
임계경로
lhd

임계경로

- 참조 사이트

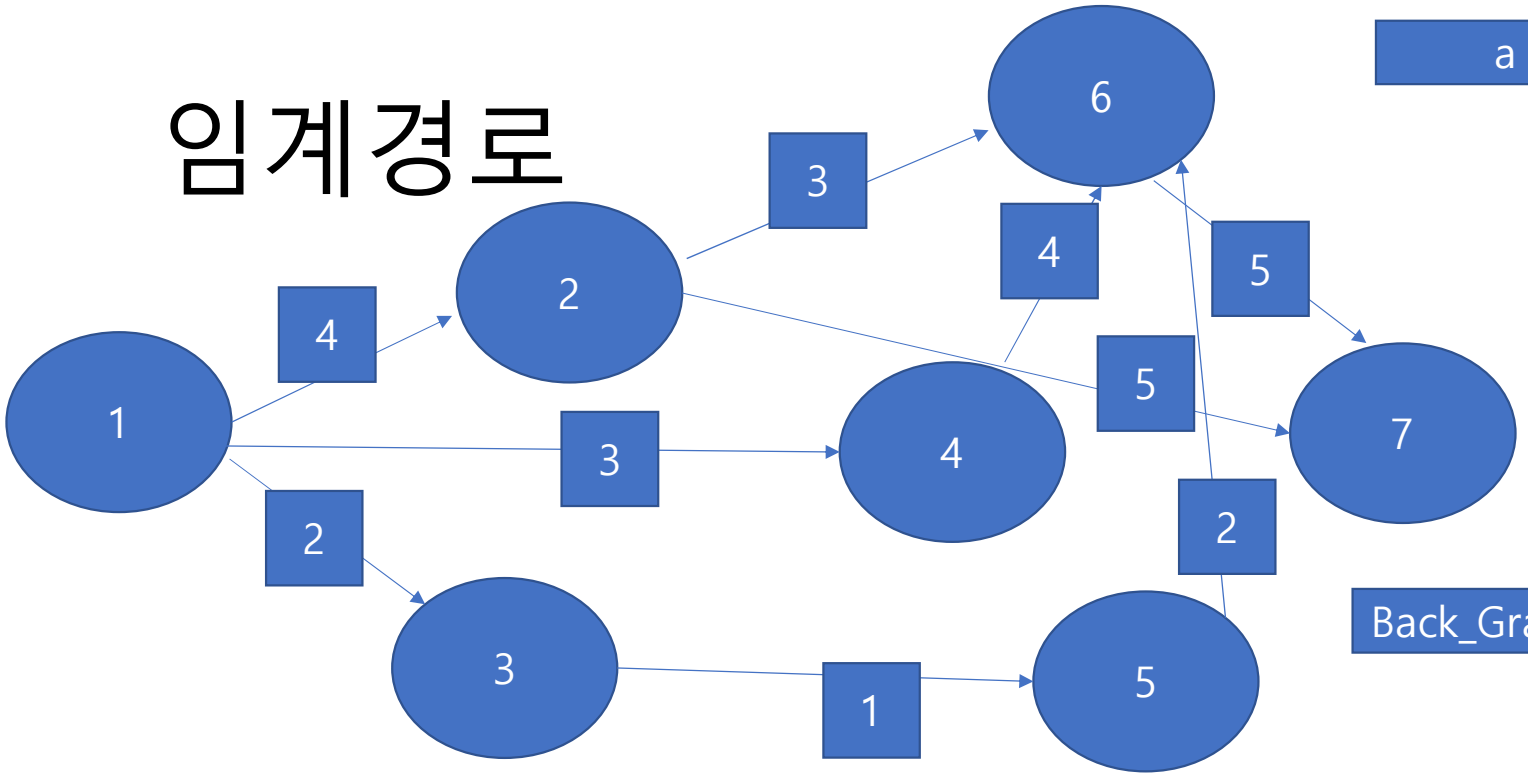
위상정렬

- 사이클이 발생하면 위상정렬을 수행할 수 없다.
- 시작점이 존재해야 만 발생한다.
- Stack
- Queue -> 더 많이 사용
- 집입차수는 특정한 노드가 주어졌을 때 다른 노드가 들어노는 개수를 의미한다.

위상정렬

- 진입차수가 0인 정점을 큐에 삽입합니다.
- 큐에서 원소를 꺼내 연결된 모든 간선을 제거합니다.
- 간선 제거 이후에 진입차수가 0이 된 정점을 큐에 삽입합니다.
- 큐가 빌 때 까지 2번 ~ 3번 과정을 반복합니다. 모든 원소를 방문하기 전에 큐가 빈다면 사이클이 존재하는 것이고 , 모든 원소를 방문했다면 큐에서 꺼낸 순서가 위상 정렬의 결과입니다.

