S.KIMMEL

Dijkstra Heap

$X[v]=0;\ A[v]=\infty;\ B[v]=\phi;\ \forall\ v\in V$
$v.key=\infty$ for all $v\in V$
$v.p=\phi$ for all $v\in V$
$s.key=0.$

$O(n)$

Heapify all $v\in V$ $\quad O(n\log n)$

while (Heap is not empty)

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

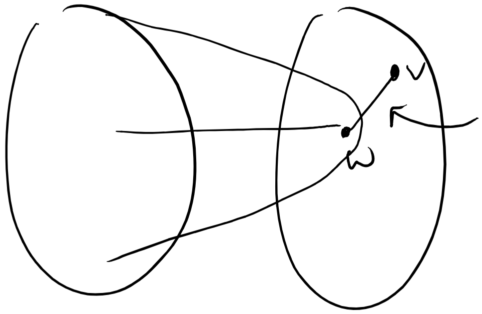
How many times does this loop run?
n times

$O(m)\rightarrow$ How many times does this check happen over whole algorithm?
What is cost?
$O(\log n)$

(See next page)

S.KIMMEL

Only need to update if



Only gets updated when w or v gets pulled into X. Only happens once for each edge.

Adding it all up:

$$O(n) + O(n\log n) + O(n\log n) + O(m\log n)$$

removal of elements

update of elements

$$\Rightarrow O((n+m)\log n)$$

$$O(m\log n)$$

Much better than FOR loop approach which was $O(nm)$