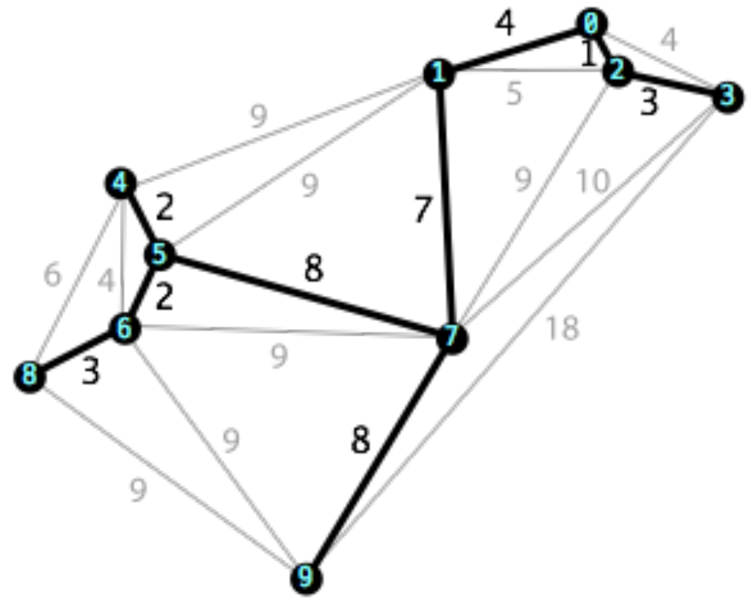
Minimum Spanning Tree

A spanning tree of an undirected graph G is a subgraph of G that is a tree containing all the vertices of G

The minimum spanning tree (MST) for a weighted undirected graph is a spanning tree with minimum weight

Prim's algorithm can be used

- Greedy algorithm
- Selects an arbitrary starting vertex
- Chooses new vertex guaranteed to be in MST
- $O(n^2)$
- Prim's algorithm is iterative

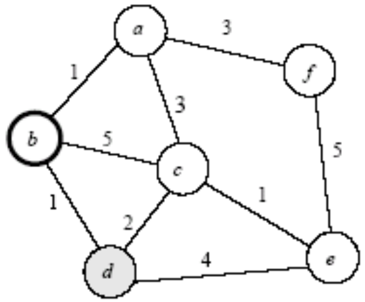


Prim's Minimum Spanning Tree Algorithm

```
PRIM MST(V, E,w, r ) { // V: vertexes, E: edges; w: weights, r: initial vertex
  VT := { r };
  d[r] := 0;
  for all v ∈ (V - VT )
    if edge (r, v) exists then d[v] := w(r, v);
    else d[v] := ∞;
  while VT ≠ V
      find a vertex u such that d[u] := min{d[v]|v ∈ (V - VT )};
      VT := VT ∪ {u};
      for all v ∈ (V - VT )
        d[v] := min{d[v],w(u, v)};
}
```

Prim’s Minimum Spanning Tree Algorithm

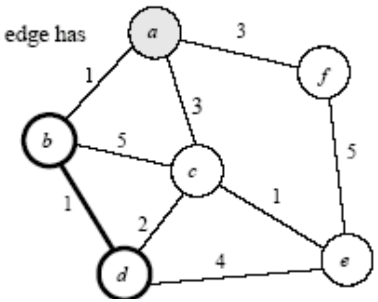
(a) Original graph



	a	b	c	d	e	f
d[]	1	0	5	1	∞	∞

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
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

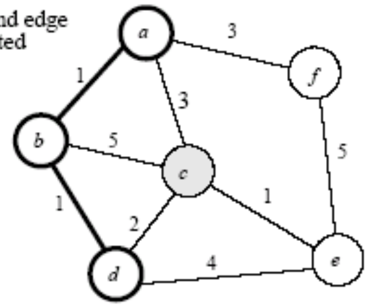
(b) After the first edge has been selected



	a	b	c	d	e	f
d[]	1	0	2	1	4	∞

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
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

