Graph Basic

그래프 기초

競技プログラミングの鉄則 – 米田優峻

KPSC Algorithm Study 25/2/27 Thu.

by Haru_101

Graph Basic

- 그래프를 구성하는 방법
- 그래프를 탐색하는 방법
- 그래프 상에서의 최단 거리 알고리즘 – 다익스트라
- 이 3가지를 배워보겠습니다.

Graph Basic

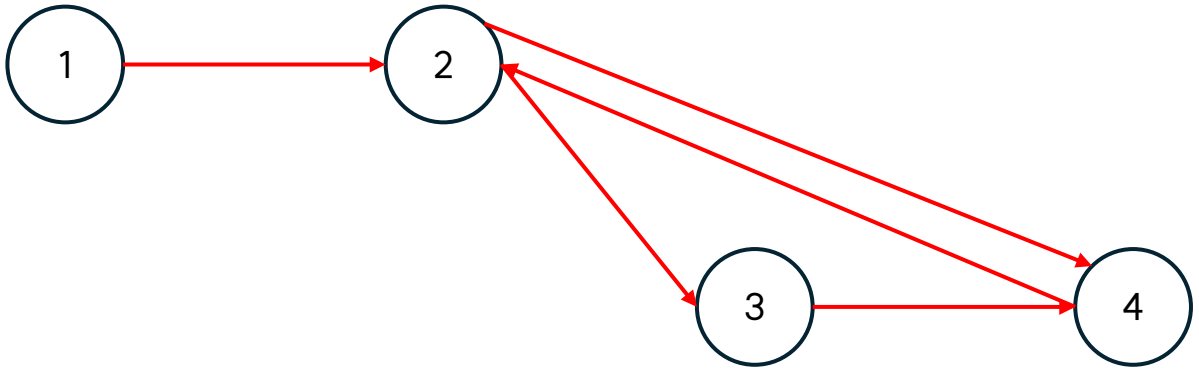
- 그래프의 간선(선분)은 양방향 혹은 단방향으로 다른 두 노드를 연결시켜줍니다.

- 이 그래프를 표현하는 방법은 대표적으로 인접행렬, 인접리스트 방법이 있습니다.

- 인접행렬 방식은 행렬에서 i번 노드에서 j번 노드로 갈 수 있는 간선이 존재하면 i행 j열의 값을 $True$로, 아니라면 $False$의 값을 가지는 행렬을 의미합니다.

Graph Basic

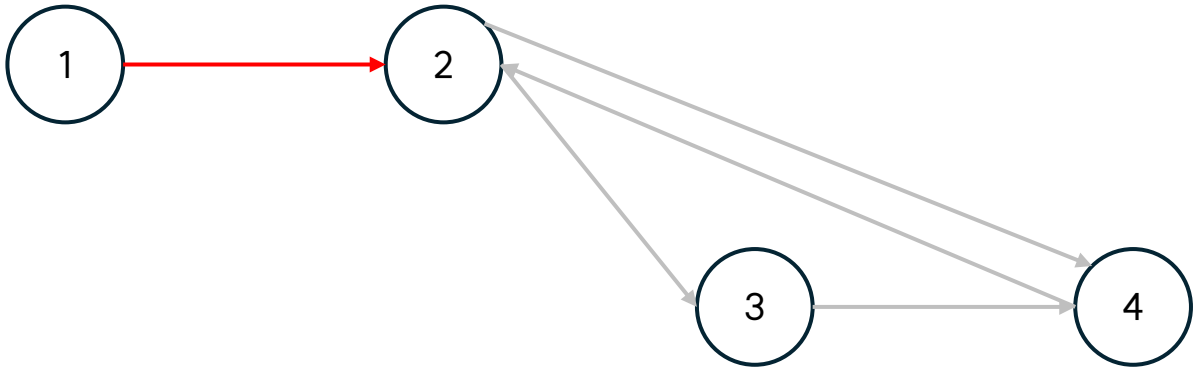
- 아래 그래프를 한번 인접행렬로 나타내봅시다.



i, j	1	2	3	4
1				
2				
3				
4				

Graph Basic

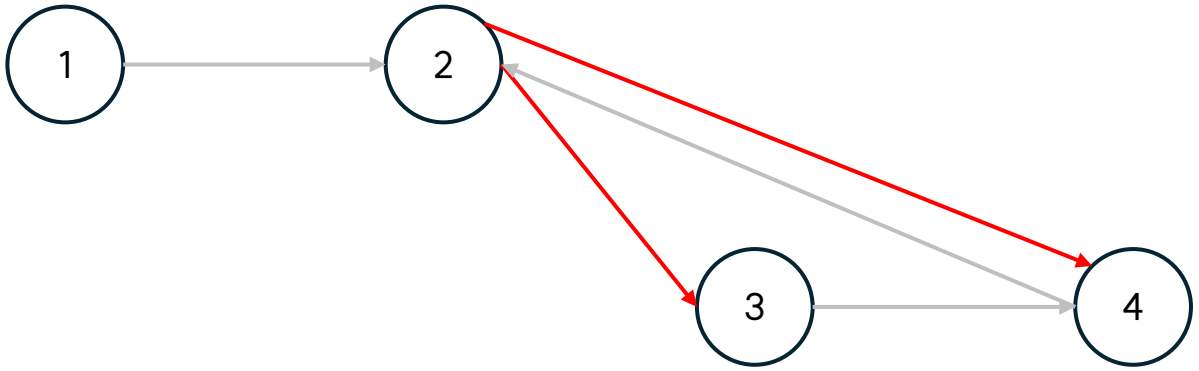
- 아래 그래프를 한번 인접행렬로 나타내봅시다.



