PART 1) Exam Preparation.

- Single Source Path problems → given a weighted directed graph find the minimum-weights path from a single vertex s to all other vertices.

- Negative weight → Finding the maximum weight paths. Rather than create new Algorithms, It is possible to negate every edge and find the minimum-weight path as normal.

* Minimum-weight paths for negated edges = Maximum-weight paths for non-negated edges

- Shortest path tree → If there are no negative cycles, we can represented with a shortest path tree. Such a tree contains exactly one shortest path from s to each v.

= Memory needed $\Theta(n)$

2 arrays

① $d[v]$ ← optional weight of shortest path
② $PARENT[v]$, $v \in V$, $\Theta(n)$

Disadvantage of shortest path tree.

① only one path for each reachable vertex can be store
② $O(n^2)$

Relaxation technique ① INITIALization

+

② RELAX.

* Algorithms following this pattern differ in their termination condition and the order that edges are relaxed.

INITIALIZATION :

```
INITIALIZATION (G, S)
    ~~PARENT~~
    d[S] ← 0 ; PARENT[S] ← NIL
    for each node v ∈ V - {S} do
        d[v] ← ∞ ; PARENT[v] ← NIL
```

* INITIALIZATION is a $\Theta(n)$ operation.
← If input is large it can be too slow.

∴ to avoid this, we can initialize other vertices as they are discovered.

RELAXATION :

```
RELAX(u, v, w)
  for each edge (u, v) ∈ E do
    if d[v] > d[u] + w(u,v) then
        d[v] ← d[u] + w(u,v) ; PARENT[v] ← u
```

* $\Theta(1)$
PARENT[v] ← u because of inequality of transitivity
$\delta(s,v) \le \delta(s,u) + w(u,v)$

RELAXation properties

① d[v] will only decrease.

② d[v] either ∞ or weight of some path from s to v

③ $d[v] \ge \delta(s,v)$

BELLMAN-FORD : It is ~~relax based~~ relaxation based algorithm for finding shortest paths of a graph.
It permits a negative weight

```
BELLMANFORD (G, w, S)
  INITIALIZATION (G, S)
① for i ← 1 to n-1 do
    for each edge (u,v) ∈ E do
      RELAX(u, v, w)

② for each edge (u,v) ∈ E do
    if d[v] > d[u] + w(u,v) then
      return False ( Negative cycle reachable from Source s)
  return True { No negative cycle v ∈ V, d[v] = δ(s,v)
```

BELLMAN-FORD correctness :

BELLMAN FORD returns False = Negative cycle, or True = No negative cycle.

BELLMAN FORD running time
① INITIALIZATION $\theta(n)$
② RELAX $\theta(1)$
③ BELLMAN-FORD running time $\theta(nm)$
" worst case $\Omega(nm)$

Let's improving the running time.

The $\theta(nm)$ running times comes from the first section of the algorithm, so we focus our efforts there :

① we could try to (decrease the number of iterations)

1) If no effective relaxation operations take place in a given iteration the loop can terminate early. because shortest path were already computed.

2) If PARENT Array has a cycle the terminate early with return false.

② reduce work done in each iteration.

↳ Store active node in a FIFO Queue.

```
BF-with-FIFO (G, w, s)                         θ(nm)
  INITIALIZATION (G, s)                        θ(nm)
   Q ← ∅ ; Enque (Q, s) do
   while Q ≠ ∅ do
   u ← head (Q) ; Deque(Q)
   for each node v ∈ adj(u) do
    relax (u, v, w)   ←————————  if (check_cycle(parent)) :
    return true.                   return False
```

1) BF

2) BF with FIFO

3) CYCLE C PARENT)

4) $r(P) \le r(Q)$

5) $w(p) = w(u_1, u_2) + w(u_2, u_3) \cdots w(u_{k-1}, u_k)$

6) RELAXTion $\begin{cases} \text{InitiAlIZation} \\ \text{RELAX} \end{cases}$

Lec 1) 위 알고리즘 다 꼭 암기!