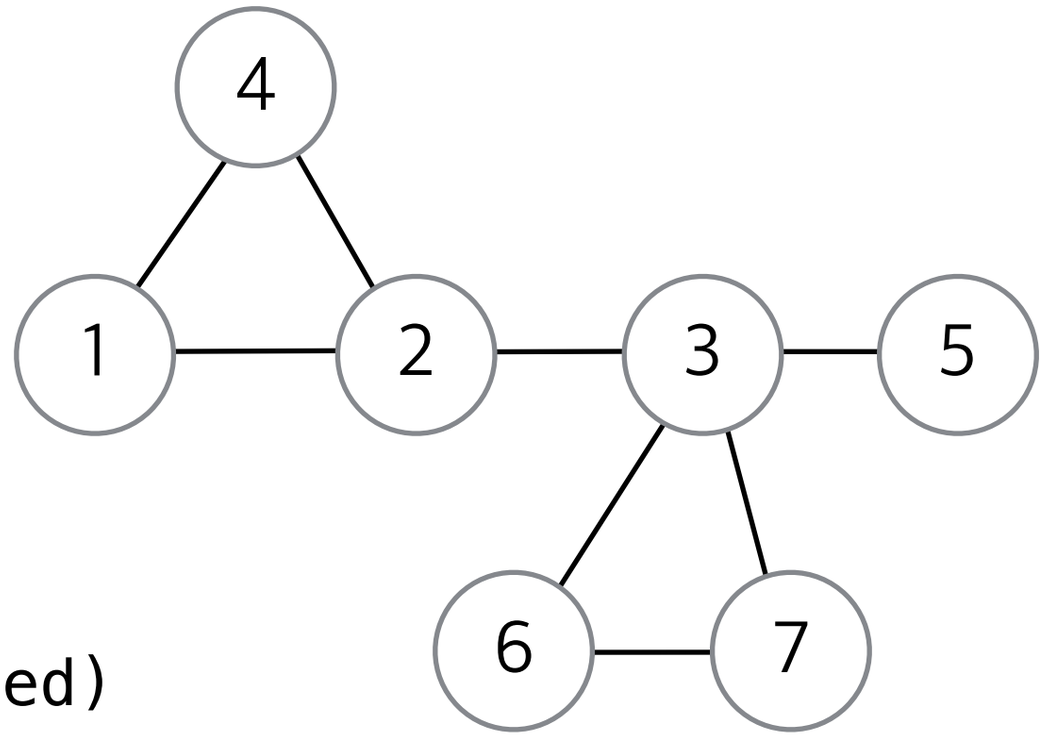
깊이우선탐색(DFS)의 예제

정점과 간선으로 이루어진 자료구조



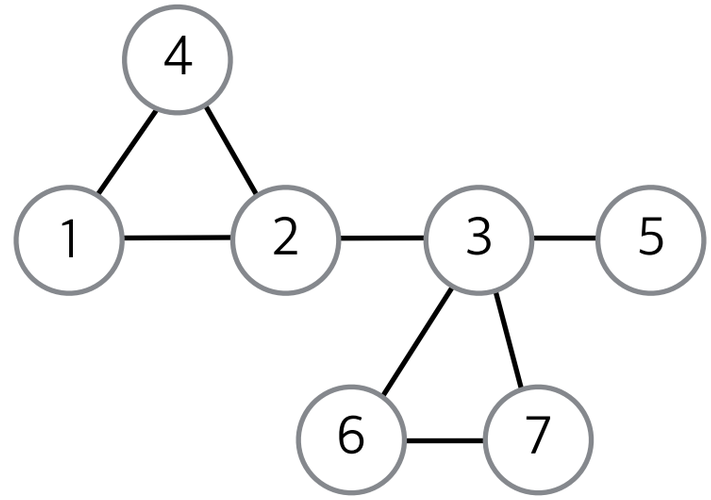
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```



깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```



visited= [F, T, F, F, F, F, F, F]

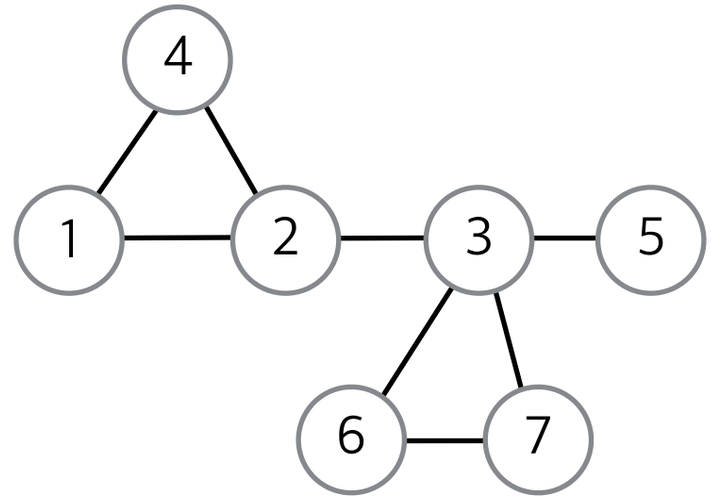
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

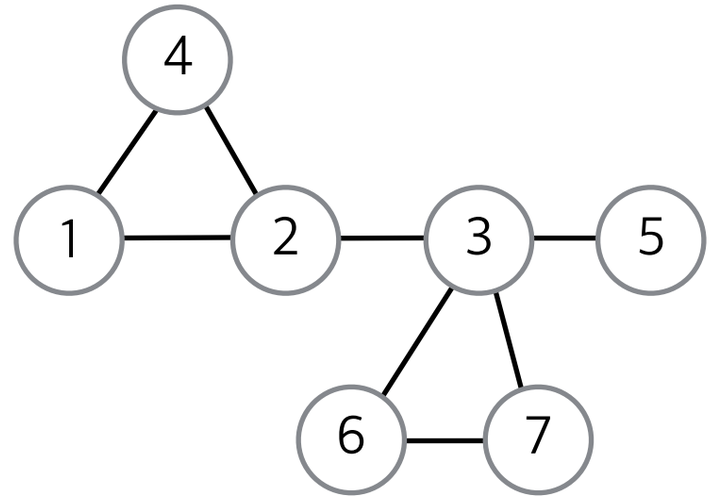
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = null



visited= [F, F, F, F, F, F, F, F]

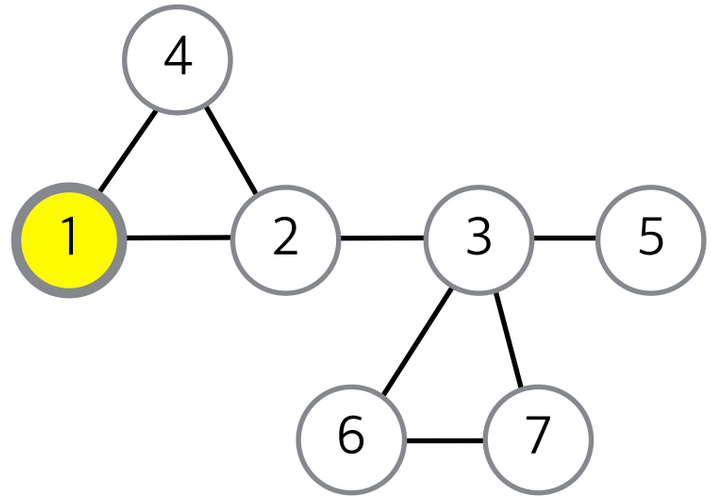
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

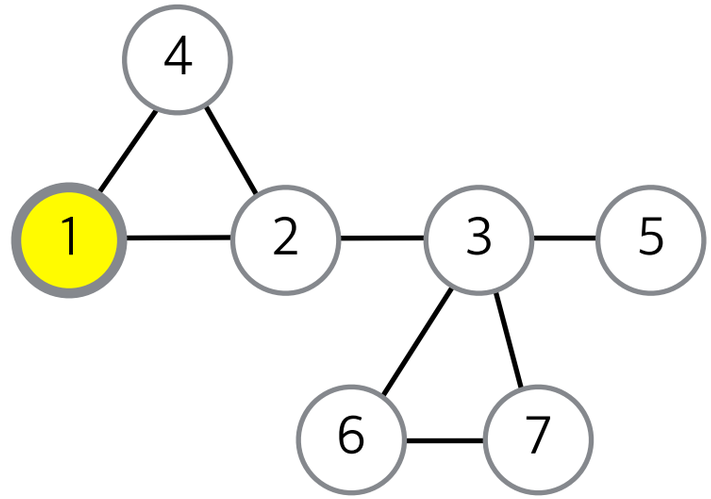
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1]



visited= [F, T, F, F, F, F, F, F]

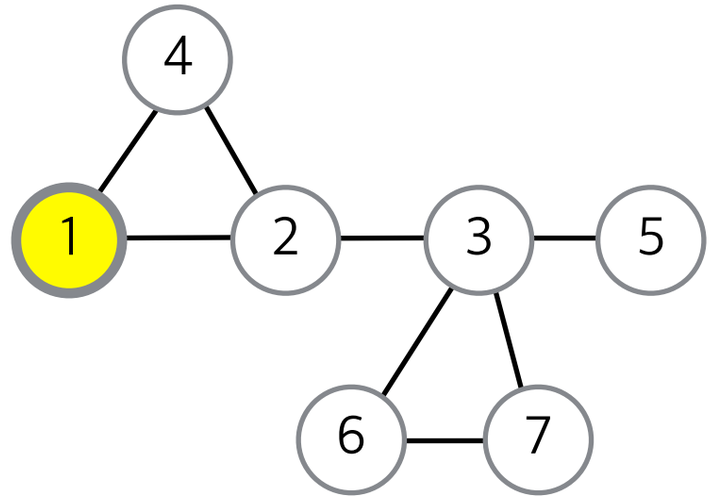
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2



visited= [F, T, F, F, F, F, F, F]

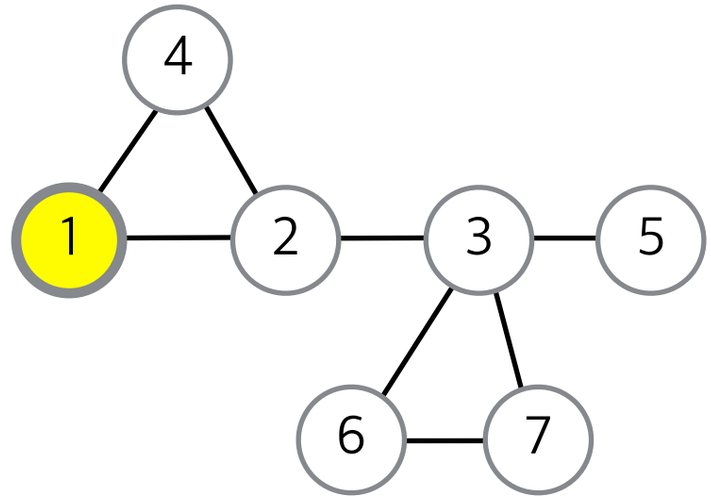
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

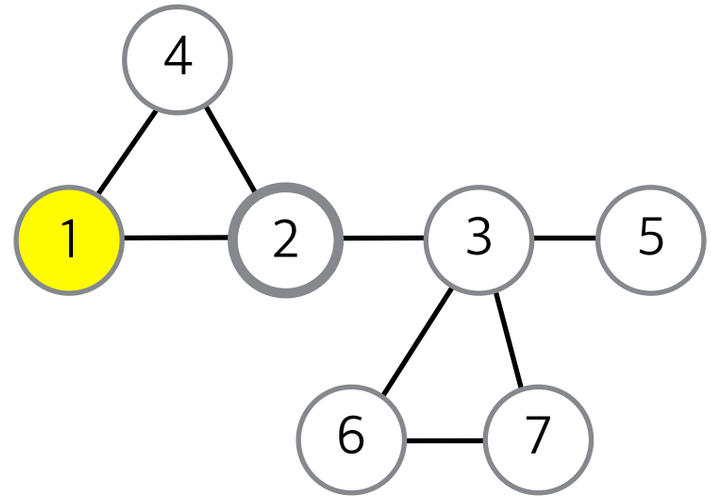
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

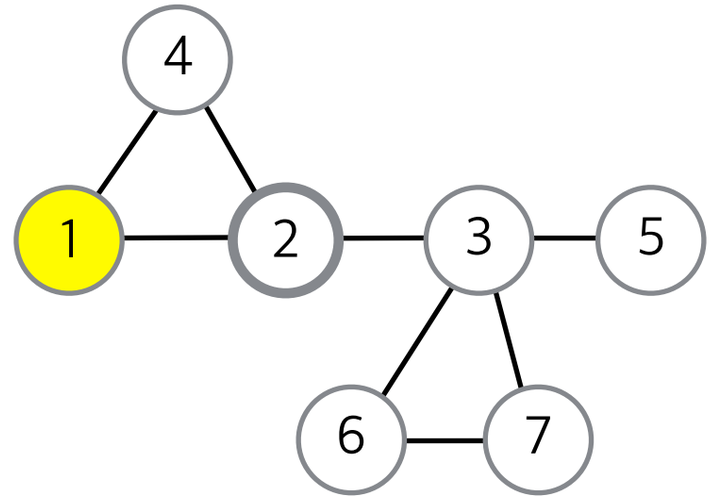
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = null



visited= [F, T, F, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

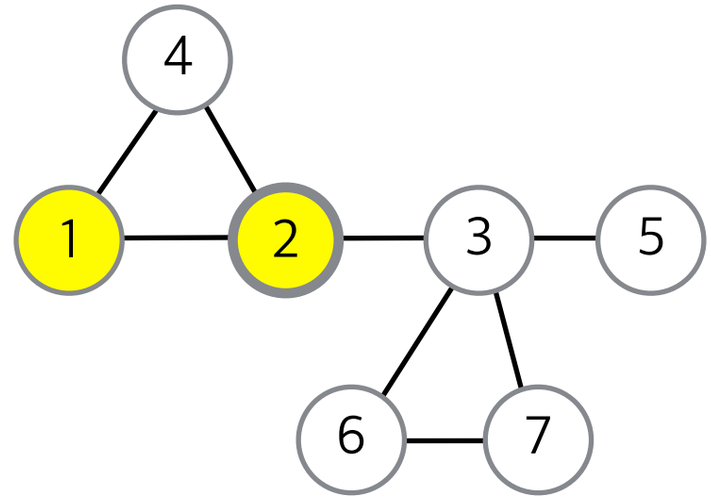
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = null



visited= [F, T, **T**, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

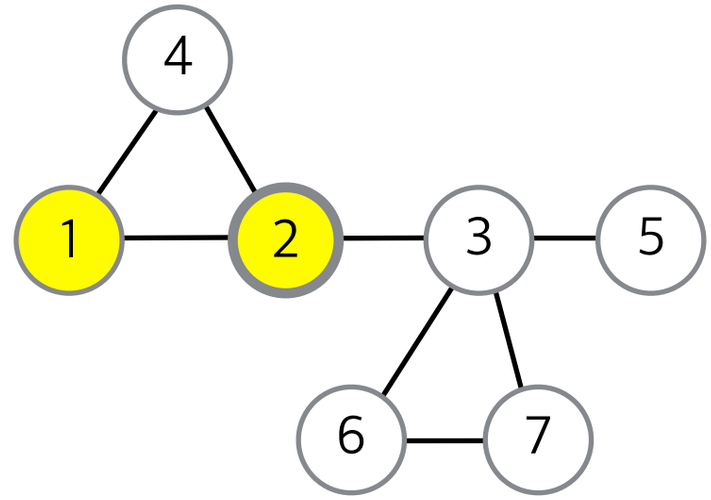
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2]



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

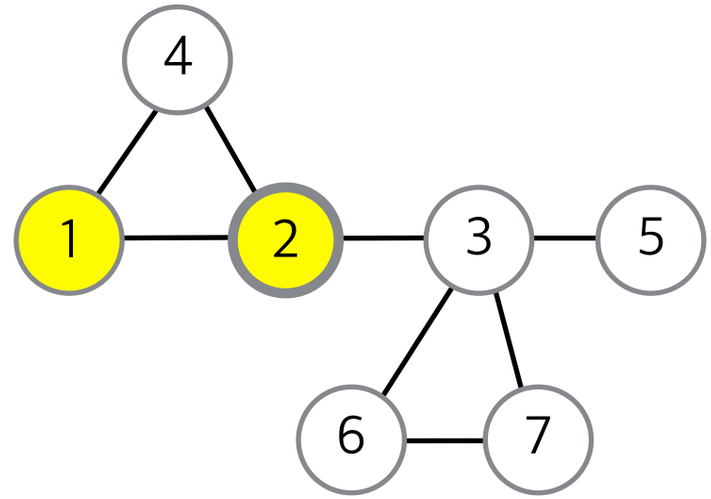
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

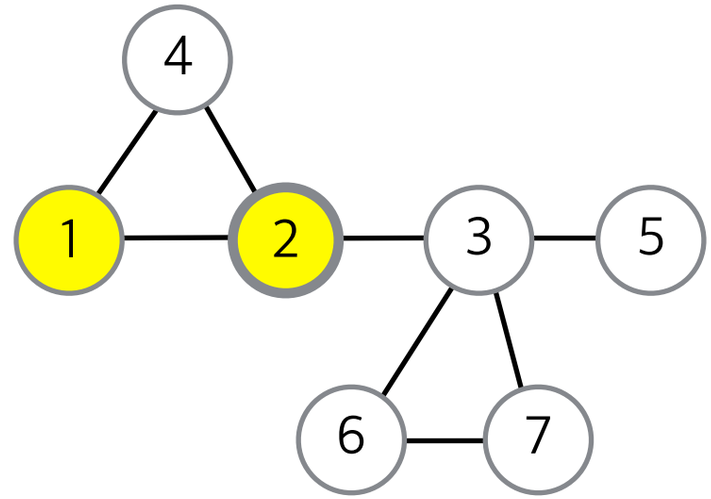
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

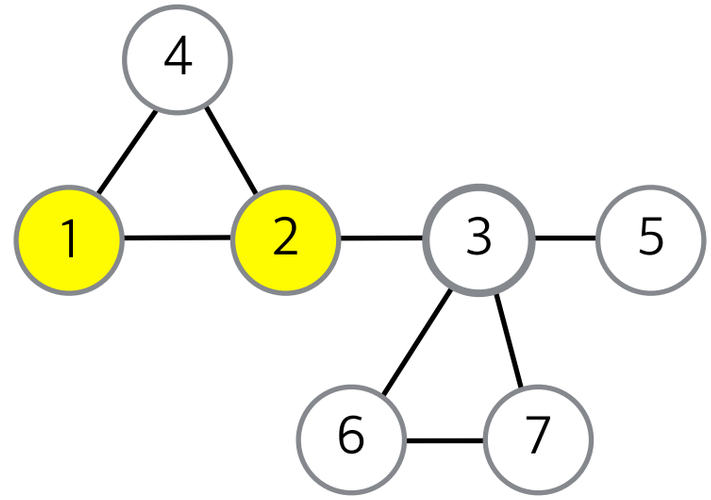
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = null



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

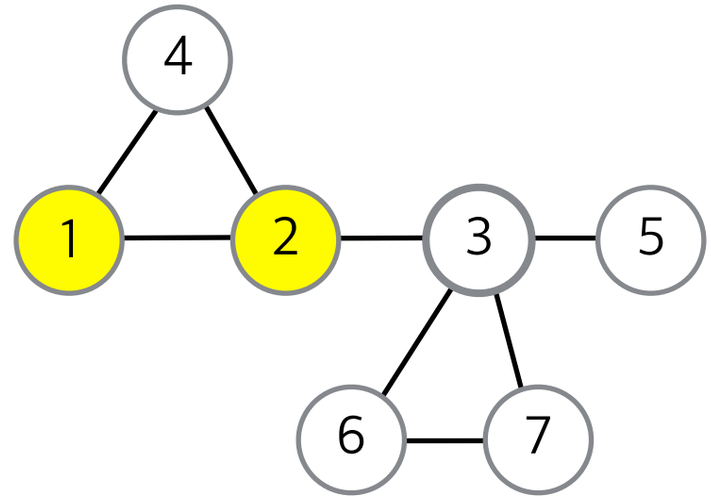
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = null



visited= [F, T, T, F, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

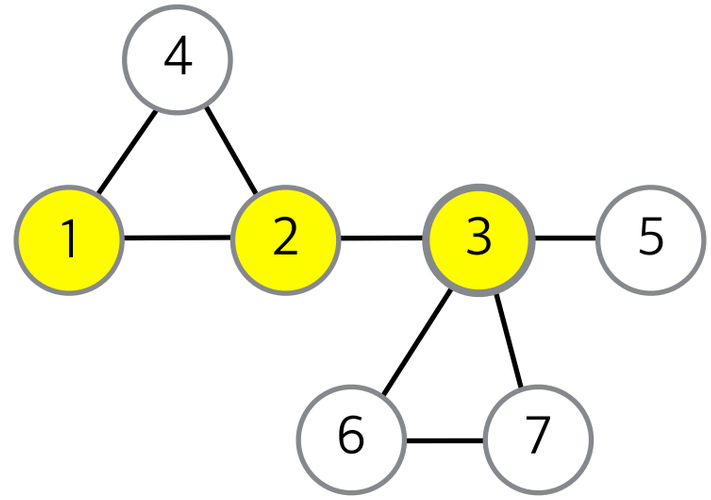
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = null



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

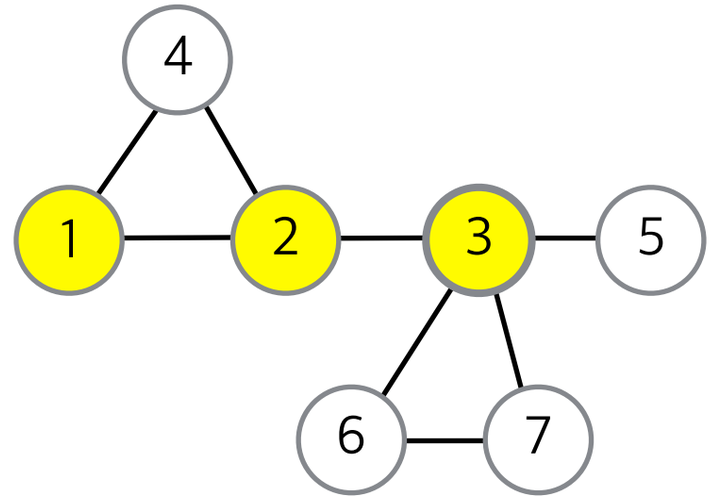
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3]



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

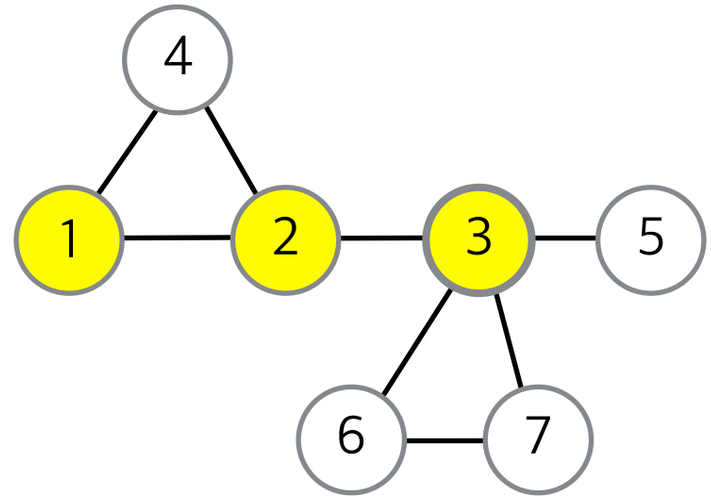
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 2



