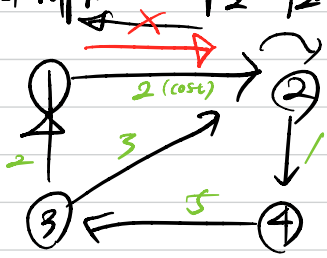
Graph 자료 구조와 기본연산



무방향 Graph

$G = (V, E)$

V: 정점 노드들 / E: 엣지 간선

$|V| = n$ 노드의 갯수 $\quad |E| = m$ 엣지 갯수만큼

가중치 표현 가능

〈1. 인접행렬〉

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$n \times n$

① memory : $O(n^2)$

② $(u, v) \in E$? (u와 v가 연결 되나?) → $G[u][v] = 1$ or ≥ 1 → $O(1)$

③ u에 인접한 모든 노드 방문

for v in range(1, n+1):
  do with G[u][v] ↳ 존재 하면 1

→ $O(n)$

(모든 경우 무조건 $O(n)$)

④ 새로운 간선 (u, v): 삽입

→ $G[u][v] = 1$

→ $O(1)$

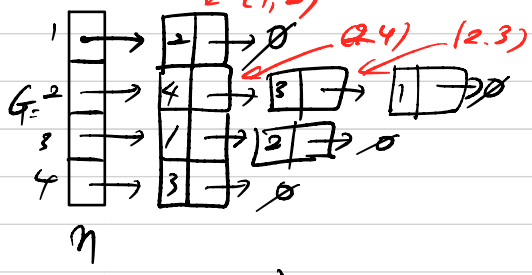
⑤ (u, v) 삭제:

→ $G[u][v] = 0$

→ $O(1)$

$O(n^2)$

〈2. 인접리스트〉



n

$O(n+m)$ (엣지 수만큼)

② $(u, v) \in E$? → $G[u]$.Search(v) (인접리스트) → $O(n)$ 노드의 갯수 만큼

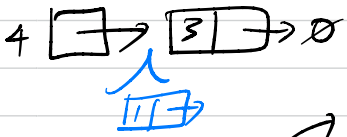
③ u에 인접한 모든 노드 방문

for each edge in G[u]:

O(인접 노드 수 만큼)

(인접수 만큼 반복)

④ 새로운 간선 (u, v): 삽입



G[u].pushFront(v) → $O(1)$

Python: append

⑤ (u, v) 삭제:

→ $x = G[u]$.Search(v)

G[u].remove(x) ↳ 삭제할 노드

$O(n)$

$O(n+m)$

∴ Sparse : 노드 갯수에 비해서 edge 갯수가 적을때 → 인접행렬 비추천

∴ Dense : 〃 edge 갯수가 많을때 → 인접행렬 추천.