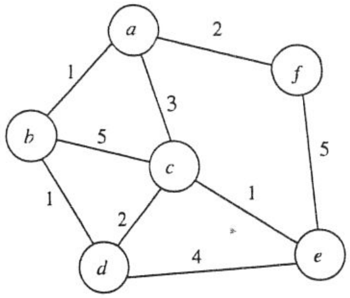
• $G(V, E, w, r)$

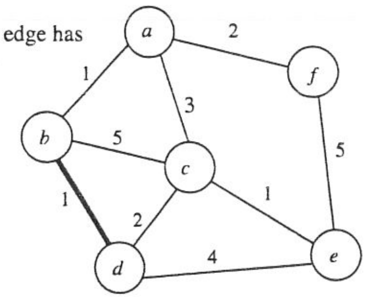
(a) Original graph



	a	b	c	d	e	f
d[]	1	0	5	1	∞	∞

	a	b	c	d	e	f
a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

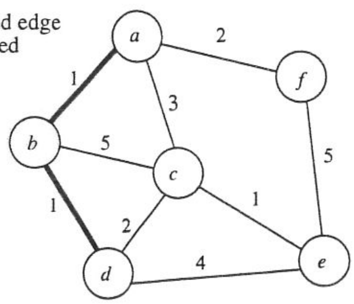
(b) After the first edge has been selected



	a	b	c	d	e	f
d[]	1	0	2	1	4	∞

	a	b	c	d	e	f
a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

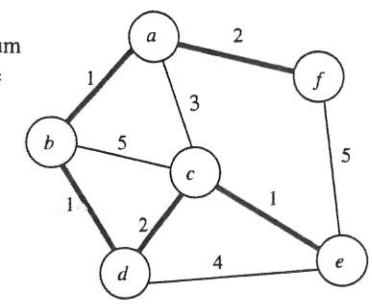
(c) After the second edge has been selected



	a	b	c	d	e	f
d[]	1	0	2	1	4	2

	a	b	c	d	e	f
a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

(d) Final minimum spanning tree



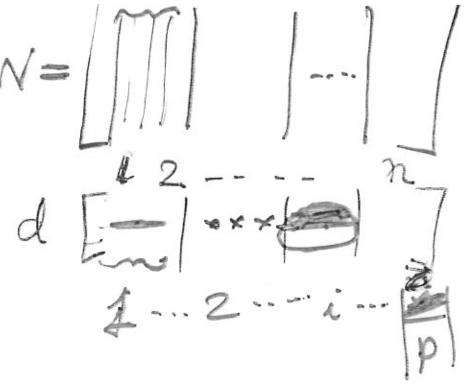
	a	b	c	d	e	f
d[]	1	0	2	1	1	2

	a	b	c	d	e	f
a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

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.

• $\Theta(n^2)$

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, <u>except</u> for each $v \in (V - V_T)$

→ Dijkstra's <u>stores</u> $l[u]$, the min cost to reach u from s via V_T
  <u>ie</u> $l[v] = \min\{l[v], l[u] + w(u, v)\}$ is the update step

→ Prim's <u>stores</u> $d[u]$, cost of min-cost edge connecting vertex in V_T to u

• Parallel formulation of both <u>algorithms</u> are the same

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

→ <u>Source-partitioned parallel formulation</u> : uses <u>n processors</u> only
  $T_p = \Theta(n^2)$; $S = \Theta(n^3)/\Theta(n^2)$; $E = \Theta(1)$; $\text{Isoefficiency}_{\text{concurr}} = \Theta(p^3)$
  $\parallel p \approx \Theta(W^{1/3})$
  very poor scalability

→ Use <u>source-parallel formulation</u> : <u>p $(> n)$ processors</u>
  $\frac{p}{n}$ procs / partition ; each partition runs the 1-source shortest path algo

Can <u>use $\Theta(n^2)$ processors efficiently</u>

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

$E = \frac{1}{1 + \Theta(p \log p / n^2)}$

Isoefficiency (communication) $= \Theta((p \log p)^{1.5})$

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

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→all BC of u
- Each P_i updates values of $d[v]$ 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 $= \Theta(n/p)$

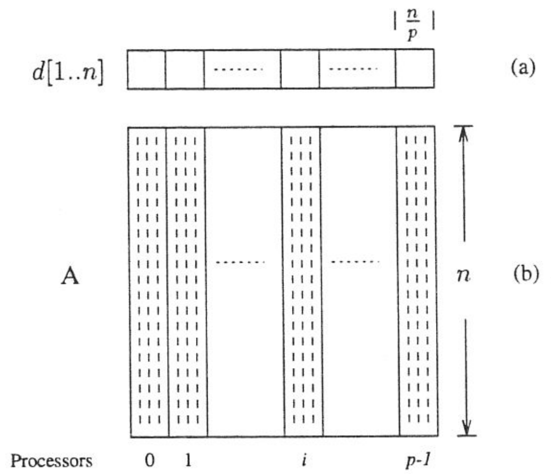


Figure 7.6 The partitioning of the distance array d and the adjacency matrix A among p processors. Copyright (r) 1994 Benjamin/Cummings Publishing Co.

