1. Dijkstra's Algorithm

For Single Source Shortest Paths (SSSP)

Recall SSSP:

Input: A graph $G=(V,E)$ with non-negative weights on the edges, and a source $s \in V$.

Goal: Find a shortest path from s to any $v \in V$.

Remarks:

- We saw that BFS solves the case where all weights are 1.
- One can also consider negative weights, and there are algorithms for this extension, like Bellman-Ford, running in time $\Theta(|V| \cdot |E|)$.
- In some cases we might only care about $s \in V$ and one $t \in V$. The time of algorithms for this "single pair" variant is <u>not</u> asymptotically faster.

Dijkstra's Algorithm

Idea: Recall BFS uses a Queue to track the vertices to explore. Now use a Priority Queue instead!

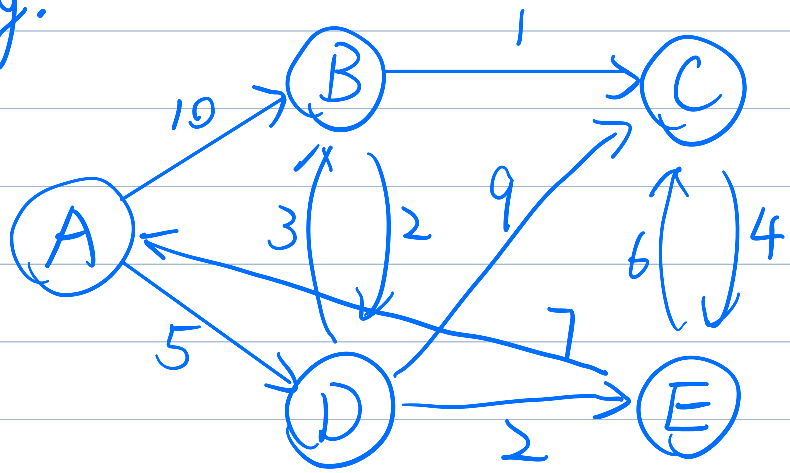
```
Dijkstra(G,w,s):
    For each v∈V:
        dist(v)=∞
        parent(v)=⊥
    dist(s)=0
    For each v∈V:
        PQ.Push(v, dist(v)) // Add v w/ dist(v) as the key
    While !PQ.IsEmpty():
        u=PQ.ExtractMin()
        For v∈Adj[u]:
            If dist(v)>dist(u)+w(u,v):
                dist(v)=dist(u)+w(u,v)
                parent(v)=u
                PQ.DecreaseKey(v, dist(v))
```

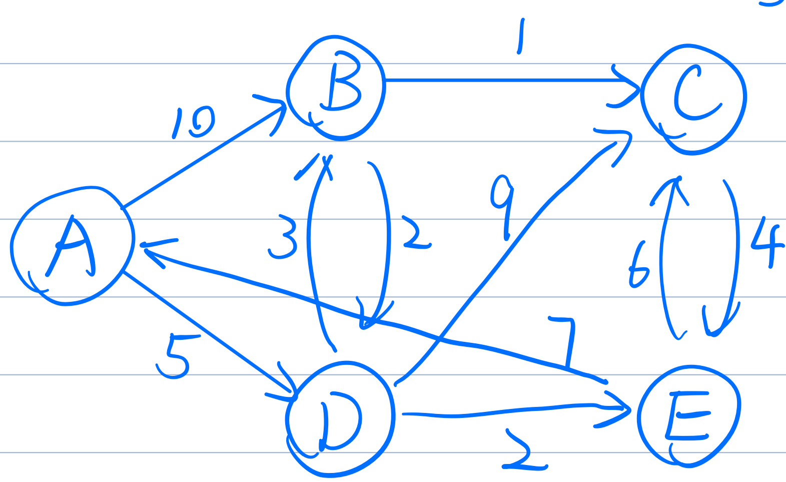
Very similar to Prim's we saw last lecture!

Q: Why can we run line 13 w/o checking if v is still in PQ?

A: It's guaranteed, otherwise dist(v) < dist(u).