i, j	1	2	3	4
1		1		
2				
3				
4				

Graph Basic

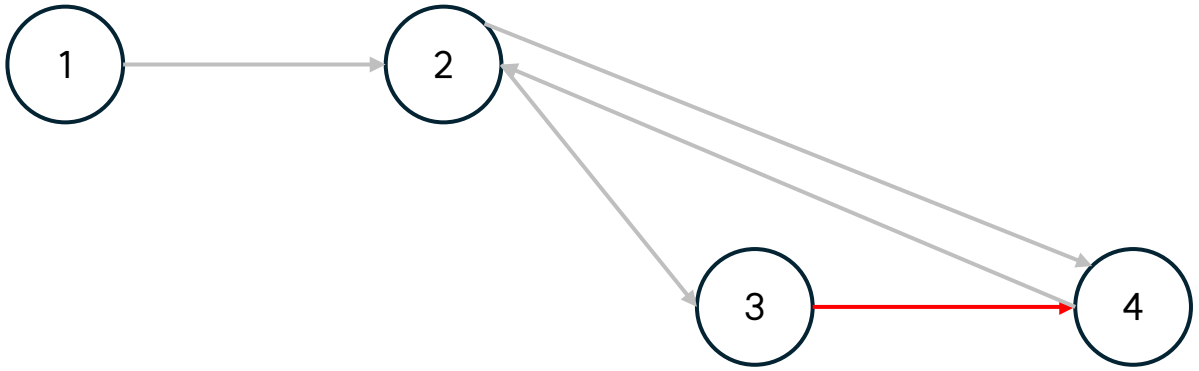
- 아래 그래프를 한번 인접행렬로 나타내봅시다.



i, j	1	2	3	4
1		1		
2			1	1
3				
4				

Graph Basic

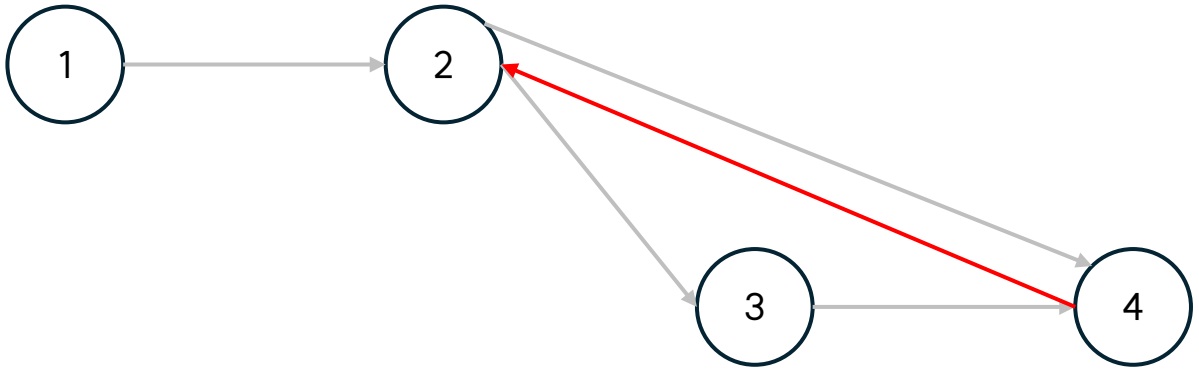
- 아래 그래프를 한번 인접행렬로 나타내봅시다.



i, j	1	2	3	4
1		1		
2			1	1
3				1
4				

Graph Basic

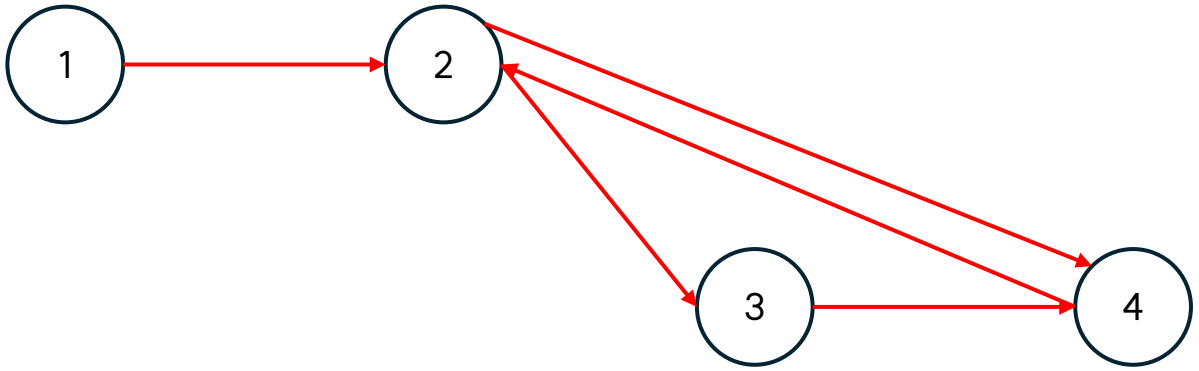
- 아래 그래프를 한번 인접행렬로 나타내봅시다.



i, j	1	2	3	4
1		1		
2			1	1
3				1
4		1		

Graph Basic

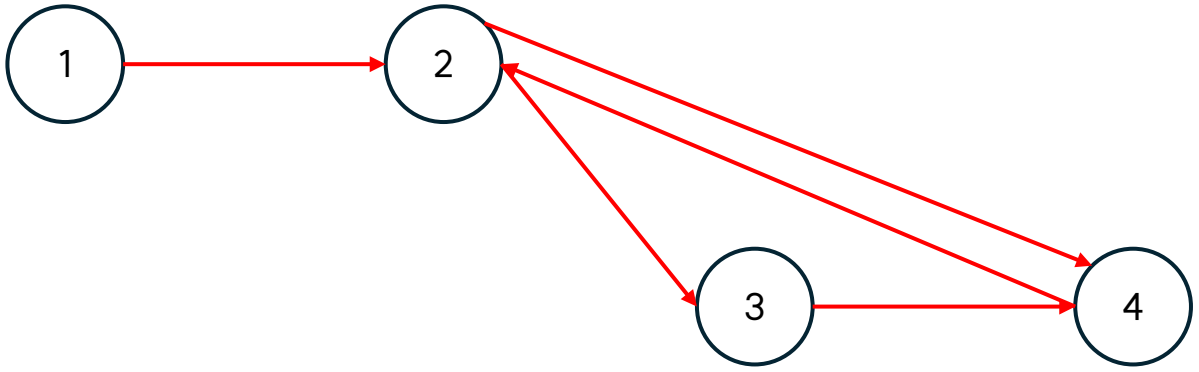
- 아래 그래프를 한번 인접행렬로 나타내봅시다.