HC: $T_P = \overbrace{\Theta\left(\frac{n^2}{p}\right)}^{\text{computation}} + \overbrace{\Theta(n \log p)}^{\text{communication}}$

$S = \dfrac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n \log p)}$

$E = \dfrac{1}{1 + \Theta((p \log p)/n)}$

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

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

Mesh: $T_P = \Theta(n^2/p) + \Theta(n\sqrt{p})$

$S = \dfrac{\Theta(n^2)}{\Theta(n^2/p) + \Theta(n\sqrt{p})}$

$E = \dfrac{1}{1 + \Theta(p^{1.5}/n)}$

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

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

Isoefficiency (due to communication) $= \Theta(p^3$

Source-parallel formulation of Dijkstra's algorithm

$n \times \sqrt{\frac{p}{n}}$

→ single-source algo. on each submesh : $T_p = \underbrace{\Theta(n^3/p)}_{\text{computation}} + \underbrace{\Theta(\sqrt{np})}_{\text{communic}}$

$S = \dfrac{\Theta(n^3)}{\Theta(n^3/p) + \Theta(\sqrt{np})}$, $E = \dfrac{1}{1 + \Theta(p^{1.5}/n^{2.5})}$

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

$\text{Isoefficiency}_{\text{communic}} = \Theta(p^{1.8})$

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

$\sqrt{\frac{p}{n}}$

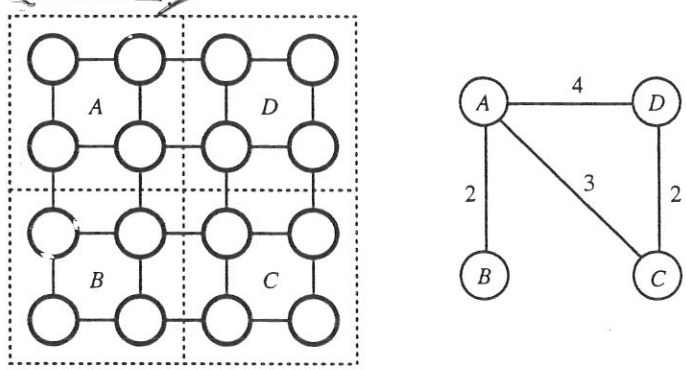


Figure 7.8 Partitioning a $\sqrt{p} \times \sqrt{p}$ mesh into n submeshes, each of size $\sqrt{p/n} \times \sqrt{p/n}$. In this example, $p = 16$ and $n = 4$. Each of the $\sqrt{p/n} \times \sqrt{p/n} = 2 \times 2$ meshes solves the single-source shortest paths problem for a given source vertex. In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

Hypercube :

// $p > n$, each ~~processor~~ partition handles $\frac{p}{n}$ processors

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

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

$$E = \frac{1}{1 + \Theta((p \log p)/n^2)}$$

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

$\text{Iso-efficiency}_{\text{communic}} = \Theta((p \log p)^{3/2})$ $\text{Isoefficiency}_{\text{concurr}} = \Theta(p^{3/2})$

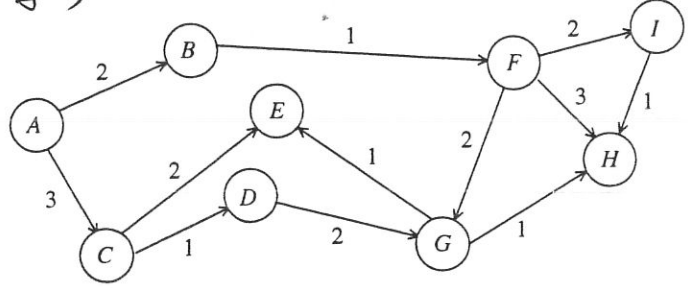
- $d_{ij}^{(k)} = \min_{1 \le m \le n} \{ d_{im}^{(k-1)} + w(v_m, v_j) \}$

- $D^{(k)}$ 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, ie, $A^2, A^4, A^8, \ldots$

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



eg: n = 20

$D^{(0)} \leftarrow A$

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

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

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

- On a HC w/ n^3 processors, $n \times n$ matrix mult in $\Theta(\log n)$ [see MM chapt.]