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

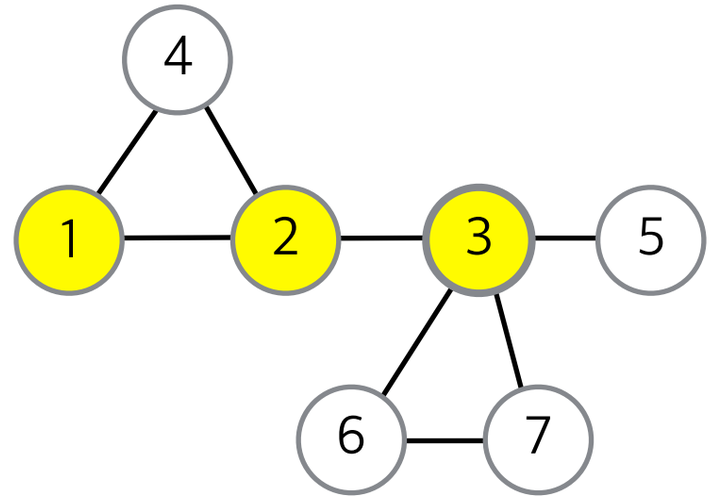
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 2



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

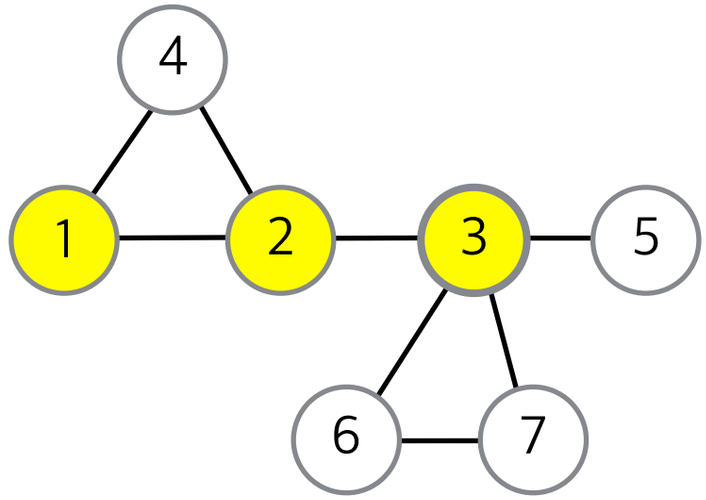
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

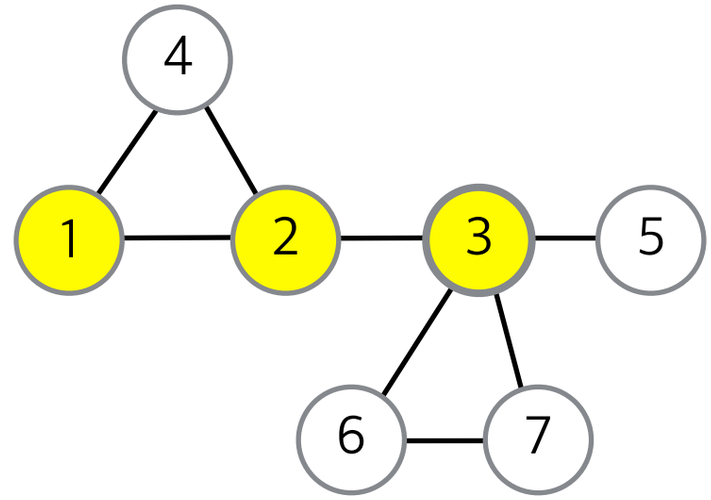
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

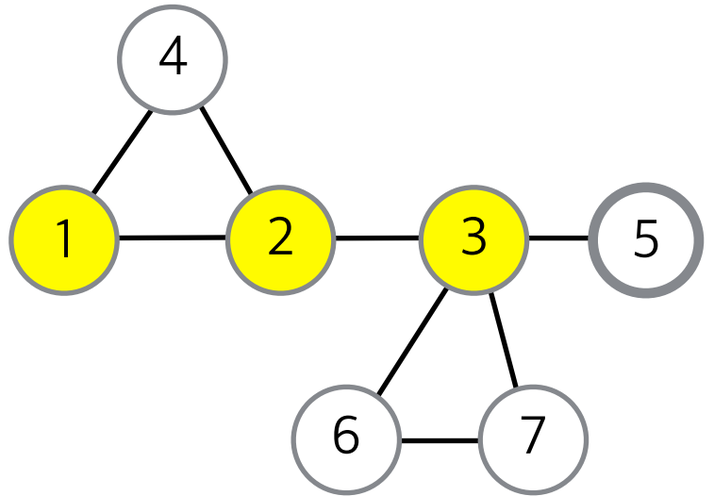
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = null



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

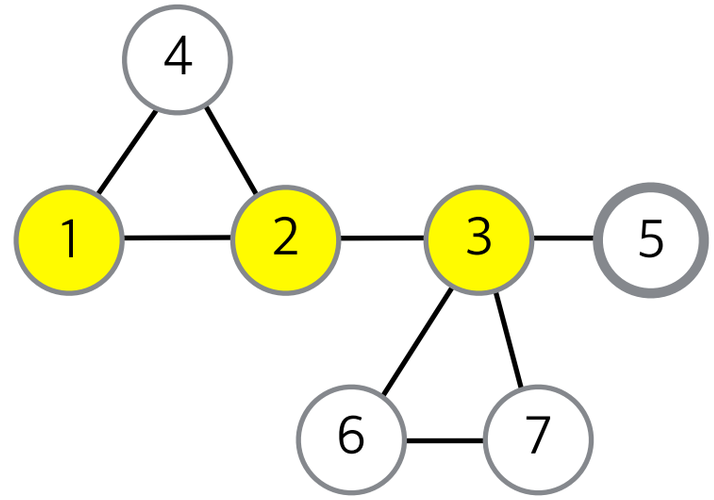
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = null



visited= [F, T, T, T, F, F, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

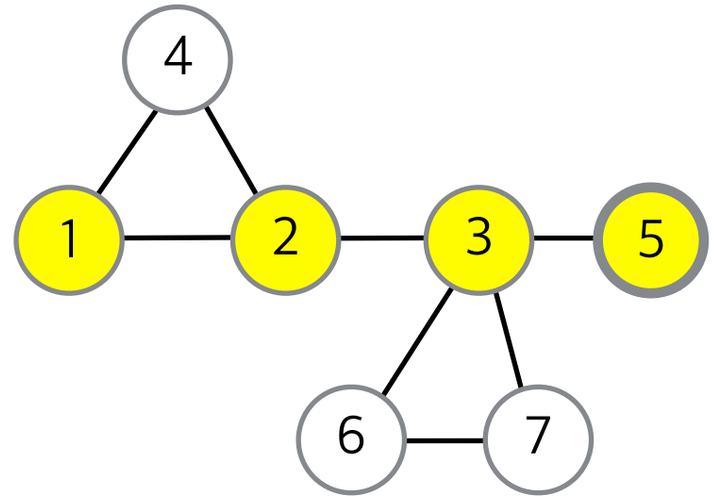
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = null



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

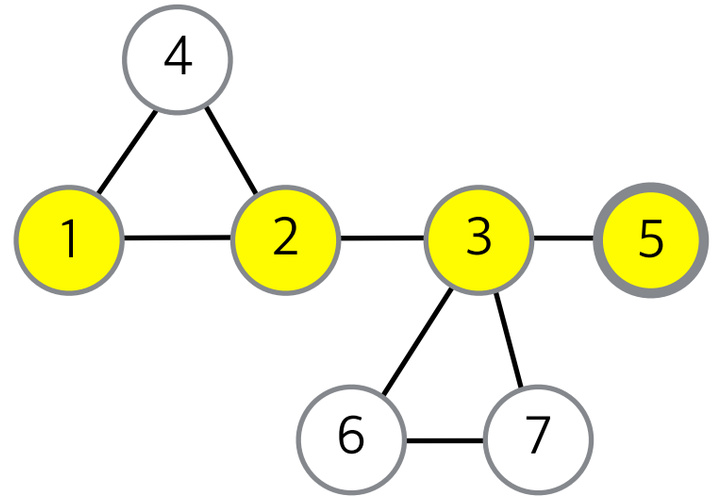
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

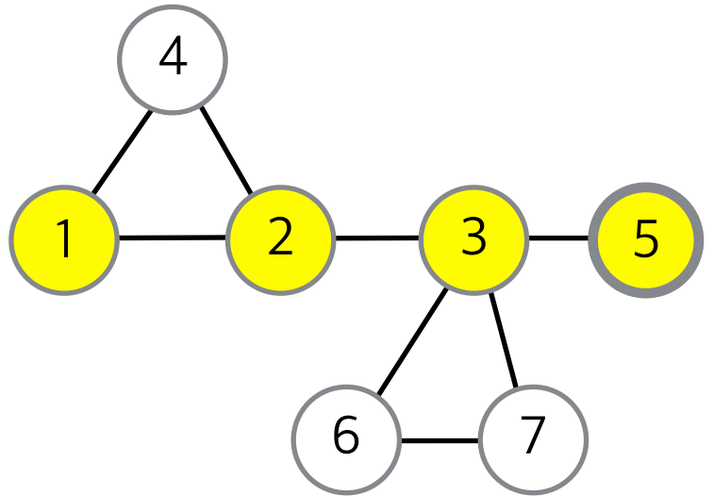
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

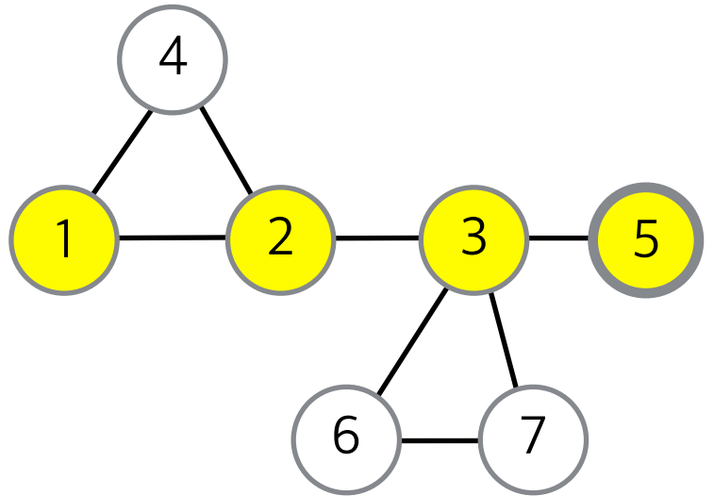
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

→   return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

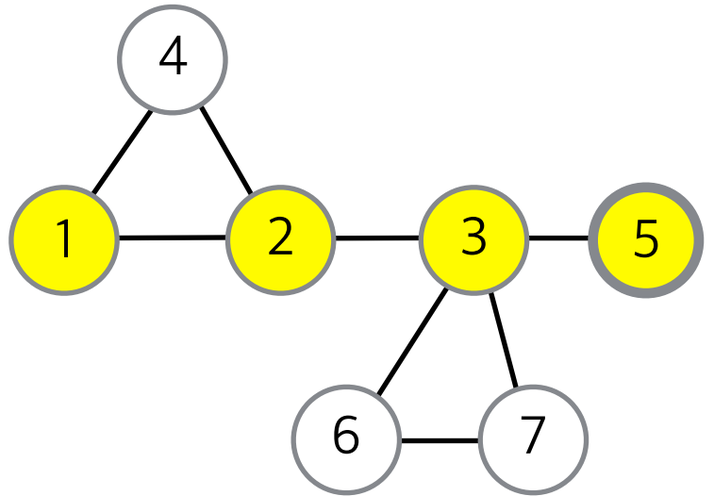
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

→   return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

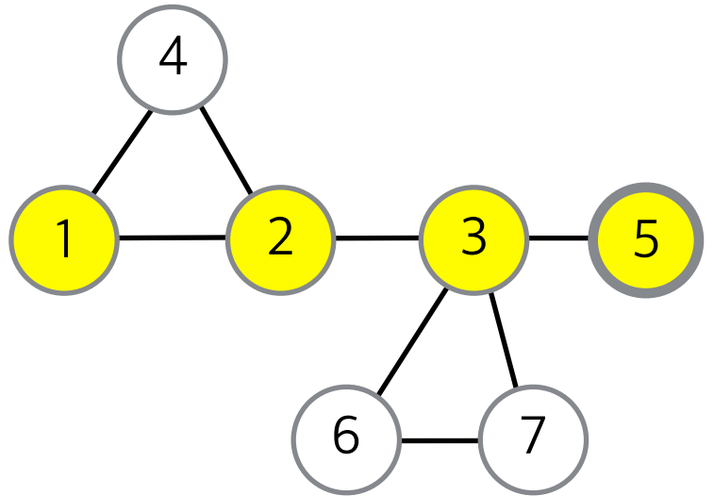
result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

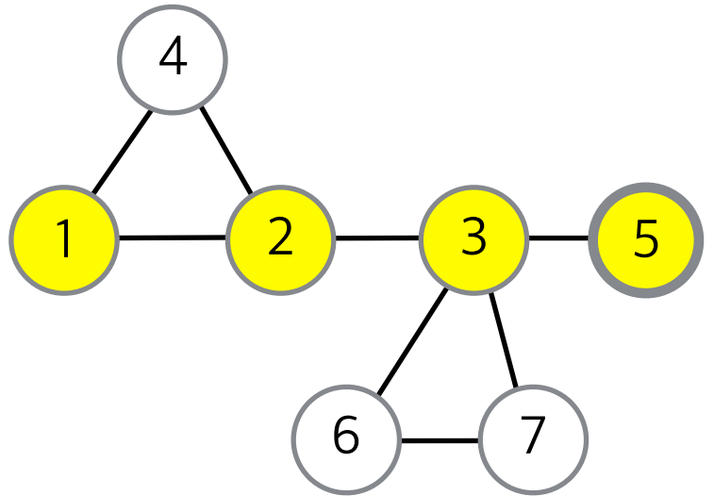
DFS(graph, 3, visited)

result = [3], v = 5

DFS(graph, 5, visited)

result = [5], v = 3

[5]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

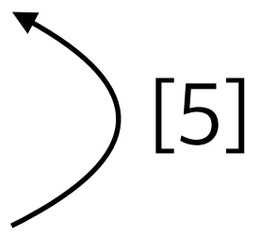
result = [1], v = 2

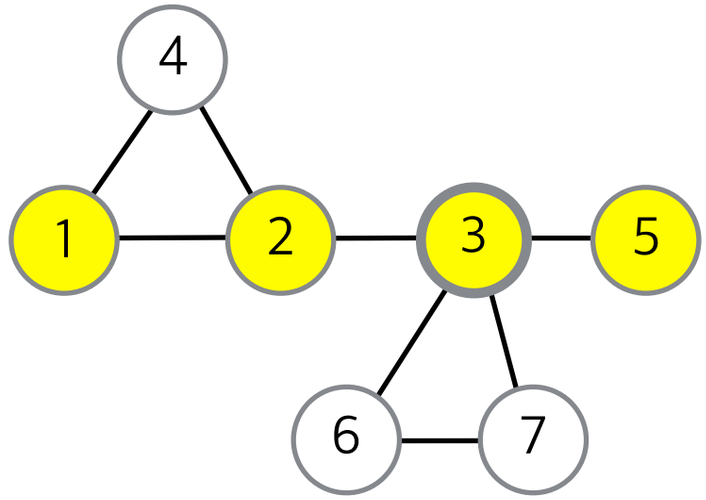
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3], v = 5





visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

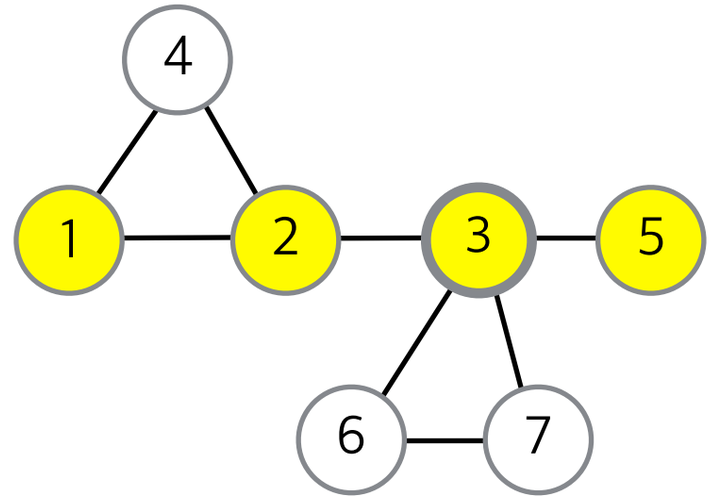
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 5



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

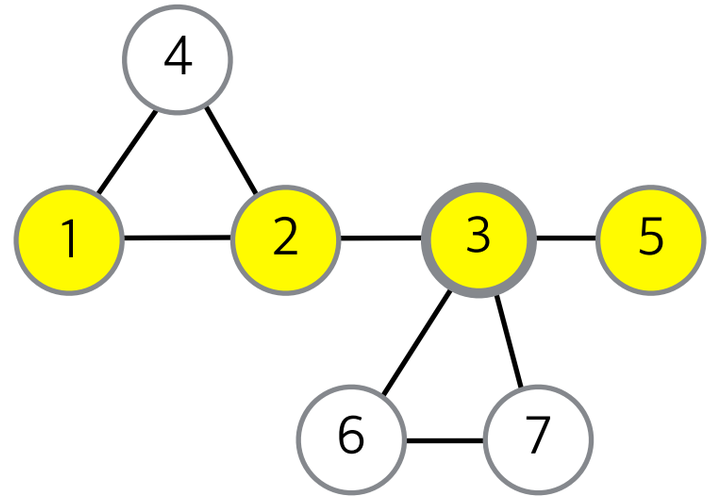
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

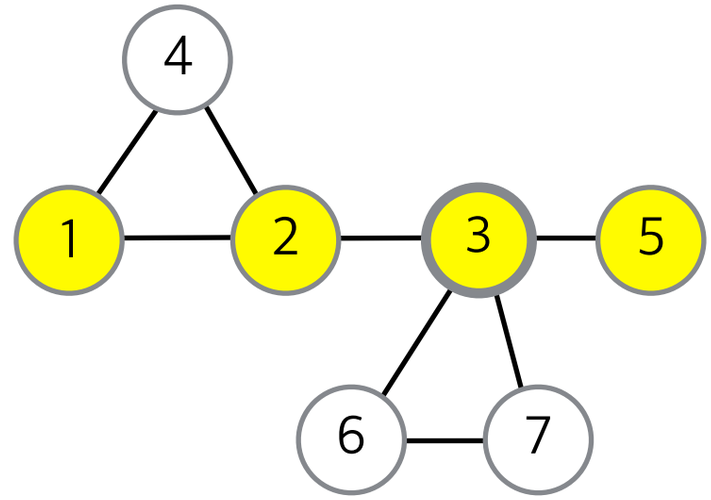
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

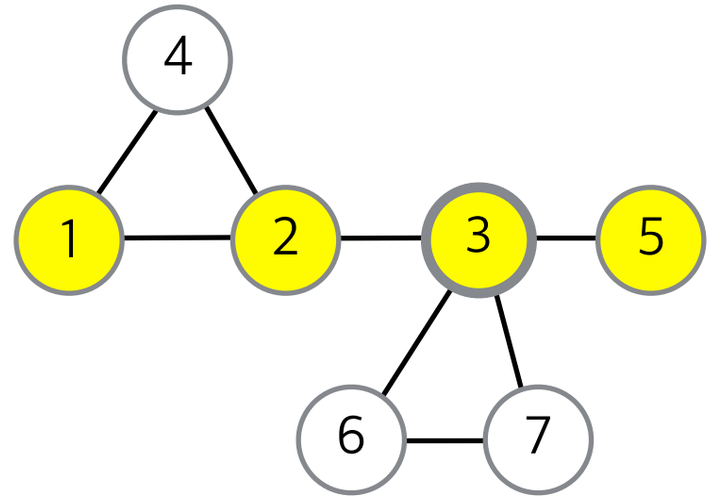
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

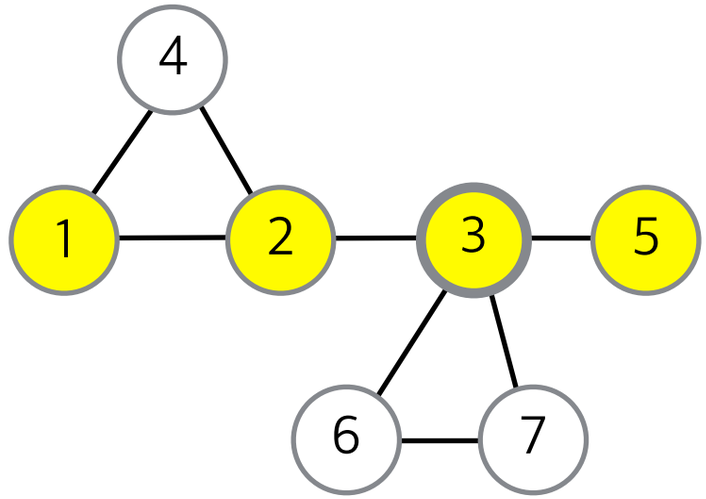
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6

DFS(graph, 6, visited)가 return하는 값은 ?



