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 \( d[u] \) values of the vertices in \( V - V_T \) are then updated.

Copyright (c) 1994 Benjamin/Cummings Publishing Co.
procedure PRIM_MST (V,E,w,r)

V_T := {r}

d[r] := 0

for all v E (V-V_T) do
    if edge (r,v) exists set d[v] := w(r,v)
    else set d[v] := infinity

while V_T != V do
  find vertex u such that d[u] = min{d[v] | v E (V-V_T)}
  V_T := V_T U {u}

for all v E (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 algorithm except for each v E (V-V_T)

→ Dijkstra’s stores l[u], the min cost to reach u from s via V_T
  ie 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: Θ(n^3)

→ Source-partitioned parallel formulation: uses n processors only

T_p = Θ(n^2) ; S = Θ(n^2)/Θ(n^2) ; E = Θ(1) ; Isoefficiency = Θ(p^-3)

very poor scalability

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

Can use Θ(n^2) processors efficiently

T_p = Θ(n^3/p) + Θ(n log p)

Isoefficiency (communication) = Θ((p log p)^{1/5})

E = \frac{1}{1+Θ(p log p/n^2)}

Isoefficiency (concurrency) = Θ(p^{-1.5})
Parallel Formulation: block-striped partitioning

- In each iteration, \( P_i \) computes \( d_i[u] = \min \{ d_i[v] \mid v \in (V - V_T) \cap V_i \} \)
- 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 \( d[i] \) 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 of partitioning](image)

Figure 7.6 The partitioning of the distance array \( d \) and the adjacency matrix \( A \) among \( p \) processors.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

\[
\begin{align*}
HC: \quad T_P & = \Theta\left(\frac{n^2}{P}\right) + \Theta(n \log p) \\
S & = \frac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n \log p)} \\
E & = \frac{1}{1 + \Theta((p \log p)/n)}
\end{align*}
\]

For cost-optimality, \( (p \log p)/n = O(1) \)

Isoefficiency (due to communication) is \( \Theta(p^2 \log^2 p) \)

\[
\begin{align*}
Mesh: \quad T_P & = \Theta\left(\frac{n^2}{p}\right) + \Theta(n \sqrt{P}) \\
S & = \frac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n \sqrt{P})} \\
E & = \frac{1}{1 + \Theta\left(\frac{p^{1.5}}{n}\right)}
\end{align*}
\]

For cost-optimality, \( p^{1.5}/n = O(1) \)

\( O(n^{2/3}) \) processors can be used efficiently

Isoefficiency (due to communication) = \( \Theta(p^2) \)
Source-parallel formulation of Dijkstra's algorithm

→ single-source algo. on each submesh: \( T_p = \Theta\left(\frac{n^3}{p}\right) + \Theta\left(\sqrt{n}p\right) \)

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

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

For cost-optimality: \( \frac{p^{1.5}}{n^{2.5}} = O(1) \Rightarrow \text{can use } O(n^{1.65}) \text{ processors efficiently} \)

Isoefficiency \(\text{communicate} = \Theta(p^{1.8}) \)

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

\[ \sqrt{\frac{p}{n}} \]

\[ \frac{p}{n} \]

Figure 7.8: Partitioning a \( \sqrt{p} \times \sqrt{p} \) mesh into \( n \) submeshes, each of size \( \sqrt{\frac{p}{n}} \times \sqrt{\frac{p}{n}} \). In this example, \( p = 16 \) and \( n = 4 \). Each of the \( \sqrt{\frac{p}{n}} \times \sqrt{\frac{p}{n}} = 2 \times 2 \) 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.

Hypercube:

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

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

\[ E = \frac{1}{1 + \Theta\left(\frac{n \log p}{n^2}\right)} \]

For cost optimality: \( \frac{n \log p}{n^2} = \Theta(1) \)

Iso-efficiency \(\text{communicate} = \Theta\left(\left(\frac{n \log p}{n^2}\right)^{3/2}\right) \)

Iso-efficiency \(\text{concurrency} = \Theta\left(p^{3/4}\right) \)

// \( p > n \), each process handles \( \frac{p}{n} \) processes.
\[ d_{ij}^{(k)} = \min_{1 \leq m \leq n} \{ d_{im}^{(k-1)} + \omega(v_m, v_j) \} \]

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

- Total \( \frac{\log(n-1)}{\log(2)} \) steps, i.e., \( A^1, A^2, A^3, \ldots \)

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

\[ D^{(0)} = A \]

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

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

**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/ \( n^3 \) processors, \( n \times n \) matrix mult in \( \Theta(\log n) \) \[ \text{[see MM chapt]} \]

\[ S = \frac{\Theta(n^3)}{\Theta(\log^2 n)} \quad ; \quad E = \frac{1}{\Theta(\log^2 n)} \]
FLOYD's algorithm

\[ d_{ij}^{(k)} = \begin{cases} \omega(u_i, v_j) & \text{if } k = 0 \\ \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \} & \text{if } k \geq 1 \end{cases} \]

* 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

\[ d_{ij}^{(k)} := \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \} \]

Figure 7.9 (a) Matrix \( D^{(k)} \) partitioned by block checkboard- ing into \( \sqrt{p} \times \sqrt{p} \) subblocks, and (b) the square subblock of \( D^{(k)} \) assigned to processor \( P_{i,j} \).

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

Parallel formulations: to compute portion of \( D^{(k)} \), need corresponding segments of \( k^{th} \) row and column of \( D^{(k-1)} \)
• $k^{th}$ 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.
  → $\sqrt{p} \times \sqrt{p}$ virtual mesh → $p$-proc HC
  → each row or column of mesh = HC of $\sqrt{p}$ proc
  → in each iteration, $k^{th}$ row & $k^{th}$ column do 1 → all BC along a column or row.

  (each such proc has $n/\sqrt{p}$ elements of $k^{th}$ row or col)
  → BC requires $\Theta((n \log p)/\sqrt{p})$ time.

Figure 7.10 (a) Communication patterns used in the block-checkerboard partitioning. When computing $d_{k,i}^{(k)}$, 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.

\[ T_p = \Theta(n^3/p) + \Theta(n^2 \log p) \]
\[ S = \frac{\Theta(n^3)}{\Theta(n^3) + \Theta((n^2 \log p)/\sqrt{p})} \]
\[ E = \frac{1}{1 + \Theta((\sqrt{p} \log p)/n)} \]

With pipelining, $T_p = \Theta(n^3/p) + \Theta(n)$

\[ \text{For cost-optimality, } \sqrt{p} \log p/n = o(1) \]
\[ \text{Isoefficiency} = \Theta(p^{1/5} \log^3 p) \]
\[ \text{Isoefficiency} = \Theta(p^{1/5}) \]
\[ W = o(n^3); \# \text{procs} = O(n^2) = p \]