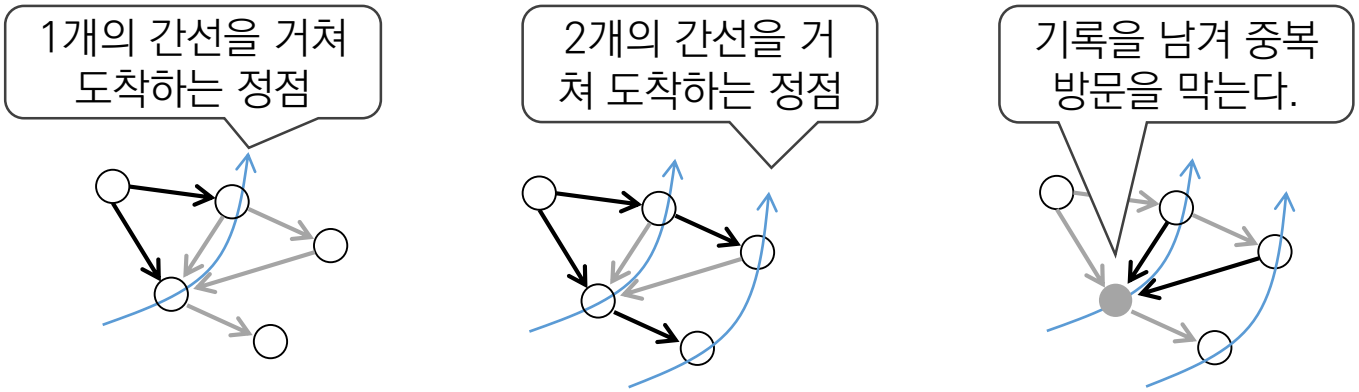
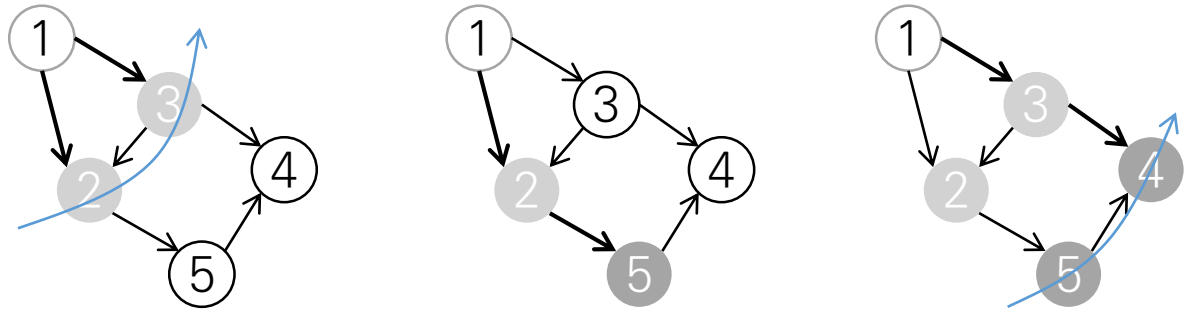
- BFS (너비 우선 탐색)
  - 시작 정점부터 거쳐가는 간선의 수가 같은 순서로 탐색하는 방식.



  - 시작에서 n개의 간선을 지나 도착하는 정점의 인접 정점은 n+1개의 간선을 지나 도착하게 됨.
  - 거리가 n인 정점들을 처리할 때 n+1인 인접 정점들을 저장함.
  - 거리 n인 정점들을 처리하면, 저장해둔 n+1 정점들을 꺼내 처리함.

- BFS (너비 우선 탐색)
  - 시작 정점부터 거쳐가는 간선의 수가 같은 순서로 방문.



