Lecture 30: Graph Algorithms

Agenda:

- Single-source shortest paths
- Bellman-Ford's algorithm for general case

Reading:

- Textbook pages 588 – 592

Dijkstra's SSSP algorithm (recall):

- $d[v]$ — weight of the shortest path from source s to v

  if no such path, set to ∞

- Idea in Dijkstra's algorithm:
  - greedily grows an SSSP tree
  - ensures that when adding a vertex, its shortest path in the current (induced) subgraph is determined
  - records for every non-tree vertex v its best parent tree vertex $p[v]$

  Note: very similar to Prim's MST algorithm (the min-priority queue implementation)

- Pseudocode (use $d[v]$ as the key):

```
procedure dijkstra(G, w, s)          **digraph G = (V, E)

for each v ∈ V(G) do                 **initialization
    d[v] ← ∞
    p[v] ← NIL
d[s] ← 0
Q ← V(G)
while Q ≠ ∅ do
    u ← ExtractMin(Q)                **s dequeued first
    for each v ∈ Adj[u] do
        if d[u] + w(u, v) < d[v] then
                                     **update v, no matter if v ∈ Q
            p[v] ← u
            decrease-key(v, d[u] + w(u, v))
                                     **d[v] ← d[u] + w(u, v)
```

Dijkstra's SSSP algorithm — proof of maintenance:

- (`while`) Loop Invariant: for every $v \in S$, $d[v]$ records the weight of the shortest path from s to v in graph G

- Maintenance (vertex u dequeued)

  $dist[u]$ — weight of a shortest path from s to u in G:

  — must show that at the end of loop body, $d[u] = dist[u]$

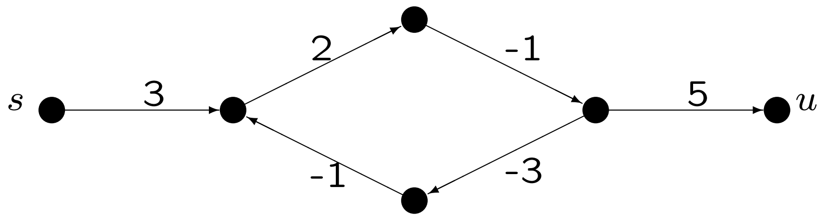
  Let $P = (s, v_1, v_2, \ldots, v_{k-1}, u)$ be any shortest path from s to u in graph G:

  - y — first vertex in P but not in S
  - x — the vertex before y in P
  - $dist[y] \leq dist[u]$ — y on the path
  - $d[y] \geq d[u]$ — min-priority queue
  - $d[y] = dist[y]$ — since $x \in S$
  - conclusion: $d[u] \leq dist[u]$

- When there are negative weight edges. we fail to claim:

  — $dist[y] \leq dist[u]$ — y on the path

- Another problem: negative weight cycles



What is the weight of a shortest path from s to u ???

Bellman-Ford's SSSP algorithm for the general case:

- General case — edge weights could be negative
- Output:
  1. if there is a negative weight cycle, report it
  2. otherwise report all the $dist[u]$ values and associated paths
- Idea in the algorithm:

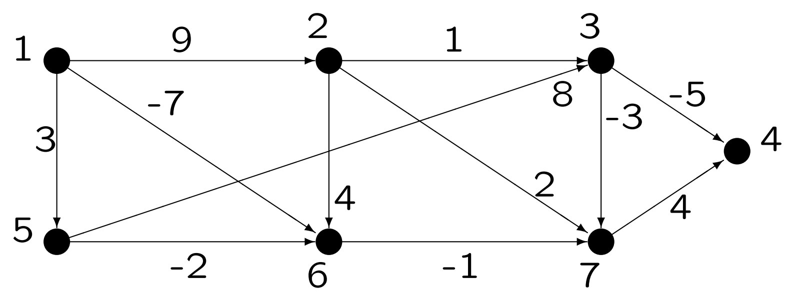
  If there is no negative weight cycle reachable from s, then every s-to-u shortest path contains at most $n-1$ edges; also is true that when all s-to-u shortest paths are discovered, for every edge (u, v) there must be $d[v] \le d[u] + w(u, v)$.

  It follows that $d[v]$ can be reduced $n-1$ times in order to have value $dist[v]$, but no more.
- Pseudocode:

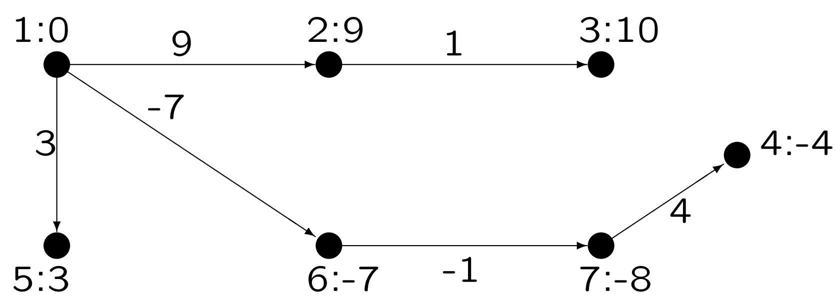
```
procedure bellman-ford(G, w, s)          **digraph G = (V, E)

for each v ∈ V(G) do                     **initialization
    d[v] ← ∞
    p[v] ← NIL
d[s] ← 0

for i ← 1 to n − 1                       **n = |V(G)|
    for each edge (u, v) ∈ E(G) do
        if d[u] + w(u, v) < d[v] then    **update d[v]
            p[v] ← u
            d[v] ← d[u] + w(u, v)
for each edge (u, v) ∈ E(G) do
    if d[u] + w(u, v) < d[v] then        **there is a negative cycle
        return FALSE
return TRUE
```

Bellman-Ford's SSSP algorithm — analysis:

- An example:



`bellman-ford`$(G, 1)$:



- Correctness: textbook pages 589 – 591

- Running time:

  1. initialization: $\Theta(n)$
  2. updating $d[v]$: $\Theta(n \times m)$
  3. checking existence of negative cycles: $\Theta(m)$

  Conclusion: $\Theta(nm)$ time (assuming adjacency list graph representation)

Have you understood the lecture contents?

well	ok	not-at-all	topic
□	□	□	SSSP problem
□	□	□	shortest path problem variants
□	□	□	Dijkstra's algorithm: idea
□	□	□	where Dijkstra's fails? why?
□	□	□	Bellman-Ford's algorithm: idea
□	□	□	execution & analysis