17.3.7
Dijkstra’s algorithm
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action

![Graph showing Dijkstra algorithm in action](image_url)
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Example: Dijkstra algorithm in action
Improved Algorithm

1. Main work is to compute the $d'(s, u)$ values in each iteration.
2. $d'(s, u)$ changes from iteration $i$ to $i + 1$ only because of the node $v$ that is added to $X$ in iteration $i$.

Initialize for each node $v$, $dist(s, v) = d'(s, v) = \infty$
Initialize $X = \emptyset$, $d'(s, s) = 0$
for $i = 1$ to $|V|$ do
    // $X$ contains the $i - 1$ closest nodes to $s$,
    // and the values of $d'(s, u)$ are current
    Let $v$ be node realizing $d'(s, v) = \min_{u \in V - X} d'(s, u)$
    $dist(s, v) = d'(s, v)$
    $X = X \cup \{v\}$
    Update $d'(s, u)$ for each $u$ in $V - X$ as follows:
    $$d'(s, u) = \min\left( d'(s, u), \ dist(s, v) + \ell(v, u) \right)$$

Running time: $O(m + n^2)$ time.
1. $n$ outer iterations and in each iteration following steps
2. updating $d'(s, u)$ after $v$ is added takes $O(deg(v))$ time so total work is $O(m)$.
**Improved Algorithm**

1. **Main work** is to compute the $d'(s, u)$ values in each iteration.
2. $d'(s, u)$ changes from iteration $i$ to $i + 1$ only because of the node $v$ that is added to $X$ in iteration $i$.

```plaintext
Initialize for each node $v$, $\text{dist}(s, v) = d'(s, v) = \infty$
Initialize $X = \emptyset$, $d'(s, s) = 0$
for $i = 1$ to $|V|$ do
    // $X$ contains the $i - 1$ closest nodes to $s$,
    // and the values of $d'(s, u)$ are current
    Let $v$ be node realizing $d'(s, v) = \min_{u \in V - X} d'(s, u)$
    $\text{dist}(s, v) = d'(s, v)$
    $X = X \cup \{v\}$
    Update $d'(s, u)$ for each $u$ in $V - X$ as follows:
    $$d'(s, u) = \min\left(\text{dist}(s, v) + \ell(v, u)\right)$$
```

**Running time:** $O(m + n^2)$ time.

1. $n$ outer iterations and in each iteration following steps
2. updating $d'(s, u)$ after $v$ is added takes $O(\deg(v))$ time so total work is $O(m)$
Improved Algorithm

Initialize for each node $v$, $\text{dist}(s, v) = d'(s, v) = \infty$
Initialize $X = \emptyset$, $d'(s, s) = 0$
for $i = 1$ to $|V|$ do
  // $X$ contains the $i - 1$ closest nodes to $s$,
  // and the values of $d'(s, u)$ are current
  Let $v$ be node realizing $d'(s, v) = \min_{u \in V - X} d'(s, u)$
  $\text{dist}(s, v) = d'(s, v)$
  $X = X \cup \{v\}$
  Update $d'(s, u)$ for each $u$ in $V - X$ as follows:
  $$d'(s, u) = \min\left(d'(s, u), \text{dist}(s, v) + \ell(v, u)\right)$$

Running time: $O(m + n^2)$ time.

1. $n$ outer iterations and in each iteration following steps
2. updating $d'(s, u)$ after $v$ is added takes $O(\deg(v))$ time so total work is $O(m)$ since a node enters $X$ only once
3. Finding $v$ from $d'(s, u)$ values is $O(n)$ time
Dijkstra’s Algorithm

1. eliminate \( d'(s, u) \) and let \( \text{dist}(s, u) \) maintain it
2. update \( \text{dist} \) values after adding \( v \) by scanning edges out of \( v \)

\[
\begin{align*}
\text{Initialize for each node } v, \quad & \text{dist}(s, v) = \infty \\
\text{Initialize } X = \emptyset, \quad & \text{dist}(s, s) = 0 \\
\text{for } i = 1 \text{ to } |V| \text{ do} \\
& \quad \text{Let } v \text{ be such that } \text{dist}(s, v) = \min_{u \in V - X} \text{dist}(s, u) \\
& \quad X = X \cup \{v\} \\
& \quad \text{for each } u \text{ in } \text{Adj}(v) \text{ do} \\
& \quad \quad \text{dist}(s, u) = \min(\text{dist}(s, u), \text{dist}(s, v) + \ell(v, u))
\end{align*}
\]

Priority Queues to maintain \( \text{dist} \) values for faster running time

1. Using heaps and standard priority queues: \( O((m + n) \log n) \)
2. Using Fibonacci heaps: \( O(m + n \log n) \).
Dijkstra’s Algorithm

1. Eliminate $d'(s, u)$ and let $dist(s, u)$ maintain it
2. Update $dist$ values after adding $v$ by scanning edges out of $v$

```
Initialize for each node $v$, $dist(s, v) = \infty$
Initialize $X = \emptyset$, $dist(s, s) = 0$
for $i = 1$ to $|V|$ do
    Let $v$ be such that $dist(s, v) = \min_{u \in V \setminus X} dist(s, u)$
    $X = X \cup \{v\}$
    for each $u$ in $\text{Adj}(v)$ do
        $dist(s, u) = \min\left(dist(s, u), dist(s, v) + \ell(v, u)\right)$
```

Priority Queues to maintain $dist$ values for faster running time

1. Using heaps and standard priority queues: $O((m + n) \log n)$
2. Using Fibonacci heaps: $O(m + n \log n)$. 

59 / 76
THE END

... (for now)