i, j	1	2	3	4
1		1		
2			1	1
3				1
4		1		

Graph Basic

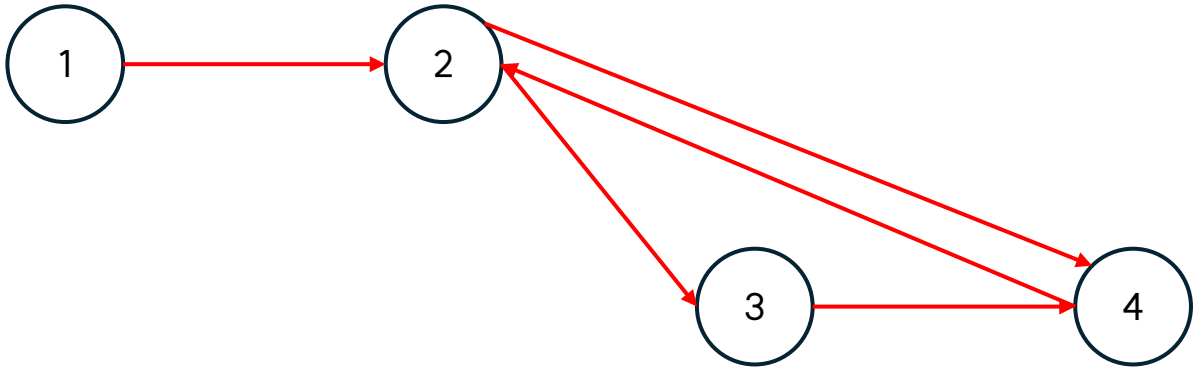
- 인접행렬의 장점은 $A[i][j]$가 1인지(True) 검사하면 해당 간선을 통해 이동할 수 있음을 한번에 알기 쉽습니다.
- 하지만, 노드의 개수가 10^5개와 같은 개수를 가진다면 $(10^5 \times 10^5)$크기의 배열을 만들어야 하므로 메모리 초과가 나기 쉽습니다.



i, j	1	2	3	4
1		1		
2			1	1
3				1
4		1		

Graph Basic

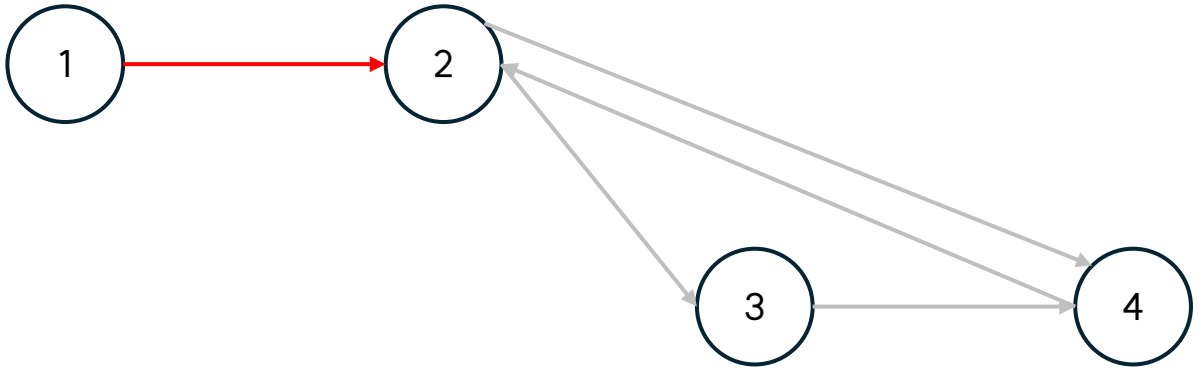
- 이제 인접리스트를 활용한 그래프 표현을 알아보겠습니다.
- 먼저, 각 정점에 대한 크기가 유동적인 벡터를 만듭니다.



i				
1				
2				
3				
4				

Graph Basic

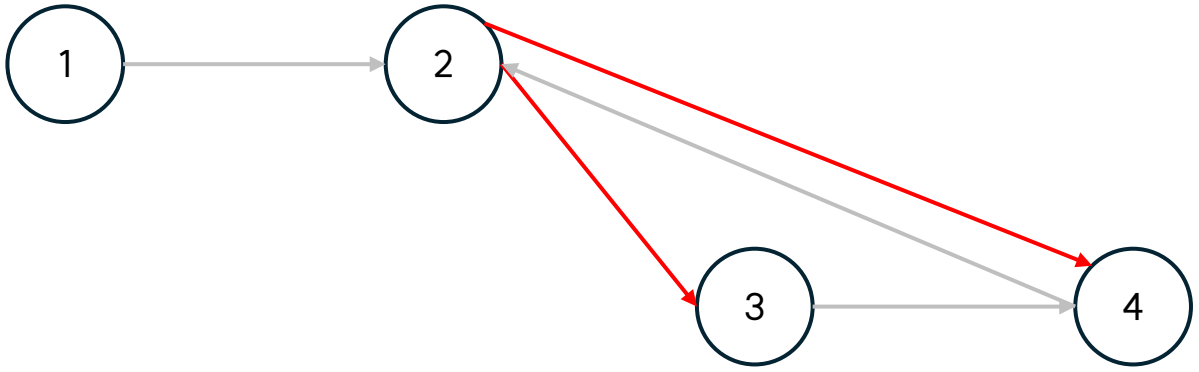
- 그 다음부터는, 각 정점에 대해 연결되어 있는 정점에 대해 번호를 해당 정점 벡터에 넣습니다.