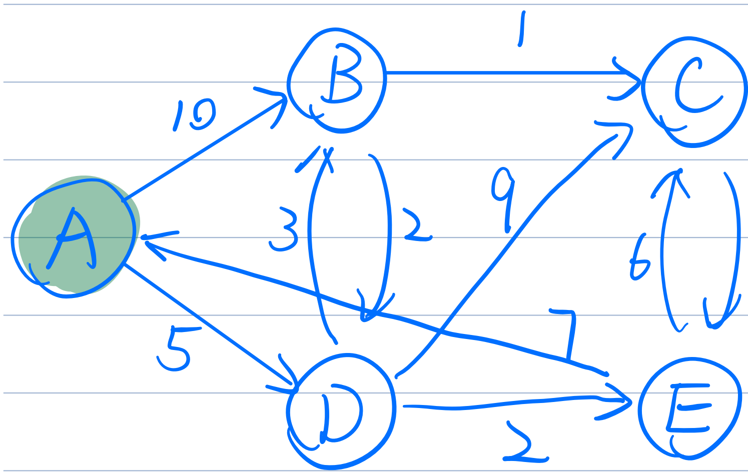
e.g.





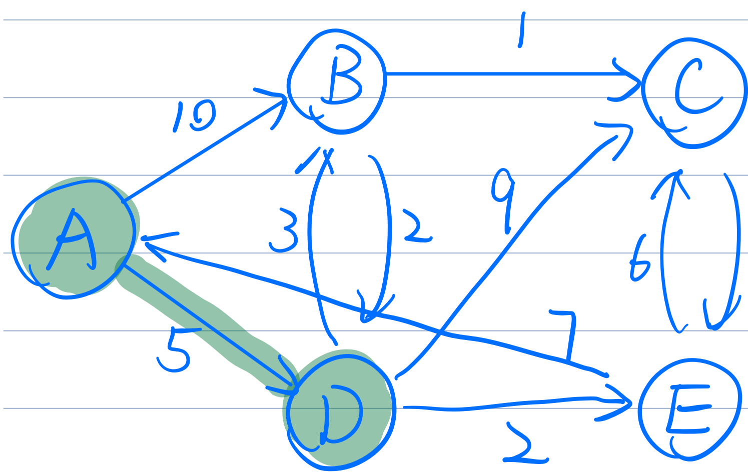
S

	parent	dist
A	⊥	0
B	⊥	∞
C	⊥	∞
D	⊥	∞
E	⊥	∞



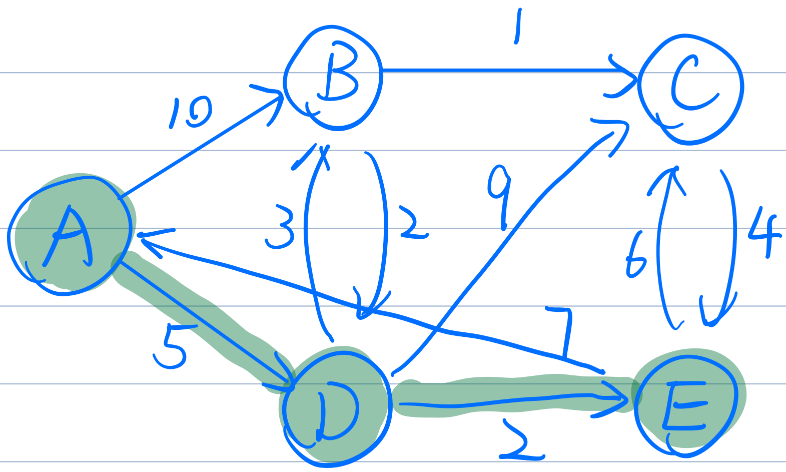
S

	parent	dist
A	$\perp$	0
B	A	10
C	$\perp$	∞
D	A	5
E	$\perp$	∞



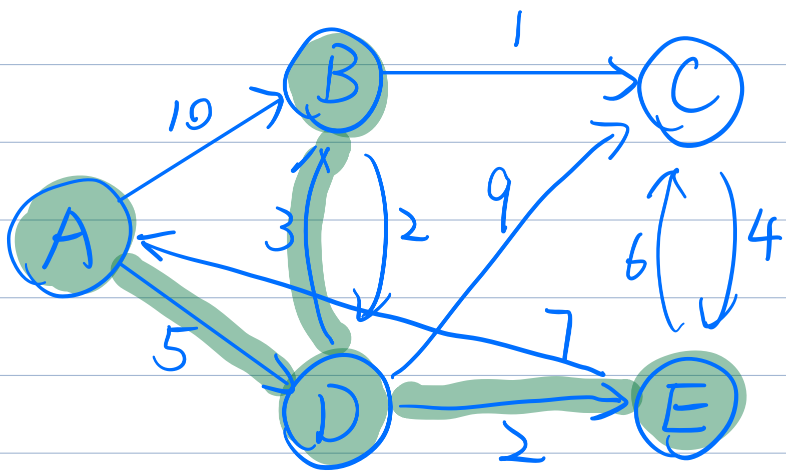
S

	parent	dist
A	$\perp$	0
B	D	8
C	D	14
D	A	5
E	D	7



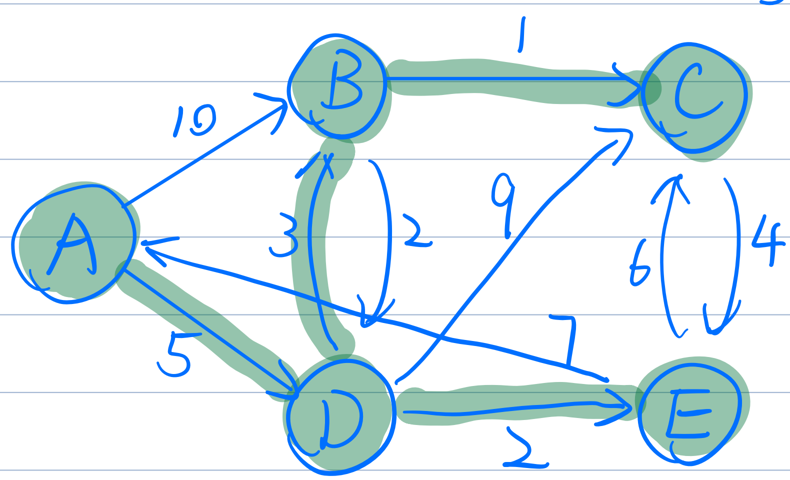
s

	parent	dist
A	⊥	0
B	D	8
C	E	13
D	A	5
E	D	7



s

	parent	dist
A	⊥	0
B	D	8
C	B	9
D	A	5
E	D	7



s → A

	parent	dist
A	$\perp$	0
B	D	8
C	B	9
D	A	5
E	D	7

<u>Runtime</u>: $|V| + |V| \times T(\text{Push}) + |V| \times T(\text{ExtMin}) + |E| \cdot T(\text{DecKey})$

w/ Binary Heap, runtime is $\Theta((|V|+|E|)\log|V|)$.
w/ Fibonacci Heap, runtime is $\Theta(|E|+|V|\log|V|)$.

<u>Correctness</u>: Let $\delta(u,v)$ be weight of shortest path from u to v. We prove the following theorem.

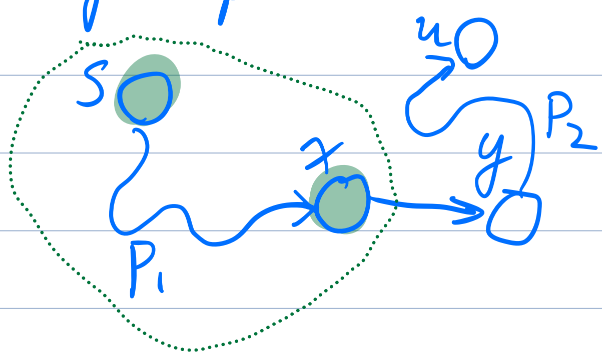
<u>Thm</u>: Dijkstra's algorithm terminates with $\text{dist}(u)=\delta(s,u)$ for all $u \in V$.

Proof Sketch:

Loop Invariant: each vertex v that is no longer in PQ (i.e. green) satisfies $dist(v) = \delta(s,v)$.

