Python Algorithm class (Graph - 5) (위상정렬 및 최장경로)

nathan29849 · 2021년 5월 23일

Graph algorithm python 위상정렬 최장경로

Python Algorithm class

목록 보기 21/27

5. Graph 위상정렬

> (1) 위상정렬
> (2) 에지에 가중치가 있는(weighted) DAG에서 최장경로(longest path) 문제

(1) 위상정렬 (topological sort)

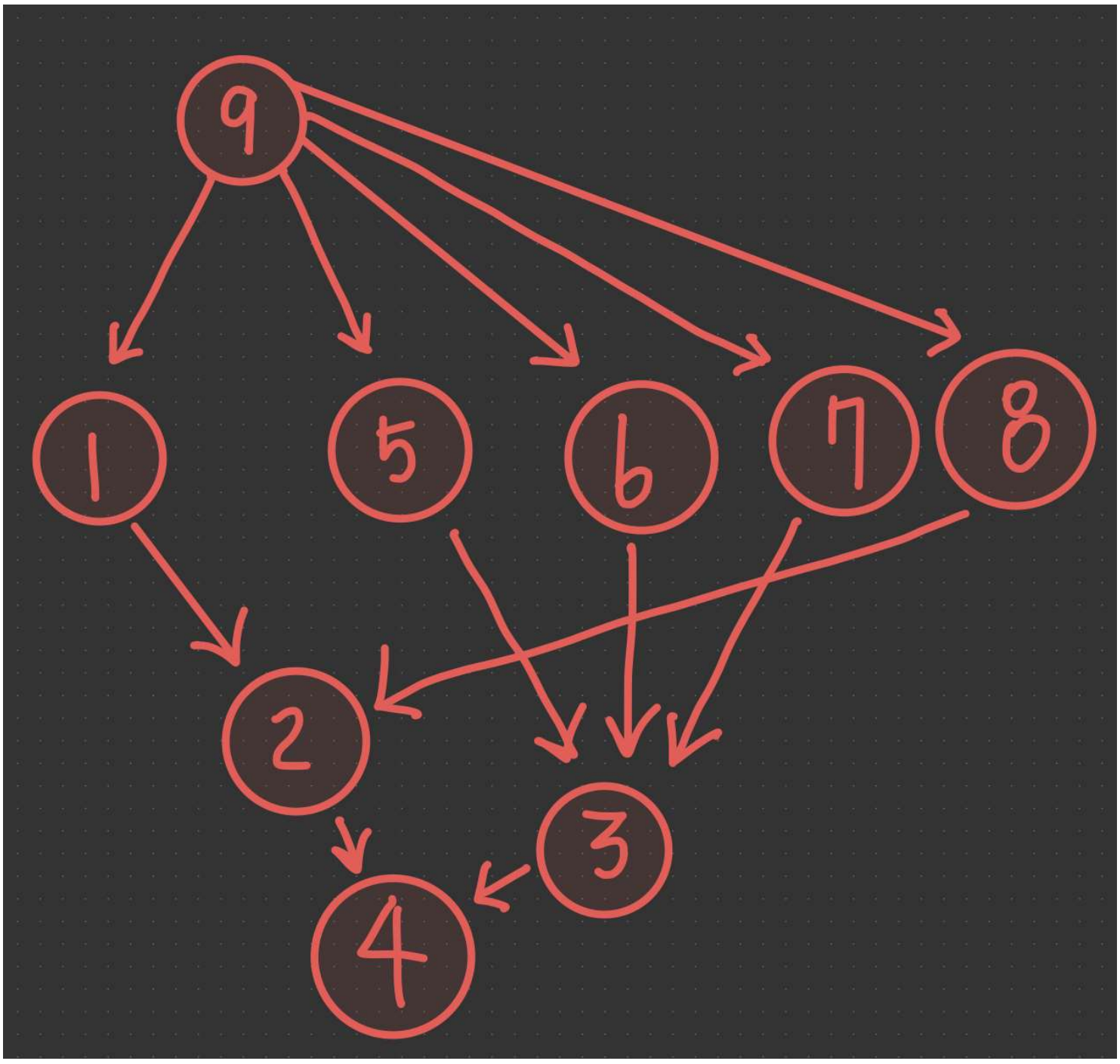
위상정렬 설명

- 그래프의 위상순서(topological order) : DAG G = (V, E)의 정점들에 다음과 같이 1부터 n(정점 수)까지 번호를 부여
  - (v, w)가 E(G)(그래프 에지)에 속하면 v의 번호가 w의 번호보다 작다.
- DAG G = (V,E)의 위상정렬 : V의 정점들을 다음 조건을 만족하면서 일렬로 나열하는 것
  - (v, w)가 E(G)에 속하면 v가 w보다 앞서 나와야 한다.
- 방향 그래프 G가 사이클을 가지고 있으면 G는 위상정렬을 할 수 없다(DAG만 가능)
  - DAG : Directed Acyclic Graph(방향 있는, 사이클 없는 그래프)

종속 태스크 스케쥴링

- 종속 태스크 스케줄링 : n개의 태스크들과 이들 태스크들 사이의 종속관계가 주어져 있다.(방향 그래프로 표현)
  - 두 태스크 t1, t2에 대해 t2가 t1에 종속되다
  - 즉, t1이 끝나야만 t2를 시작할 수 있다.
- 태스크들의 종속 관계를 만족하는 태스크들의 수행 순서를 구하라
- 예시

태스크	태스크 번호	종속 태스크
옷 고르기	1	9
옷 입기	2	1, 8
아침식사	3	5, 6, 7
출발	4	2, 3
커피 만들기	5	9
토스트 만들기	6	9
주스 따르기	7	9
샤워하기	8	9
잠깨기	9	-