i				
1	2			
2				
3				
4				

Graph Basic

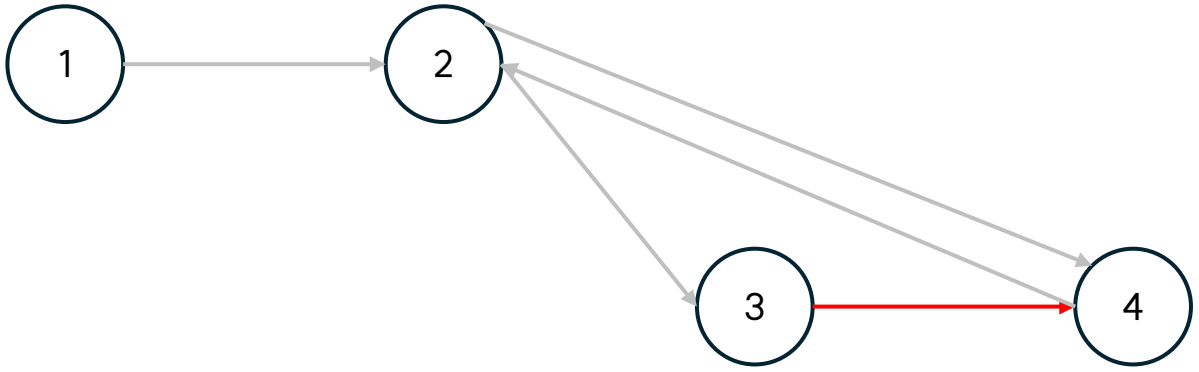
- 그 다음부터는, 각 정점에 대해 연결되어 있는 정점에 대해 번호를 해당 정점 벡터에 넣습니다.



i				
1	2			
2	3	4		
3				
4				

Graph Basic

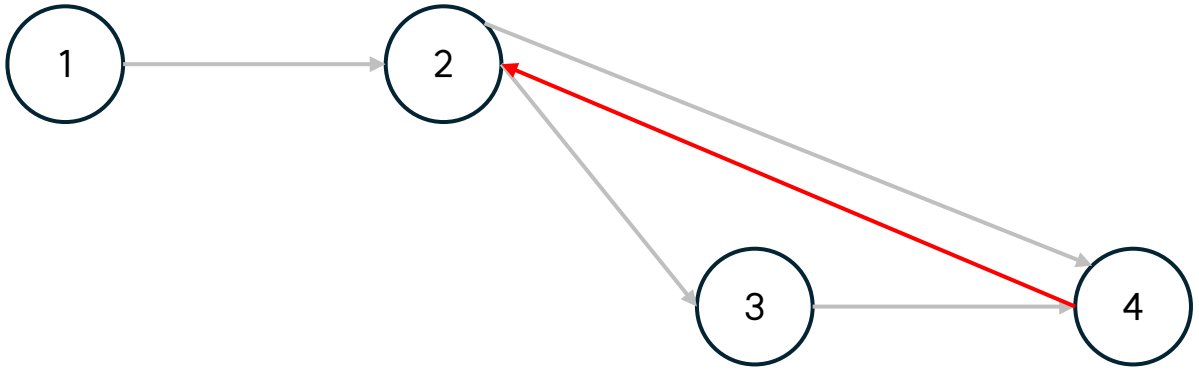
- 그 다음부터는, 각 정점에 대해 연결되어 있는 정점에 대해 번호를 해당 정점 벡터에 넣습니다.



i				
1	2			
2	3	4		
3	4			
4				

Graph Basic

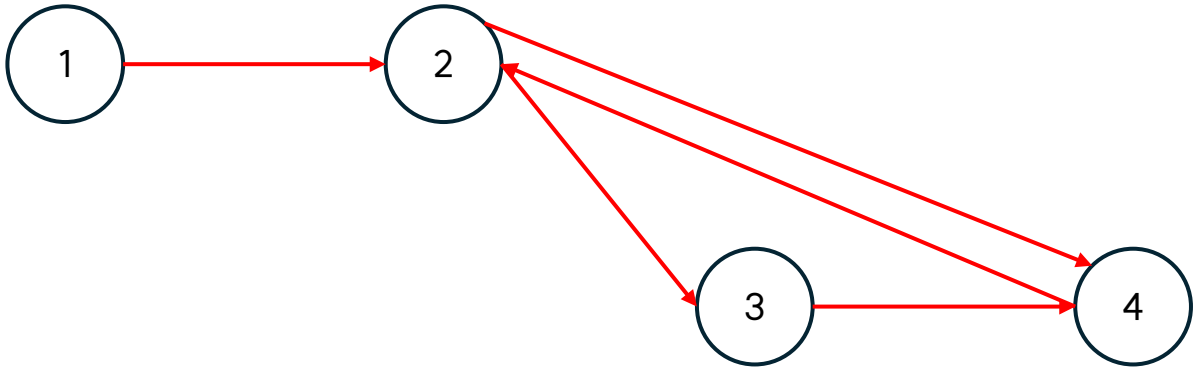
- 그 다음부터는, 각 정점에 대해 연결되어 있는 정점에 대해 번호를 해당 정점 벡터에 넣습니다.



