Prim’s algorithm using adjacency lists & binary heap: $\Theta(|E| \log n)$

$\rightarrow$ better if $|E| = O(n^2 / \log n)$

Complexity of adjacency-list based algorithms: $\Omega(n + |E|)$

- difficult to achieve even work distribution & low comm overhead for random sparse graphs
- focus on grid-like graphs

![Figure 7.15 Examples of sparse graphs: (a) a linear graph, in which each vertex has two incident edges; (b) a grid graph, in which each vertex has four incident vertices; and (c) a random sparse graph.](image-url)
Johnson\textunderscore SSPP $(V, E, s)$

$Q := V$

for all $v \in Q$ do

$\ell[v] := \infty$

$\ell[s] := 0$

while $Q \neq \emptyset$ do

$u := \text{extract\_min}(Q)$;

for each $v \in \text{Adj}[u]$ do

if $v \in Q$ and $\ell[u] + \omega(u, v) < \ell[v]$ then

$\ell[v] := \ell[u] + \omega(u, v)$

\begin{figure}[h]
\centering
\begin{subfigure}[b]{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{graph_a.png}
\caption{(a)}
\end{subfigure}
\hfill
\begin{subfigure}[b]{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{graph_b.png}
\caption{(b)}
\end{subfigure}
\caption{A street map (a) can be represented by a graph (b). In the graph shown in (b), each street intersection is a vertex and each edge is a street segment. The vertices of (b) are the intersections of (a) marked by dots. Copyright (c) 1994 Benjamin/Cummings Publishing Co.}
\end{figure}

- $Q$ implemented as a binary min-heap
  - each update in $O(\log n)$

- Complexity = $O(|E| \log n)$

- To parallelize, distribute the priority queue
while \( P_i \) is extracting \( v \) from \( Q_i \), \( P_j \) may extract \( u \) from \( Q_j \) such that \( l[v] < l[u] \)

\[ \rightarrow \text{if } \exists \text{ path } (v, u), \text{ cost of } (s-v, u) < \text{ cost } (s, u) \]

\[ \Rightarrow \text{vertices may not be extracted in nondecreasing order of } l \]

\[ \Rightarrow \text{computation must be redone} \]

**source** = \( 'a' \); each vertex assigned to separate processor.

\( P_{1,0} \) picks path \( <a, d> \) from local \( Q_s \).

Later, it discovers shorter path \( <a, b, e, d> \)

---

**Figure 7.17** A grid graph.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
**Figure 7.18** The wave of activity in the priority queues.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

---

**Figure 7.19** Mapping the grid graph (a) onto a mesh (b) by using the block-checkerboard mapping. In this example, $n = 16$ and $\sqrt{p} = 4$. The shaded vertices are mapped onto the shaded processor.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- Assign $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ vertices to each processor, given $n \times n$ grid
- # busy processors = # processors intersected by wave
- Let $W$ be overall work done by sequential algorithm

\[
S_{\text{max}} = \frac{W}{W/\sqrt{p}} = \sqrt{p} \quad ; \quad E_{\text{max}} = 1/\sqrt{p}
\]
Figure 7.20 Mapping the grid graph (a) onto a mesh (b) by using the cyclic-checkerboard mapping. In this example, \( n = 16 \) and \( \sqrt{p} = 4 \). The shaded graph vertices are mapped onto the correspondingly shaded mesh processors.

Figure 7.21 Mapping the grid graph (a) onto a linear array of processors (b). In this example, \( n = 16 \) and \( p = 4 \). The shaded vertices are mapped onto the shaded processor.

Block-striped mapping: on avg, \( \frac{p}{2} \) processors are busy

\[
S = \frac{p}{2} \quad ; \quad E = \frac{1}{2}
\]

However, cannot use more than \( O(n) \) proce
Figure 7.22  The number of busy processors as the computational wave propagates across the grid graph.
Copyright (c) 1994 Benjamin/Cummings Publishing Co.