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

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

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

- difficult to achieve even work distribution & low comm ovhd for random sparse graphs

$\therefore$ focus on grid-like graphs

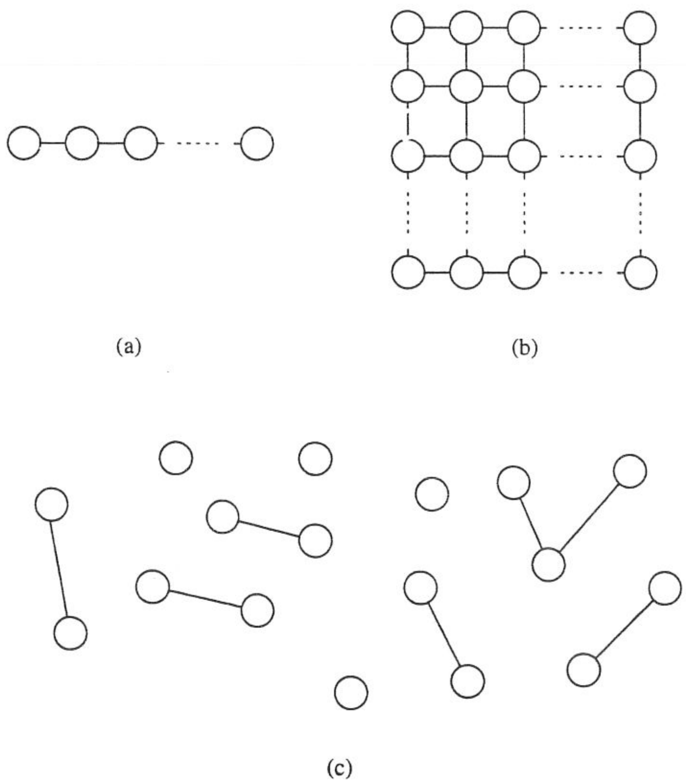


Figure 7.15 Examples of sparse graphs: (a) a linear graph, in which each vertex has two incident edges; (b) a grid graph, in which each vertex has four incident vertices; and (c) a random sparse graph.

Johnson_SSSP (V, E, s)

/* modification of Dijkstra's SSSP */

```
Q := V
for all v ∈ Q do
    l[v] := ∞
l[s] := 0
while Q ≠ ∅ do
    u := extract_min(Q);
    for each v ∈ Adj[u] do
        if v ∈ Q and l[u] + w(u,v) < l[v] then
            l[v] := l[u] + w(u,v)
```

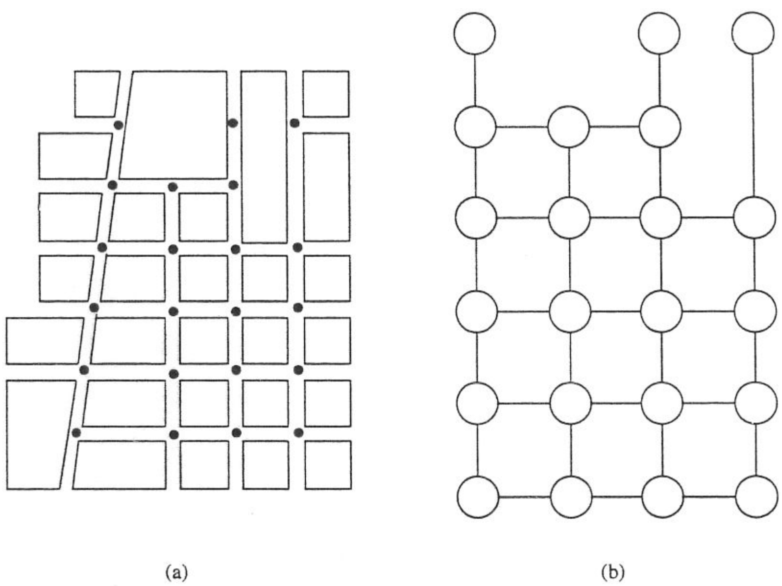


Figure 7.16 A street map (a) can be represented by a graph (b). In the graph shown in (b), each street intersection is a vertex and each edge is a street segment. The vertices of (b) are the intersections of (a) marked by dots.

- Q implemented as a binary min-heap
  → each update in $O(\log n)$
- Complexity $= \Theta(|E| \log n)$
- 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 $l[v] < l[u]$

  → If $\exists$ path (v, u), cost of $(s-v, u) <$ cost (s, u)

  ⇒ vertices may not be extracted in nondecreasing order of l

  ⇒ computation must be redone

eg source = 'a' ; each vertex assigned to separate processor

$P_{1,0}$ picks path $\langle a, d \rangle$ from local Q.

Later, it discovers shorter path $\langle a, b, e, d \rangle$

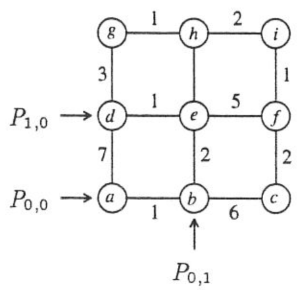


Figure 7.17 A grid graph. Copyright (r) 1994 Benjamin/Cummings Publishing Co.

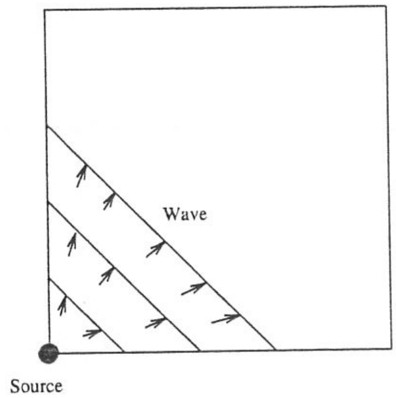


Figure 7.18 The wave of activity in the priority queues.