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

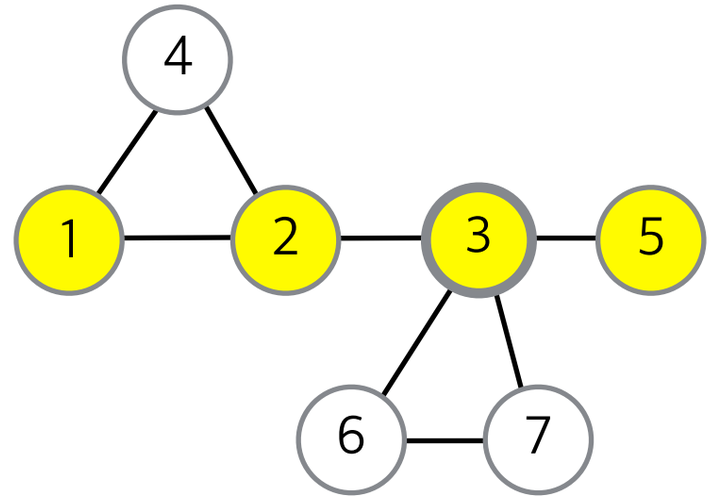
result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5], v = 6

DFS(graph, 6, visited)

[6, 7]



visited= [F, T, T, T, F, T, F, F]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

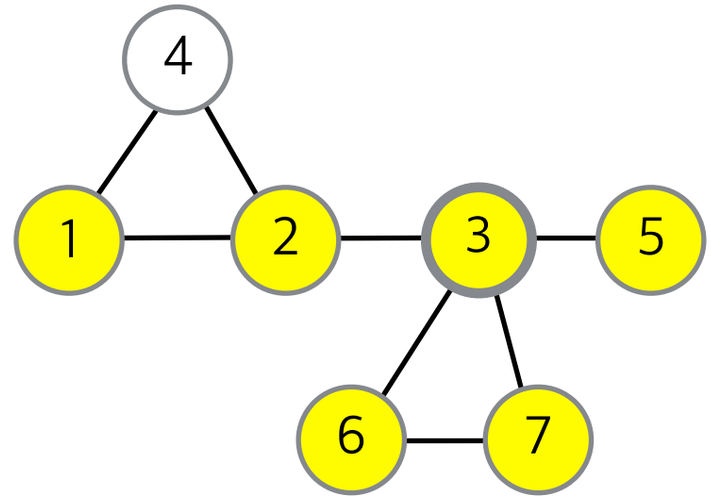
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 6



visited= [F, T, T, T, F, T, **T**, **T**]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

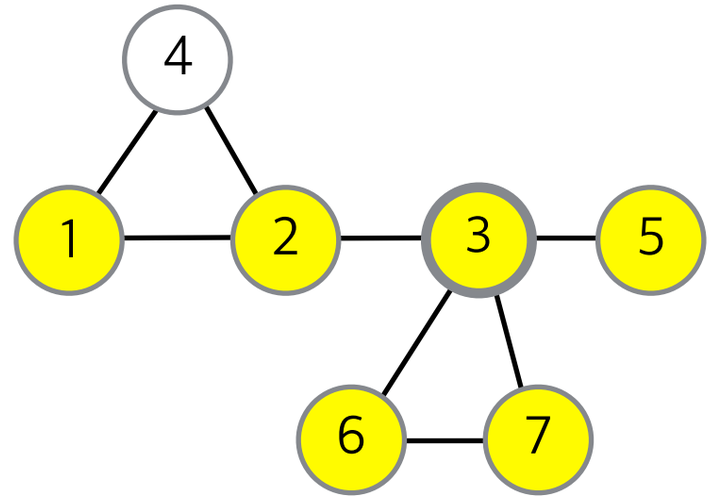
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

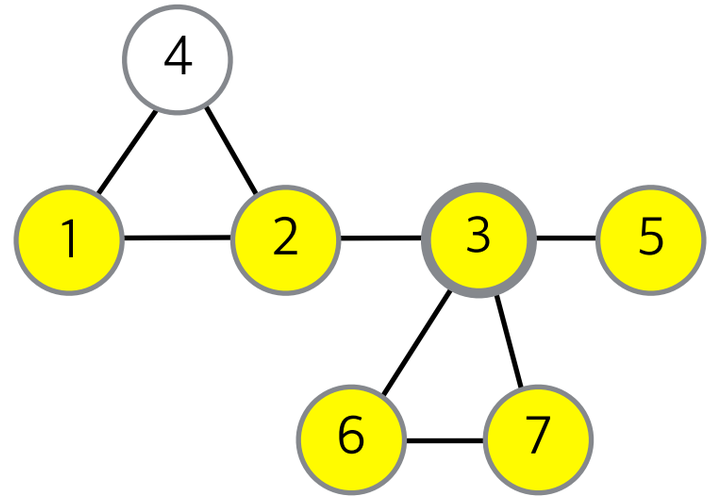
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

→   return result
```

DFS(graph, 1, visited)

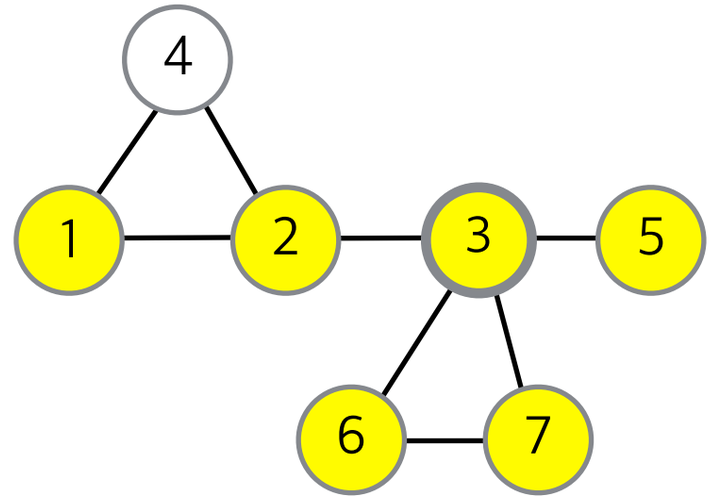
result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

→   return result
```

DFS(graph, 1, visited)

result = [1], v = 2

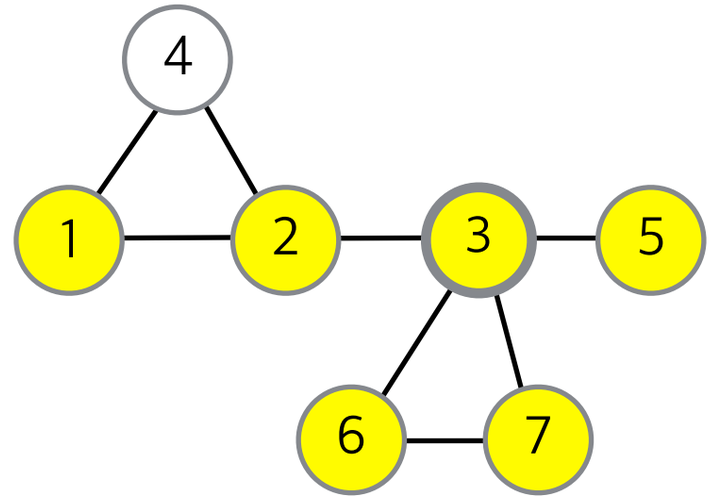
DFS(graph, 2, visited)

result = [2], v = 3

DFS(graph, 3, visited)

result = [3, 5, 6, 7], v = 7

[3, 5, 6, 7]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

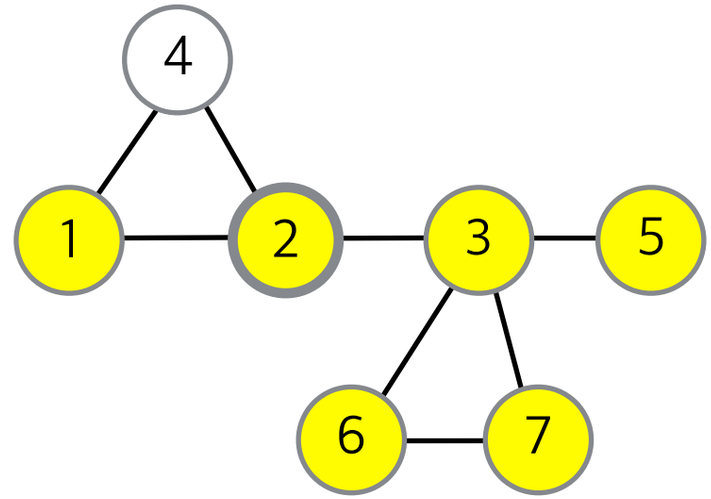
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2], v = 3

[3, 5, 6, 7]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

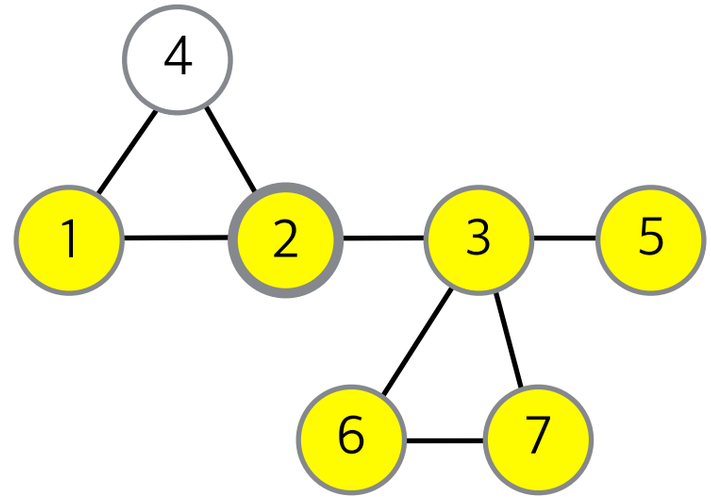
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 3



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

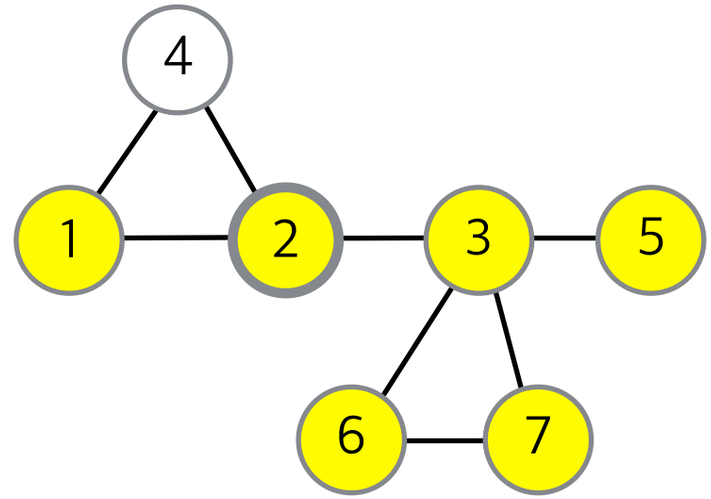
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

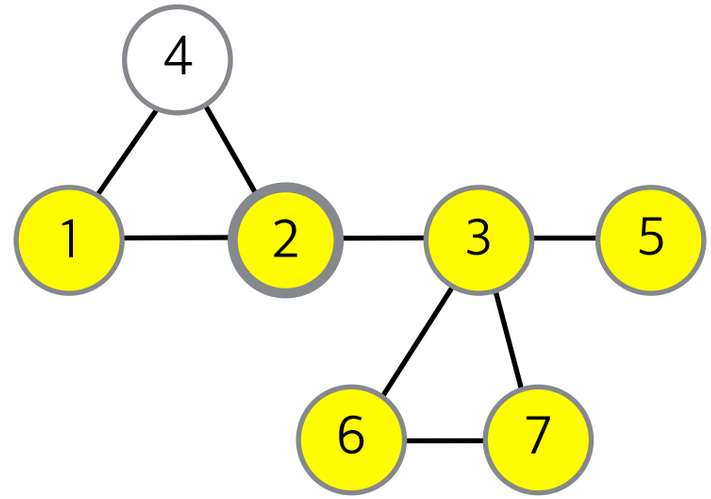
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

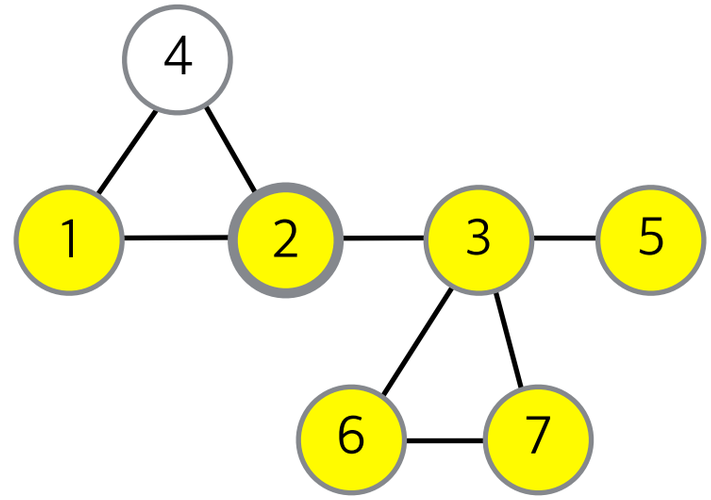
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4

DFS(graph, 4, visited)가 return하는 값은 ?



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

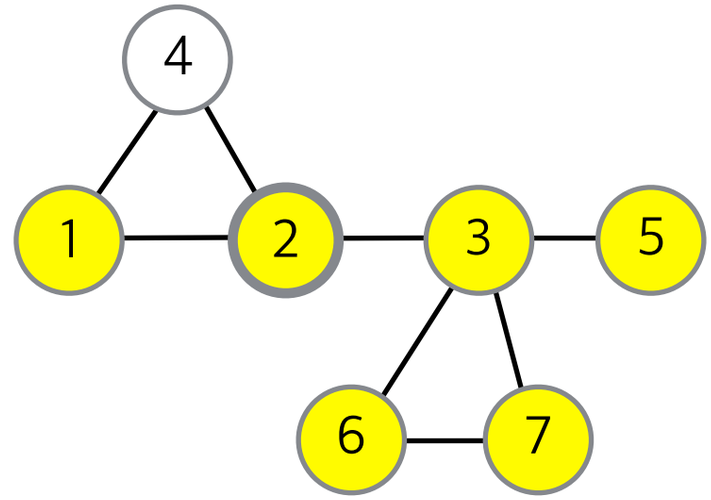
DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7], v = 4

[4]



visited= [F, T, T, T, F, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

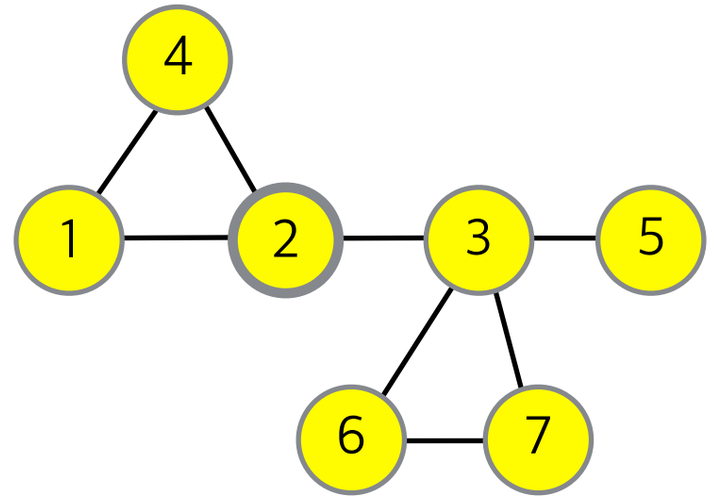
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, **T**, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

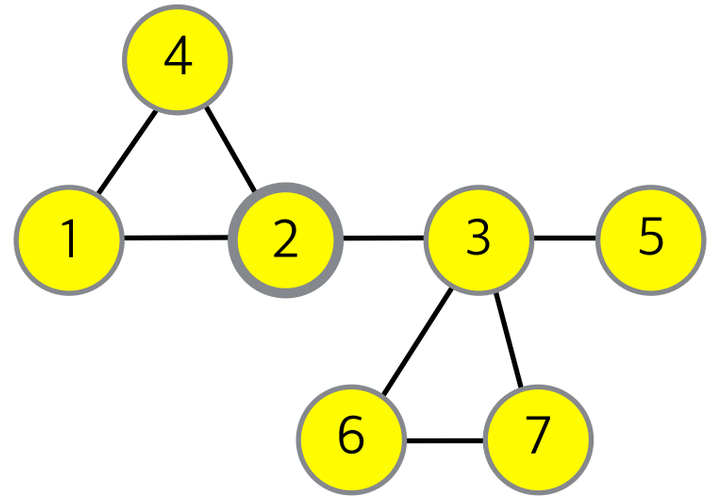
```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

→   return result
```

DFS(graph, 1, visited)

result = [1], v = 2

DFS(graph, 2, visited)

result = [2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

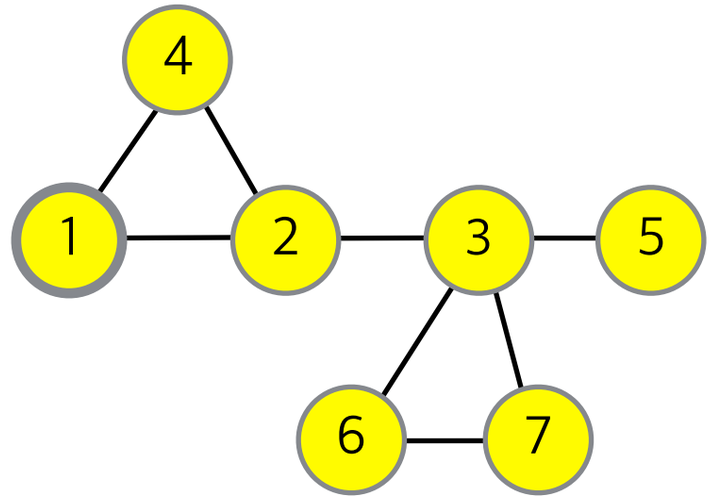
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1], v = 2

[2, 3, 5, 6, 7, 4]



visited= [F, T, T, T, T, T, T, T]

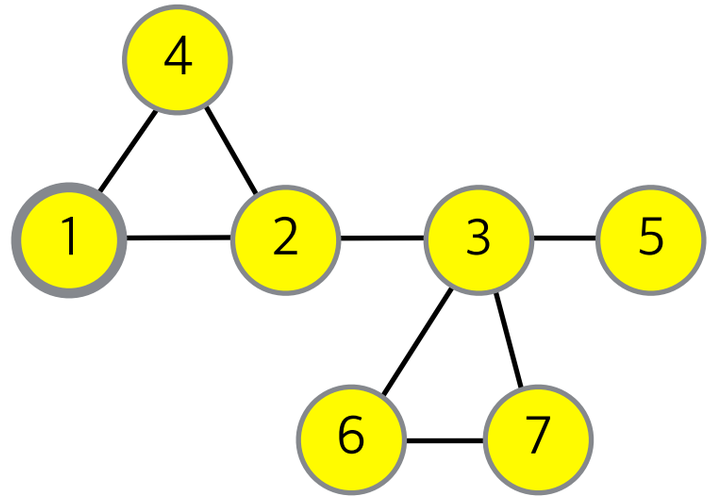
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 2



visited= [F, T, T, T, T, T, T, T]

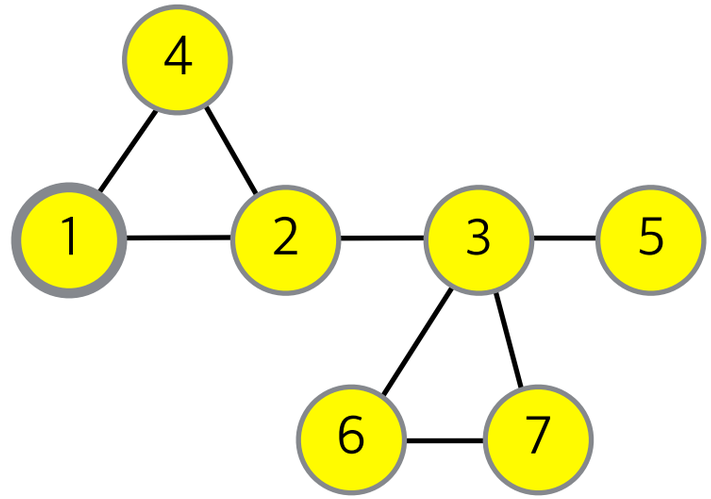
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

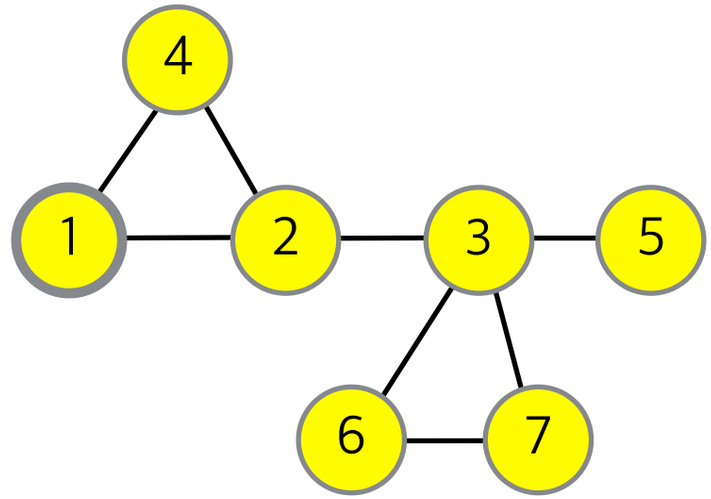
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

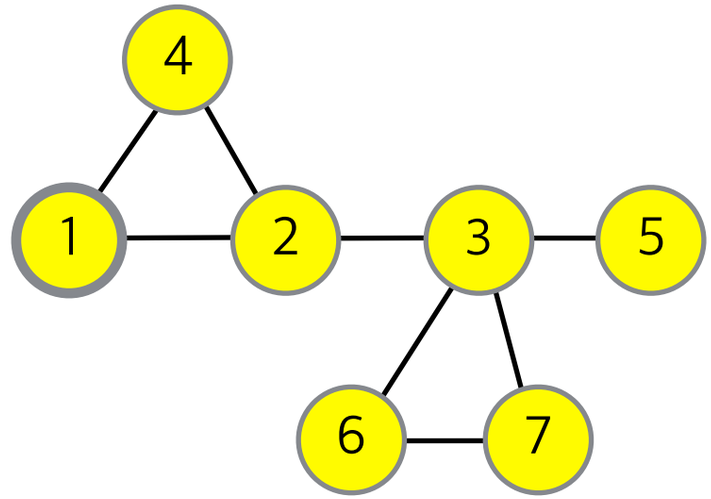
graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

→   return result
```

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

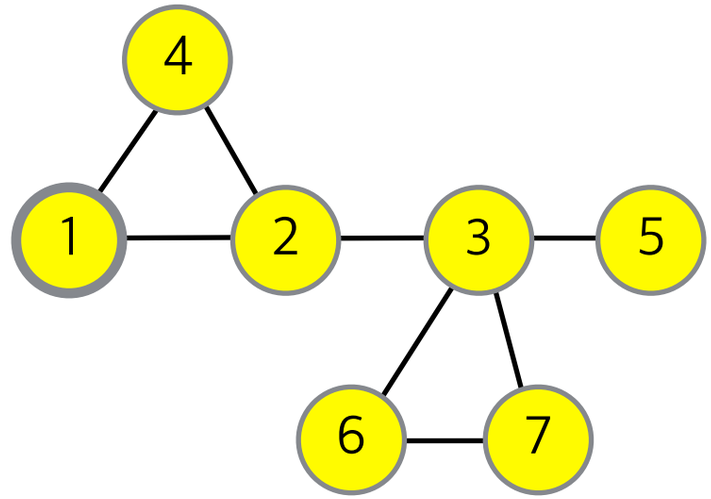
깊이 우선 탐색 (DFS) 구현