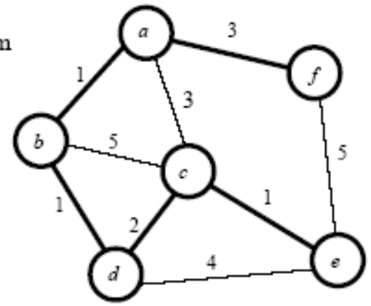
(c) After the second edge has been selected



	a	b	c	d	e	f
d[]	1	0	2	1	4	3

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
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	3

	a	b	c	d	e	f
a	0	1	3	∞	∞	3
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

Parallel Formulation of Prim's Algorithm

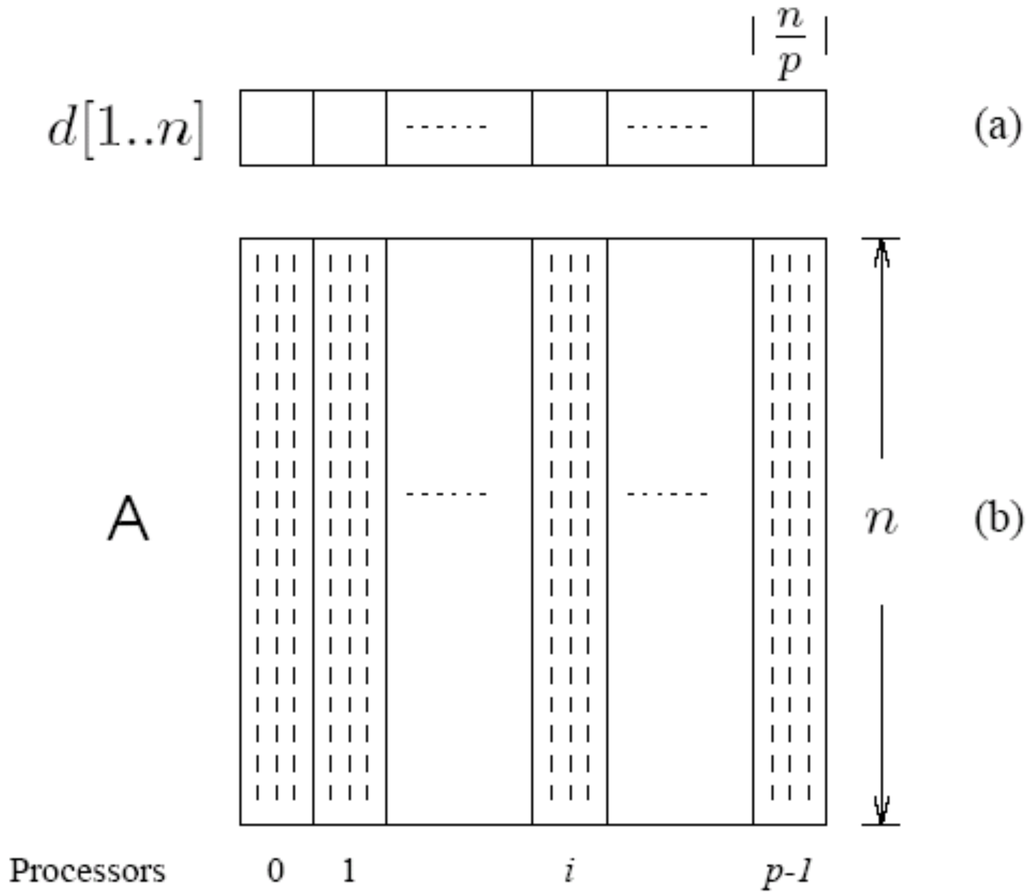
Difficult to perform different iterations of the **while** loop in parallel because *d[v]* may change each time

Can parallelize each iteration though

- Partition vertices into *p* subsets V_i, $i=0,\ldots,p-1$
- Each process P_i computes $d_i[u]=min\{d_i[v] \mid v \in (V-V_T) \cap V_i\}$
- Global minimum is obtained using all-to-one reduction
- New vertex is added to V_T and broadcast to all processes
- New values of *d[v]* are computed for local vertex

Prim's Algorithm: Parallel Formulation

The partitioning of the distance array d and the adjacency matrix A among p processes.



Prim's Algorithm: Parallel Formulation

The cost to select the minimum entry and update the *d* vector is $O(n/p)$.

The cost of a broadcast is $O(\log p)$.

So, the parallel time per iteration is $O(n/p + \log p)$.

The total parallel time (for n vertexes) is given by $O(n^2/p + n \log p)$.

The corresponding isoefficiency is $O(p^2 \log^2 p)$.