i				
1	2			
2	3	4		
3	4			
4	2			

Graph Basic

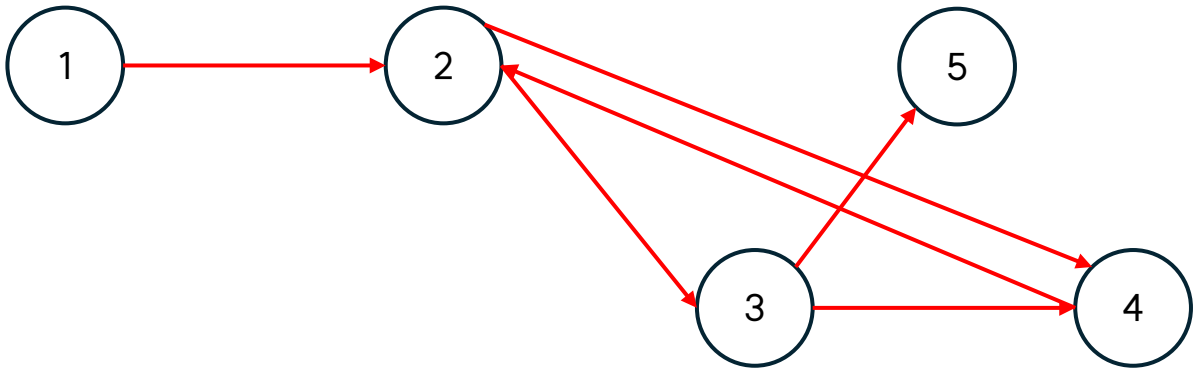
- 이렇게 하면 직접적으로 $A[i][j]$가 1인지 알 수는 없지만, 메모리 용량이 절약되는 장점이 있습니다.



i				
1	2			
2	3	4		
3	4			
4	2			

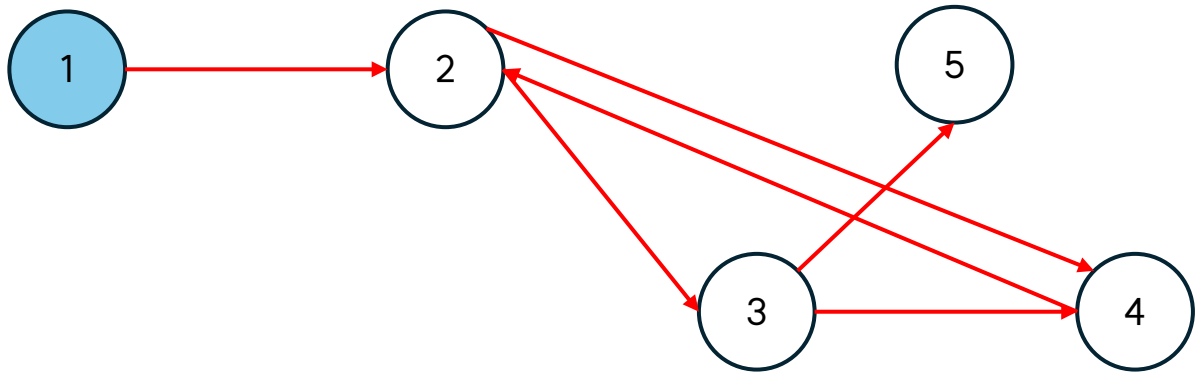
Graph Basic

- 이제 그래프를 순회하는 방법을 알아보겠습니다.
- 대표적으로는 깊이 우선 탐색(DFS), 너비 우선 탐색(BFS)가 있습니다.
- 먼저 DFS의 작동 방식에 대해 알아봅시다. (정점 1에서 시작한다고 가정합시다)



Graph Basic

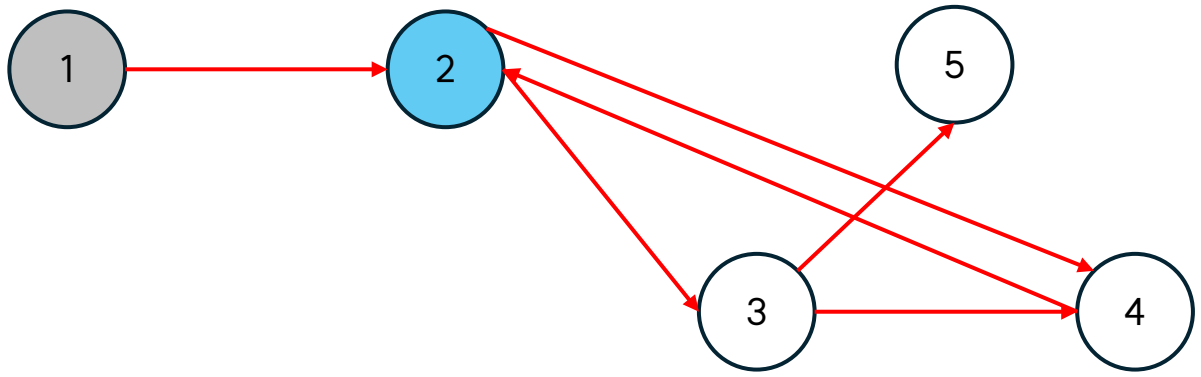
- 먼저 시작 정점이 1이니 현재 정점은 1이라고 합시다.
- 그 다음, 현재 정점에 대해 방문을 했다고 visited 배열에 표시합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있는 정점에 대해 먼저 등장한 정점으로 이동합니다. (예시에서는 2)



Visited	1	2	3	4	5
T/F	T				

Graph Basic

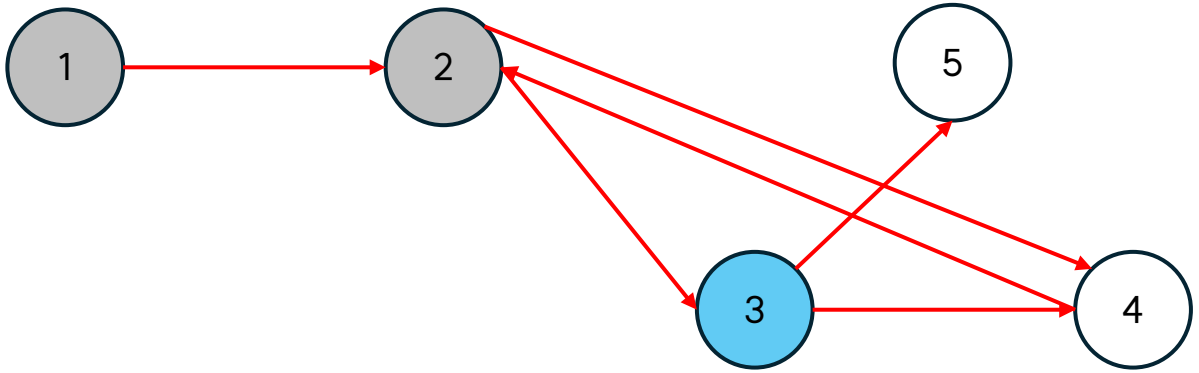
- 그 다음, 현재 정점에 대해 방문을 했다고 visited 배열에 표시합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있는 정점에 대해 먼저 등장한 정점으로 이동합니다. (예시에서는 3)