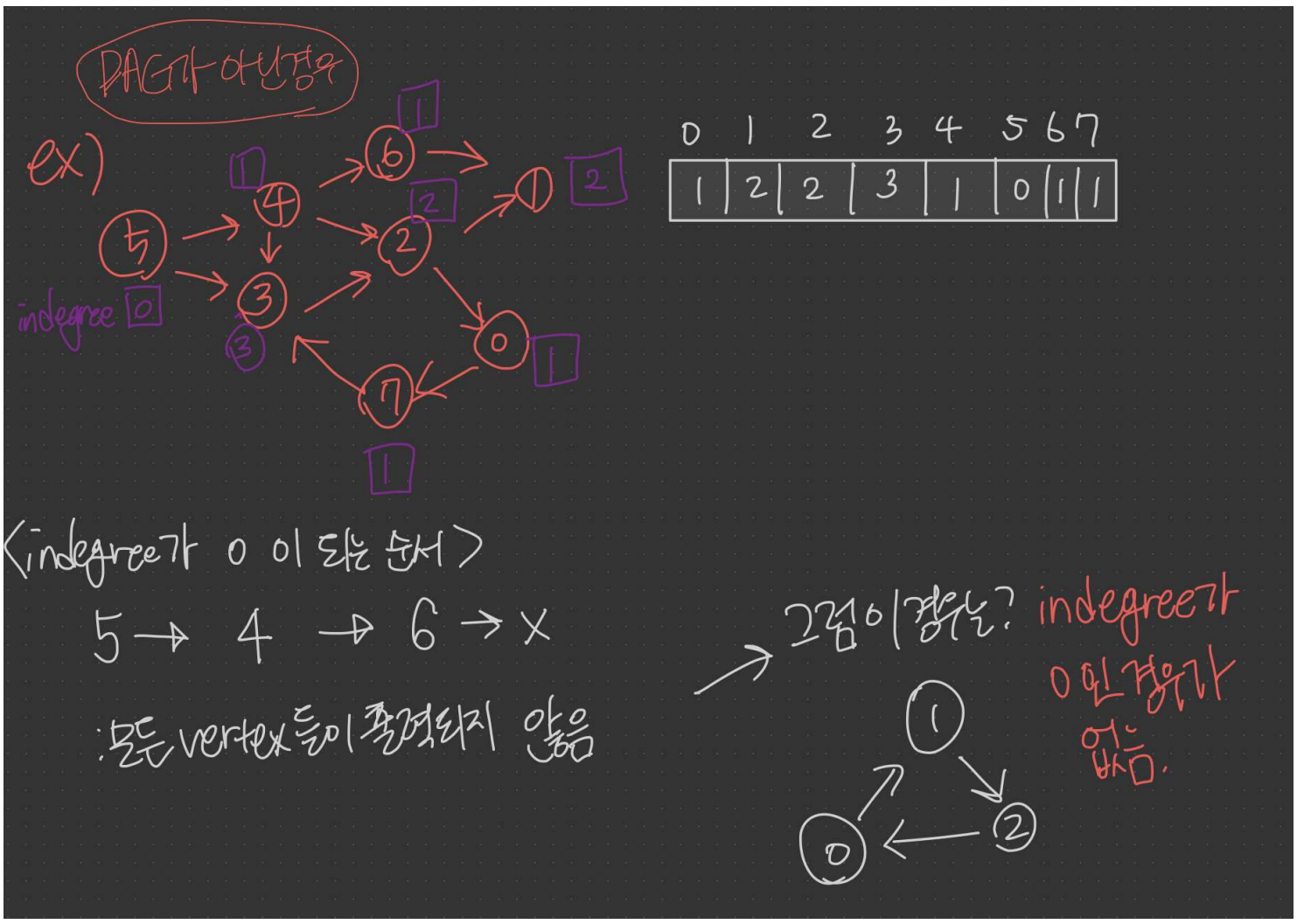
위상정렬의 결과 : 9, 1, 5, 6, 7, 8, 2, 3, 4
위상정렬의 결과2 : 9, 8, 7, 6, 5, 1, 2, 3, 4
(위와 같이 위상정렬의 결과는 다양할 수 있다.)

진입분지수를 이용한 위상정렬

- 정점의 분지수(=에지)를 `degree`라고 한다.
- 방향이 있는 그래프에서는 정점의 분지수(정점으로 들어오는 에지)를 `indegree`라고 한다.

> - indegree[v] : 정점 v로 들어오는 에지의 수(정점 v의 진입분지수)
> - 각 정점의 진입분지수(indegree)를 구하여 0인 정점을 v를 출력하고 v에서 나가는 모든 에지(v,w)에 대하여 w의 진입분지수를 1만큼 감소시킨다.
> - 위 과정을 반복한다.

- 단, 모든 vertex들이 출력되지 않으면 cycle이 있다고 판단한다. (DAG여야 함)



깊이우선탐색(DFS)을 이용한 위상정렬, 진입분지수를 이용한 위상정렬(BFS)