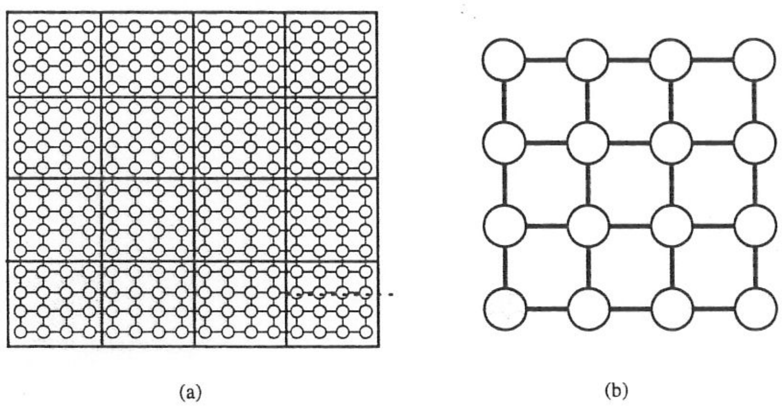
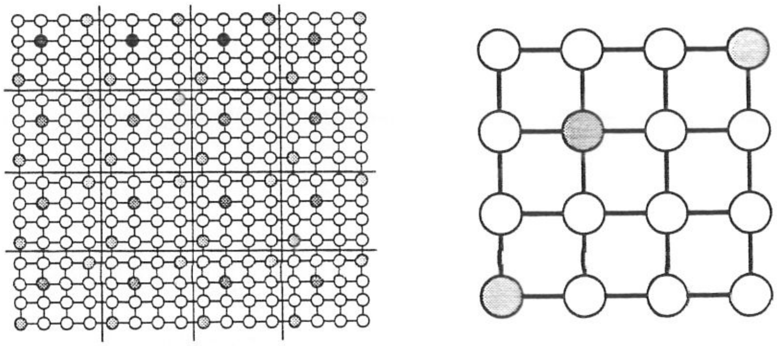


Figure 7.19 Mapping the grid graph (a) onto a mesh (b) by using the block-checkerboard mapping. In this example, $n = 16$ and $\sqrt{p} = 4$. The shaded vertices are mapped onto the shaded processor.

- 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

$$S_{max} = \frac{W}{W/\sqrt{p}} = \sqrt{p} \quad ; \quad E_{max} = 1/\sqrt{p}$$

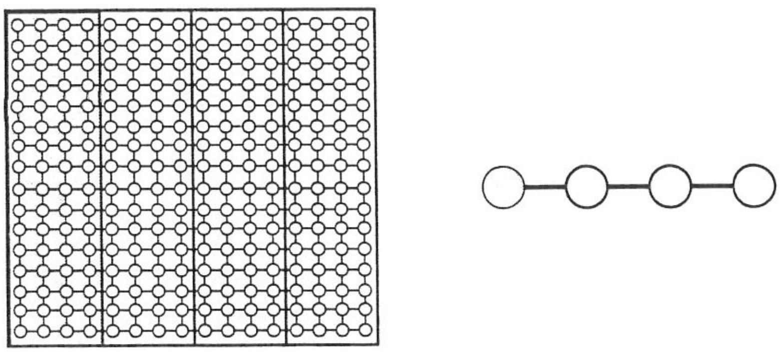
• each processor responsible for vertices that belong to different parts of the grid graph
$\Longrightarrow$
procs remain busy most of the time, but higher communic.



(a) (b)

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.

[as adjacent vertices are assigned to separate processors, more communic necessary each time P_i extracts a node from Q_i]



(a) (b)

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
$S = \frac{p}{2}$; $E = \frac{1}{2}$. However, cannot use more than $O(n)$ procs

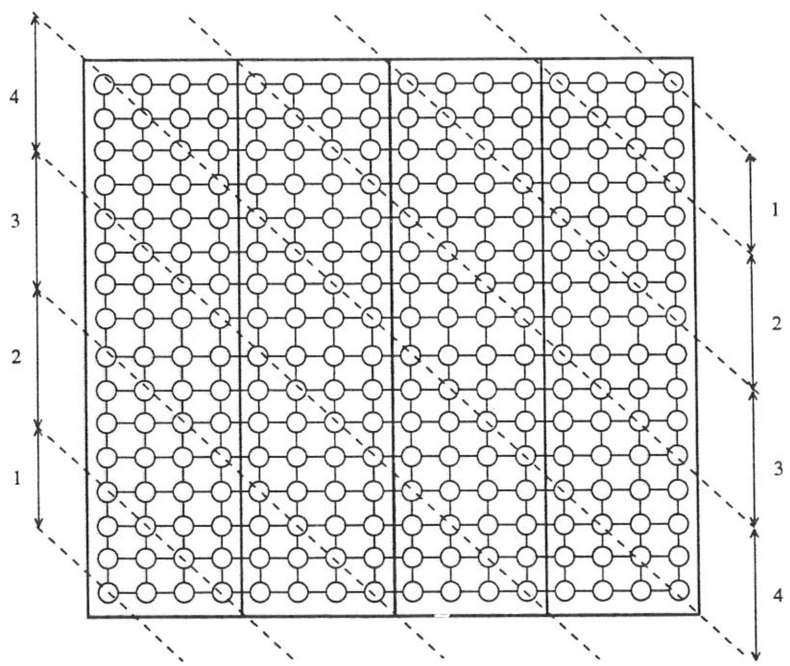


Figure 7.22 The number of busy processors as the computational wave propagates across the grid graph.