Visited	1	2	3	4	5
T/F	T	T			

Graph Basic

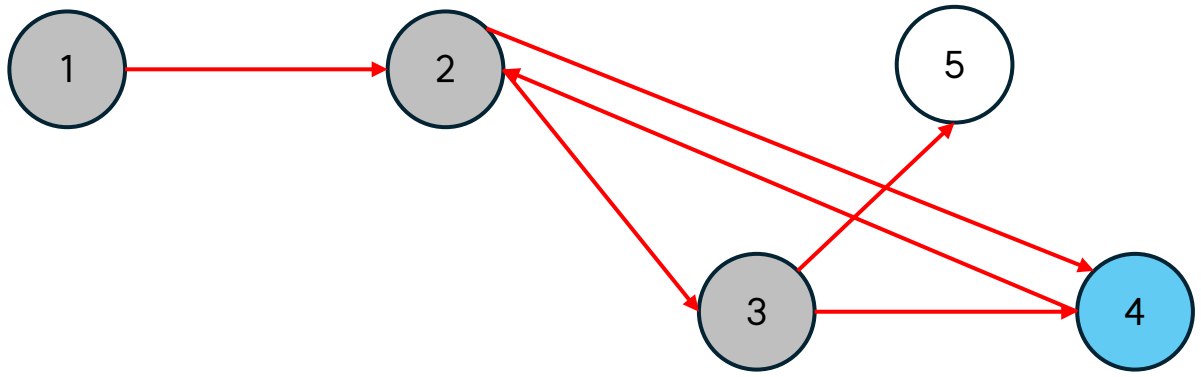
- 그 다음, 현재 정점에 대해 방문을 했다고 visited 배열에 표시합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있는 정점에 대해 먼저 등장한 정점으로 이동합니다. (예시에서는 4)



Visited	1	2	3	4	5
T/F	T	T	T		

Graph Basic

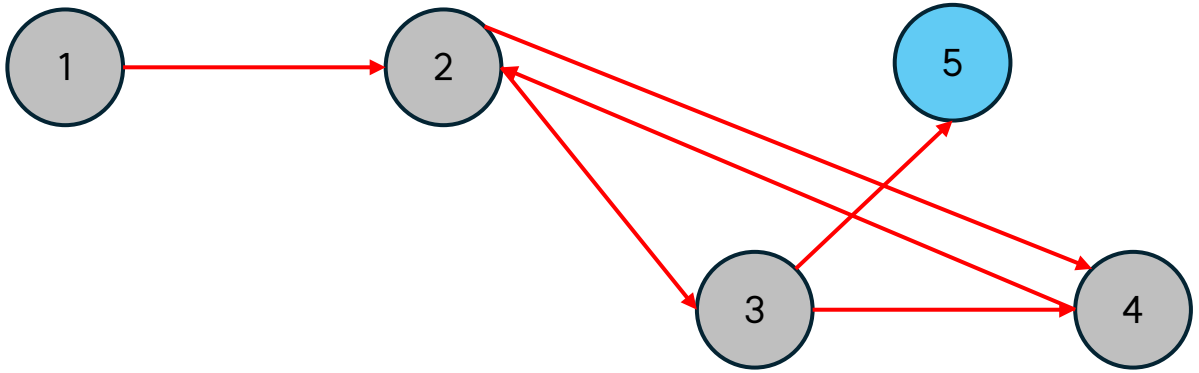
- 그 다음, 현재 정점에 대해 방문을 했다고 visited 배열에 표시합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있는 정점은 없으나, 3에서 5로 갈 수 있기 때문에 3에서 갈 수 있는 다음 정점인 5로 이동합니다.



Visited	1	2	3	4	5
T/F	T	T	T	T	

Graph Basic

- 그 다음, 현재 정점에 대해 방문을 했다고 visited 배열에 표시합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 이렇게 DFS 순회가 끝났습니다. 방문순서는 1->2->3->4->5 가 됩니다. (구현마다 다를 수 있음)

