CS 340 – lecture 24 – Nov 02

$O(|V| + |E|)$ running time (worst-case) topol. sort

- getting initial list of in-degrees $\rightarrow O(|V| + |E|)$
- "remove" $|V|$ vertices
- make $|E|$ many in-degree updates

$\Bigg\} O(|V| + |E|)$

4.3 Shortest-Path Algorithms

Definition 17. Let $G = (V, E)$ be a weighted graph with cost function $c: E \to \mathbb{R}$, and let $p = [v_1, v_2, \ldots, v_n]$ be a path in G. The weighted length of p is

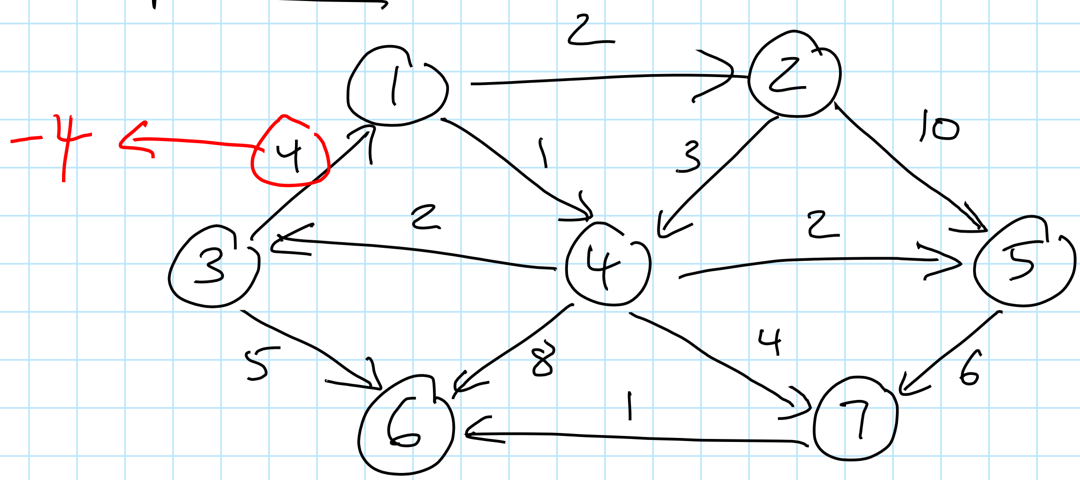
$$c(v_1, v_2) + c(v_2, v_3) + \ldots + c(v_{n-1}, v_n) = \sum_{i=1}^{n-1} c(v_i, v_{i+1})$$

Single-source shortest path problem (SSSPP)
weighted / unweighted

given: graph $G = (V, E)$ weighted / unweighted
vertex $s \in V$ (source)

task: for each vertex $v \in V$, find a path from s to v in G that has the smallest weighted length / length

Example 44.



- shortest weighted path from 1 to 6 : [1, 4, 7, 6] (weighted length = 6)

- shortest path from 1 to 6 : [1, 4, 6] (length 2)
- there are two shortest paths from 2 to 7

- If we change the weight on (3, 1) to −4 instead of 4, then the cycle [1, 4, 3, 1] has weighted length −1
  and the path [1, 4, 3, 1, 4, 7, 6] : 5

  [1, 4, 3, 1, 4, 3, 1, 4, 7, 6] : 4

  $[\underbrace{1, 4, 3, \ldots, 1, 4, 3,}_{1{,}006 \text{ times}} 1, 4, 7, 6]$: −1,000

⇝ there is no shortest weighted path

⇝ SSSPP only meaningful if there are no negative-cost cycles.

4.3.1 Unweighted Shortest Paths

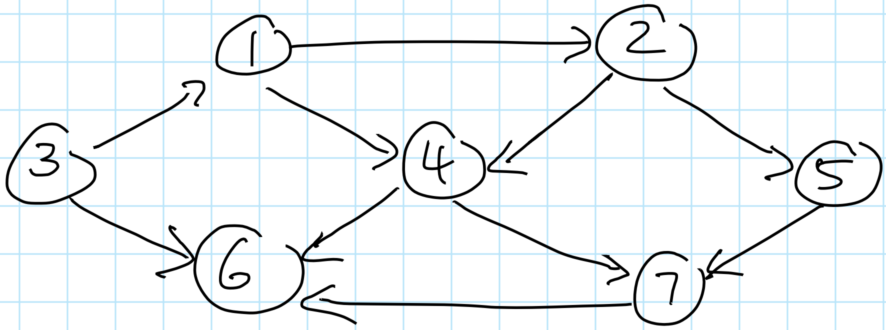
idea:
- first determine the [SP for the] set of vertices V_s closest to s
- then determine the [SP for the] not yet visited vertices closest to those in V_s
- ...

⇝ "Breadth-First Search" BFS

BFS allows us to store, for each vertex v, only store the length of the SP from s to v, plus the previous vertex on such an SP
(we reconstruct the whole path from that information)

```
Queue <Vertex> q;
for each Vertex v  { v.distance = ∞ };            O(|V|)
s.distance = 0;                                  } O(1)
q.enqueue (s);                                   }
while (!q.isEmpty())                             } if graph given as adjacency list:
{                                                } O(|V|+|E|)
    Vertex v = q.dequeue();
    for each Vertex w with (v,w) ∈ E
        if (w.distance == ∞)
        {   w.distance = v.distance + 1;
            w.previous = v;
            q.enqueue (w);
        }
}
```

Example 45 (see Handout 5 on UR Courses)



$s = 3$
source

vertex	distance	previous
~~1~~	$\cancel{\infty}$ 1	— 3
~~2~~	$\cancel{\infty}$ 2	— 1
~~3~~	0	—
~~4~~	$\cancel{\infty}$ 2	— 1
~~5~~	∞ 3	— 2
~~6~~	$\cancel{\infty}$ 1	— 3
~~7~~	∞ 3	— 4

Q: ~~3~~, ~~1~~, ~~6~~, ~~2~~, ~~4~~, ~~5~~, ~~7~~

reconstruct SP from s (=3) to 5.

previous (5) = 2, previous (2) = 1,

previous (1) = 3 = s

$\rightsquigarrow$ [3, 1, 2, 5]