```
def DFS(graph, x, visited) :
    visited[x] = True
    result = [x]

    for v in graph[x] :
        if visited[v] == False :
            result = result +
                     DFS(graph, v, visited)

    return result
```

[1, 2, 3, 5, 6, 7, 4]

DFS(graph, 1, visited)

result = [1, 2, 3, 5, 6, 7, 4], v = 4



visited= [F, T, T, T, T, T, T, T]

graph[1] = [2, 4]
graph[2] = [1, 3, 4]
graph[3] = [2, 5, 6, 7]
graph[4] = [1, 2]
graph[5] = [3]
graph[6] = [3, 7]
graph[7] = [3, 6]

그래프 순회

깊이우선탐색 (Depth First Search)

스택을 이용하여 그래프를 순회하는 방법

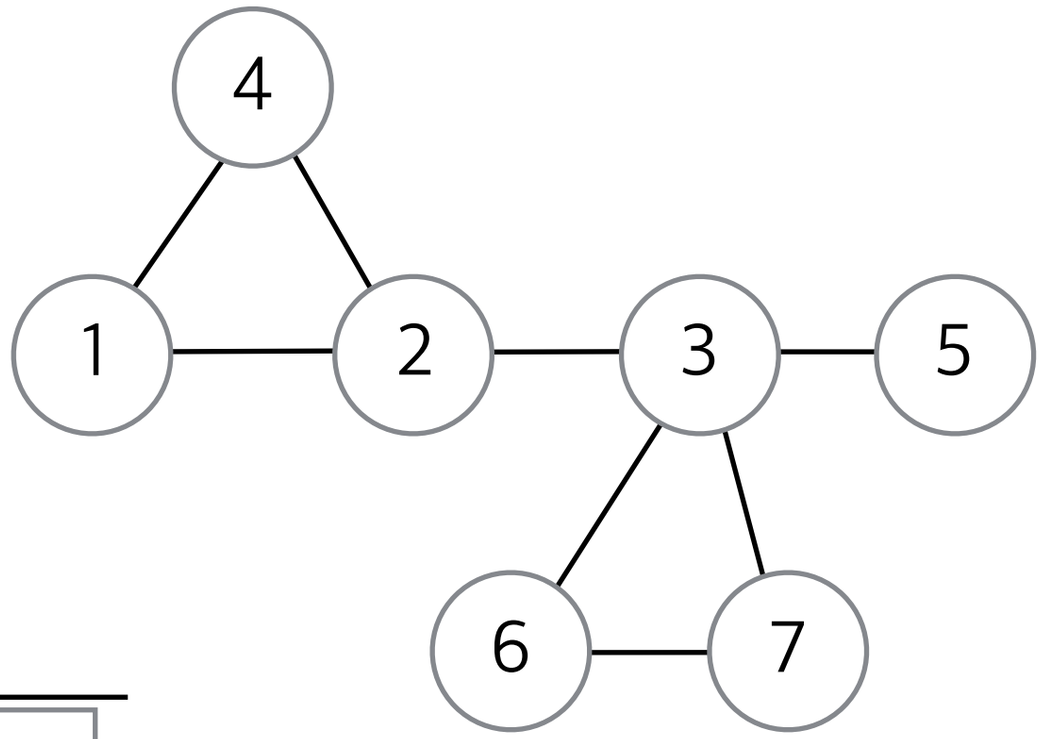
너비우선탐색 (Breadth First Search)

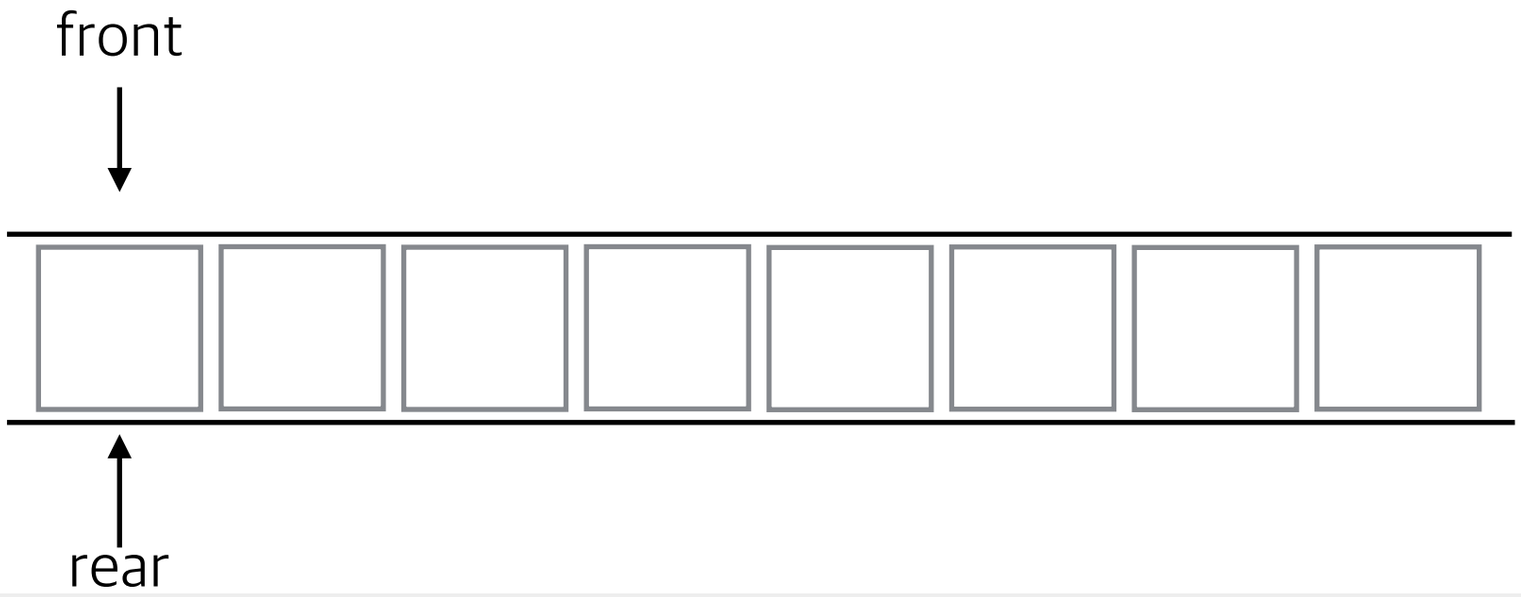
큐를 이용하여 그래프를 순회하는 방법

너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. Node 선택
2. 인접한 노드 색칠

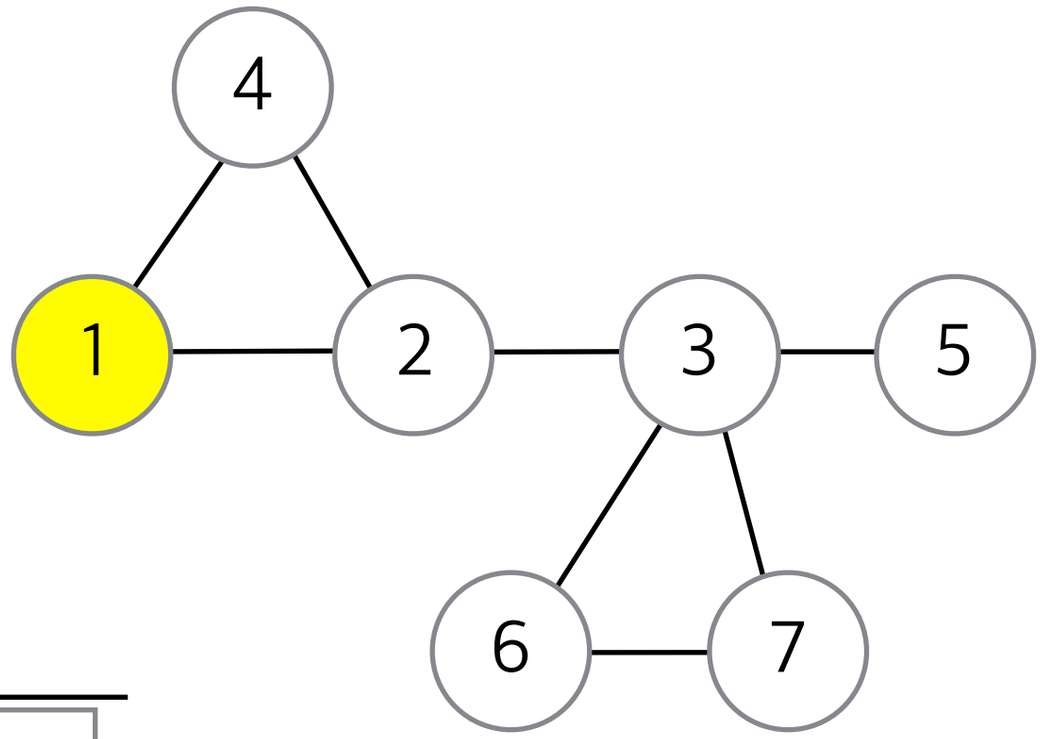




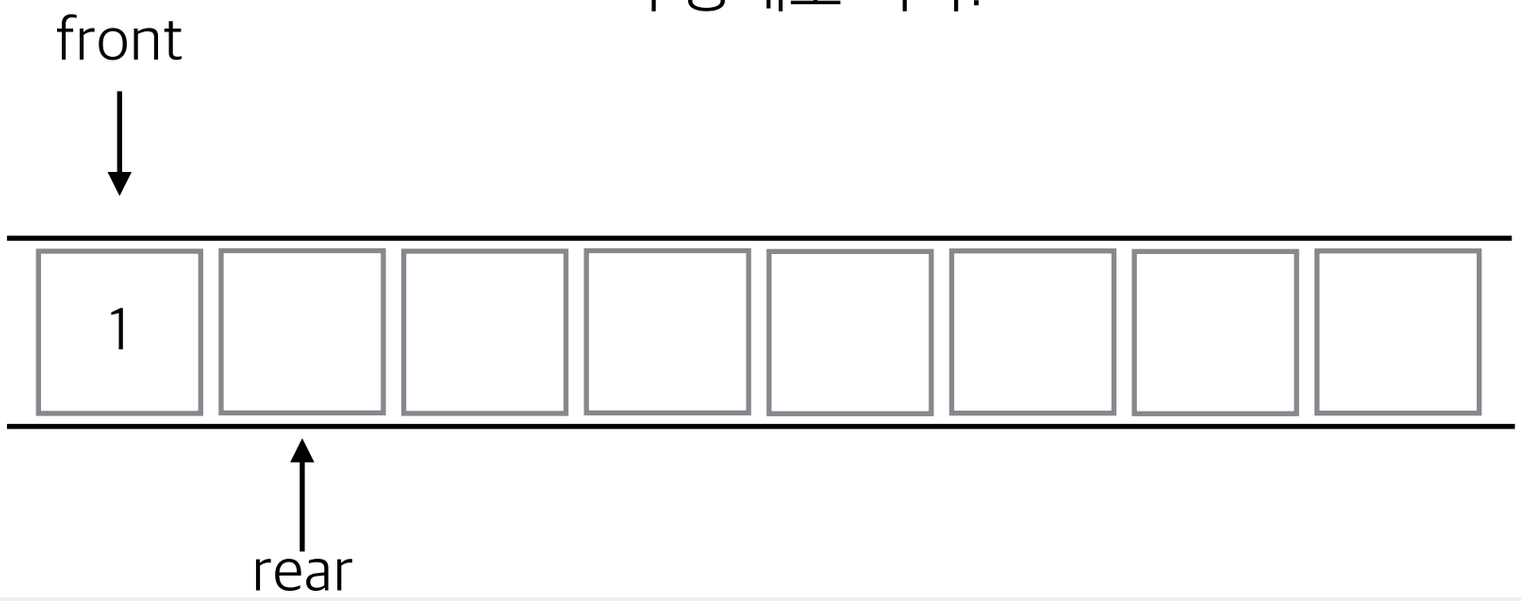