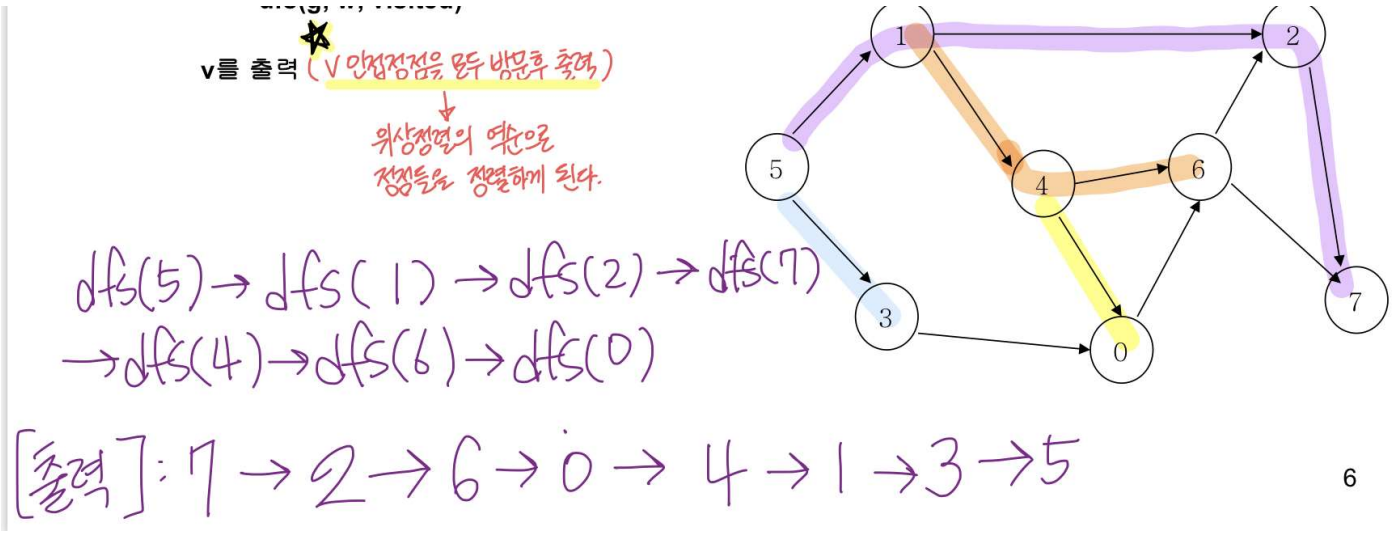
- Depth First 골격을 이용하여 위상정렬(역순으로 정점을 출력한다.)

```
Algorithm dfs(g, v, visited) // g: Graph, v: start vertex
   visited[v] = true // visit v

   // v와 인접한 각 정점 w에 대하여
   if(!visited[w]) // if w is unvisited
      dfs(g, w, visited)

   // v를 출력(v 인접 정점을 모두 방문 후 출력 -> 위상정렬의 역순으로 정점들을 정렬하게 된다.)
   print(v)
```

- 만약 Topological Sort로 출력하고 싶다면, v를 stack S에 push 후 하나씩 출력하면 된다.



- 파이썬 코드로 구현

```
from collections import deque

class Node:
    def __init__(self, vertex):
        self.vertex = vertex
        self.link = None

class Graph:
    def __init__(self, size):
        self.adjList = [None]*size
        self.visited = [False]*size
        self.n = size
        self.indegree = [0]*size
        self.Q = deque()

    def add_edge(self, v1, v2):
        new_node = Node(v2)
        new_node.link = self.adjList[v1]
        self.adjList[v1] = new_node
        self.indegree[v2] += 1

    def indegreeM(self):
        for v in range(self.n):
            if(self.indegree[v] == 0):
                self.Q.append(v)
        temp = []
        while(self.Q):
            print("Q:", self.Q)
            V = self.Q.popleft()
            temp.append(V)
            node = self.adjList[V]
            while node is not None:
                self.indegree[node.vertex] -= 1
                if(self.indegree[node.vertex] == 0):
                    self.Q.append(node.vertex)
                node = node.link

        for i in range(len(temp)-1):
            print(temp[i], end="->")
        print(temp[-1])

    def dfs(self, v):
        self.visited[v] = True
        node = self.adjList[v]
        while node != None:
            w = node.vertex
            if self.visited[w] is False:

                self.dfs(w)
            node = node.link
        print(v, end =' ')

    def printGraph(self):
        for v in range(self.n):
```

```
            print(v, end=": ")
            current = self.adjList[v]
            while current is not None:
                print(current.vertex, end=' ')
                current = current.link
            print()

    def revTopologicalSort(self):
        for v in range(self.n):
            if self.visited[v] is False:
                self.dfs(v)

n, m = [int(x) for x in input().split()]
g = Graph(n)
for i in range(m):
    v1, v2 = [int(x) for x in input().split()]
    g.add_edge(v1, v2)


g.indegreeM()
# g.printGraph()
# g.revTopologicalSort()
```

