procedure PRIM_MST (V, E, w, r)

V_T := \{r\}

d[r] := 0

for all \ v \in (V - V_T) do

if edge \ (r, v) \ exists \ set \ d[v] := w(r, v)
else \ set \ d[v] := \infty

while \ V_T \neq V \ do

find vertex \ u \ such \ that \ d[u] = \min \{d[v] \mid \ v \in (V - V_T)\}

V_T := V_T \cup \{u\}

for all \ \ v \in (V - V_T) \ do

\[ d[v] = \min \{d[v], w(u,v)\} \]

Dijkstra's single-source shortest paths algorithm is identical to Prim's MST algo, except for each \ v \in (V - V_T)

→ Dijkstra's stores \ l[u] \ the min cost to reach \ u \ from \ s \ via \ V_T

i.e. \ \ l[v] = \min \{l[v], l[u] + w(u,v)\} \ is \ the \ update \ step

→ Prim's stores \ d[u] \ cost of min-cost edge connecting vertex in \ V_T \ to \ u

* Parallel formulation of both algorithms are the same

* Dijkstra's all-pairs shortest paths: \ \Theta(n^3)

→ Source-partitioned parallel formulation: uses \ n \ processors only

\[ T_p = \Theta(n^2) \ ; \ S = \Theta(n^3) / \Theta(n^2) \ ; \ E = \Theta(1) \ ; \text{good efficiency} \implies \Theta(p^3) \]

very poor scalability

→ Use source-parallel formulation: \ p \ (>n) \ processors

\[ \frac{p}{n} \ \text{procs/partition; each partition runs the 1-source shortest path algo} \]

Can use \ \Theta(n^2) \ processors efficiently
Figure 7.5  Prim's minimum spanning tree algorithm. The MST is rooted at vertex $b$. During each iteration, the minimum cost edge connecting a vertex in $V_T$ to a vertex in $V - V_T$ is selected and the corresponding vertex is added to $V_T$ (shown shaded in the distance array $d$). The $d[v]$ values of the vertices in $V - V_T$ are then updated.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
Parallel Formulation: block-striped partitioning

- In each iteration, $P_i$ computes $d_i[u] = \min\{d_i[v] \mid v \in (V_i - V_i) \cap V_i\}$
- Then do single-node accumulation at $P_0$ to find global minimum $d_i[u]
- P_0$ does $1 \rightarrow$ all BC of $u$
- Each $P_i$ updates values of $d_i[u]$ for its local vertices
- (Each $P_i$ needs to store the columns of the weighted adjacency matrix for $V_i$ assigned to it.
- Space = $\Theta(n^2/p)$; computation in each iteration = $\Theta(n/p)$

![Diagram showing partitioning of the distance array $d$ and the adjacency matrix $A$ among $p$ processors.](image)

Figure 7.6: The partitioning of the distance array $d$ and the adjacency matrix $A$ among $p$ processors. Copyright (c) 1994 Benjamin/Cummings Publishing

\[
\begin{align*}
\text{Computation} & \quad \Theta\left(\frac{n^2}{p}\right) + \Theta(n \log p) \\
\text{Communication for cost-optimality, } (p \log p)/n & \quad = O(1) \\
\text{Isoefficiency (due to communication)} & \quad = \Theta(p^2 \log^2 p) \\
HC: T_p & \quad = \Theta\left(\frac{n^2}{p}\right) + \Theta(n \log p) \\
S & \quad = \frac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n \log p)} \\
E & \quad = \frac{1}{1 + \Theta\left(\frac{p \log p}{n}\right)} \\
\text{Mesh: } T_p & \quad = \Theta\left(\frac{n^2}{p}\right) + \Theta(n \sqrt{p}) \\
S & \quad = \frac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n \sqrt{p})} \\
E & \quad = \frac{1}{1 + \Theta\left(\frac{p^{1.5}}{n}\right)} \\
\end{align*}
\]

For cost-optimality, $p^{1.5}/n = O(1) \Rightarrow O(n^{1.5})$ processors can be used efficiently

Isoefficiency (due to communication) = $\Theta(p^3)$
\[ d_{ij}^{(k)} = \min_{1 \leq m \leq n} \{ d_{im}^{(k-1)} + \omega(v_m, v_j) \} \]