너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. Node 선택
2. 인접한 노드 색칠



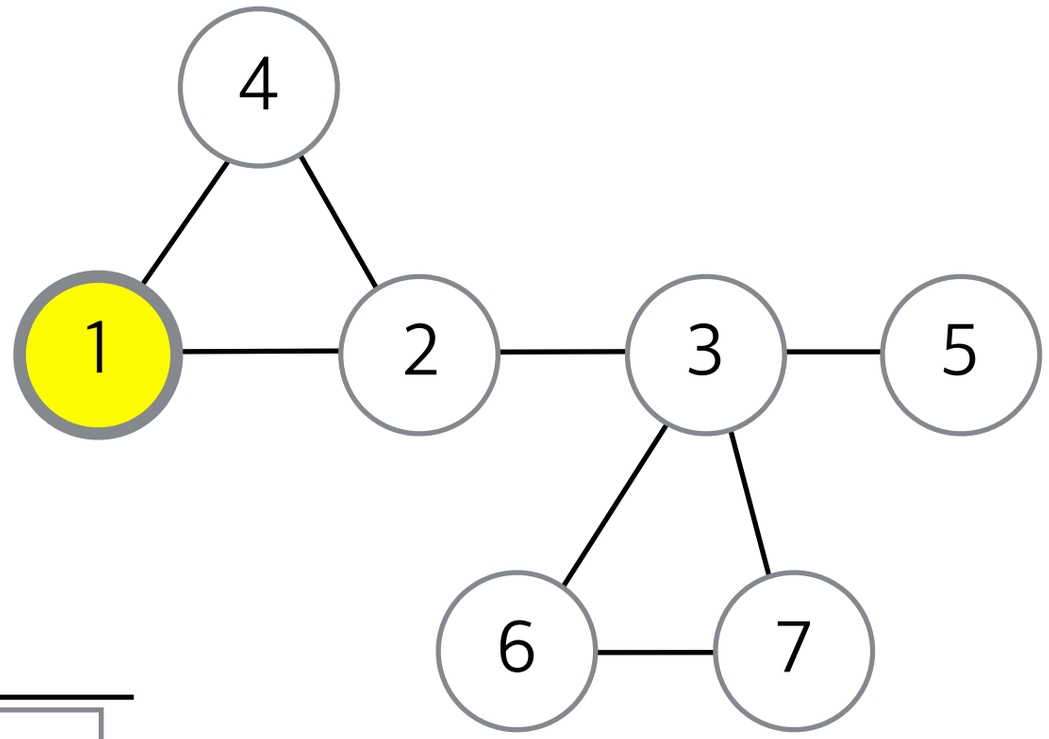
이 상태로 시작!

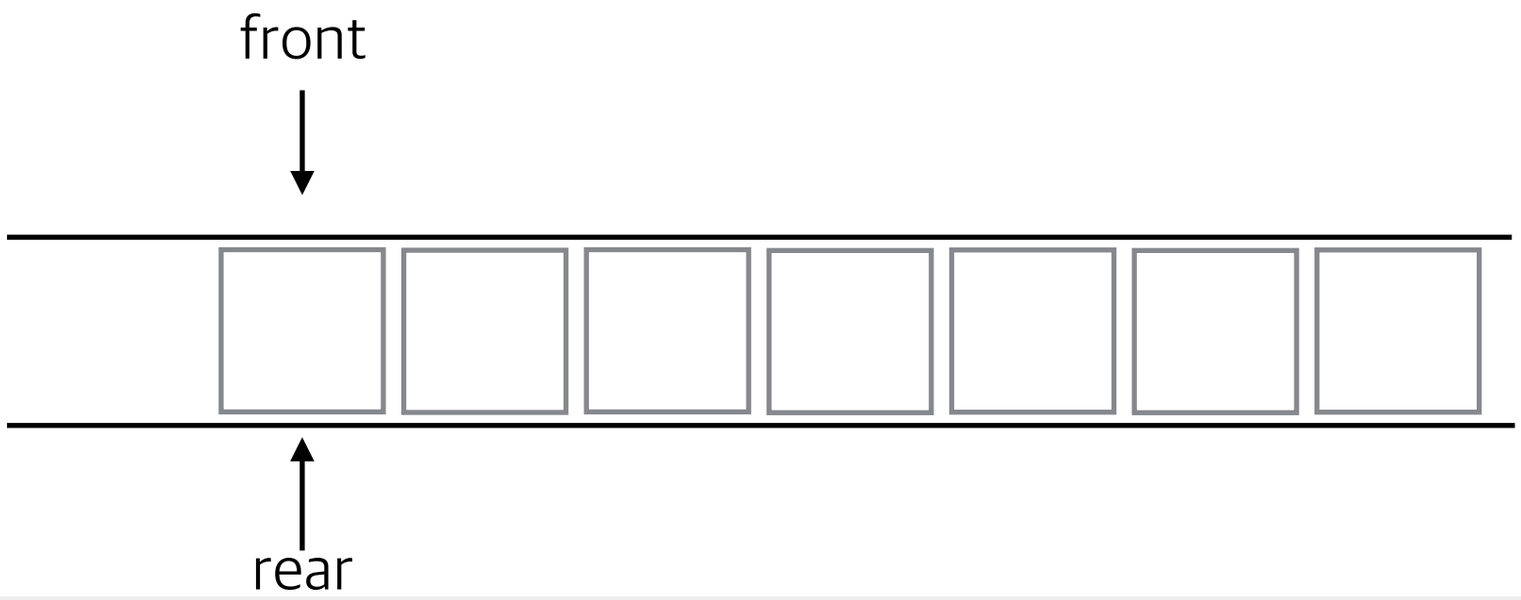


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. **Node 선택**
2. 인접한 노드 색칠

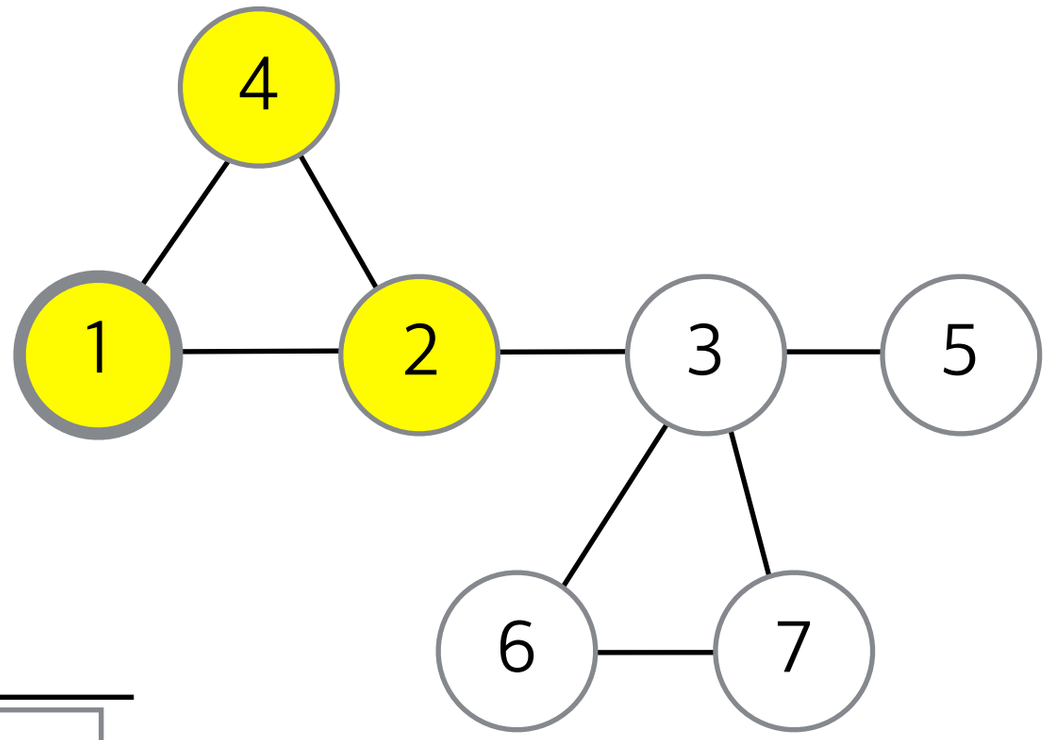


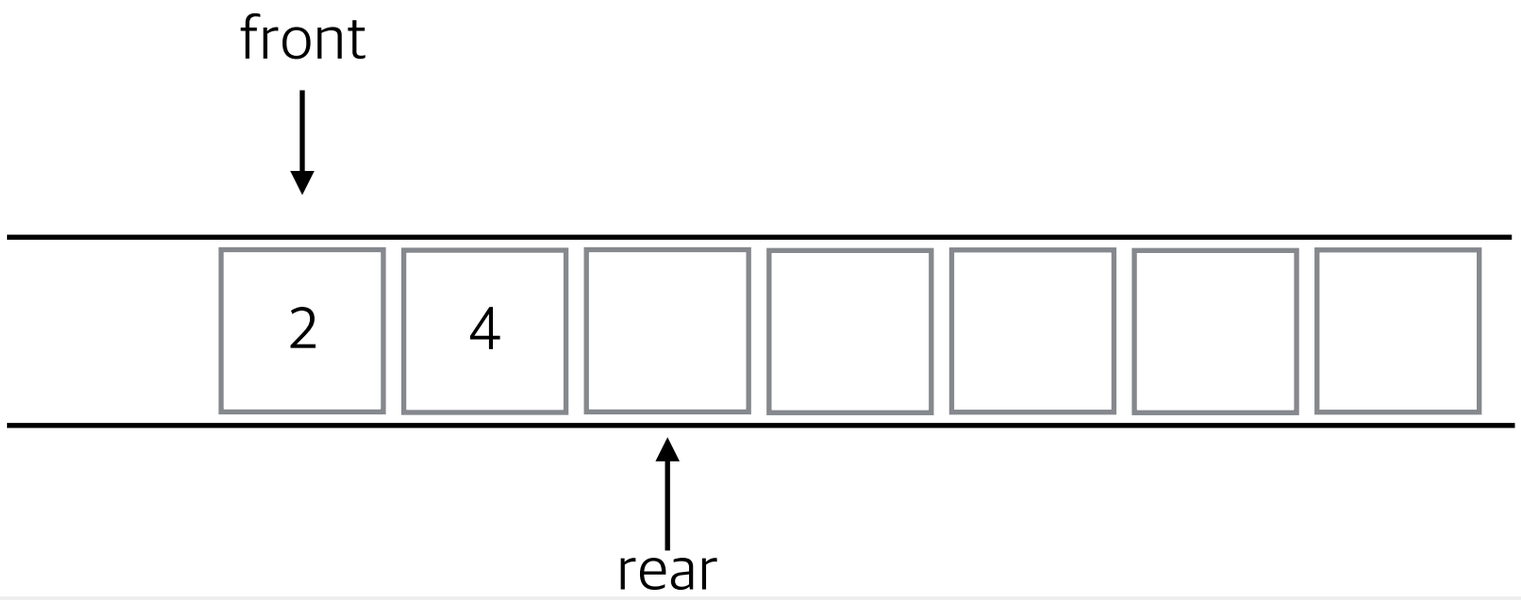


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. Node 선택
2. **인접한 노드 색칠**

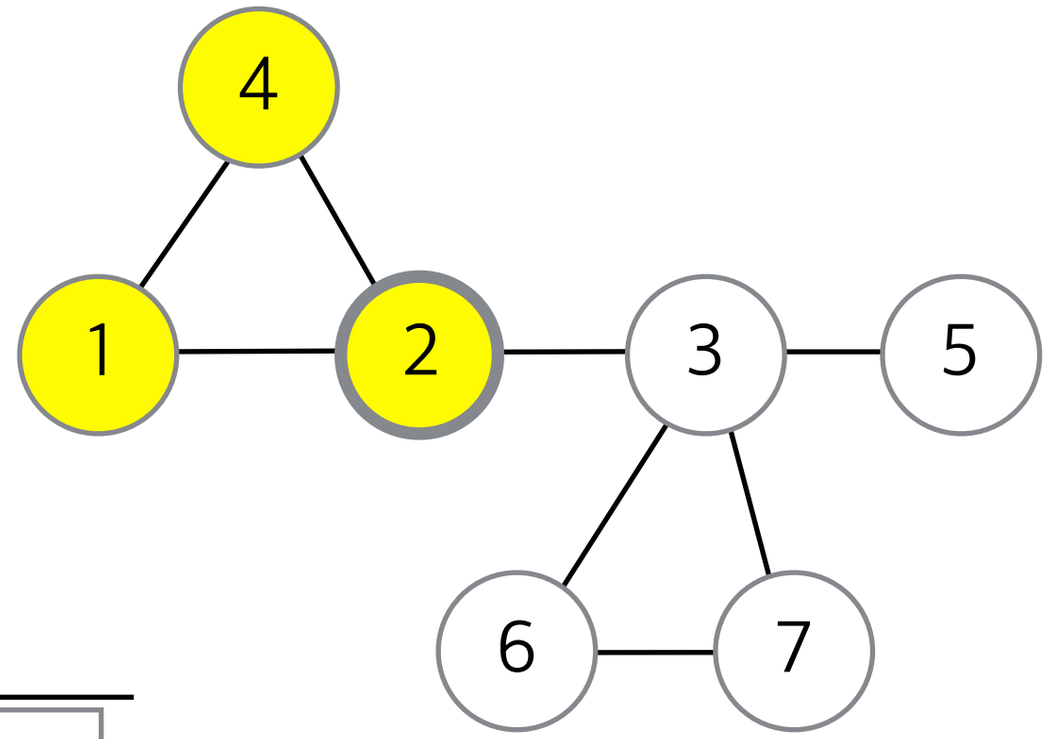


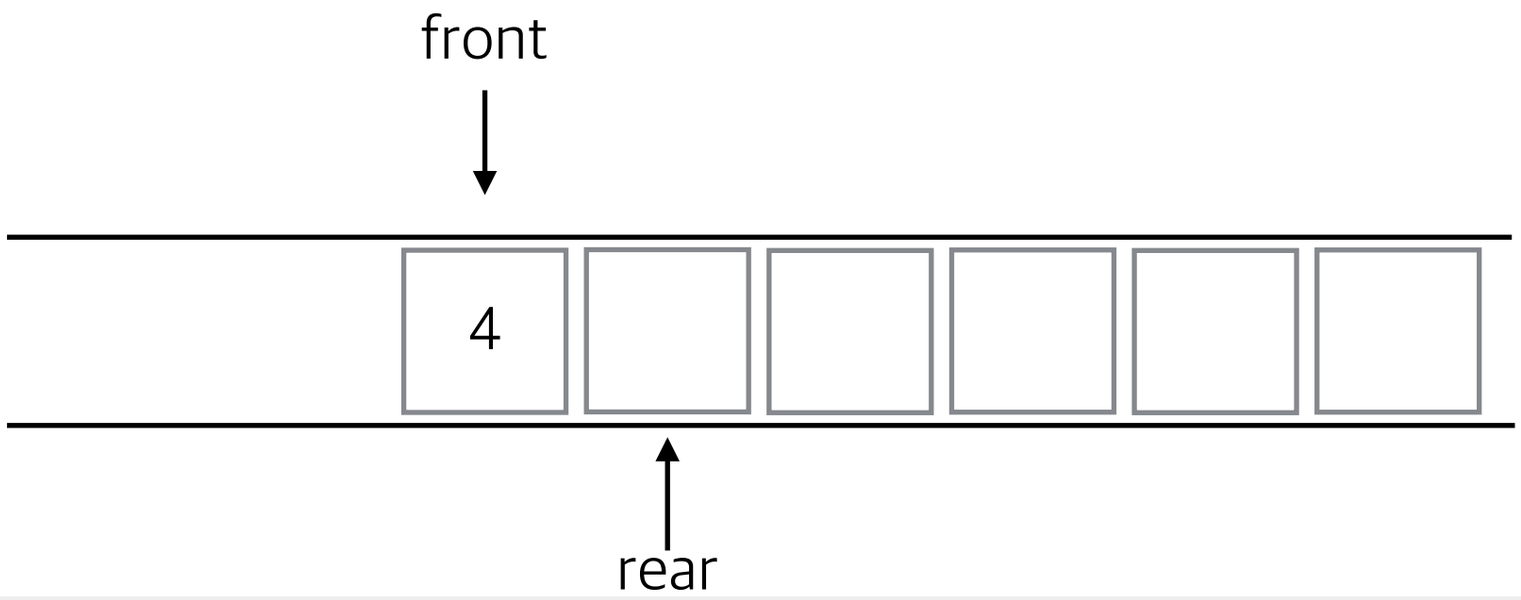


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. **Node 선택**
2. 인접한 노드 색칠

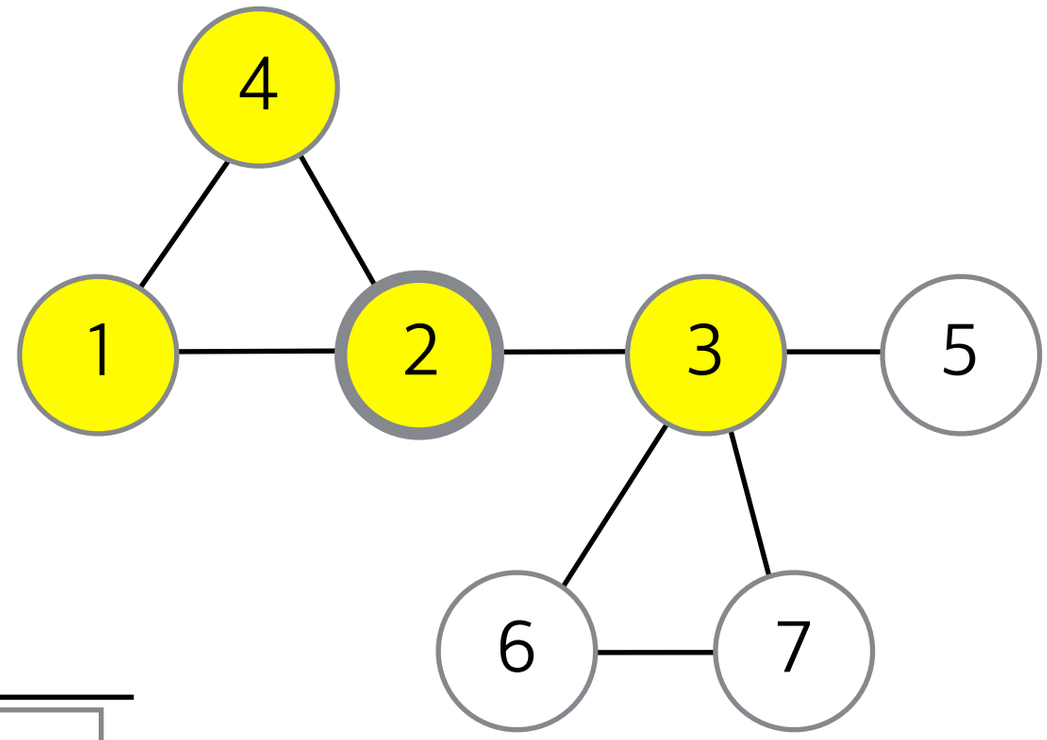


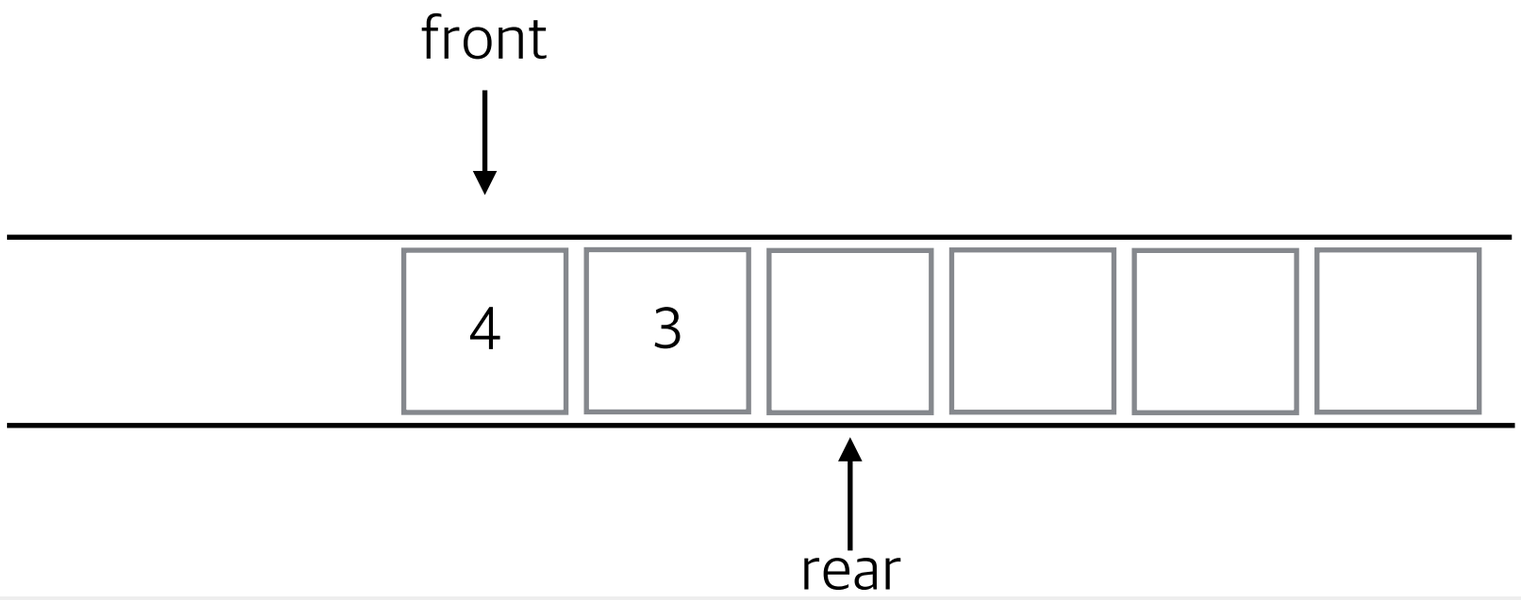


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. Node 선택
2. **인접한 노드 색칠**

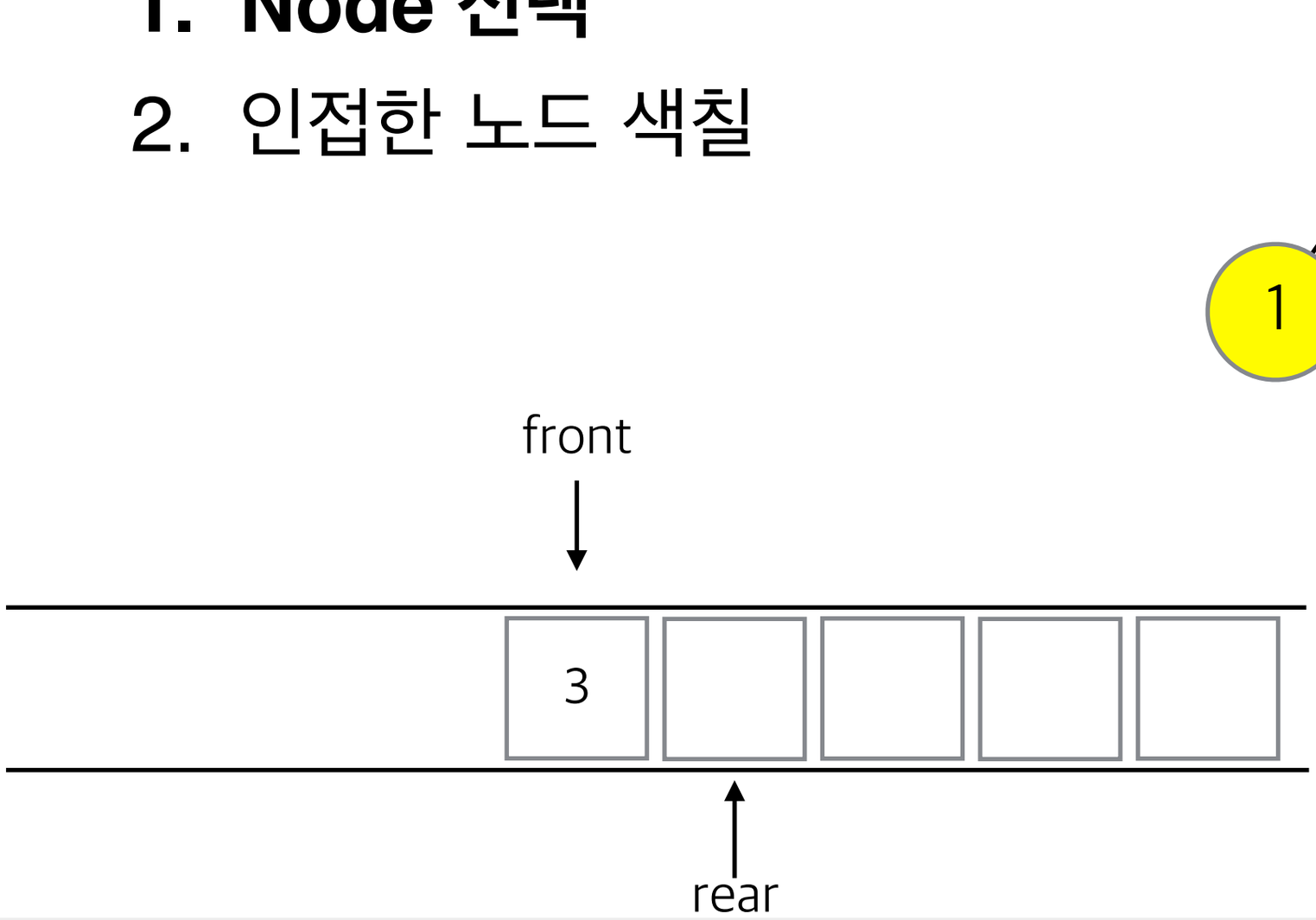


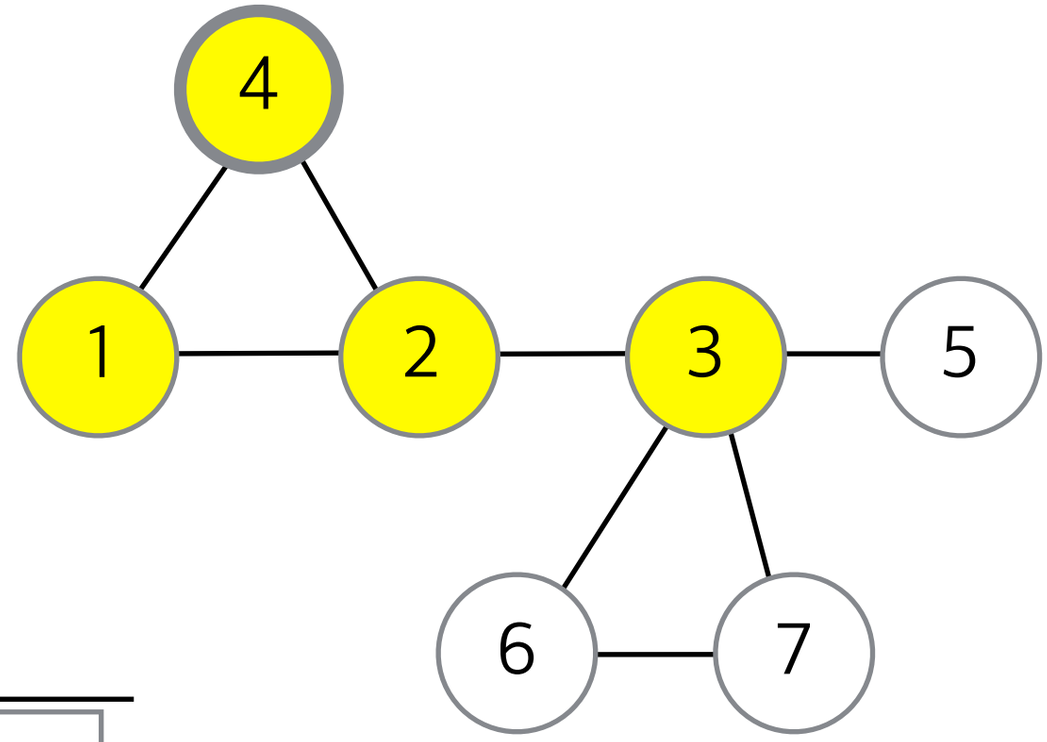


