CS 340 – lecture 25 – Nov 04

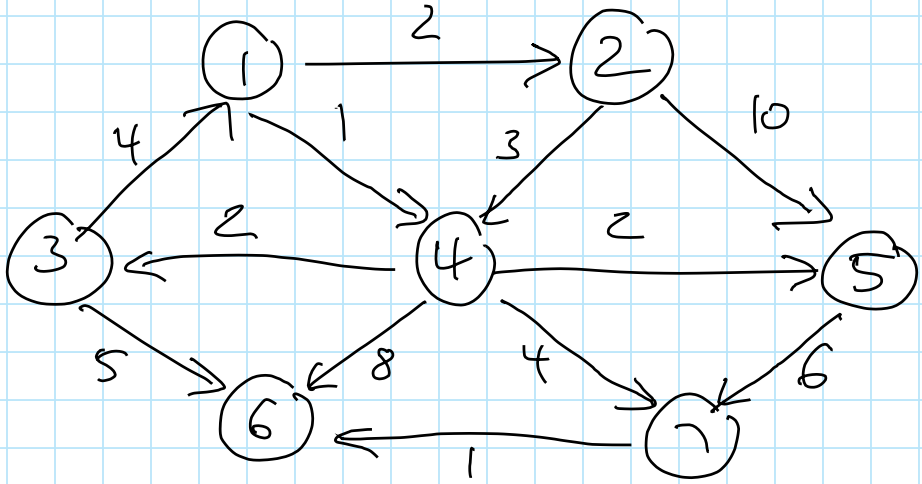
4.3.2 Weighted SP – Dijkstra's Algorithm

assumption: no negative edge costs

IDEA:
- similar to unweighted SP, but no queue; declare handled vertices "known" (instead of dequeued)
- be greedy: select vertex v with smallest current distance from source among all "unknown" vertices and mark it "known"
- adjust distance of adjacent "unknown" vertices

Example 41,

$s = 1$



distance(1) = 0, all other v: distance(v) = ∞

greedy: mark "1" as "known"
adjust distance(2) = 2, distance(4) = 1

greedy: mark "4" as "known"

adjust distance(3) = 3

distance(5) = 3

" (6) = 9

" (7) = 5

greedy: mark "2" as "known"

adjust distance(5) ? No!

because path [1, 2, 5] has cost of 12,

but current distance(5) = 3,

indicating that a path with cost 3 has been found before.

greedy: mark 3 "known"

(note: could also pick 5)

adjust distance(6) = 8

...

Why does this work?

O(|V|) for each Vertex v {v.distance = ∞; v.known = false}

O(1) s.distance = 0;

do forever

{

over time in total O(|V|²) {Vertex v = unknown vertex with smallest distance;

(if no such vertex exists {break;})

v.known = true; //see "dequeued" in BFS

over time in total O(|E|) {for each Vertex w with [(v,w) ∈ E && !w.known]

if (w.distance > v.distance + c(v,w))

{

w.distance = v.distance + c(v,w)

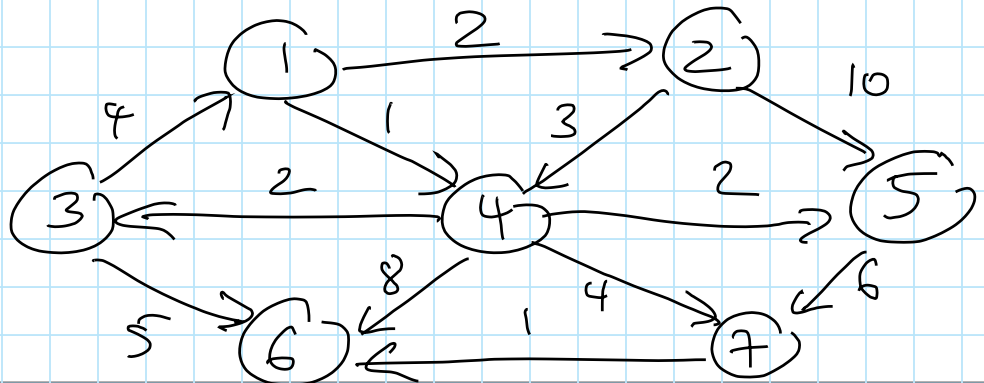
w.previous = v;

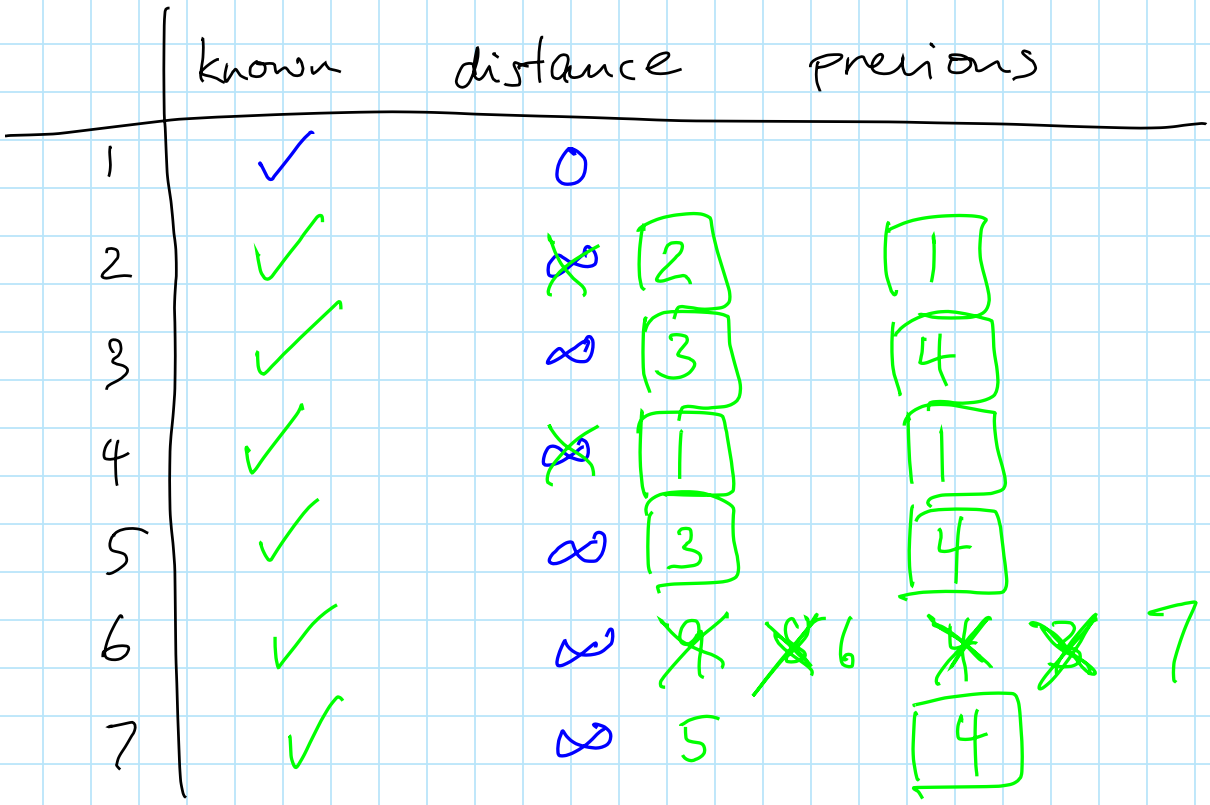
}

}

O(|V|²)

Example 41 ctd.




known
distance
previous
1
2
3
4
5
6
7
0
2
3
1
3
6
5
1
4
1
4
4