- \( D^{(k)} \) computed from \( D^{(k-1)} \) & \( A \) using modified matrix multiplication
  \[ c_{ij} = \min_{k=1}^{n} \{ a_{i,k} + b_{k,j} \} \], instead of \( \sum_{k=1}^{n} a_{i,k} \times b_{k,j} \)

- Total \( \lceil \log (n-1) \rceil \) steps, i.e., \( A^1, A^2, A^3, \ldots \)

- Complexity = \( \Theta(n^3 \log n) \) \( \Rightarrow \) not optimal, but has a high degree of parallelism

![Graph](image)

\[
A^1 = \begin{pmatrix}
0 & 2 & 3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 0 & 1 & 2 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0 & \infty & 2 & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 0 & \infty & 1 & \infty & 1 & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\end{pmatrix}
\]

\[
A^2 = \begin{pmatrix}
0 & 2 & 3 & 4 & 5 & 3 & \infty & \infty & \infty & \infty \\
\infty & 0 & 1 & 2 & 3 & 4 & 3 & \infty & \infty & \infty \\
\infty & \infty & 0 & 1 & 2 & 3 & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 0 & 1 & 2 & 3 & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \\
\end{pmatrix}
\]

\[
A^4 = \begin{pmatrix}
0 & 2 & 3 & 4 & 5 & 3 & 5 & 6 & 5 & \infty \\
\infty & 0 & 1 & 2 & 3 & 4 & 3 & 5 & 6 & 5 \\
\infty & \infty & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 5 \\
\infty & \infty & \infty & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & 4 & 5 \\
\infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & 4 \\
\infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \\
\end{pmatrix}
\]

\[
A^8 = \begin{pmatrix}
0 & 2 & 3 & 4 & 5 & 3 & 5 & 6 & 5 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 5 \\
\infty & 0 & 1 & 2 & 3 & 4 & 3 & 5 & 6 & 5 & 0 & 1 & 2 & 3 & 4 & 5 \\
\infty & \infty & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 5 & \infty & 0 & 1 & 2 & 3 & 4 & 5 \\
\infty & \infty & \infty & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \infty & \infty & 0 & 1 & 2 & 3 & 4 \\
\infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & 4 & 5 & \infty & \infty & \infty & 0 & 1 & 2 & 3 \\
\infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & 4 & \infty & \infty & \infty & \infty & 0 & 1 \\
\infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & 3 & \infty & \infty & \infty & \infty & \infty & 0 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & 2 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty \\
\end{pmatrix}
\]

Figure 7.7 An example of the matrix-multiplication-based all-pairs shortest paths algorithm.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

*On a HC w/ \( n^2 \) processors, \( n \times n \) matrix mult in \( \Theta(\log n) \) [see MM chap.]*

\[
S = \frac{\Theta(n^3)}{\Theta(\log^2 n)} \quad \Rightarrow \quad E = \frac{1}{\Theta(\log^2 n)}
\]
Source-parallel formulation of Dijkstra's algorithm

\[ T_p = \frac{\Theta\left(\frac{n^3}{p}\right)}{\text{computation}} + \frac{\Theta\left(\sqrt{np}\right)}{\text{communication}} \]

\[ S = \frac{\Theta(n^3)}{\Theta\left(\frac{n^3}{p}\right) + \Theta\left(\sqrt{np}\right)} \]

\[ E = \frac{1}{1 + \Theta\left(\frac{p^{1.5}}{n^{2.5}}\right)} \]

For cost-optimality, \( p^{1.5}/n^{2.5} = O(1) \) \( \Rightarrow \) can use \( O(n^{1.66}) \) processors efficiently

Isoefficiency comm = \( \Theta(p^{1.8}) \)

Isoefficiency concurrency = \( \Theta(p^{1.5}) \)

---

**Figure 7.8** Partitioning a \( \sqrt{p} \times \sqrt{p} \) mesh into \( n \) submeshes, each of size \( \sqrt{p/n} \times \sqrt{p/n} \). In this example, \( p = 16 \) and \( n = 4 \). Each of the \( \sqrt{p/n} \times \sqrt{p/n} = 2 \times 2 \) meshes solves the single-source shortest paths problem for a given source vertex. In this example, each submesh is marked by the source vertex of its single-source algorithm.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
FLOYD's algorithm

\[ d_{ij}^{(k)} = \begin{cases} \omega(v_i, v_j) & \text{if } k = 1 \\ \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \} & \text{if } k \geq 1 \end{cases} \]

Solve bottom-up in the order of increasing values of \( k \); \( \Theta(n^3) \)

\( D^{(0)} = A \)
for \( k = 1 \) to \( n \) do
  for \( i = 1 \) to \( n \) do
    for \( j = 1 \) to \( n \) do
      \[ d_{ij}^{(k)} := \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \} \]

\[ \frac{i}{\sqrt{p}}, \frac{j}{\sqrt{p}} \]

(a) Matrix \( D^{(k)} \) partitioned by block checkerboard-ing into \( \sqrt{p} \times \sqrt{p} \) subblocks, and (b) the square subblock of \( D^{(k)} \) assigned to processor \( P_{i,j} \).