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. **Node 선택**
2. 인접한 노드 색칠

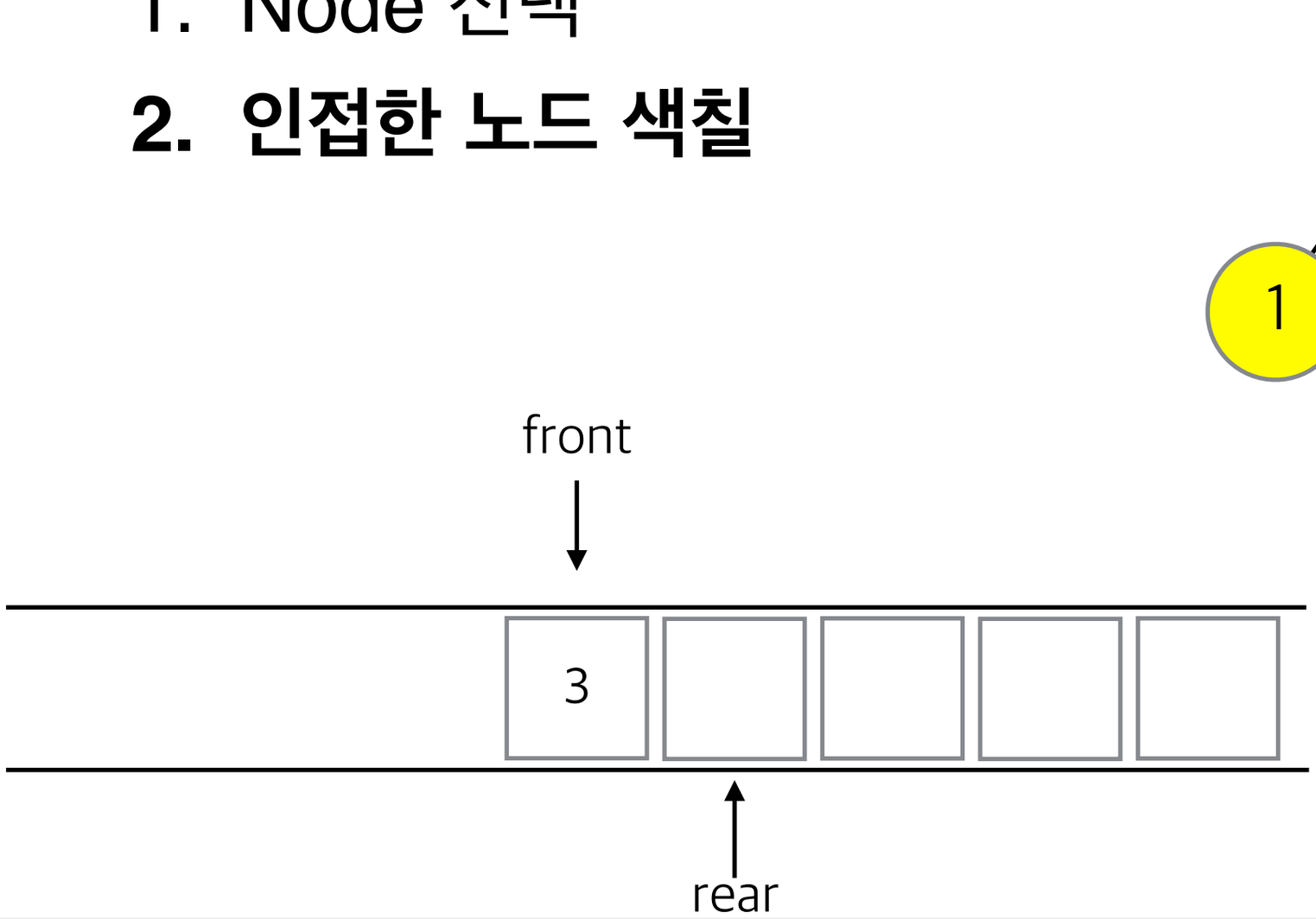


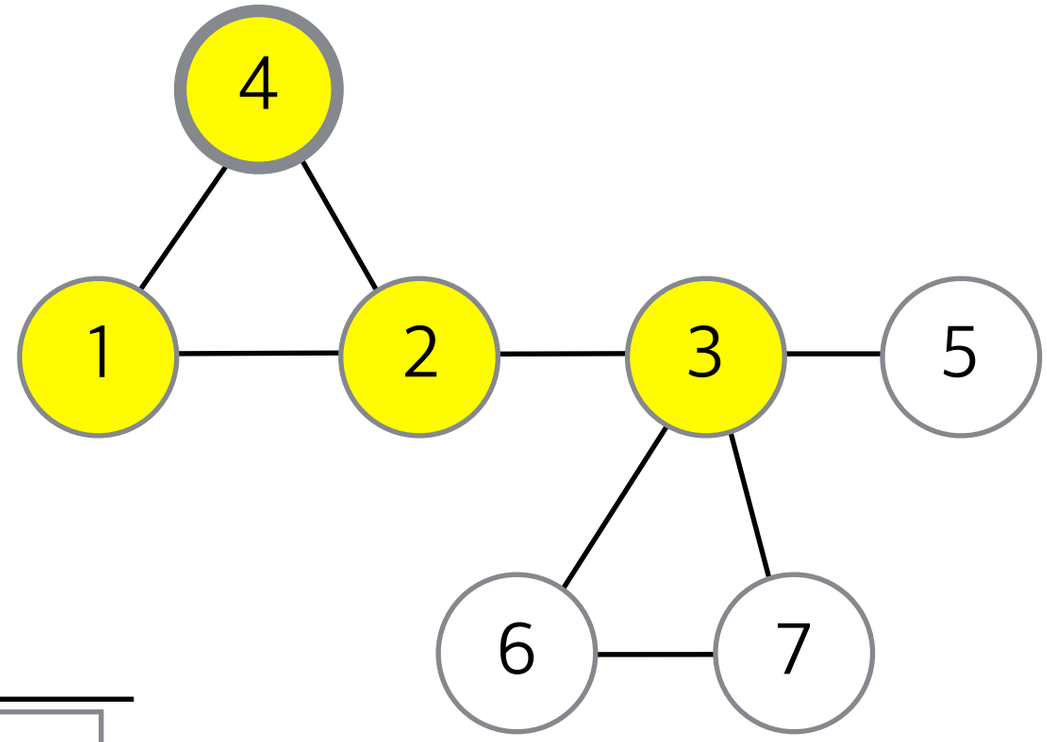


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. Node 선택
2. **인접한 노드 색칠**

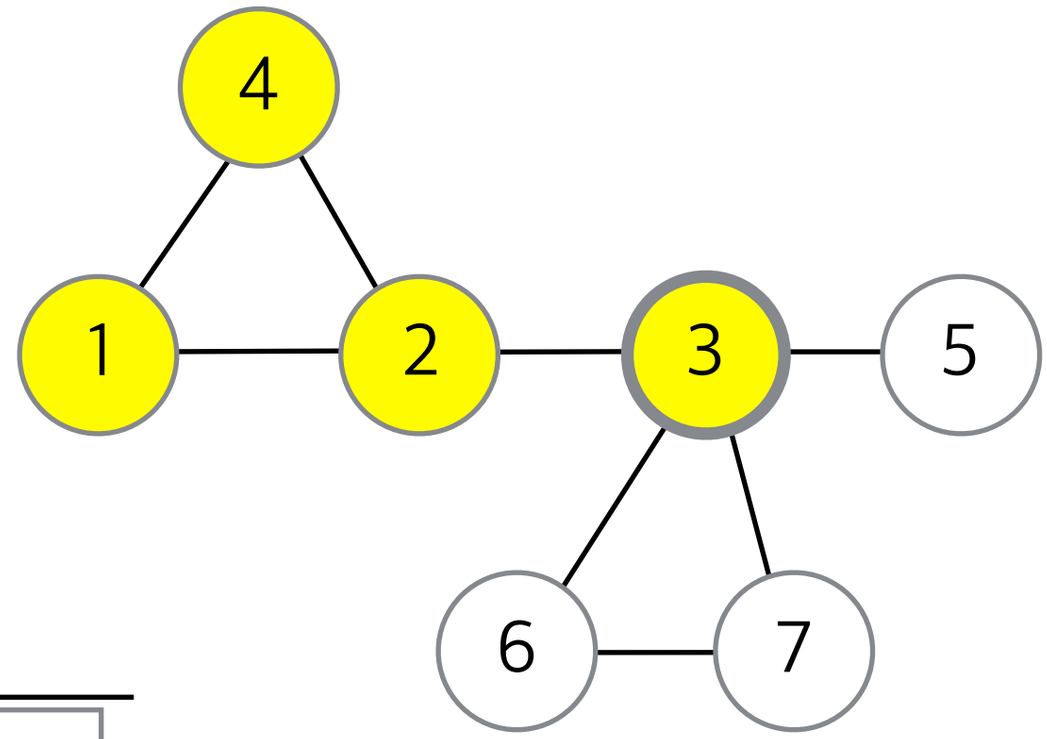


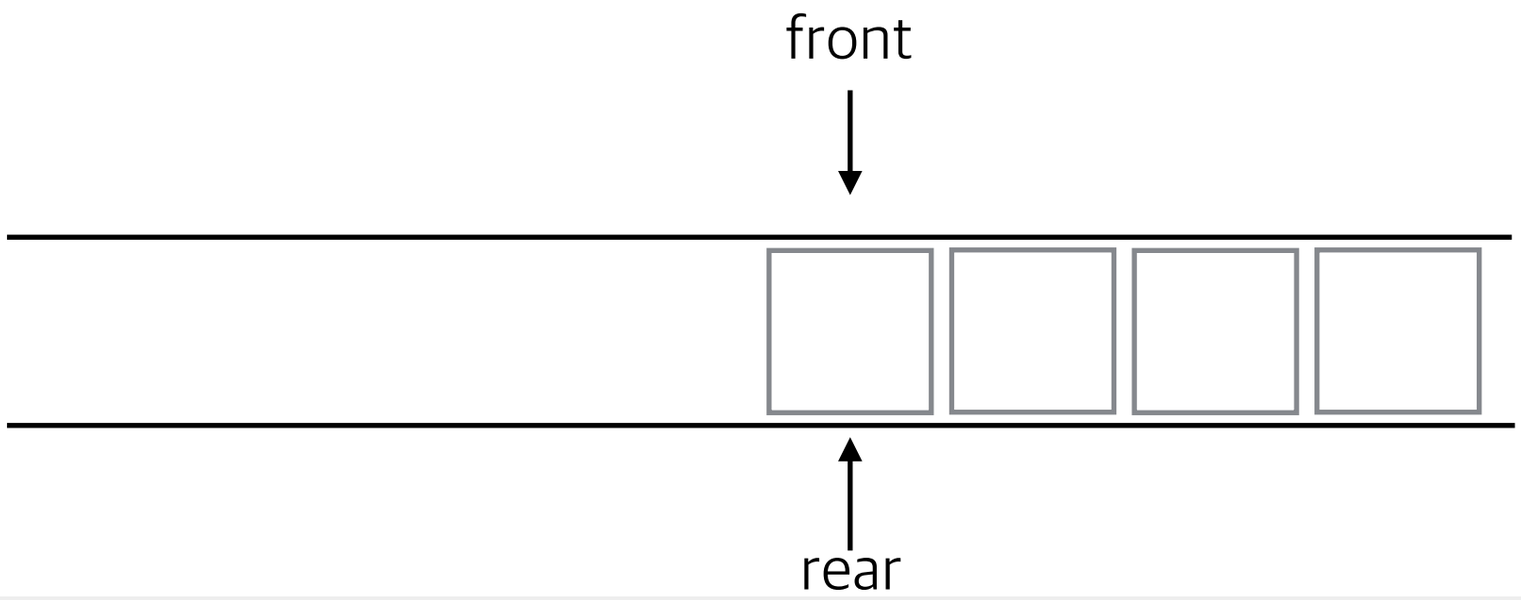


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. **Node 선택**
2. 인접한 노드 색칠

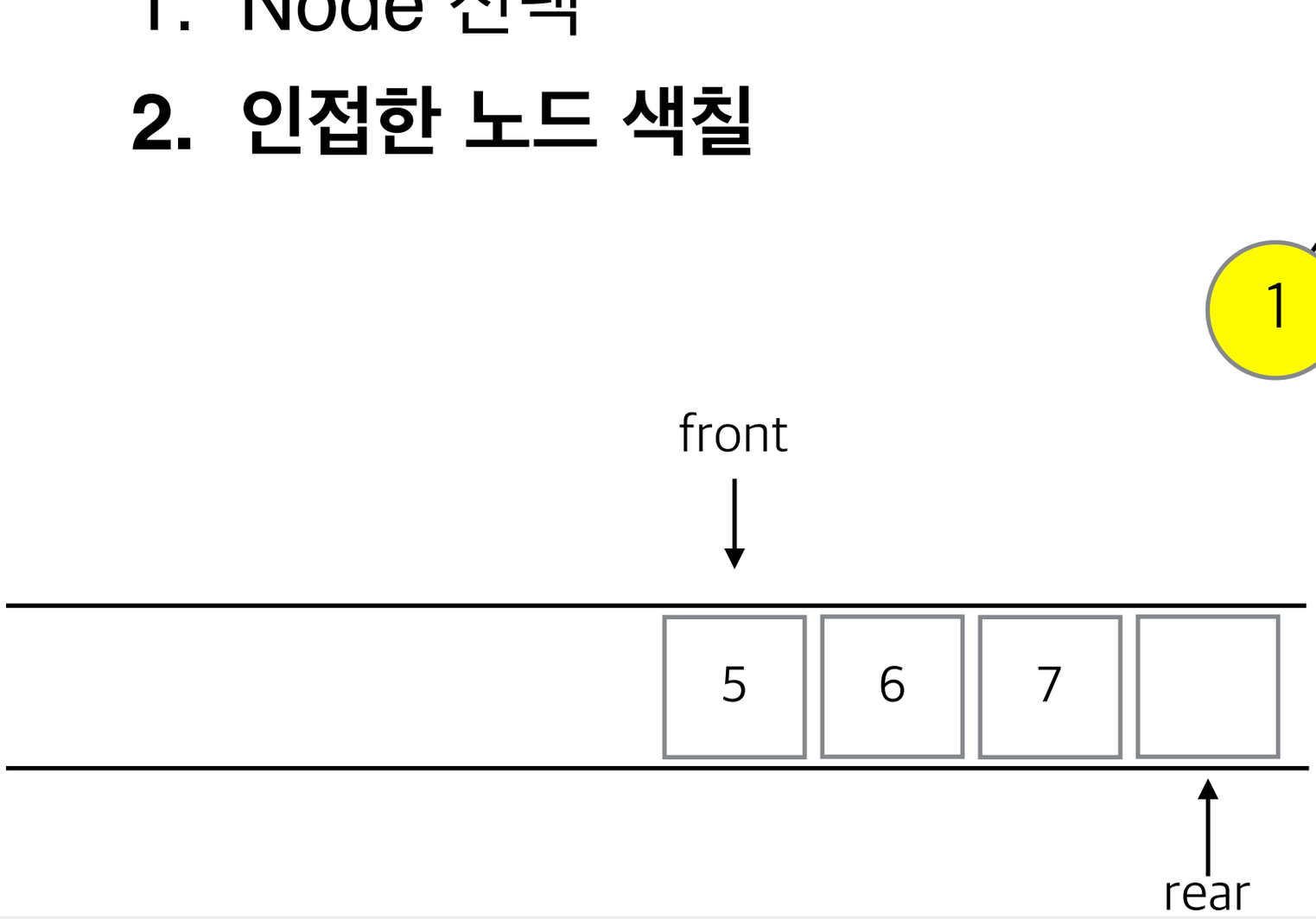


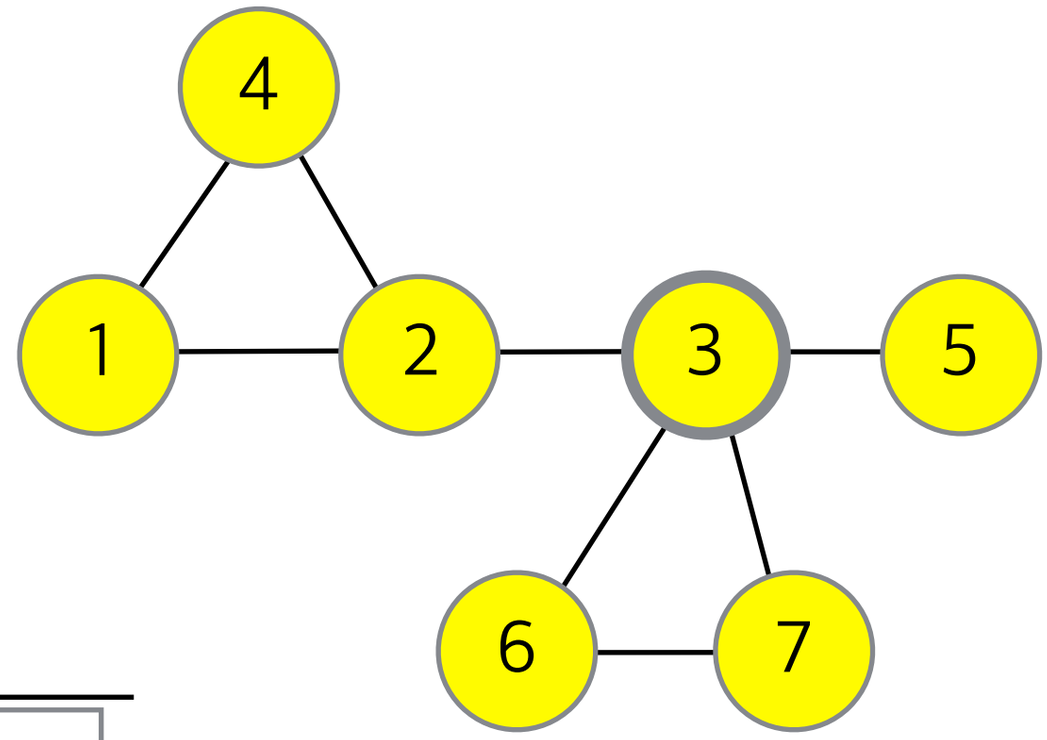


너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

1. Node 선택
2. **인접한 노드 색칠**

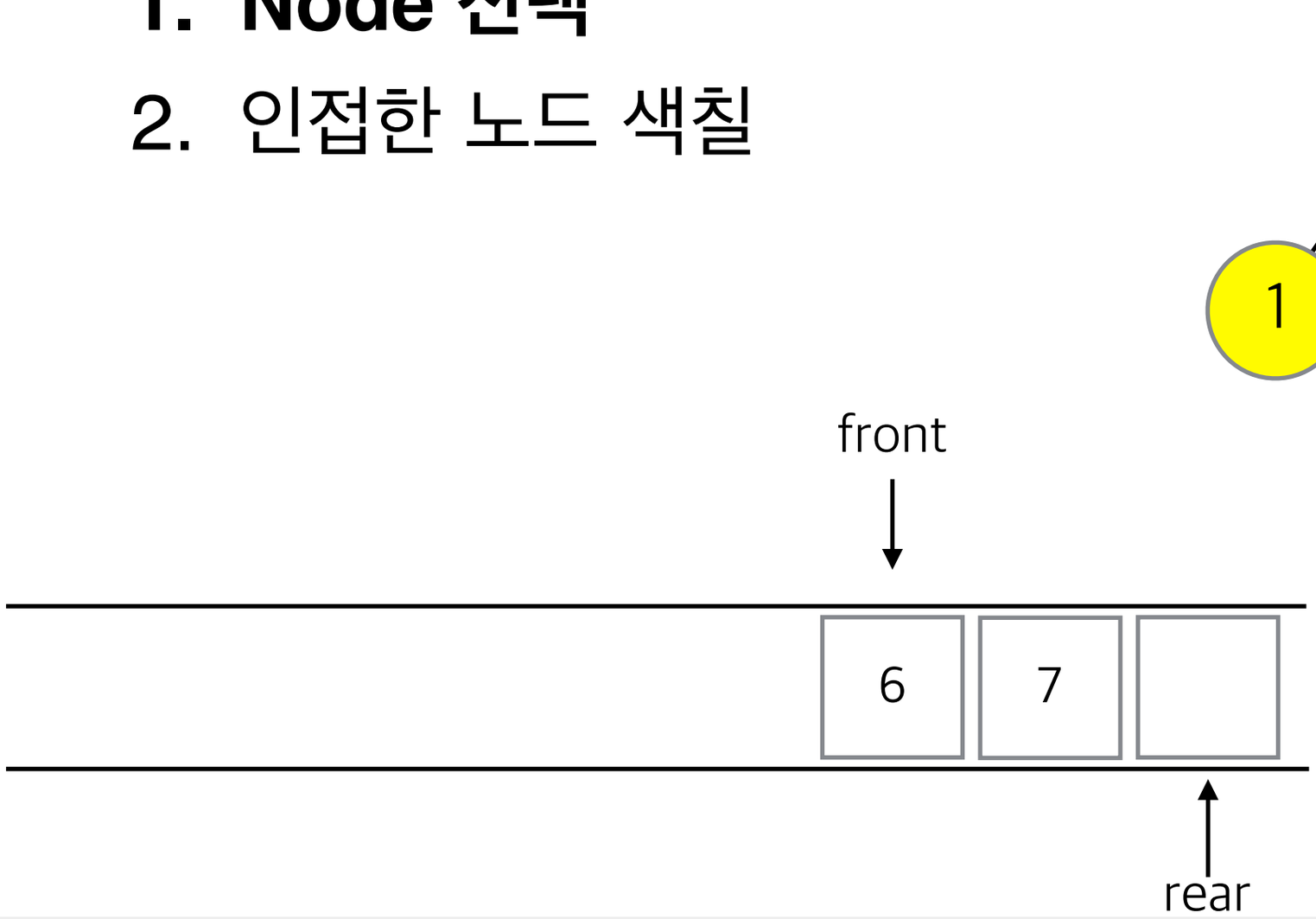




너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

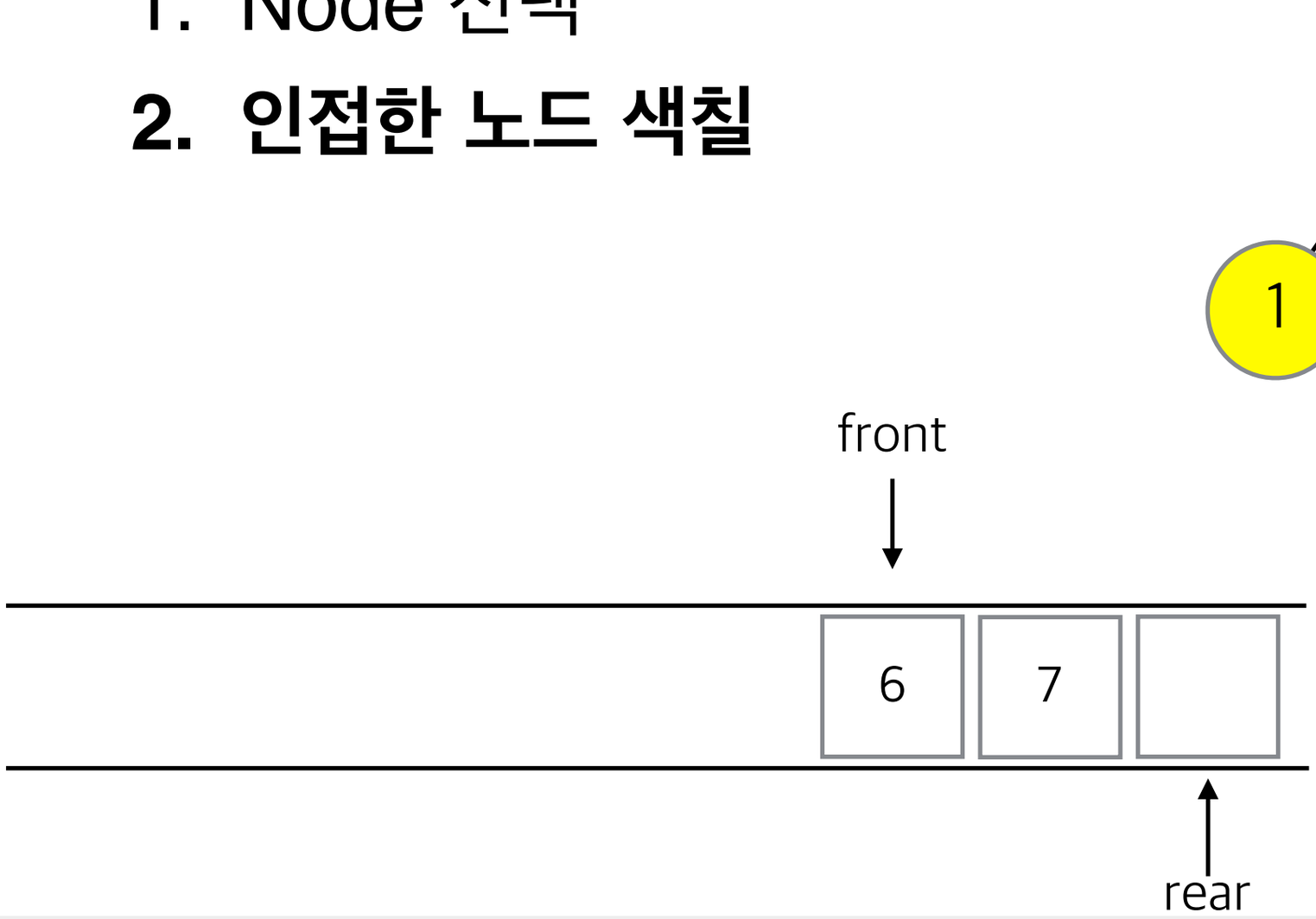
1. **Node 선택**
2. 인접한 노드 색칠



너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