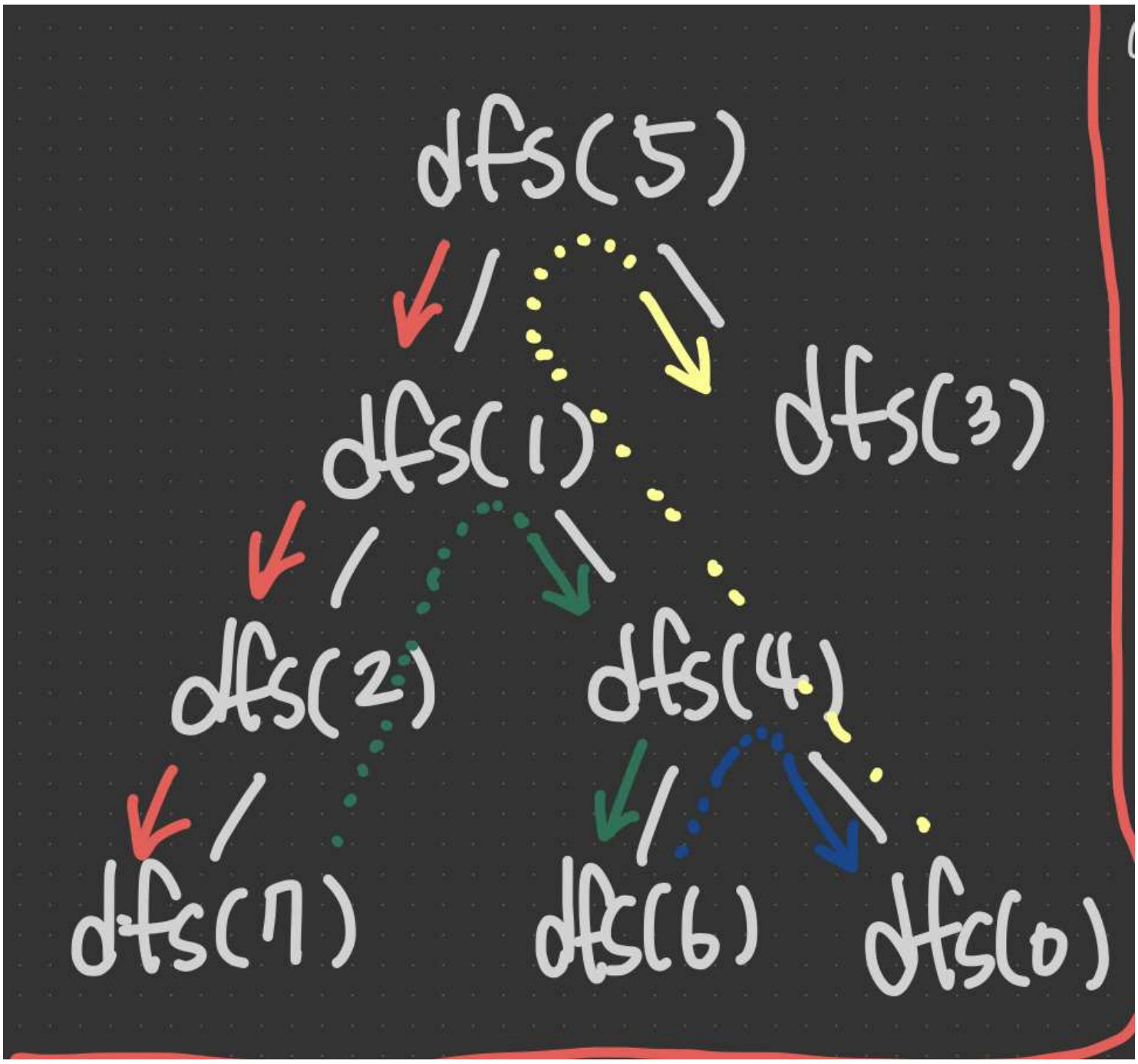
- 입력

```
8 11
5 1
1 2
1 4
4 6
6 2
6 7
2 7
4 0
0 6
3 0
5 3
```

- 코드 결과

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

# 진입 분지수를 이용한 위상정렬
5->3->1->4->0->6->2->7
```



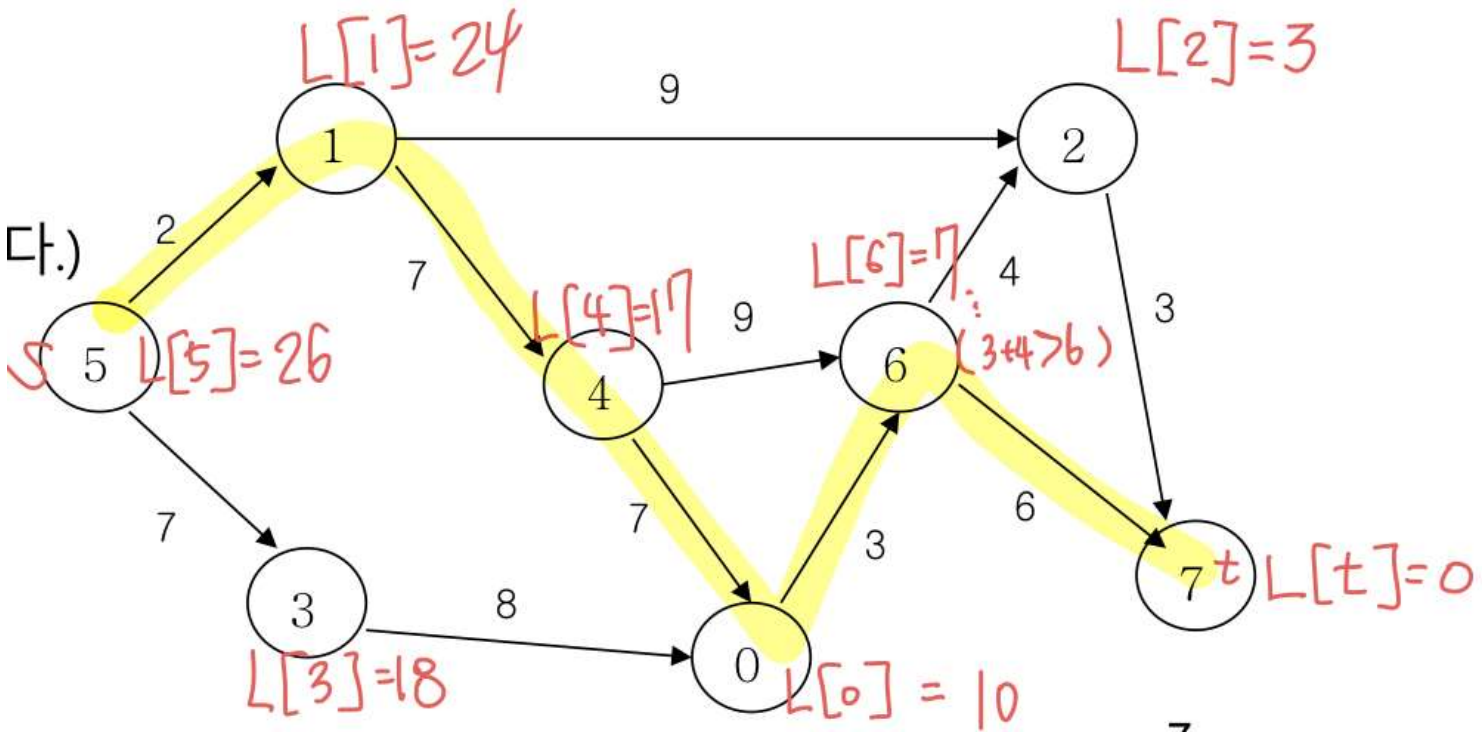
(2) 에지에 가중치가 있는 DAG에서 최장경로 문제

- 원래 잘 알려진 것은 "최단경로" 문제 (다음 주제)
- 에지에 가중치가 있는 DAG에서 두 정점 s와 t에 대하여 s로부터 t까지의 최장(가장 길이가 긴) 경로를 구하는 문제
- 일반적인 weighted 그래프에서 정점 s로부터 정점 t까지 최장 경로를 찾는 것은 매우 어려운 문제
    - 효율적인 알고리즘이 없다. (NP Complete 문제)
- DAG에서 정점 s로부터 정점 t까지의 최장(가장 길이가 긴) 경로를 찾는 것은 쉬운 문제이다.
    - **동적계획법**을 이용
- subproblem

> L[u]: u로부터 정점 t까지 가는 최장경로의 길이 (재귀적으로 구할 것임)
> Out(u) = {v|(u,v) ∈ E} // u로부터 인접한 정점들의 집합
> L[u] = max{L[u], L[v] + weight(u,v)} // u → v
> // L[v] : v에서부터 t까지가는 최장 경로
> // wight(u, v) : u에서부터 v까지의 가중치
> v ∈ Out(u) // Out(u)에 있는 v를 모두 확인한다고 생각하면 됨. 그 중 최댓값이 L[u]가 됨.
> L[u]를 **동적계획법**으로 구한다.

> L[t] = 0으로 초기화 한다.
> (나머지 정점들의 L[]은 -∞로 초기화 한다.)
> 깊이 우선 탐색을 이용하여 구한다.

- 예 (s = 5, t = 7)



- Python code

