Lec 11) Single Source Shortest-path problem.

- $G = (V, E)$ $|V| = n$ number of nodes

  $|E| = n$ number of edges

- (n, n) → Corresponding to size of input
  we will analysis performance of Algorithms with this.

- we use term of weight then distance because there is a negative distance. So we use a more general term.

- Single Source Shortest Path problem
  = finding the path of smallest total weight.

- point to point problem can solved by single-source path Algorithm.

- page 3) $w(p) = w(v_1, v_2) + w(v_2, v_2) \cdots + w(v_k, v_{k+1})$

- The shortest path weight(Distance) from u to v :

$$\delta(u,v) = \begin{cases} \min w(p) \text{ over all } p \text{ from } u \text{ to } v \\ +\infty \text{ if there is no path from } u \text{ to } v \end{cases}$$

- A shortest Path $w(p) = \delta(u,v)$

  * we can have Multiple shortest paths but Only one shortest path weight.

page 4)

- A subpath of a shortest path is a shortest path.

- Triangle inequality : $\delta(s,v) \leq \delta(s,u) + w(u,v)$

page 5) Negative weight in an application.

- (x,y) edge = $r(x,y)$ rate of (x,y)
- (x,y) 의 weight = $w(x,y) = -\ln(r(x,y))$
  ex) $w(S,D) = -\ln(r(S,D)) = -\ln(1.6) = -0.47$

- If $r(x,y) > 1$, then $w(x,y) < 0$.

page 7) Negative weight cycles ← our consideration 1

If there is a negative cycles then we can't find ~~Nega~~ shortest path. - There is no efficient algorithm.
= NP hard problem (computationally difficult)
If there is No negative cycle then we can find shortest path.

If there is a negative cycle just find that there is a negative cycle.

page 9) If there are multiple shortest path from S to V then just find one shortest path. ← consideration 2.

Representation of out put : Shortest path tree.

Advantage : efficient Representation, we need very small memory.
we only store parent of node so it is very efficient

Disadvantage : we have to do Computation to set a path.
trace the path. = Computation

② only give one shortest path.
→ if there are more than 1 have to we different way to represent.

Page 10) · there may be other shortest paths from s to v which are not included in this tree.

· It is efficient to represent because it only needs linear of number of nodes.

· $\Theta(n)$, n is number of nodes.
= Space requirment is only linear.

· worse case $\Theta(n^2)$

Relaxation technique. :

· we store 2 arrays

1) d[v] : shortest path estimate.

2) PARENT [v] : the current predecessor of node v.

Relaxation technique = ① Initialization + ② Relaxation operation.

① Initialization (G, s)

d[s] ← 0 ; PARENT[s] ← NIL
for each node v ∈ {V} − {s} do
d[v] ← ∞ ; PARENT {v} ← NIL.

distance is not always 0 because of negative cycle. but if there is no negative cycle always 0

이 부분만 Relax 한다. ← (u, v)

weight 없으면 항상 같은 값. ← w

② RELAX (u, v, w) { relax edge (u, v) }
if d[v] > d[u] + w(u, v) then
d[v] ← d[u] + w(u, v) ; PARENT [v] ← u.

This is A Technique not a Algorithm.
It doesn't tell you when you have to stop.

Page 14) Properties of the relaxation technique.

If we use relaxation technique.

- └ Shortest path estimate never ~~ge~~ increase. Always derease.

- beginning ∞ or equal to the weight of some path from S to v.

- $\forall$, $d[v] \geq \delta(s,v)$

- ∞ ·········(gradually)→ lower and lower → shortest path estimat

- effective relax operation → 실제로 shortest path estimate를 derease 한 relax operation.

relax operation을 할 수 있다.

- $(u,v) \in E$, $d[v] > d[u] + w(u,v)$
- $x \in V$, $d[x] > \delta(s,x)$

이 두 조건은 항상 같다. 하나가 True면 다른 하나도 True
〃 〃 False 〃 False.

우리가 더 이상 effective ~~relax~~ relax operation을 못하면 그게 바로 shortest path estimate 이다.

No Negative Cycles

- If there is no negative cycle there may be only finitely ~~many~~ effective relax operation. Means it will finish the computation but it doesn't guarantee whether it will be fast.

- The PARENT pointers always form a tree and at the end of the computation tree will be shortest path tree.

- when eventually no effective relax operation possible then

$\forall$, $d[v] = \delta(s,v)$
and
PARENT pointers form a shortest-paths tree.

Page 15)

- always exist $d[v] > d[u] + w(u,v)$
  meaning an effective RELAX operation always possible.

- THE PARENT pointers will eventually form a CYCLE.

* PARENT pointer 가 tree 인지 CYCLE 인지를 체크해서 negative CYCLE 이 있는지 확인 할 수 있다

Page 56) Exercise Checking PARENT nodes < tree ? / cycle ?

1) 각 Node 들에게 PARENT를 찾는다
2) PARENT가 있으면 그 다음 NODE로 간다.

* PARENT node 가 child node 로 가는 [illegible]

즉 [illegible] tree 이면 S로 갈 [illegible] tree
[illegible] [illegible] CYCLE

worst case $O(n^2)$