임계경로



a

0	
1	(2, 4), (3, 2), (4, 3)
2	(6, 3), (7, 5)
3	(5, 1)
4	(6, 4)
5	(6, 2)
6	(7, 5)
7	

Back_Graph

0	
1	
2	(1, 4)
3	(1, 2)
4	(1, 3)
5	(3, 1)
6	(2, 3), (4, 4), (5, 2)
7	(2, 5), (6, 5)

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

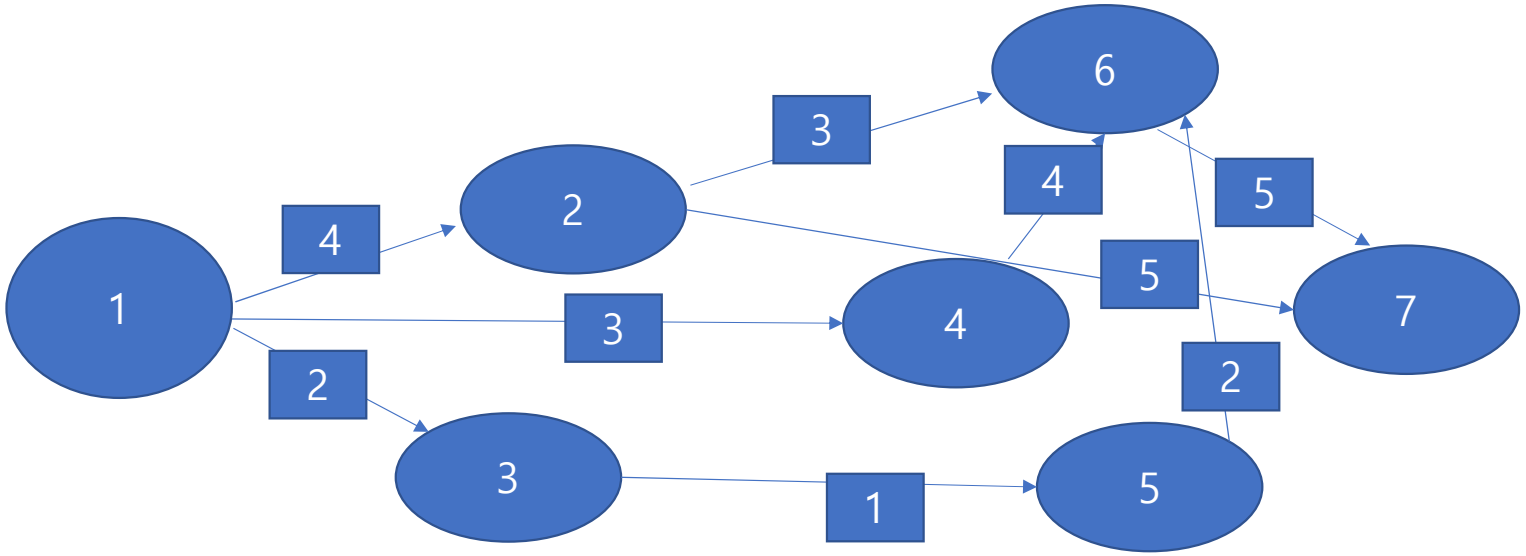
indegree

Node, time

임계경로

```
result = [0 for i in range(MAX+1)]
q = deque()
#시작점 노드를 큐에 삽입합니다.
q.append(start)
# 더이상 방문할 노드가 없을 때 까지
```

- 진입차수가 0인 것을 다 큐에 넣어준다.
- 진입차수가 0인 것이 하나밖에 없다.