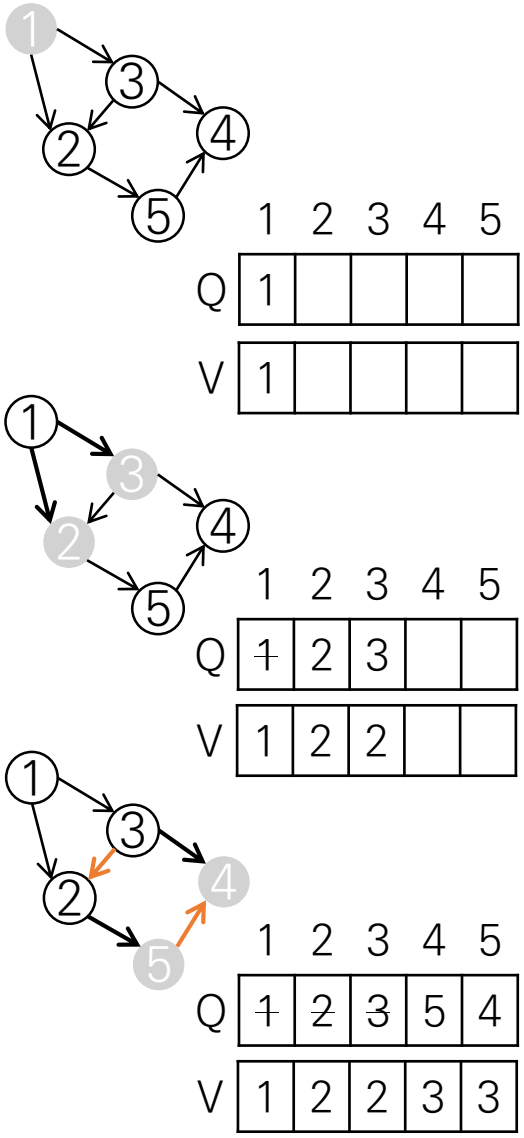
방문 순서 : 1 – 2 – 3 – 5 – 4

  - 시작 정점으로부터 거리가 같은 정점들을 큐에 저장.
  - 그 정점들의 인접 정점 끼리도 거리가 같음.
    - 인접이면서 이미 방문한 곳은 제외. (3–〉2에서 2는 이미 방문)
  - 인접 노드를 그 다음 순으로 처리되도록 큐에 저장함.

- BFS

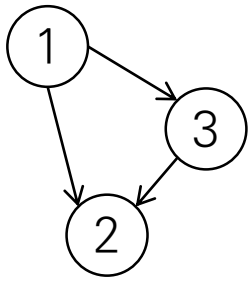
```
BFS(s)
    enQ(s)                  // 시작점 enqueue
    V[s] = 1                // 방문 표시
    while( is_not_emptyQ())
        n = deQ()
        visit(n)            // 노드에 대해 처리할 일
        for i : 1 -> N
            // i가 인접이면서 방문안한 노드면
            if( A[n][i] == 1 && V[i] == 0 )
                enQ(i)
                V[i] = V[n] + 1
```

'V[i] = 1' 대신 'V[i] = V[n] + 1'로 표시하면 인접한 정점으로 부터의 거리를 알 수 있다.



위상 정렬

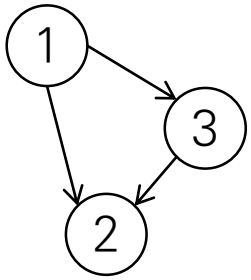
- 앞의 1, 3이 처리되어야 2번을 처리할 수 있는 경우.
  - 정점의 진입 차수를 활용.
  - 진입 차수가 0인 정점부터 시작.
  - 정점을 처리할 때 인접 정정에 처리되었음을 알림.
    - 인접 정점의 진입 차수를 하나 줄임.
    - 진입 차수가 0이 되면 다음 번에 처리할 차례가 됨.



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

I	0	2	1

A : 인접행렬
I : 진입 차수



A : 인접행렬
I : 진입 차수

```
Sort( )
    for i : 0 -> N          // 진입차수가 0이면 enQ
        if( I[i] == 0 )
            enQ(i)
    while(is_not_emptyQ())
        n = deQ( )
        do(n)               // 노드 n에서 해야할 일
        for i : 1 -> N
            if( A[n][i] == 1 )
                I[i]--    // n의 인접노드 진입차수 감소
                if( I[i] == 0 )  // 진입차수가 0이면 enQ
                    enQ(i)
```

Q	1							
Q	~~1~~	3						
Q	~~1~~	~~3~~	2					
Q	~~1~~	~~3~~	~~2~~					

I	0	2	1
I	0	2->1	1->0
I	0	1->0	0
I	0	0	0