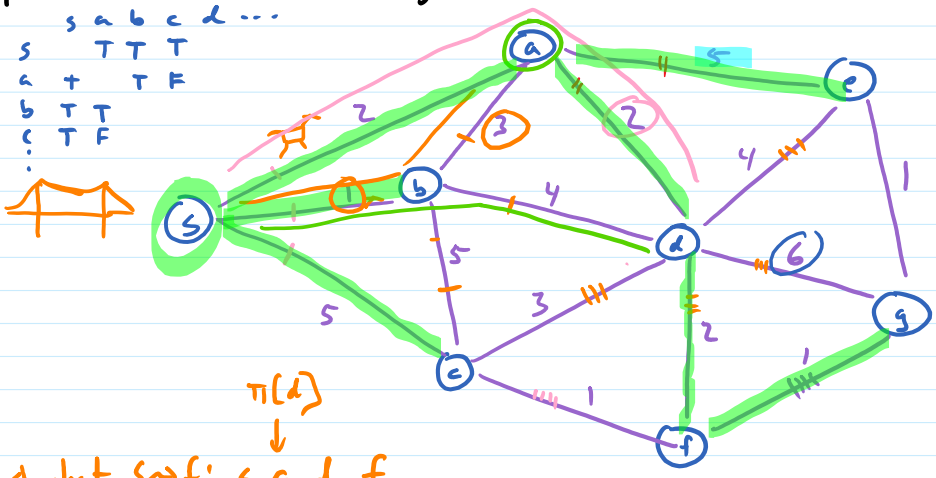
Given weighted graph with no negative-weight edges and start vertex s, find path of minimum total weight from s to all other vertices



	s	a	b	c	d	...
s	∞	2	1	5		
a	2	∞	5	∞		
b	1	5	∞			
c	5	∞				

Priority Queue

		prev = π	
1	s		0
3	a	s	2
2	b	s	1
5	c	s	5
4	d	b	4
7	e	a	7 = 2+5
6	f	d	6
8	g	d	~~10~~ 7

$\pi[d]$

shortest $s \rightsquigarrow f$: s a d f

$\pi[a]$ $\pi[f]$

s	b	a	d	c	f	e	g
0	1	2	4	5	6	7	7

Dijkstra(G, ℓ)

$S \leftarrow \{s\}$ "solved" vertices (know shortest path)

$d(s) \leftarrow 0$

while $S \neq V$

choose $v \notin S$ to minimize $d'(v) = \min_{\substack{u \in S \\ (u,v) \in E}} d(u) + \ell(u,v)$

cost of $s \rightsquigarrow u$ + cost of $u \rightarrow v$ = cost of $s \rightsquigarrow u \rightarrow v$

// # <!-- best weight of last path $s \rightsquigarrow v$ so far (using verts in S) -->

have priority queue priority of $v = d'(v)$

update d' = priority as we add vertices to S

$S \leftarrow S \cup \{v\}$

$d(v) = d'(v)$

no neg-weight edges

Given weighted G (directed or undirected), and a source vertex s, find min-weight path $s \rightsquigarrow v$ for all vertices v.

```
Dijkstra(G, l)
    S ← {s}
    d(s) ← 0
    while S ≠ V
        choose v ∉ S to minimize d'(v) = min_{u∈S, (u,v)∈E} d(u) + l(u,v)
        S ← S ∪ {v}
        d(v) = d'(v)
```

```
Q ← ∅
S ← ∅
for each vertex v
    π[v] ← NIL
    d'[v] ← ∞
d'[s] ← 0

for each vertex v
    enqueue(Q, v, d'[v])
while Q not empty
    v = extract_min(Q)
    d[v] = d'[v]
    S ← S ∪ {v}
    for (v,w) ∈ E where w ∉ S
        if d[v] + l(v,w) < d'[w]
            d'[w] = d[v] + l(v,w)
            π[w] = v
            decrease_priority(Q, w, d'[w])
```

s is 1st off Q

n iterations

use adj list to facilitate iterating through v's edges

$d[v] + l(v,w)$

$s \rightsquigarrow v \rightarrow w$

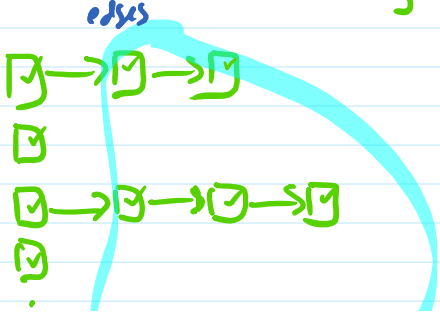
v is next-to-last on last path $s \rightsquigarrow w$ so far

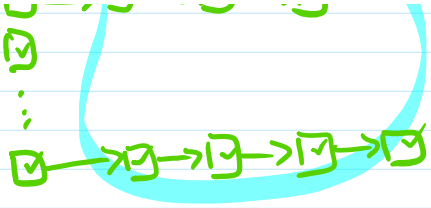
once per edge in worst case

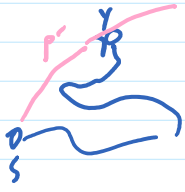
$O(n+m)$ for graph iterations

```
for each vertex v
    for each edge (v,w)
        do something
```

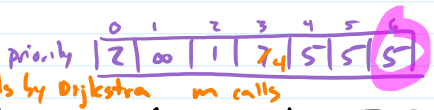
edges






Y
R
P
D
S

Priority Queue Implementations



n calls by Dijkstra m calls

	build	extract_min	decrease-priority	TOTAL
unsorted array	$\Theta(n)$	$\Theta(n)$ search entire array	$\Theta(1)$	$O(n^2+m) = O(n^2)$
sorted array	$O(n \log n)$	$\Theta(1)$ wrap-around style remove 1st element	$O(n)$ find elt to change, swap it to new loc	$O(n+mn) = O(mn)$ assuming all reachable
binary heap balanced BST	$O(n)$ $O(n \log n)$	$O(\log n)$	$O(\log n)$	$O(n \log n + m \log n) = O(m \log n)$ assuming all reachable
fib heap		$O(\log n)$ amortized	$O(1)$	$O(n \log n + m)$