```
import sys

class Node:
    def __init__(self, vertex):
        self.vertex = vertex
        self.link = None

class Graph:
    def __init__(self, size):
        self.adjList = [None]*size
        self.visited = [False]*size
        self.n = size
        self.Long = [-sys.maxsize]*size # 음의 무한대로 설정
        self.weight = {}    # 가중치 에지를 표현하기 위해 리스트가 아닌 딕셔너리 자료형 사용

    def add_edge(self, v1, v2, w):
        new_node = Node(v2)
        new_node.link = self.adjList[v1]
        self.adjList[v1] = new_node
        self.weight[str(v1)+str(v2)] = w


    def findLongestPath(self, v, t):    # v 시작점, t 목적지
        if v == t:
            self.Long[v] = 0
        node = self.adjList[v]
        while node is not None:
            w = node.vertex
            self.findLongestPath(w, t)
            self.Long[v] = max(self.Long[v], self.Long[w]+self.weight[str(v)+str(w)])
            node = node.link


    def printGraph(self):
        for v in range(self.n):
            print(v, end=": ")
            current = self.adjList[v]
            while current is not None:

                print(current.vertex, end=' ')
                current = current.link
```

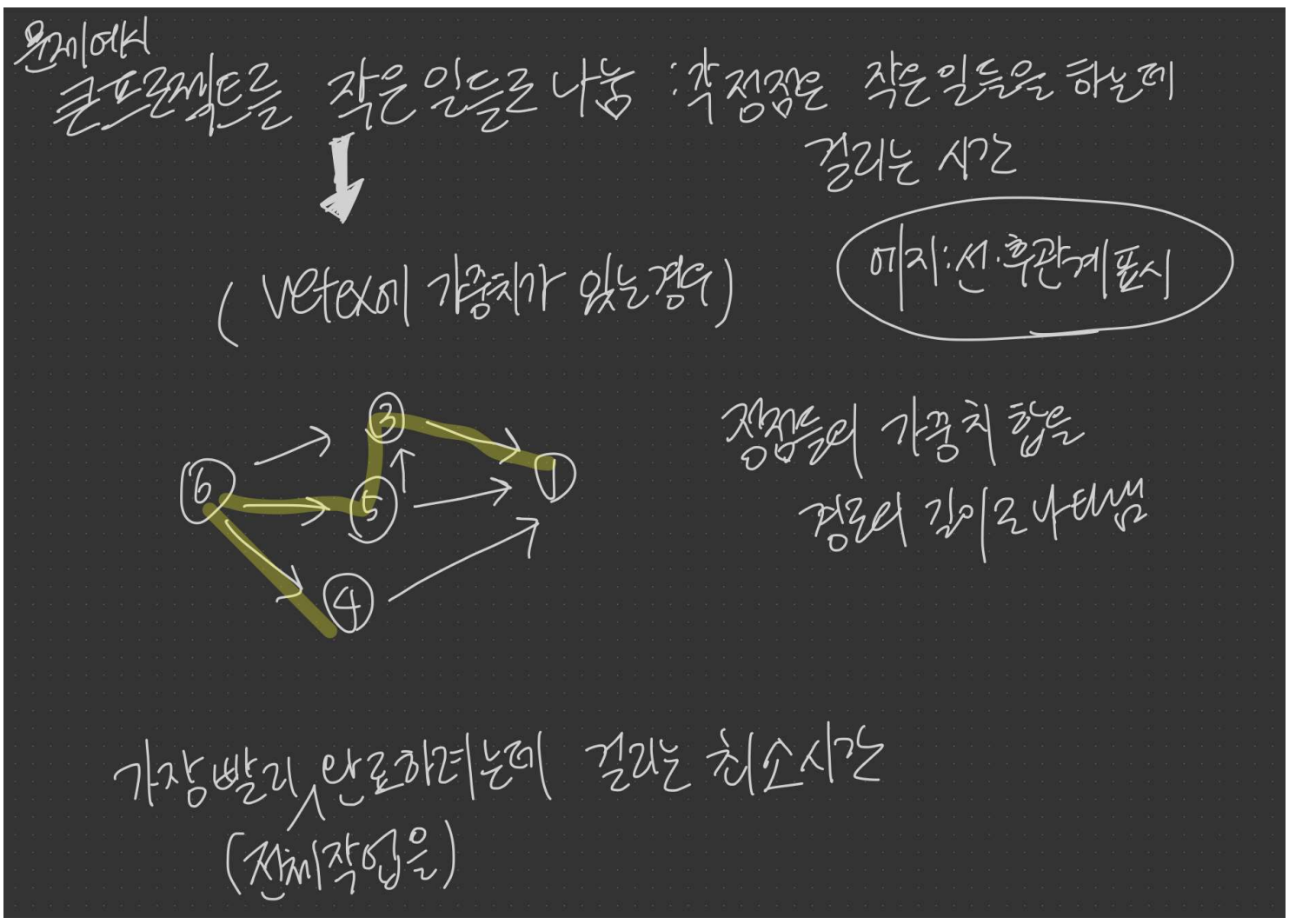
```
            print()



