Lecture18_ShortestPaths2

Tuesday, October 20, 2020 4:21 PM

Unweighted graph

Suppose that w(u, v) = 1 for all (u, v) ∈ E. Can Dijkstra's algorithm be improved?

The PQ we have does not have to be a true PQ. Reason: whenever we found a new improved path, we change the key value or the distance. Therefore, this component won't have dramatic change.

- Use a simple FIFO queue instead of a priority queue.

Correctness of BFS


```
while Q ≠ ∅
    do u ← DEQUEUE(Q)
        for each v ∈ Adj[u]
            do if d[v] = ∞
                then d[v] ← d[u] + 1
                    ENQUEUE(Q, v)
```


Key idea:

The FIFO Q in breadth-first search mimics the priority queue Q in Dijkstra.

- **Invariant:** v comes after u in Q implies that $d[v] = d[u]$ or $d[v] = d[u] + 1$.

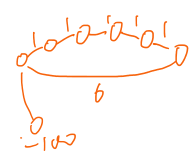
Running time: O(V + E)

- Works for all edge weight are same
- Limitation of Dijkstra's algorithm? No negative edge weight.

Determine eight negative cycle exist

Question: remove negative edge weight by adding x to all edge weight, where -x is the smallest edge weight in the graph

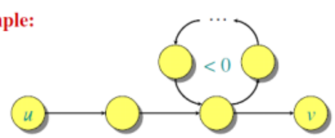
- Do we still have the same problem?? Do they have same shortest path?
- Design a counter example showing that this idea would not work
  No! the problem changes!! The shortest path changes!!
  The addition may affect the path multiple times
- This modification based on for each path, we add number of edges times x to the shortest path.



Negative-weight cycles

If a graph G = (V, E) contains a negative weight cycle, then some shortest paths may not exist.

Example:



Bellman-Ford algorithm: Finds all shortest-path lengths from a source s ∈ V to all v ∈ V or determines that a negative-weight cycle exists.

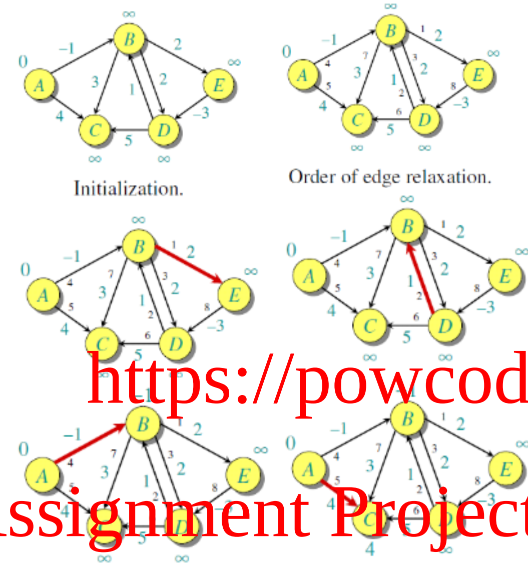
- Figure out if there exist negative cycles

```
d[s] ← 0
for each v ∈ V – {s}          } initialization
    do d[v] ← ∞

for i ← 1 to |V| – 1
    do for each edge (u, v) ∈ E
        do if d[v] > d[u] + w(u, v)            } relaxation
            then d[v] ← d[u] + w(u, v)         } step
for each edge (u, v) ∈ E
    do if d[v] > d[u] + w(u, v)
        then report that a negative-weight cycle exists
```

At the end, $d[v] = \delta(s, v)$, if no negative-weight cycles.
Time = $O(VE)$.

// do relaxation when found better path
How many times we do relaxation? In which order we do?
Does not matter (matters for Dijkstra)
N - 1 pass maximum. (at most n - 1 edges for n vertices)
After n -1 relaxations, if it still can do relaxations, then there is negative cycle.*



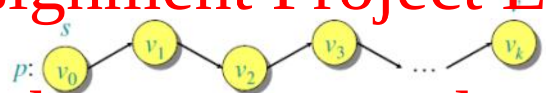
Initialization. Order of edge relaxation.

- One node can be update more than one time in single iteration
- If one whole pass doesn't change any ting, done

Correctness:

Theorem. If G = (V, E) contains no negative weight cycles, then after the Bellman-Ford algorithm executes, d[v] = δ(s, v) for all v ∈ V.

Proof. Let $v \in V$ be any vertex, and consider a shortest path p from s to v with the minimum number of edges.



Since p is a shortest path, we have

$$\delta(s, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i).$$

Initially, $d[v_0] = 0 = \delta(s, v_0)$, and $d[v_0]$ is unchanged by subsequent relaxations (because of the lemma from *Shortest Paths I* that $d[v] \geq \delta(s, v)$).

- After 1 pass through E, we have $d[v_1] = \delta(s, v_1)$.
- After 2 passes through E, we have $d[v_2] = \delta(s, v_2)$.

⋮

- After k passes through E, we have $d[v_k] = \delta(s, v_k)$.

Since G contains no negative-weight cycles, p is simple. Longest simple path has $\leq |V| - 1$ edges. □

Corollary. If a value d[v] fails to converge after |V| – 1 passes, there exists a negative-weight cycle in G reachable from s.