Time Complexity

- Initialization: \( O(1) \) or \( O(n) \)
- Each vertex placed in \( Q \) at most once
  - removed at most once

For each edge:
- all constant time except update (increase priority)

\[
\begin{align*}
\text{initialize PQ to have space for all } n \text{ vertices} & \quad O(n) \left( T_E(n) + T_q(n) \right) \\
\text{time to insert in } Q & \quad T_q(n) \\
\text{time to extract Max in } Q & \quad T_E(n) \\
\text{time to update} & \quad O(mT_U(n))
\end{align*}
\]
Dijkstra's + Prim's Algorithm

Greedy Tree Builder

Time Complexity

- Initialization: $O(1)$ or $O(n)$
- Each vertex placed in Q at most once + removed at most once: $O(n \left( \frac{T_{I}(n)}{T_{I}(n)} + T_{E}(n) \right)$
- For each edge: all constant time except update (increase priority): $O(m T_{O}(n))$

Initializes the PQ to have space for all $n$ vertices.
\[
O(nT_{\text{I}}(n) + nT_{\text{E}}(n) + mT_{\text{U}}(n)) \quad \text{Adj. List rep.}
\]

- Insert \(\max\) # vertices in PQ
- Extract Max
- Update (increase priority)

Binary heap

- \(T_{\text{I}}(n), T_{\text{E}}(n), T_{\text{U}}(n)\) \(O(\log n)\)
- Time \(O(m \log n)\) assume \(m \geq n\)

Fibonacci heap

- \(T_{\text{I}}(n) + T_{\text{U}}(n)\) when increase priority \(O(1)\) amortized
- Worst-case cost of \(O(m + n \log n)\)