n, m = [int(x) for x in input().split()]
g = Graph(n)
for i in range(m):
    v1, v2, w = [int(x) for x in input().split()]
    g.add_edge(v1, v2, w)
# g.printGraph()
g.findLongestPath(5, 7)
print(g.Long)


# 8 11
# 5 1 2
# 5 3 7
# 1 4 7
# 1 2 9
# 3 0 8
# 4 0 7
# 4 6 9
# 0 6 3
# 6 2 4
# 6 7 6
# 2 7 3

#[10, 24, 3, 18, 17, 26, 7, 0]
```



nathan

나는 날마다 모든 면에서 점점 더 나아지고 있다.

이전 포스트
Python Algorithm class (Grap...

다음 포스트
Python Algorithm class (Grap...

0개의 댓글

댓글을 작성하세요

댓글 작성

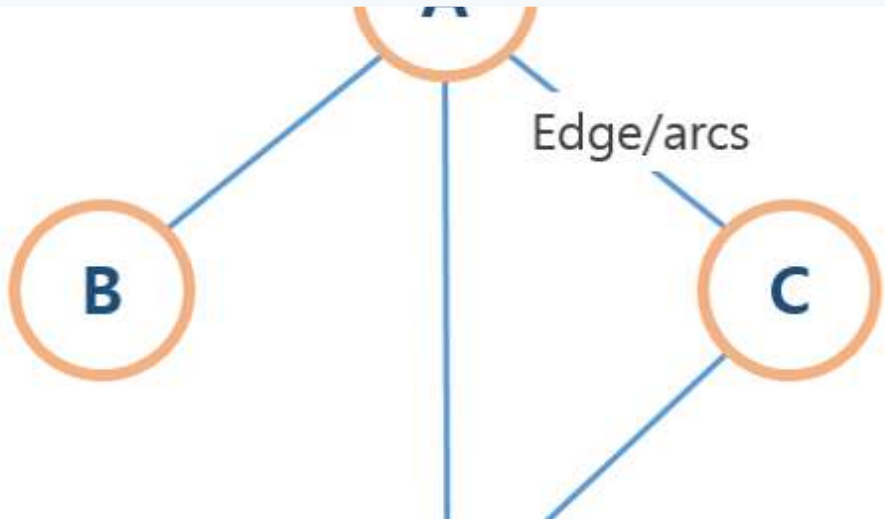
관심 있을 만한 포스트

[Algorithm] 위상 정렬(Topology Sort)

위상 정렬이란? 위상 정렬은 순서가 정해져있는 작업 차례로 수행해야 할 때, 그 순서를 결정해주는 알고리즘이다. 순차적으로 어떤 그래프가 형성되어있을 때, 그 그래프를 하나의 경로로 나타내는 것이다. 하나의 경로만 존재하는 것이 아니라, 여러 나열 방법이 있을 수 있기 때문에 답은 다양하게 나올 수 있는 알고리즘이다. DAG(Directed Acycl...

2020년 2월 6일 · 0개의 댓글

by **max9106** ❤ 0

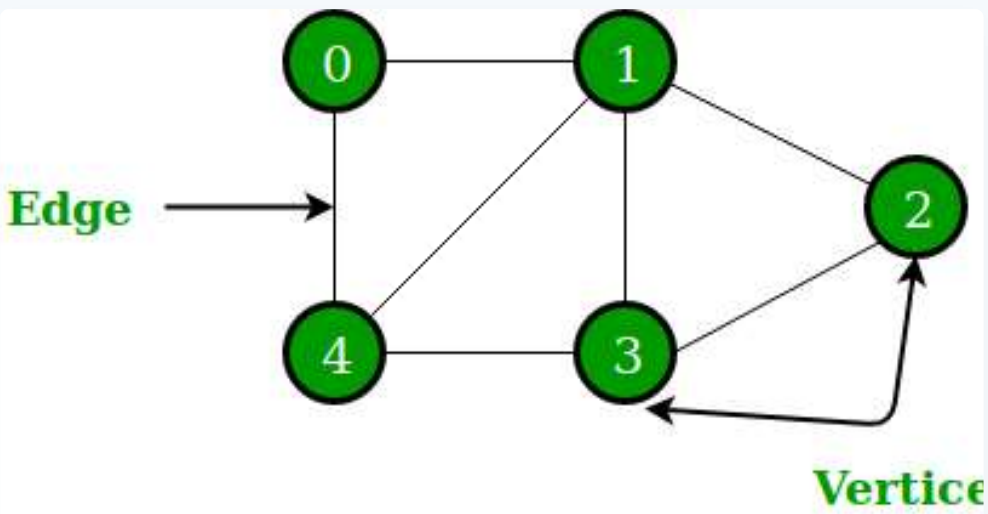


[자료구조]그래프

그래프는 노드와 간선으로 이루어진 자료구조입니다