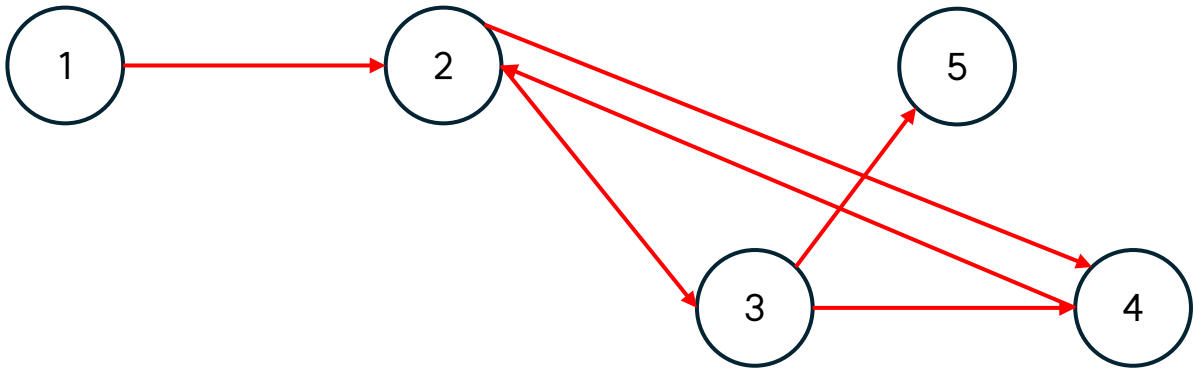


Visited	1	2	3	4	5
T/F	T	T	T	T	T

Graph Basic

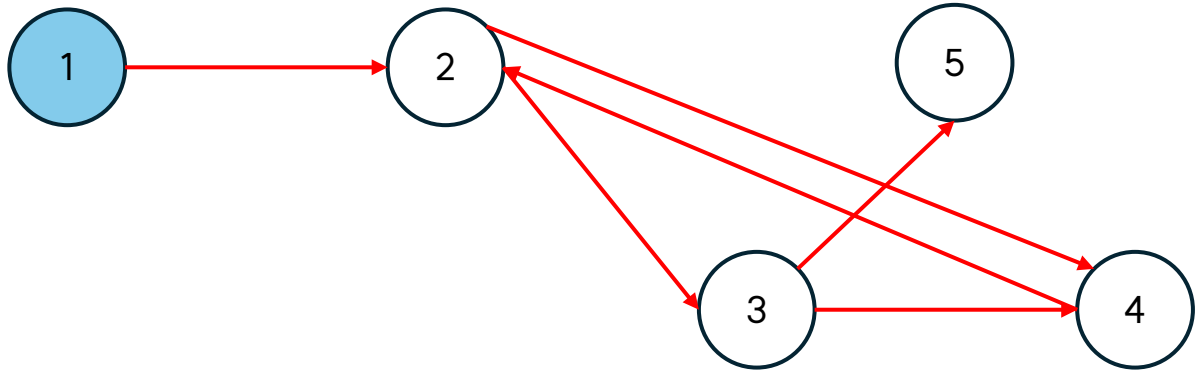
- 이제 BFS로 순회하는 방법을 알아봅시다.

- 먼저 queue를 준비합니다. 이 queue는 '처리해야할 정점'을 담아놓은 queue라고 보시면 됩니다.



Graph Basic

- 먼저 시작 정점이 1이니 현재 정점은 1이라고 합시다.
- 그 다음, 현재 정점에 대해 방문을 했다고 visited 배열에 표시합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있으며 아직 방문하지 않은 모든 정점에 대해 queue에 넣습니다. (현재 예시에서는 2)

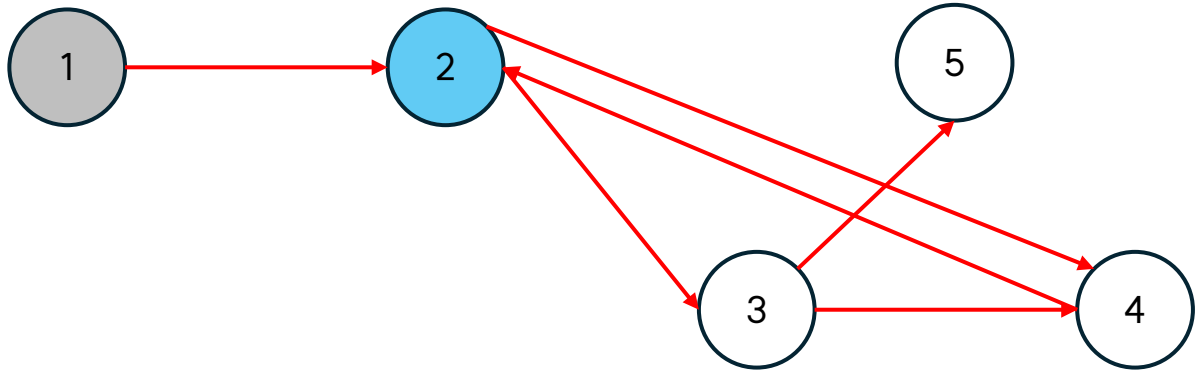


Visited	1	2	3	4	5
T/F	T				

queue	2				

Graph Basic

- 그 다음, queue의 가장 맨 앞에 있는 정점을 현재 정점으로 정하고, visited를 True로 합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있으며 아직 방문하지 않은 모든 정점에 대해 queue에 넣습니다. (현재 예시에서는 3, 4)

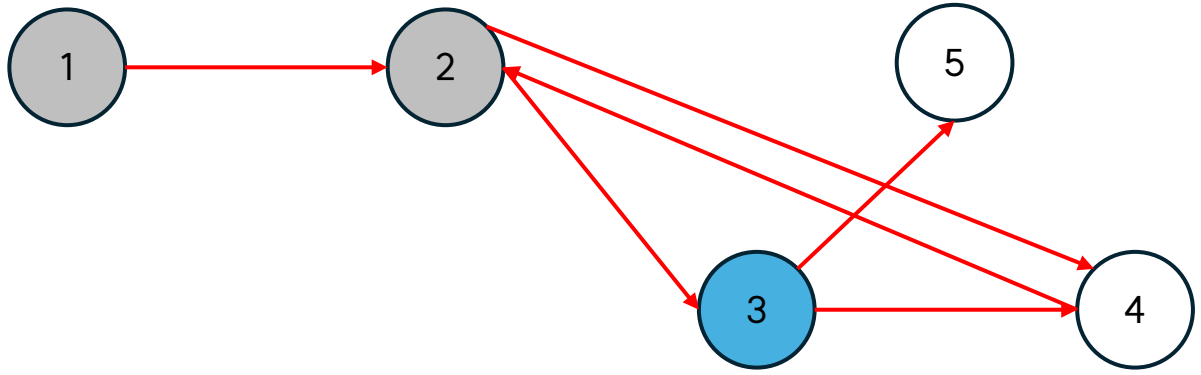


Visited	1	2	3	4	5
T/F	T	T			

queue	3	4			

Graph Basic

- 그 다음, queue의 가장 맨 앞에 있는 정점을 현재 정점으로 정하고, visited를 True로 합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있으며 아직 방문하지 않은 모든 정점에 대해 queue에 넣습니다. (현재 예시에서는 4, 5)