  $S = \dfrac{\Theta(n^3)}{\Theta(\log^2 n)}$; $E = \dfrac{1}{\Theta(\log^2 n)}$

FLOYD's algorithm

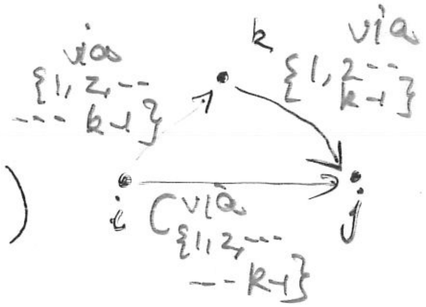
$$d_{i,j}^{(k)} = \begin{cases} w(v_i, v_j) & \text{if } k = \phi \\ \min\left\{ d_{i,j}^{(k-1)},\ d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \right\} & \text{if } k \geq 1 \end{cases}$$

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

space $= \Theta(n^2)$

$D^{(0)} = A$

for $k = 1$ to n do
  for $i = 1$ to n do
    for $j = 1$ to n do
      $d_{i,j}^{(k)} := \min\left(d_{i,j}^{(k-1)},\ d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \right)$



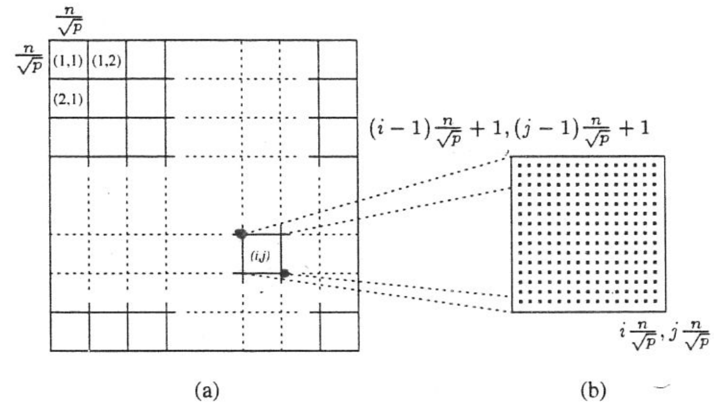


Figure 7.9 (a) Matrix $D^{(k)}$ partitioned by block checkerboarding into $\sqrt{p} \times \sqrt{p}$ subblocks, and (b) the square subblock of $D^{(k)}$ assigned to processor $P_{i,j}$.

Total of p blocks

Parallel formulations: to compute portion of $D^{(k)}$, need corresp. 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
  - $\rightarrow \sqrt{p} \times \sqrt{p}$ virtual mesh $\longleftrightarrow$ p-proc HC
  - $\rightarrow$ each row or column of mesh $\equiv$ HC of $\sqrt{p}$ procs
  - $\rightarrow$ in each iteration, k^{th} row & k^{th} column do $1 \rightarrow$ all BC along a column or row

    (each such proc has $n/\sqrt{p}$ elements of k^{th} row or col)

    $\Longrightarrow$ BC requires $\Theta((n \log p)/\sqrt{p})$ time

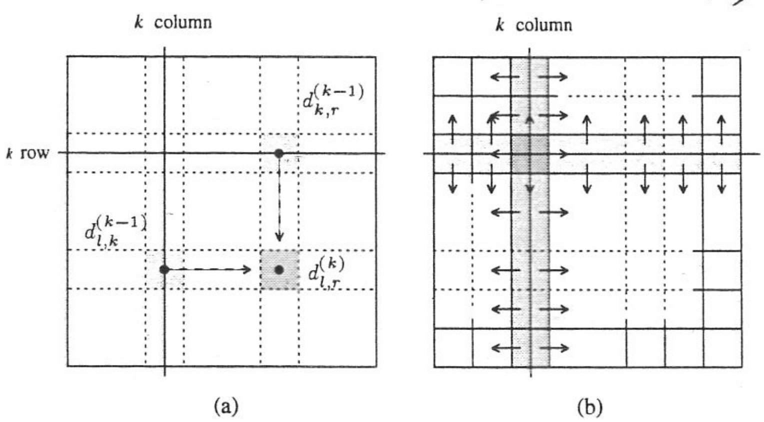


Figure 7.10 (a) Communication patterns used in the block-checkerboard partitioning. When computing $d_{i,j}^{(k)}$, information must be sent to the highlighted processor from two other processors along the same row and column. (b) The row and column of $\sqrt{p}$ processors that contain the k^{th} row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

$\rightarrow \Theta(n^2/p)$ to compute values of $D^{(k)}$ in block

$$T_p = \Theta(n^3/p) + \Theta\left(\frac{n^2}{\sqrt{p}} \log p\right)$$

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

$$E = \frac{1}{1 + \Theta((\sqrt{p} \log p)/n)}$$

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

$\text{Isoefficiency}_{\text{comm}} = \Theta(p^{1.5} \log^3 p)$

$\text{Isoefficiency}_{\cancel{\text{comput}}\ \text{Concurrency}} = \Theta(p^{1.5})$

$W = \Theta(n^3)$; # procs $= O(n^2) = p$

with pipelining, $T_p = \Theta\left(\frac{n^3}{p}\right) + \Theta(n)$