2020년 4월 20일 · 0개의 댓글

by **gimtommang11** ❤ 4



Java Graph, DFS, BFS

그래프를 이해한다.

2020년 6월 12일 · 0개의 댓글

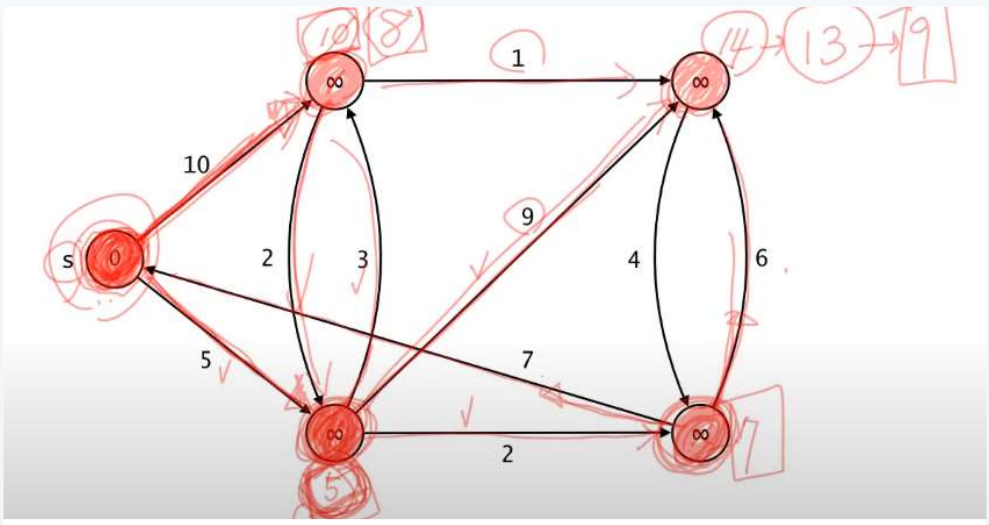
그래프 알고리즘 정리

그래프? 정점과 간선들로 이루어진 집합. 즉 트리 역시 그래프에 속한다고 할 수 있다. 그래프를 표현하는 세가지 방법 1. 간선 리스트 말그대로 배열에 간선들을 저장한다. 가장 간단하게 구현되지만 한 정점의 간선에...

2019년 8월 13일 · 1개의 댓글

by **agugu95** ❤ 0

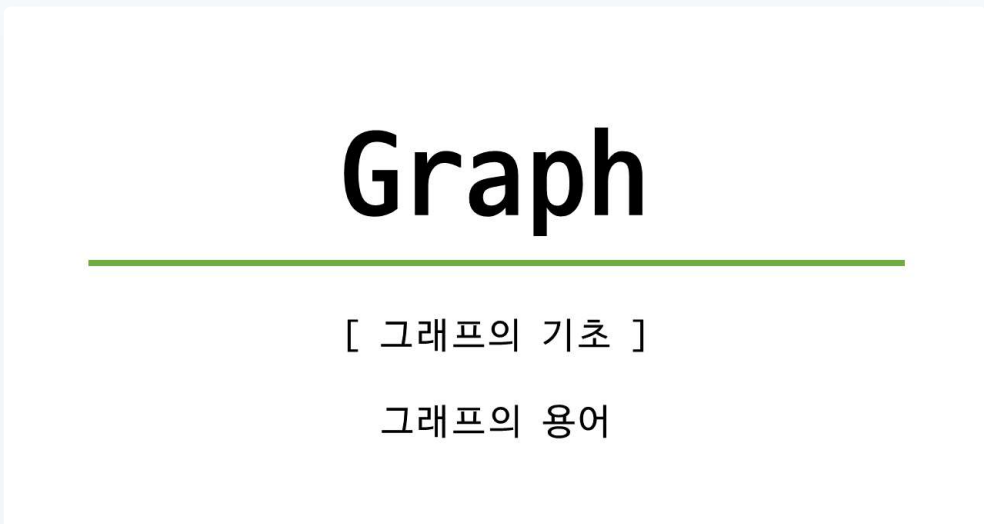
by **wan088** ❤ 0



그래프 최단 경로와 다익스트라(Dijkstra) 알고리즘

다익스트라 알고리즘을 이해한다.

2020년 6월 21일 · 0개의 댓글



[그래프] 그래프 기초

그래프는 자료 구조의 일종이다. 정점(Node, Vertex)와 간선(Edge)로 이루어져 있으며, 정점의 집합은 보통 V, 간선의 집합은 E로 표시한다. 정점 A에서 B로 가는 경로는 다양하게 말할 수 있다.A -> BA -> C -> BA -> C...

2020년 3월 20일 · 0개의 댓글

by **agugu95** ❤ 1

by **kjh107704** ❤ 0

최단경로 - (2) 벨만-포드(Bellman-Ford) 알고리즘

벨만-포드 알고리즘은 앞서 살펴본 다익스트라 알고리즘과 같이 그래프에서 시작 점점으로부터 다른 모든 정점까지의 최단 경로를 찾기 위한 알고리즘이다. 하지만 중요한 차이점이 있는데, 벨만-포드 알고리즘은 그래프 내에 음수 가중치를 갖는 간선이 있는 경우에도 활용할 수 있...

2020년 10월 4일 · 0개의 댓글

