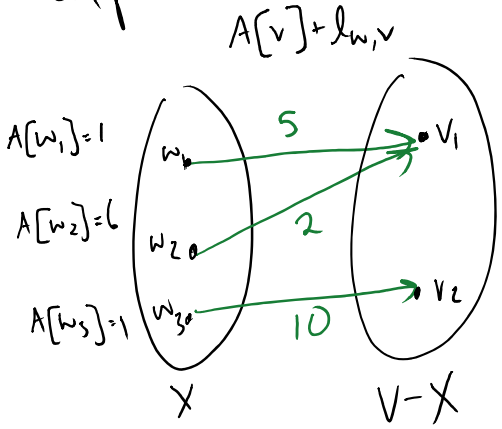
Instead

- Store vertices $v \in V - X$ in heap
- Key of v is

$$v.key = \min_{\substack{w \in X \\ (w,v) \in E}} \{A[w] + \ell_{w,v}\}$$

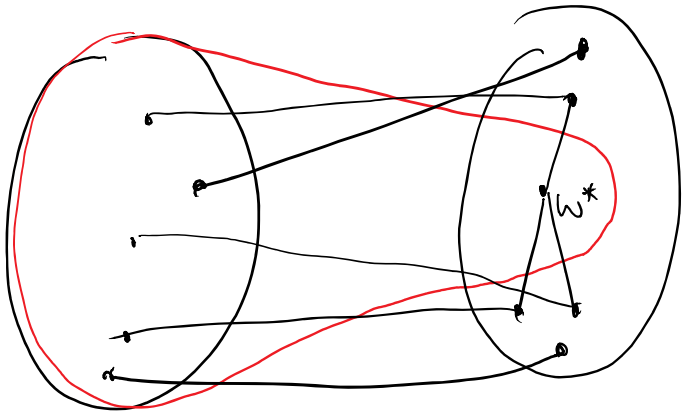


- For each element v,

$$v.p = \arg\min_{w \in X} \{A[w] + \ell_{w,v}\}$$

- If v is not directly connected to X, $k(v) = \infty$, $v.p = \emptyset$

Why better?



$\Longleftarrow$ Only need to check vertices in $A_G[w^*]$

X before w^* added

X after w^* added

Dijkstra Heap

$X[v] = 0$; $A[v] = \infty$; $B[v] = \emptyset$; $\forall\ v \in V$

$v.key = \infty$ for all $v \in V$

$v.p = \emptyset$ for all $v \in V$

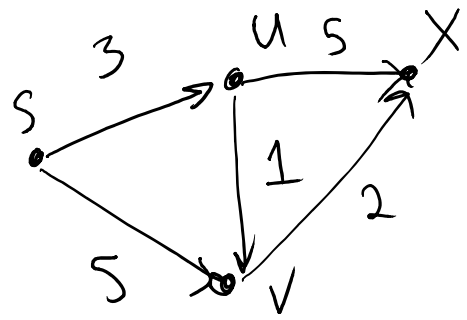
$s.key = 0$.

Heapify all $v \in V$

while (Heap is not empty)

- Let w = vertex with min key
- Remove w; $X[w] = 1$; $A[w] = w.key$;
- $B[w] = B[w.p] + (w.p, w)$;
- for $u \in A_G[w]$ && u not explored
  - Check if need to update $u.key$
  - If yes, remove and reinsert

ex:



H

	Round 1	Round 2	Round 3
s	0 | ϕ	X	
u	∞ | ϕ	3 | s	X
v	∞ | ϕ	5 | s	4 | u → X
x	∞ | ϕ	∞ | ϕ	8 | u → 6 | v

Dijkstra Heap

$X[v] = 0$; $A[v] = \infty$; $B[v] = \emptyset$; $\forall\, v \in V$

$v.key = \infty$ for all $v \in V$

$v.p = \emptyset$ for all $v \in V$

$s.key = 0$.

Heapify all $v \in V$

[while (Heap is not empty)

- Let w = vertex with min key
- Remove w; $X[w] = 1$; $A[w] = w.key$;
- $B[w] = B[w.p] + (w.p, w)$;
- for $u \in A_G[w]$ && u not explored
  - Check if need to update $u.key$
  - If yes, remove & reinsert]

How many times does this loop run?

How many times does this check happen over whole algorithm? What is cost?