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Parallel formulations: to compute portion of \( D^{(k)} \), need corresp. segments of \( k^{th} \) row and column of \( D^{(k-1)} \)
\* **Kth iteration:** each of the \( \sqrt{P} \) processors containing part of the \( K \)th row (column) send it to the \( \sqrt{P}-1 \) processors in the same column (row).

\* Embed mesh into \( p \)-proc HC with cut-through routing

\[ \rightarrow \sqrt{P} \times \sqrt{P} \text{ virtual mesh} \rightarrow p \text{-proc HC} \]

\[ \rightarrow \text{each row or column of mesh } \equiv \text{HC of } \sqrt{P} \text{ proc} \]

\[ \rightarrow \text{in each iteration, } K \text{th row & } K \text{th column do } 1 \rightarrow \text{all BC along a column or row} \]

\[ \text{(each such proc has } \frac{n}{\sqrt{P}} \text{ elements of } K \text{th row or col)} \]

\[ \Rightarrow \text{BC requires } \Theta\left(\frac{n \log p}{\sqrt{P}}\right) \text{ time} \]

![Diagram](a) ![Diagram](b)

Figure 7.10 (a) Communication patterns used in the blockcheckerboard partitioning. When computing \( d_{ij}^{(k)} \), information must be sent to the highlighted processor from two other processors along the same row and column. (b) The row and column of \( \sqrt{P} \) processors that contain the \( K \)th row and column send them along processor columns and rows.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

\[ T_p = \Theta\left(\frac{n^3}{P}\right) + \Theta\left(\frac{n^2 \log p}{\sqrt{P}}\right) \]

For cost-optimality, \( \sqrt{P} \log p/n = O(1) \)

\[ S = \frac{\Theta(n^3)}{\Theta(n^3/P) + \Theta(n^2 \log p/\sqrt{P})} \]

\[ E = \frac{1}{1 + \Theta\left(\frac{\sqrt{P} \log p}{\sqrt{P}}\right)} \]

\[ \text{Isoefficiency} \left(\text{comm}\right) = \Theta(p^{1.5} \log^3 p) \]

\[ \text{Isoefficiency} \left(\text{compute}\right) = \Theta(p^{1.5}) \]
Prim's algorithm using adjacency lists & binary heap: $\Theta(|E| \log n)$

$\rightarrow$ better if $|E| \approx O\left(\frac{n^2}{\log n}\right)$

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

![Graph examples](image)

**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.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.
\[ \text{Johnson-SSSP}(V, E, s) \]
\[ Q := V \]
\[ \text{for all } v \in Q \text{ do} \]
\[ l[v] := \infty \]
\[ l[s] := 0 \]
\[ \text{while } Q \neq \emptyset \text{ do} \]
\[ u := \text{extract-min}(Q); \]
\[ \text{for each } v \in \text{Adj}[u] \text{ do} \]
\[ \text{if } v \in Q \text{ and } l[u] + \omega(u, v) < l[v] \text{ then} \]
\[ l[v] := l[u] + \omega(u, v) \]

\textbf{Figure 7.16} 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.

- \( Q \) implemented as a binary min-heap
- each update in \( O(\log n) \)
- Complexity = \( \Theta(|E| \log n) \)
- To parallelize, distribute the priority queue
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

\[
S_{max} = \frac{W}{w/\sqrt{p}} = \sqrt{p} \quad ; \quad E_{max} = 1/\sqrt{p}
\]
Each processor responsible for vertices that belong to different parts of the grid graph remain busy most of the time, but higher communication is necessary each time \( p_i \) extracts a node from \( Q_i \).

**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.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

**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.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

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

\[
S = \frac{p}{2} \quad \text{and} \quad E = \frac{1}{2}.
\]

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 (r) 1994 Benjamin/Cummings Publishing Co.