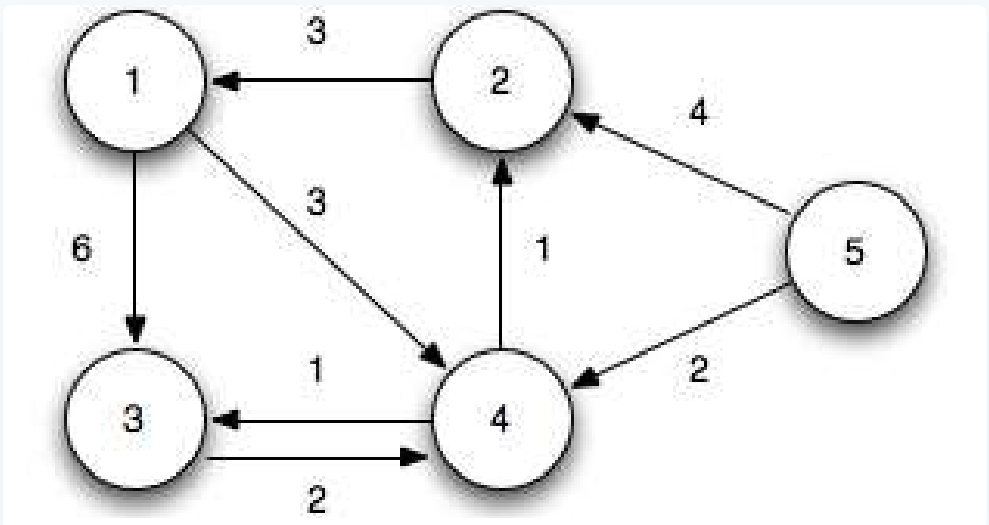
핵심 자료구조 - 그래프 : 최단 경로 : ① : BFS, 다익스트라

BFS와 다익스트라 알고리즘을 통해서 그래프의 최단경로를 구하는 것에 대해서 알아봅시다.

2021년 3월 13일 · 1개의 댓글

by **adorno10** ❤ 1

by **kasterra** ❤ 8



다익스트라 알고리즘(Dijkstra Algorithm)

다익스트라 알고리즘과 백준 1753번

2020년 11월 16일 · 0개의 댓글

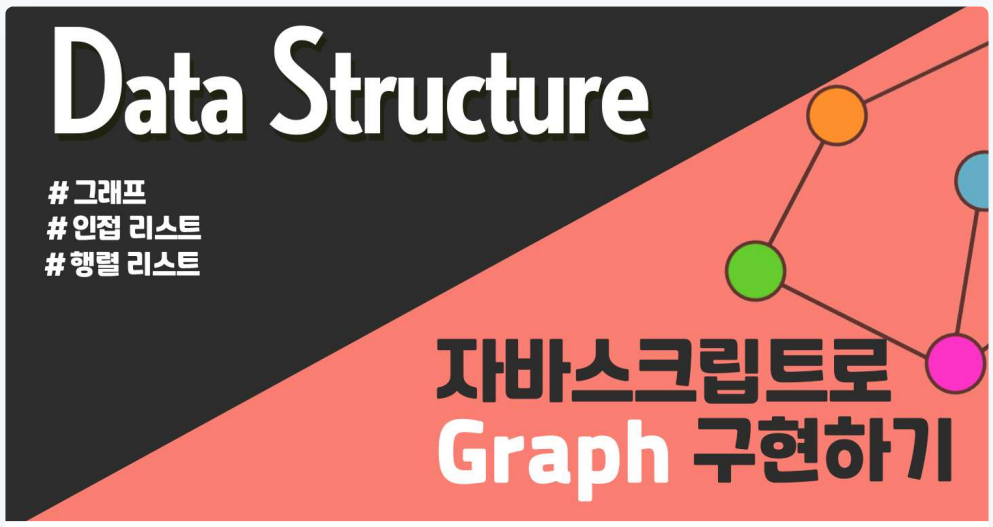
[알고리즘][그래프] 그래프의 표현과 정의

그래프 그래프의 정의 정의: 그래프 G(V,E)는 어떤 자료나 개념을 표현하는 정점(vertex)들의 집합 V와 이들을 연결하는 간선(edge)들의 집합 E로 구성된 자료구조 그래프의 종류 표현하고자하는 대상에 따라 여러가지...

2020년 8월 24일 · 0개의 댓글

by **lre12** 0

by **rachell_lee** 0

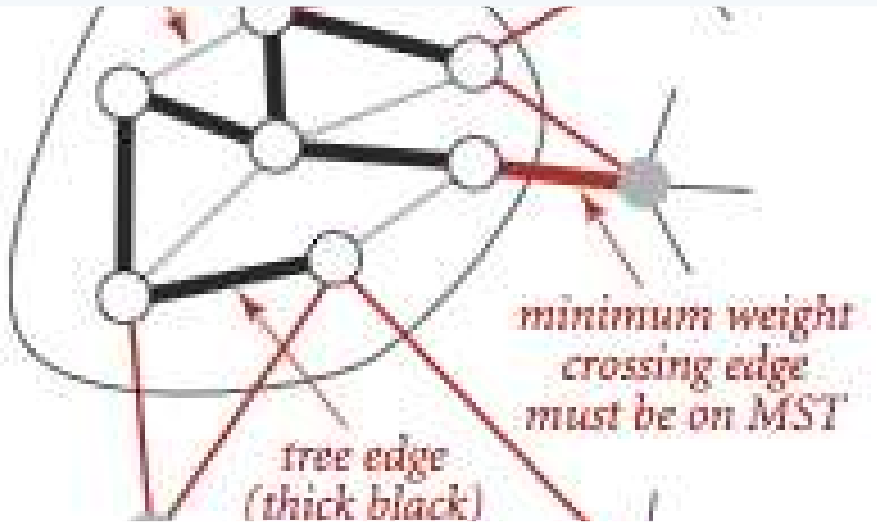


[Data Structure] 자바스크립트로 그래프 Graph 구현하기

그래프란 노드(또는 vertex라고도 부른다.)와 그 노드를 연결하는 간선(edge)을 하나로 모아 놓은 비선형 자료 구조이다. 그래프는 방향성에 따라 무방향(undirected) 그래프와 단방향(directed) 그래프로 나뉘며 간선...

2020년 10월 26일 · 0개의 댓글

by **nomadhash** 1



최소 스패닝 트리(MST)와 Prim's Algorithm(프림 알고리즘)

프림 알고리즘을 이해한다.

2020년 6월 15일 · 0개의 댓글

by **agugu95** 1