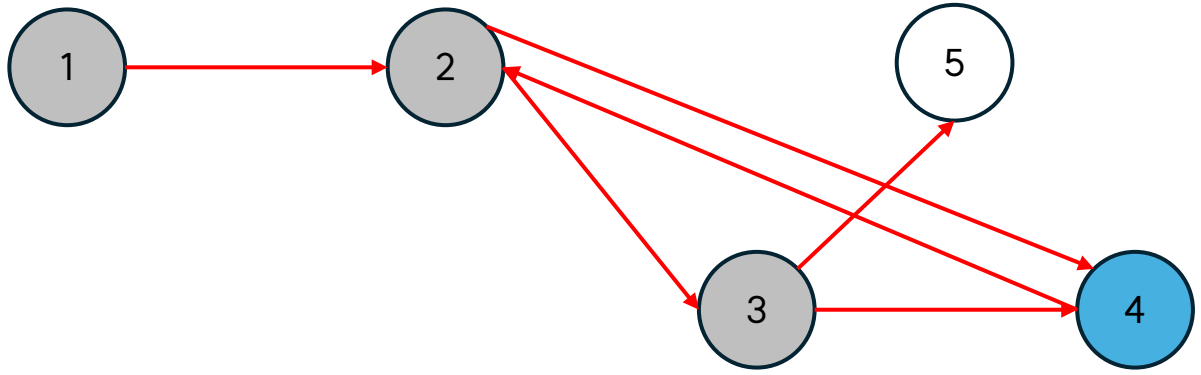


Visited	1	2	3	4	5
T/F	T	T	T		

queue	4	4	5		

Graph Basic

- 그 다음, queue의 가장 맨 앞에 있는 정점을 현재 정점으로 정하고, visited를 True로 합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있으며 아직 방문하지 않은 모든 정점에 대해 queue에 넣습니다. (현재 예시에서는 없음)

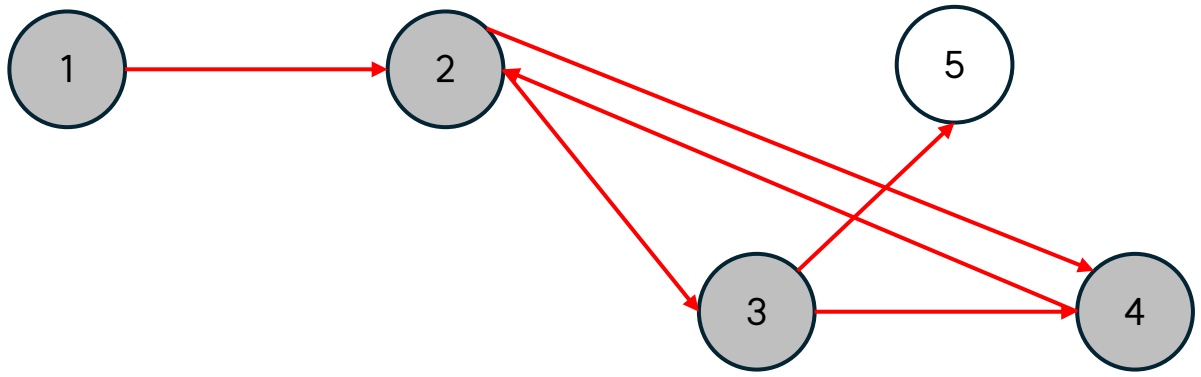


Visited	1	2	3	4	5
T/F	T	T	T	T	

queue	4	5			

Graph Basic

- 그 다음, queue의 가장 맨 앞에 있는 정점을 현재 정점으로 정하기 전에, 이미 visited[4]가 True이므로 그냥 queue에서 제거합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)

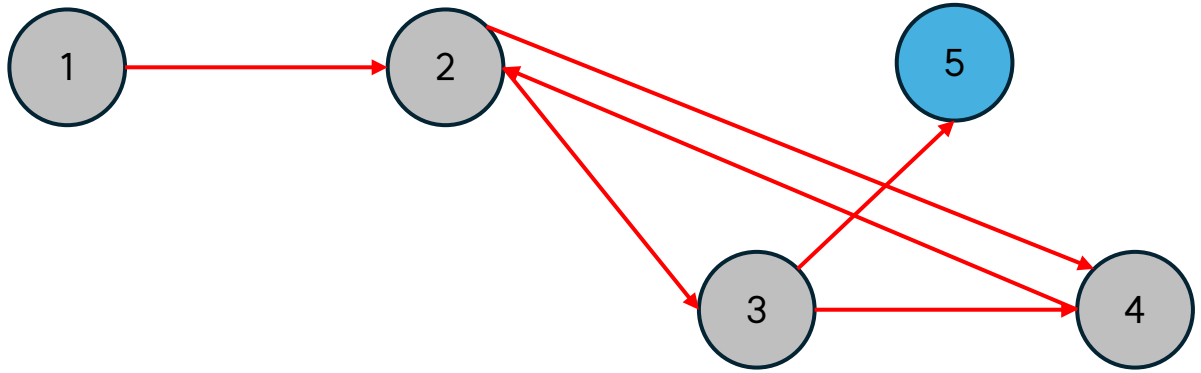


Visited	1	2	3	4	5
T/F	T	T	T	T	

queue	5				

Graph Basic

- 그 다음, queue의 가장 맨 앞에 있는 정점을 현재 정점으로 정하고, visited를 True로 합니다.
  - visited는 True면 방문했음을, False면 방문하지 않았음을 의미합니다. (빈칸은 False)
- 그 다음, 현재 정점에서 갈 수 있으며 아직 방문하지 않은 모든 정점에 대해 queue에 넣습니다. (현재 예시에서는 없음)
- 이렇게 BFS 순회가 끝났습니다.



Visited	1	2	3	4	5
T/F	T	T	T	T	**T**

queue					

KPSC
KMU PROBLEM SOLVING CLUB

Graph Basic

끝.

그동안 수고하셨습니다.