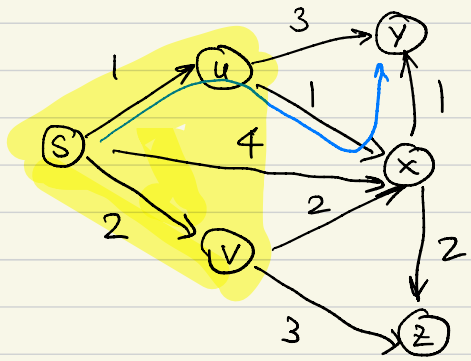
Shortest Paths.

$n = |V|$
$m = |E|$

Input: A (directed) graph $G = \langle V, E, \ell \rangle$. An edge e has length $\ell(e) \geq 0$.
A source node $s \in V$.
(The length of a path P is defined as $\ell(P) = \sum_{e \in P} \ell(e)$.)

Output: An array $d[\,]$ where
$d[u]$ = the length of the shortest path between s and u.

Example:



$d[x] = 2$ $\quad s \to u \to x$
$d[y] = 3$ $\quad s \to u \to x \to y$

Dijkstra's algorithm. [Dijkstra '59]

We maintain: $S \subseteq V$: "explored" nodes of G.
$\forall u \in S$, we have found a shortest path from s to u.

An array $d'[\,]$ such that
for every $v \notin S$.

$$d'(v) = \begin{cases} \text{the length of the shortest path from } s \text{ to } v. & \text{if } v \text{ is within one hop from } s. \\ +\infty & \text{o.w.} \end{cases}$$

$$d'(v) = \min_{\substack{e=(u,v) \\ u \in S}} d(u) + \ell(e).$$

the length of the shortest path from s to u

Example: $S =$

$d'(y) = 4$. $\quad e = (u, y)$.

$d'(x) = 2$ $\quad e = (u, x)$

Dijkstra's algorithm.

$S = \{s\}$ $\quad d(s) = 0$.

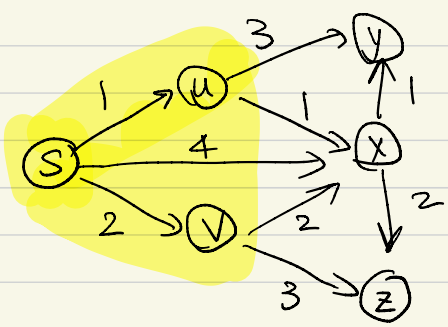
while $S \neq V$

select a node $v \notin S$ with the smallest. $d'(v)$.

$$d'(v) = \min_{\substack{e=(u,v) \\ u \in S}} d(u) + \ell(e)$$

Add v to S and set $d(v) = d'(v)$

End while.



$S = \{s\}$. $\quad d = (0, \ , \ , \ , \ , \ )$

$\qquad\qquad\qquad

Priority Queues (see Chapter 2.5 of the textbook)

We want to maintain a dynamic set S of elements (Recall arrays and linked lists) and each element e has a priority value or key $key(e)$.
(in this lecture, smaller keys represent higher priorities.)

Operations: Let H be a priority queue. (that has $\leq n$ elements at any time)

Runtime = $O(\log n)$ H.insert(x, k): insert an element x with $key(x)=k$.
$O(\log n)$ H.delete(x) : delete x.
$O(1)$ H.min() : return an element x with the smallest key.
$O(\log n)$ H.changekey(x,k) : change the key of x to k.

(We can implement a priority queue using a min-heap and a position array.)

Implementing Dijkstra.

Combine d and d' into one array d.
$\uparrow$ $u \in S$ $\quad$ $\uparrow$ $u \notin S$.

```
S = {s} ,   d(s) = 0.
For every u ∈ V , u ≠ s :
    If (s,u) ∈ E   d(u) = ℓ(s,u)   Else   d(u) = +∞.
    H.insert(u, d(u));          ← H.insert() is called O(n) times
End For.
While S ≠ V :                   ← While loop runs O(n) times
    v = H.min();   H.delete(v) :   ← In total, H.min() and H.delete()
    S = S ∪ {v}:                     are called O(n) times.
    For every edge (v, w) where w ∉ S :
        If d(v) + ℓ(v,w) < d(w):   ← Every edge appears at most
            d(w) = d(v) + ℓ(v,w)     once in this For loop.
            H.changekey(w, d(w))     In total, H.changekey()
        End If                       is called O(m) times.
    End For
End While
Return d[].
```

Overall runtime:

$$O(\underbrace{n\cdot\log n}_{\text{insert}} + \underbrace{n}_{\text{min}} + \underbrace{n\cdot\log n}_{\text{delete}} + \underbrace{m\log n}_{\text{changekey}} + \underbrace{m+n}_{\text{other operations like changing } S \text{ or } d(w)})$$
$$= O(m\cdot\log n).$$

Comments:

1. What if $\ell(u,v) < 0$? Dijkstra's algorithm no longer works!
2. Dijkstra $\approx$ continuous BFS.
3. If we use "for $v=1..n$" to find the v with the smallest $d(v)$: Runtime $O(n^2)$