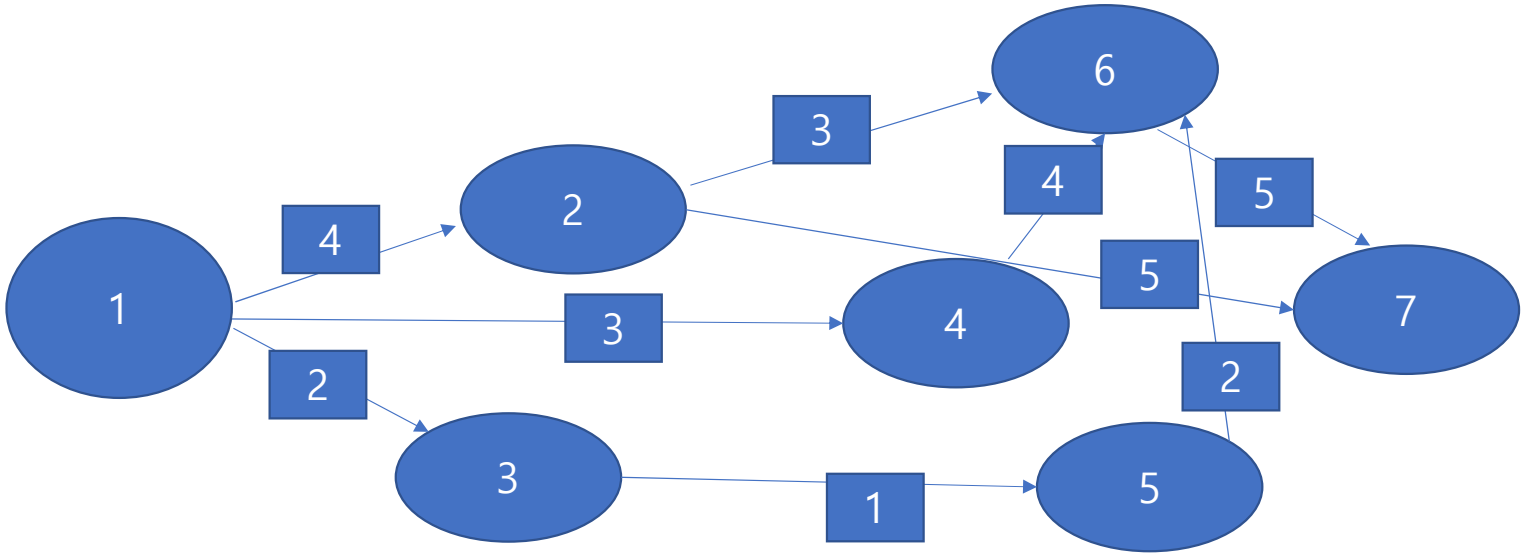


임계경로

- X가 1이고
- a[x] -> (2, 4), (3, 2), (4, 3)

```
while q:
    x = q.popleft()
    for i in range(len(a[x])):
        y = Edge(a[x][i].node, a[x][i].time)
        print("y::", y, result[x])
        if result[y.node] <= y.time + result[x]:
            result[y.node] = y.time + result[x]
        # 새롭게 진입차수가 0이 된 정점을 부여 삼입합니다.
        inDegree[y.node]-= 1
        if inDegree[y.node] == 0:
            q.append(y.node)

print(result)
```



0	
1	(2, 4), (3, 2), (4, 3)
2	(6, 3), (7, 5)
3	(5, 1)
4	(6, 4)
5	(6, 2)
6	(7, 5)
7	

임계경로

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

```
while q:
    x = q.popleft()
    for i in range(len(a[x])):
        y = Edge(a[x][i].node, a[x][i].time)
        print("y::", y, result[x])
        if result[y.node] <= y.time + result[x]:
            result[y.node] = y.time + result[x]
        # 새롭게 진입차수가 0이 된 정점을 부여 삽입합니다.
        inDegree[y.node]-= 1
        if inDegree[y.node] == 0:
            q.append(y.node)

print(result)
```

- Result계산
- i->(2, 4), (3, 2), (4, 3)
- Y=(2,4), result[1] =0
- If문에서 result[2] <= 4+0 => result[2] = 4
- Indegree하나 빼고 q에 넣기
- Y=(3,2) result[1] = 0 result[3]=2
- Y=(4,3) result[1] = 0 result[4]=3
- Q에는 (2,3,4)

0	
1	(2, 4), (3, 2), (4, 3)
2	(6, 3), (7, 5)
3	(5, 1)
4	(6, 4)
5	(6, 2)
6	(7, 5)
7	

임계경로

- Q에 2를 진행
- Q에 3을 진행
- …진행 후 결과는

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

```
while q:
    x = q.popleft()
    for i in range(len(a[x])):
        y = Edge(a[x][i].node, a[x][i].time)
        print("y::", y, result[x])
        if result[y.node] <= y.time + result[x]:
            result[y.node] = y.time + result[x]
        # 새롭게 진입차수가 0이 된 정점을 부여 삽입합니다.
        inDegree[y.node]-= 1
        if inDegree[y.node] == 0:
            q.append(y.node)

print(result)
```

1	2	3	4	5	6	7
0	4	2	3	3	7	12

0	
1	(2, 4), (3, 2), (4, 3)
2	(6, 3), (7, 5)
3	(5, 1)
4	(6, 4)
5	(6, 2)
6	(7, 5)
7	

1	2	3	4	5	6	7
0	4	2	3	3	7	12

임계경로

- Q=7
- Cur = 7
- For l, t back_graph[7] (2, 5), (6, 5)
- l,t =>(2, 5)
- `result[cur]-result[i]==t`
- `Result[7]-result[2] ==5 12-4`
- l,t =>(6, 5)
- `Result[7]-result[6] ==5 12-7=5`
- `Check[6] = 1`
- `Q=> 6`

```
#백트래킹
cnt=0 #임계경로에 속한 모든 정점의 개수
q.append(end)
print(q)
while q: #도착점에서 시작점으로
    cur=q.popleft()
    print(cur)
    print(back_graph)
    for i,t in back_graph[cur]:
        #도착점까지의 비용에서 시작점의 비용을 뺐을 때 그 간선비용과 같다면
        if result[cur]-result[i]==t:
            cnt+=1
            if check[i]==0: #큐에 한번씩만 담을 수 있도록,중복방문제거
                q.append(i)
                check[i]=1
    print(q)
print(result)
print(result[end])
print(cnt)
```

Back_Graph

0	
1	
2	(1, 4)
3	(1, 2)
4	(1, 3)
5	(3, 1)
6	(2, 3), (4, 4), (5, 2)
7	(2, 5), (6, 5)