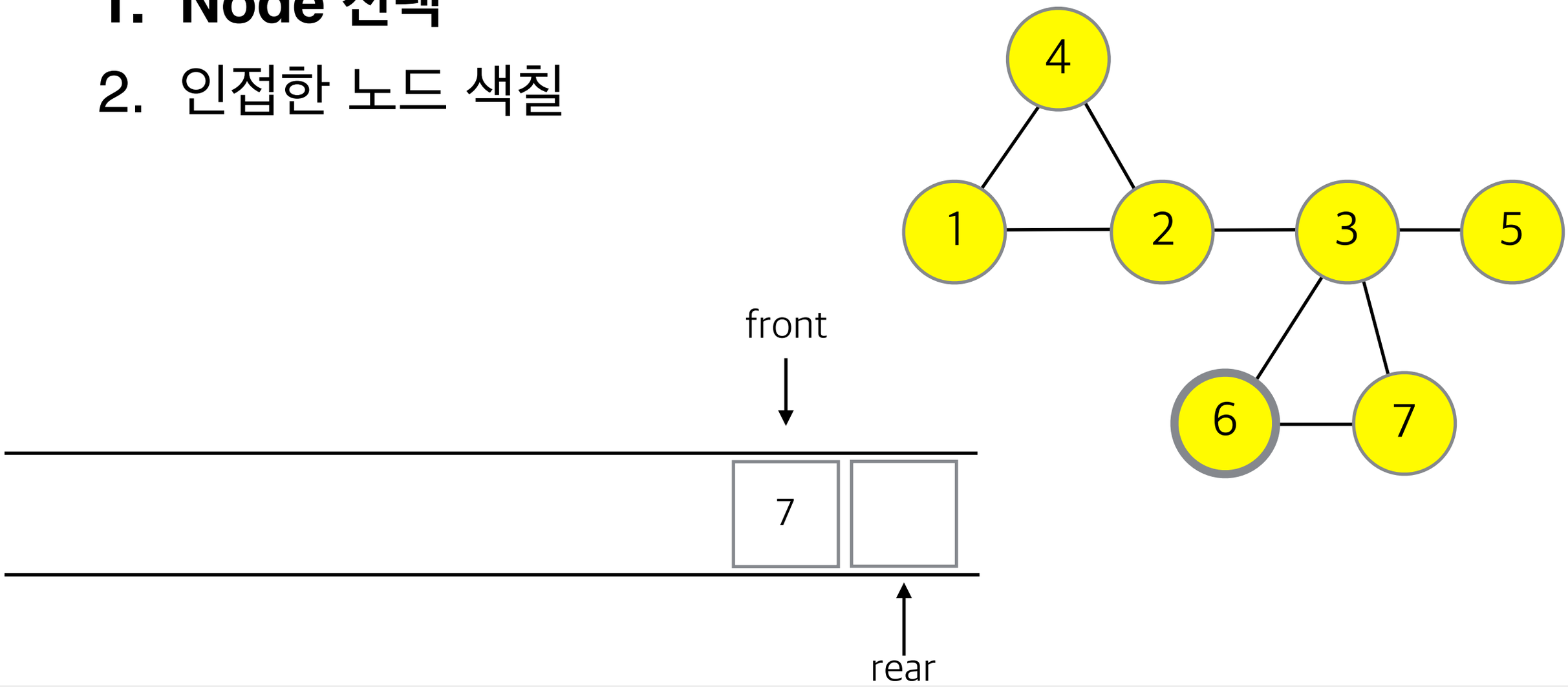
1. Node 선택
2. **인접한 노드 색칠**



너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

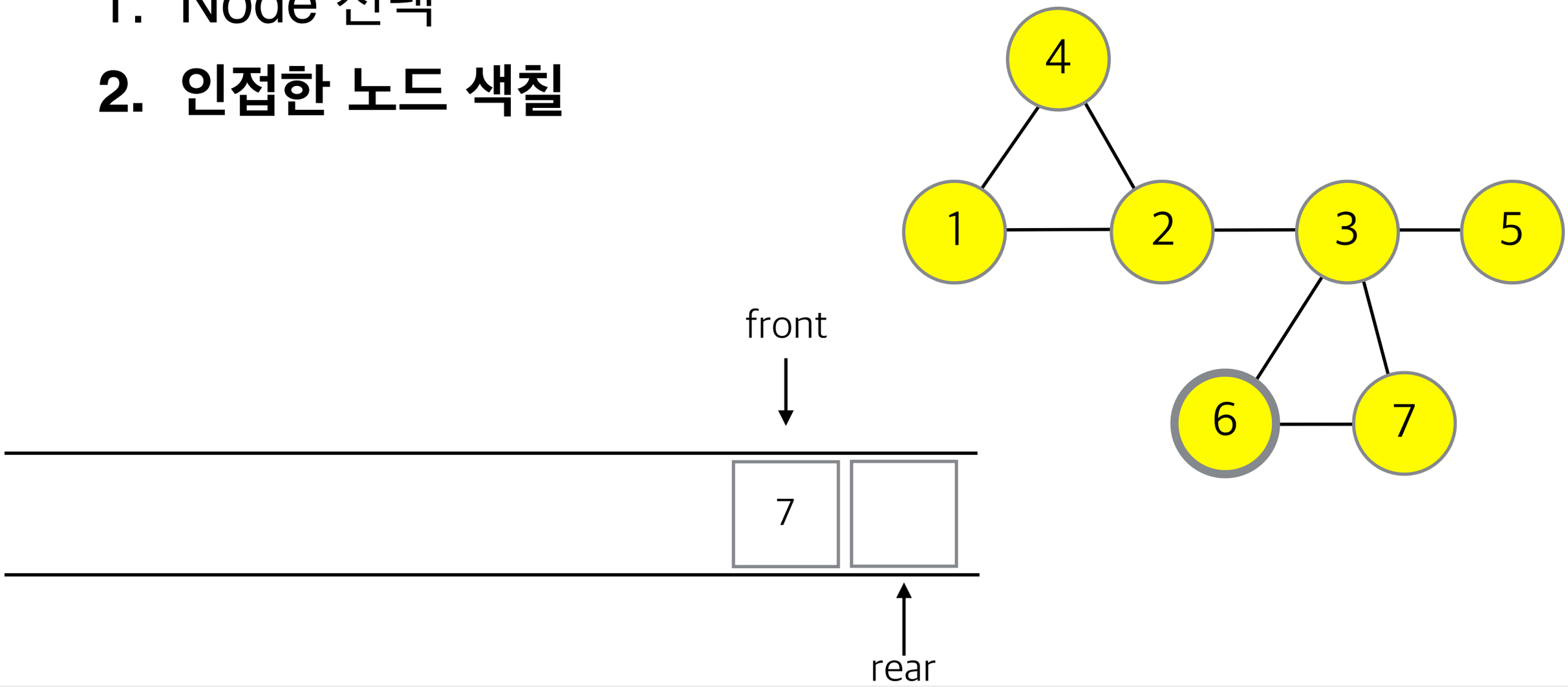
1. **Node 선택**
2. 인접한 노드 색칠



너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

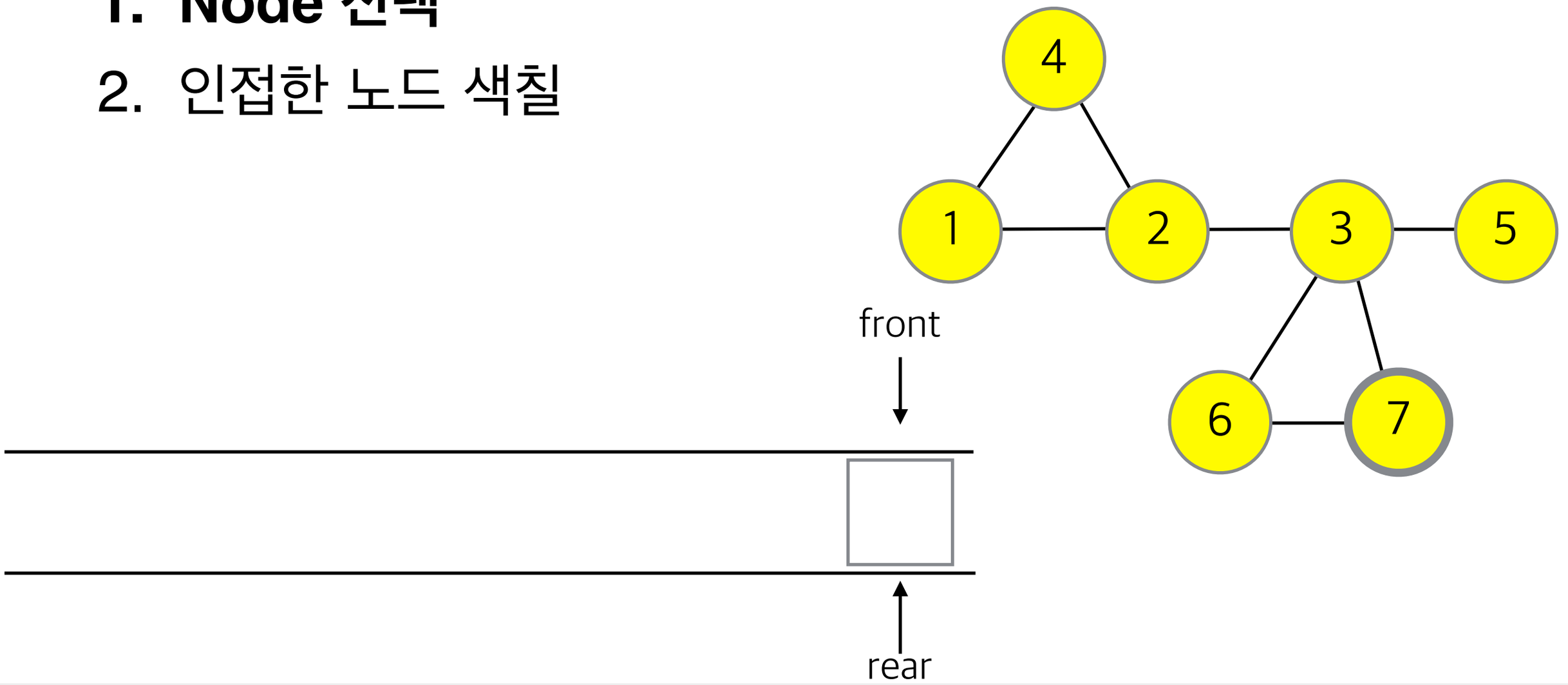
1. Node 선택
2. **인접한 노드 색칠**



너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

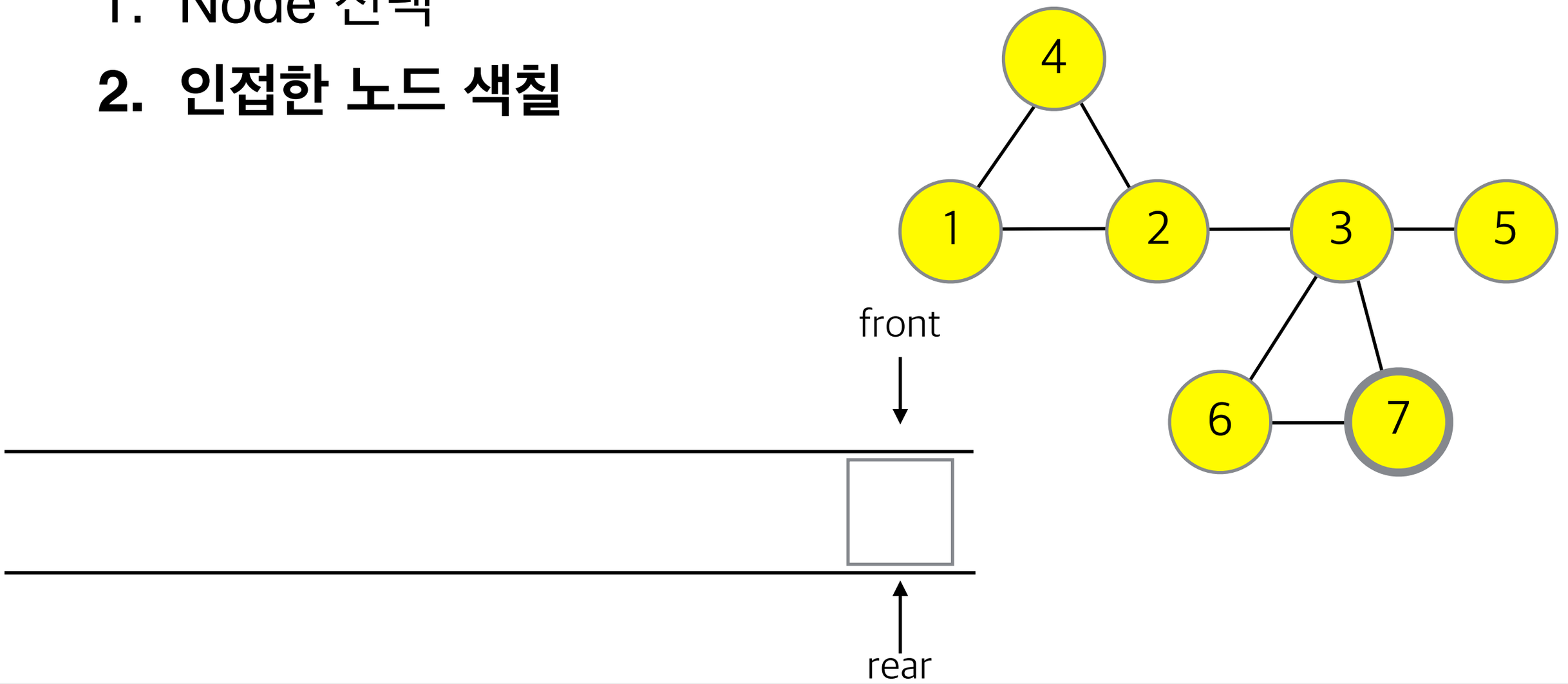
1. **Node 선택**
2. 인접한 노드 색칠



너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

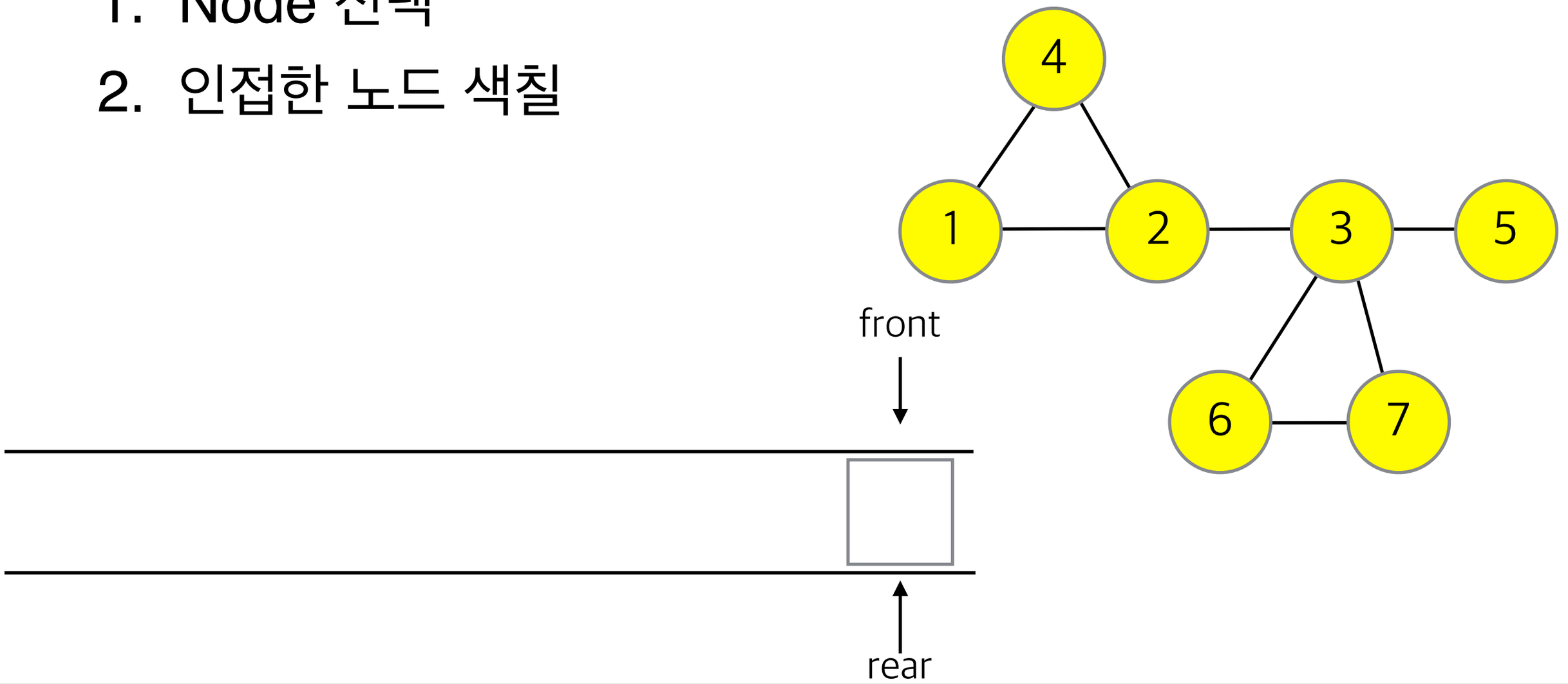
1. Node 선택
2. **인접한 노드 색칠**



너비 우선 탐색 (BFS)

인접한 노드들을 우선 모두 색칠해 두고 뻗어나간다

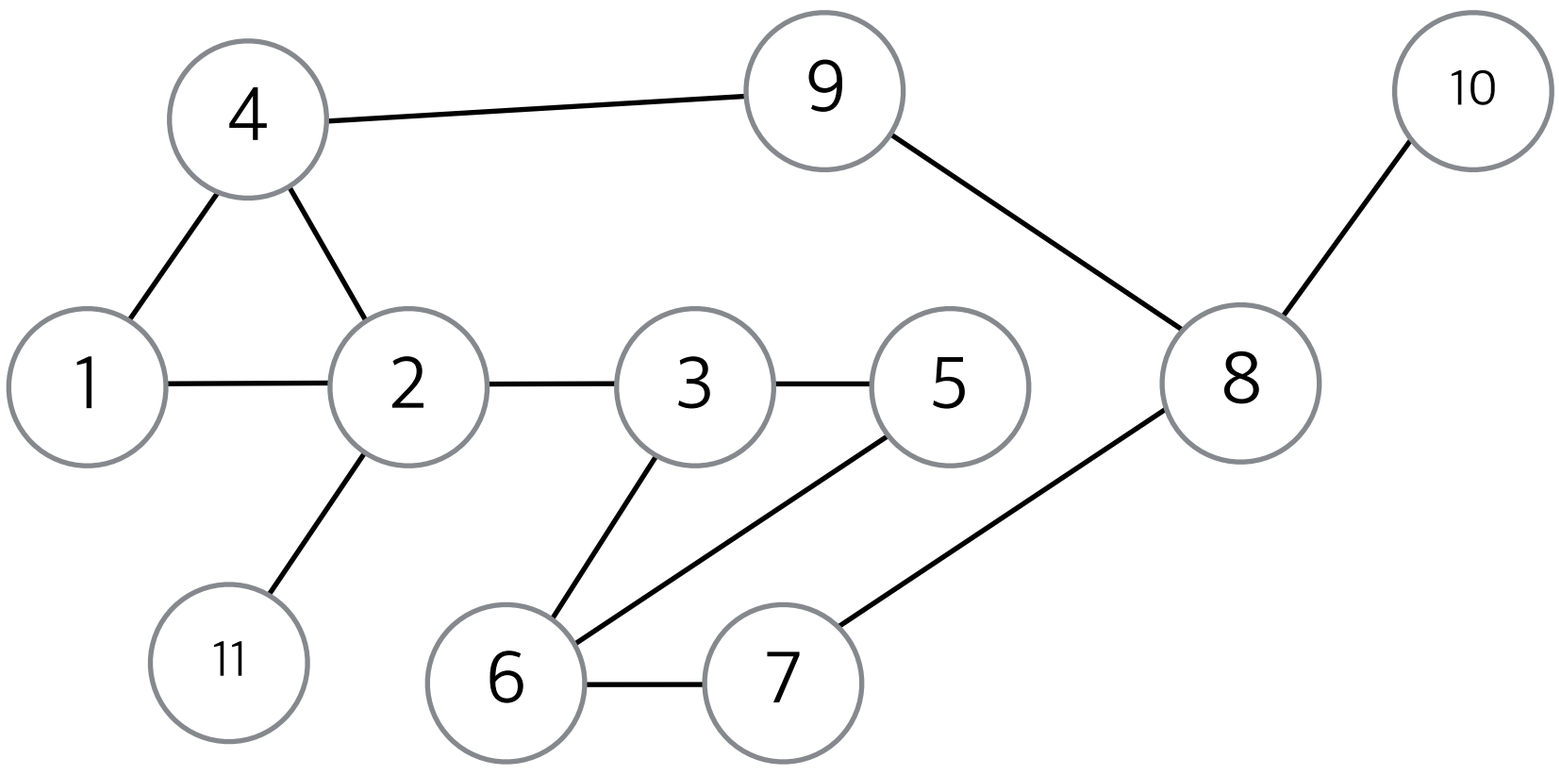
1. Node 선택
2. 인접한 노드 색칠



예제

다음 그래프에 대하여 1을 시작으로 DFS, BFS한 결과는 ?

단, 인접한 노드가 여러개일 경우 노드 번호가 작은 노드부터 방문



[문제 2] 유치원 소풍

유치원에 파벌(?)이 몇개인지,
그리고 각 파벌의 학생 수는 몇명인지를 출력

시스템 입력