Initialization: Trivially True.

Maintenance: Assume the loop inv. holds for all green vertices, and now we are about to color u as green. Consider a shortest path from s to u and let x, y be two consecutive vertices along the path s.t. x is green and y is white.



Notice the following:

① $dist(x) = \delta(s,x)$: since x is green and by loop inv.

② $dist(y) = \delta(s,y)$: this comes from

$$\delta(s,y) = \delta(s,x) + w(x,y),$$

and when we explored Adj[x], we ensured $dist(y) \le dist(x) + w(x,y)$. Therefore:

$$
\begin{aligned}
dist(y) &\le dist(x) + w(x,y) \\
&= \delta(s,x) + w(x,y) \quad (\text{by } ①) \\
&= \delta(s,y)
\end{aligned}
$$

Moreover, $dist(y)$ can never be smaller than $\delta(s,y)$, so $dist(y) = \delta(s,y)$ exactly.

③ $dist(u) = \delta(s,u)$: Notice u is extracted from PQ, so we have

$$dist(u) \le dist(y) = \delta(s,y) \le \delta(s,u).$$

This implies $dist(u) = \delta(s,u)$ as $dist(u)$ never goes below $\delta(s,u)$.

Temination: At termination, PQ is empty. so all vertices are green, and hence satisfy the property $dist(v) = \delta(s,v)$.