```
7
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
```

시스템 출력

```
3
7
8
9
```

그래프 문제가 어려운 점

그래프 문제라는 것을 파악하는 것이 어렵다

문제에는 그래프가 전혀 등장하지 않을 수도 있다

파악 후 풀이를 만들어 내는 것이 어렵다

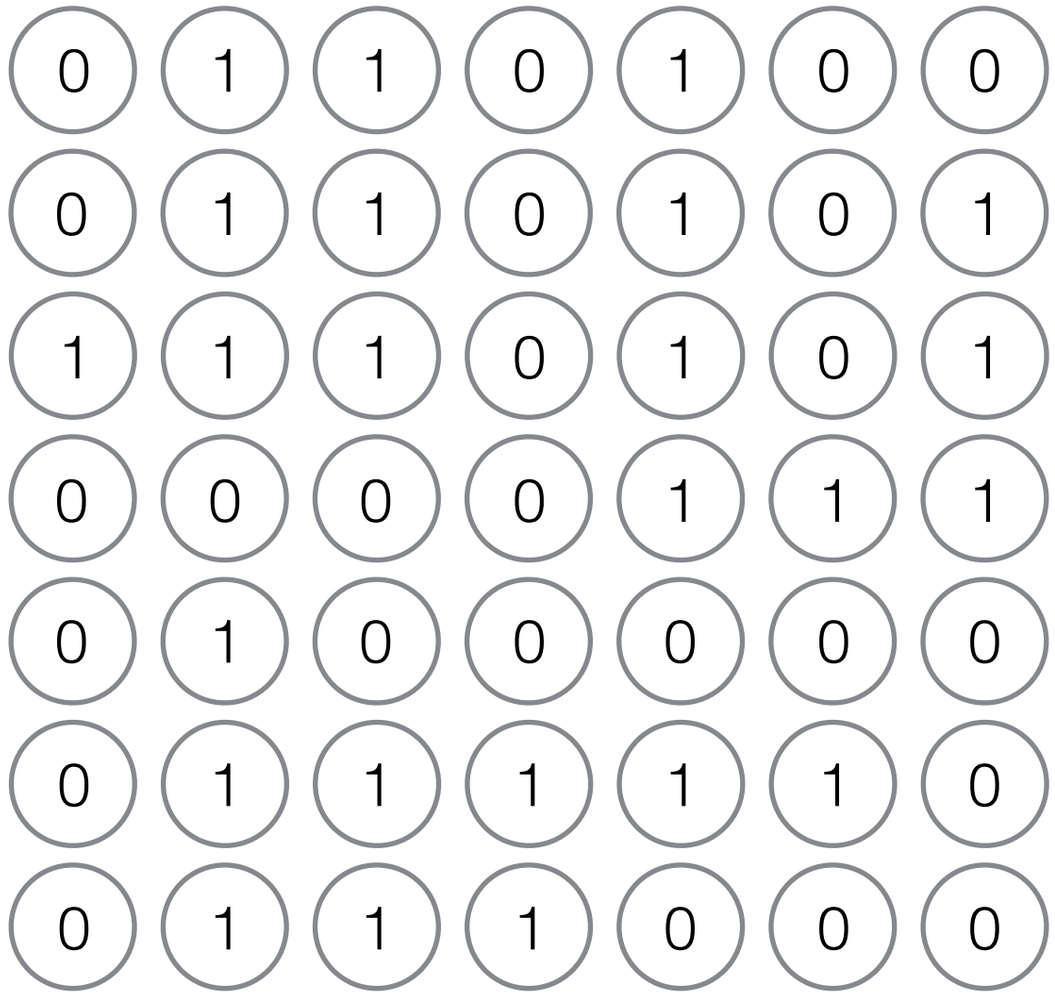
수학문제 푸는 것 (특히 증명하는 것)과 비슷하다

코딩하는 것이 어렵다

그래프 모델링

문제로부터 그래프를 만들어 내야 한다

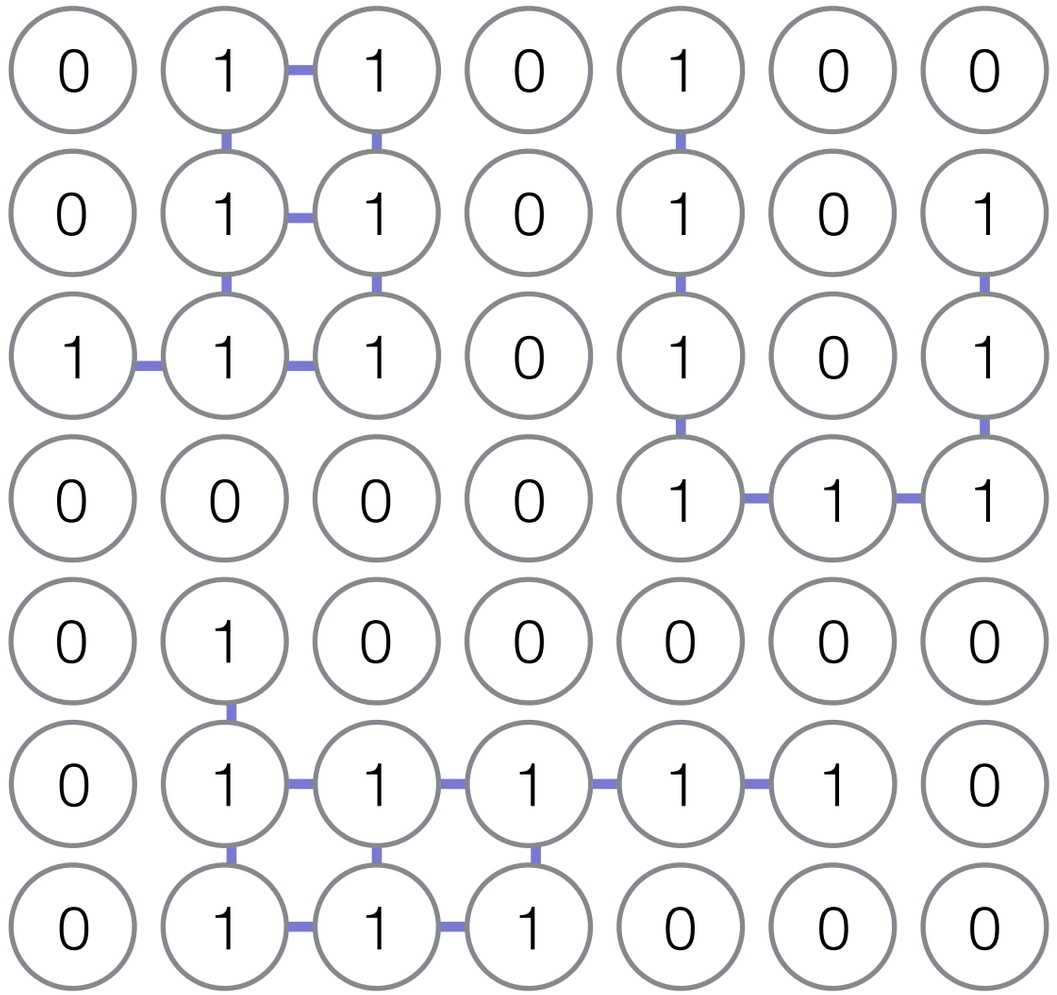
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



그래프 모델링

문제로부터 그래프를 만들어 내야 한다

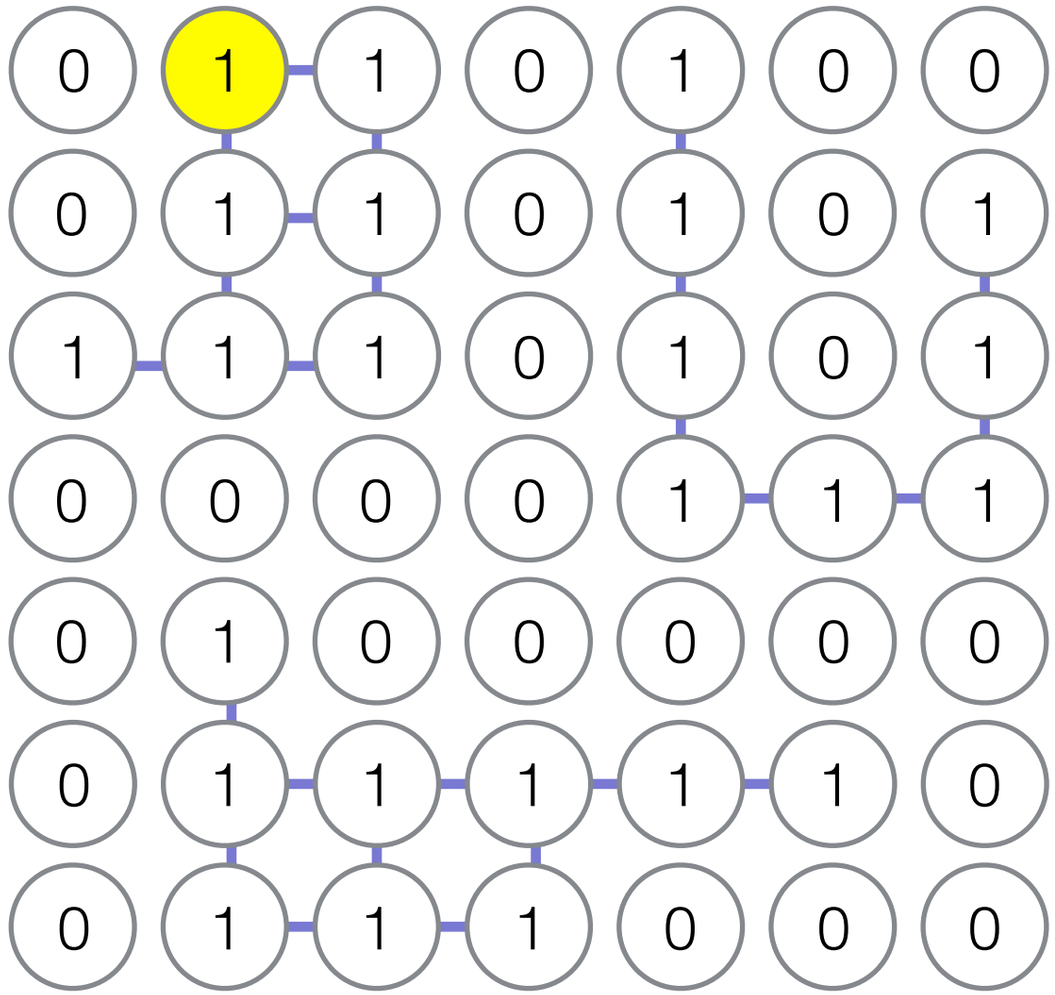
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



그래프 모델링

문제로부터 그래프를 만들어 내야 한다

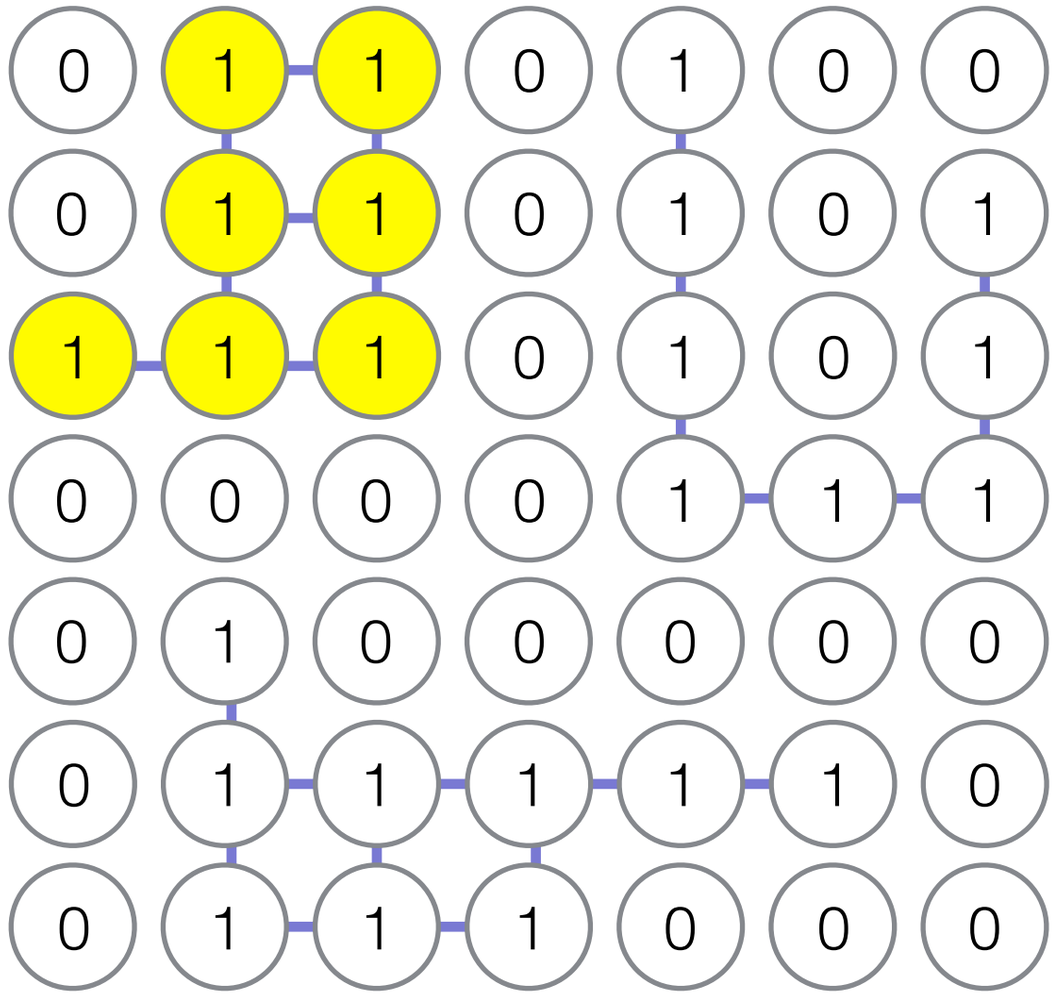
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



그래프 모델링

문제로부터 그래프를 만들어 내야 한다

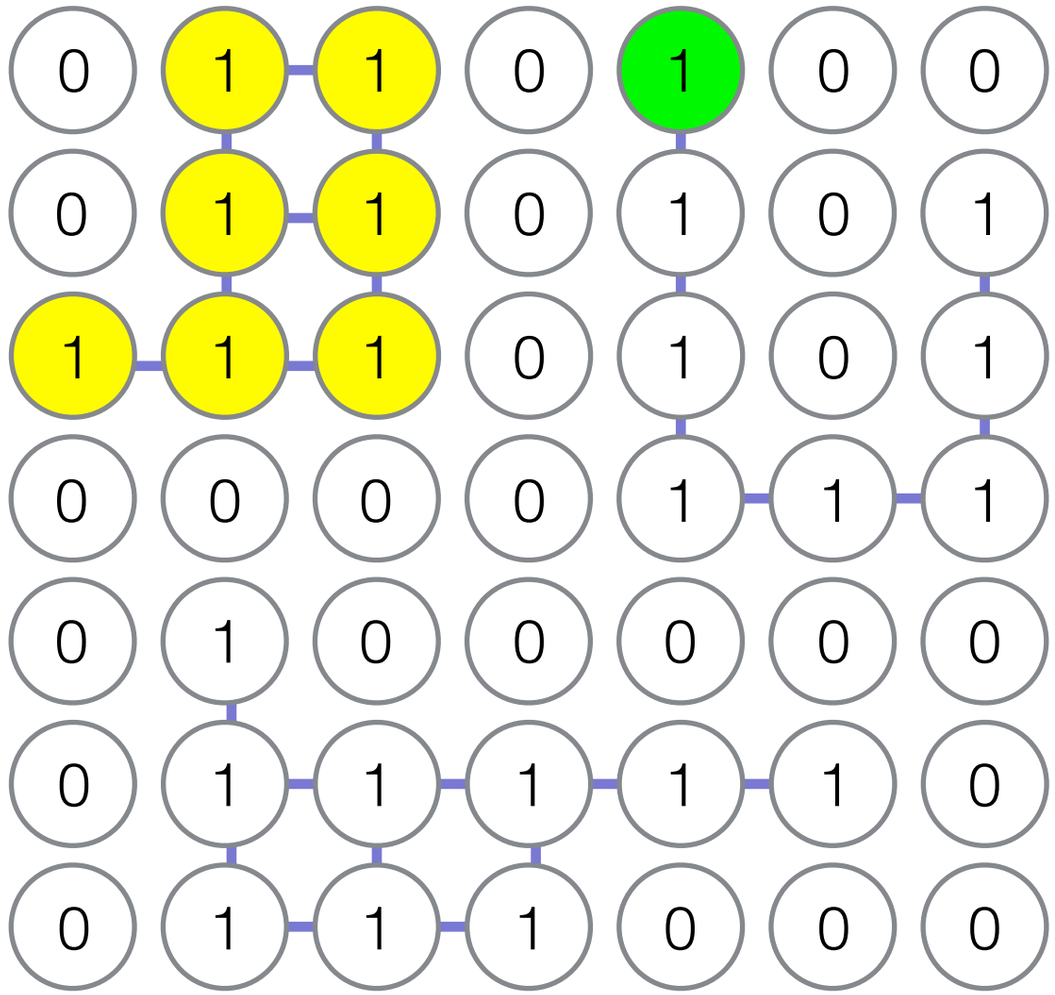
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



그래프 모델링

문제로부터 그래프를 만들어 내야 한다

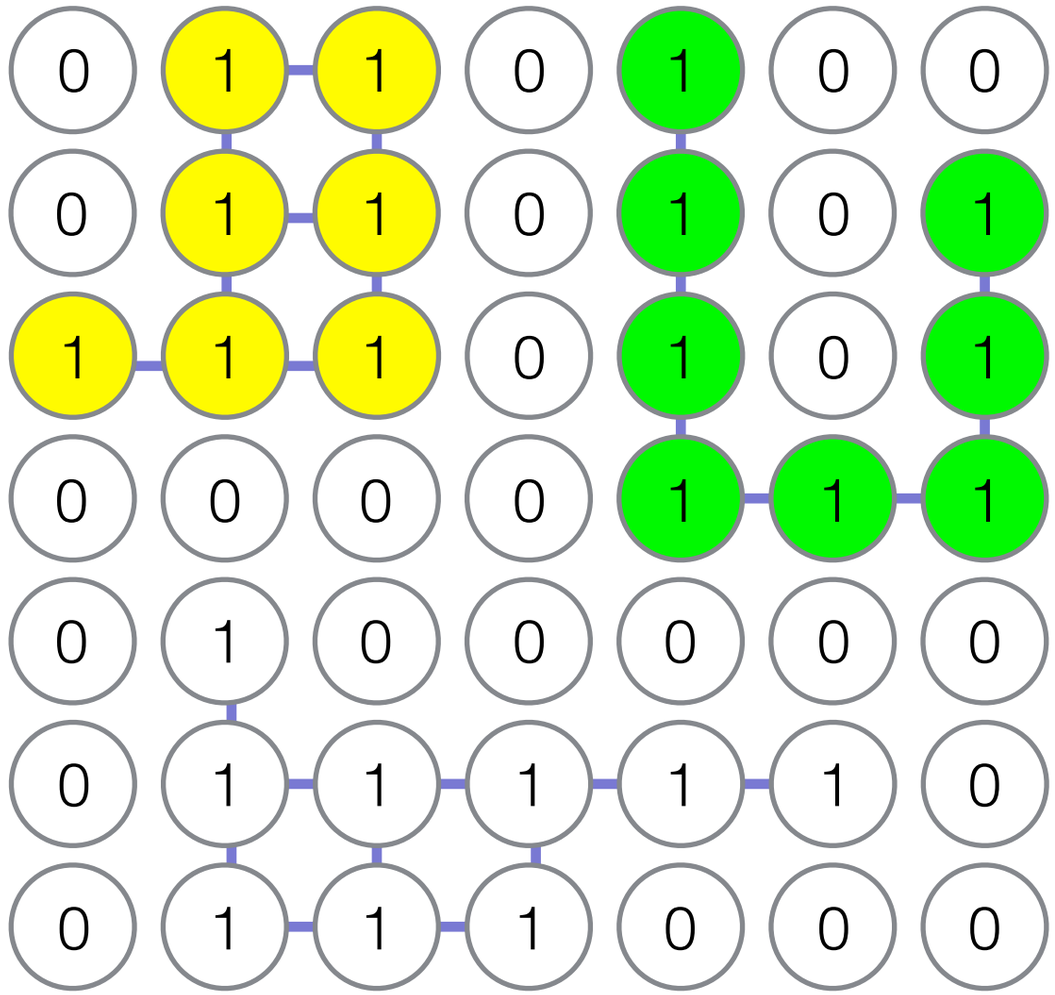
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



그래프 모델링

문제로부터 그래프를 만들어 내야 한다

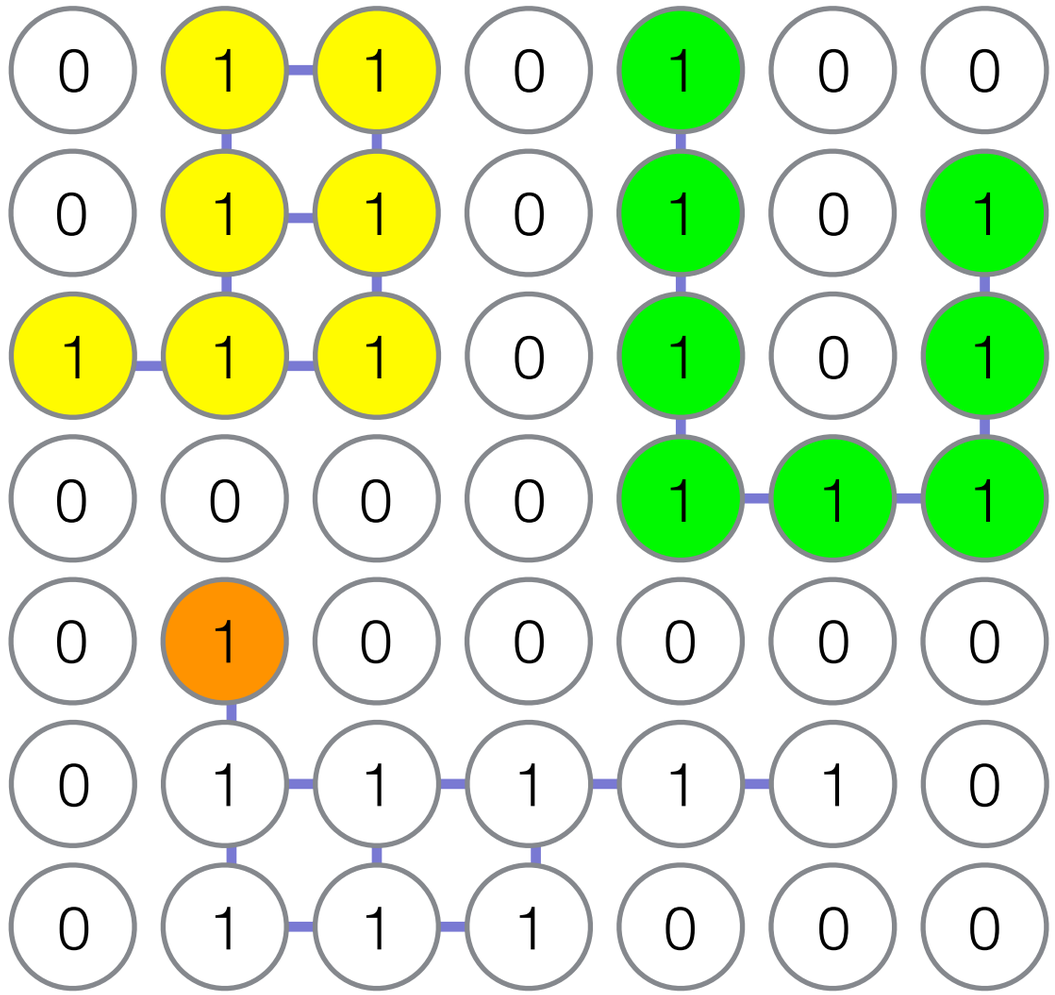
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



그래프 모델링

문제로부터 그래프를 만들어 내야 한다

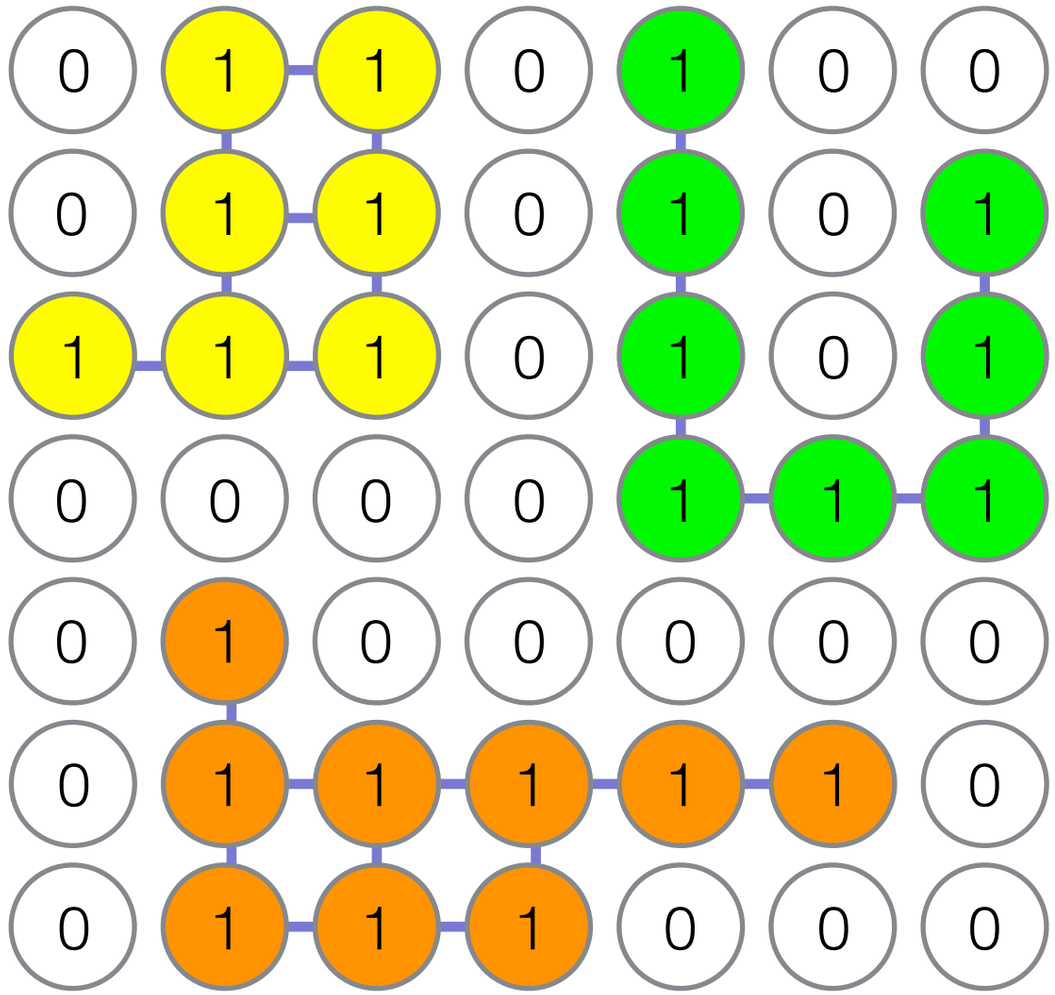
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0



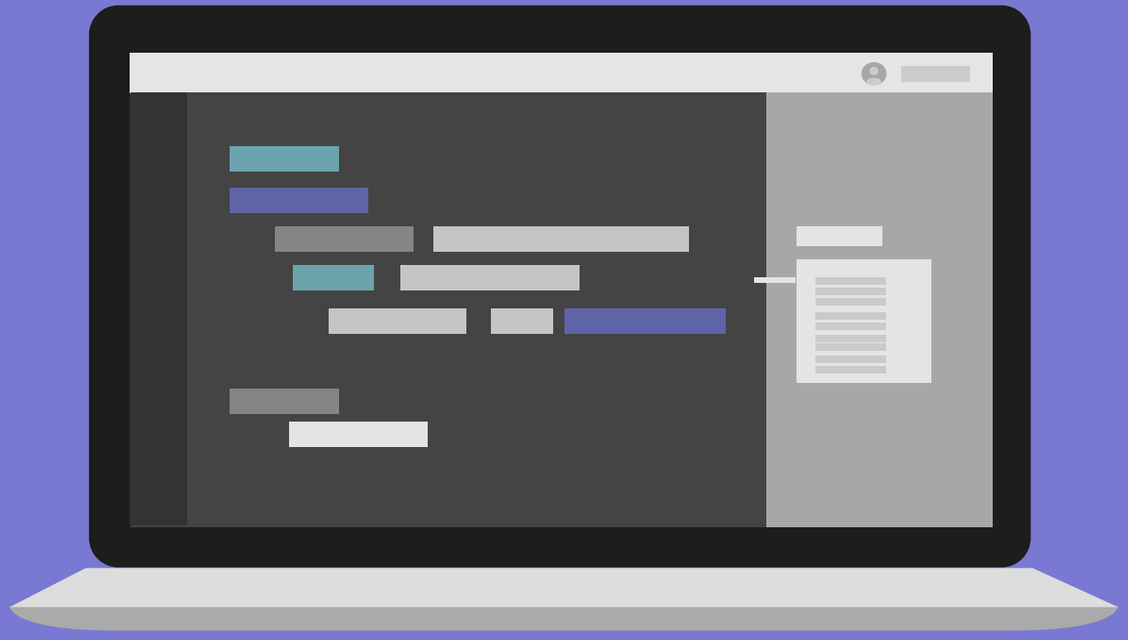
그래프 모델링

문제로부터 그래프를 만들어 내야 한다

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



[문제 2] 유치원 소풍



그래프는 왜 중요한가 ?

현실 세계의 많은 것들을 그래프로 나타낼 수 있다
즉, 그래프와 관련된 문제가 매우 많다

그래프와 관련된 수학적 정리가 매우 많다
그래프 이론이라는 분야가 따로 있다 (Graph theory)

어렵다(...)
그래프와 관련된 이론도 어렵고 구현도 어렵고...ㅠㅠ

그래프 알고리즘 배워봤다 하려면

현실 세계의 많은 것들을 그래프로 나타낼 수 있다

그래프 알고리즘 배워봤다 하려면

그래프 순회 (BFS, DFS)

최단경로 알고리즘 (Dijkstra, Floyd)

최소신장트리 구하기 (Prim, Kruskal)

그래프 알고리즘 좀 안다 하려면

그래프 순회 (BFS, DFS)

최단경로 알고리즘 (Dijkstra, Floyd)

최소신장트리 구하기 (Prim, Kruskal)

그래프 알고리즘 좀 안다 하려면

그래프 순회 (BFS, DFS)

최단경로 알고리즘 (Dijkstra, Floyd)

최소신장트리 구하기 (Prim, Kruskal)

Strongly Connected Component

Maximum Flow (Ford-Fulkerson Algorithm)

Maximum Flow-Min Cut Theorem

그래프 알고리즘의 고수다 하려면

그래프 순회 (BFS, DFS)

최단경로 알고리즘 (Dijkstra, Floyd)

최소신장트리 구하기 (Prim, Kruskal)

Strongly Connected Component

Maximum Flow (Ford-Fulkerson Algorithm)

Maximum Flow-Min Cut Theorem

그래프 알고리즘의 고수다 하려면

그래프 순회 (BFS, DFS)

최단경로 알고리즘 (Dijkstra, Floyd)

최소신장트리 구하기 (Prim, Kruskal)

Strongly Connected Component

Maximum Flow (Ford-Fulkerson Algorithm)

Maximum Flow-Min Cut Theorem

Minimum Independent Set

Maximum Vertex Cover

NP-Completeness

그래프 알고리즘의 고수다 하려면

그래프 순회 (BFS, DFS)

최단경로 알고리즘 (Dijkstra, Floyd)

최소신장트리 구하기 (Prim, Kruskal)

Strongly Connected Component

Maximum Flow (Ford-Fulkerson Algorithm)

Maximum Flow-Min Cut Theorem

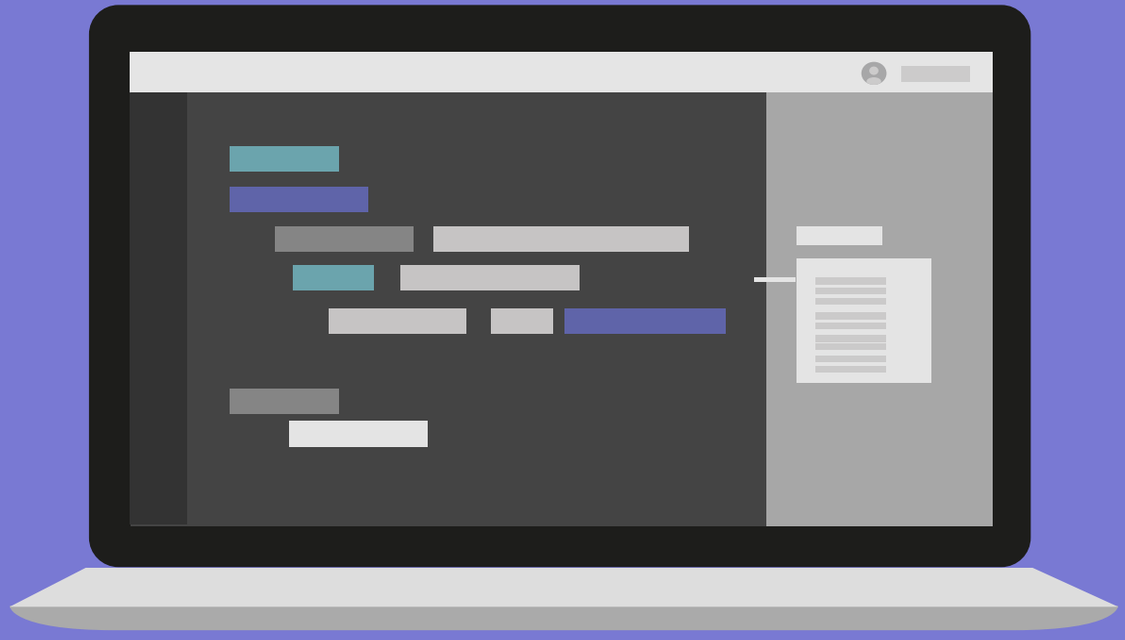
Minimum Independent Set

Maximum Vertex Cover

NP-Completeness

물론 구현까지...

퀴즈 및 설문



/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

blog.naver.com/elicer