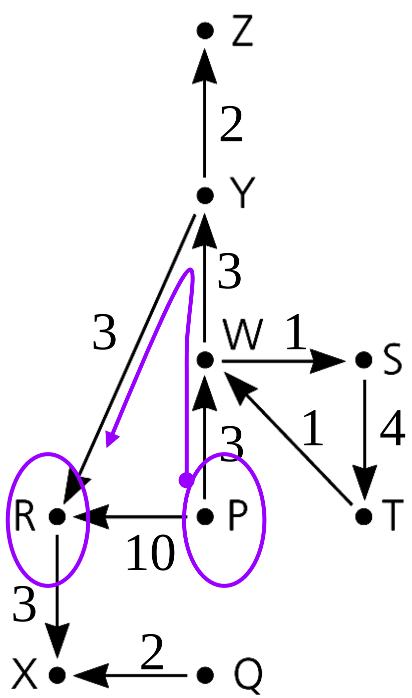
Graph App.

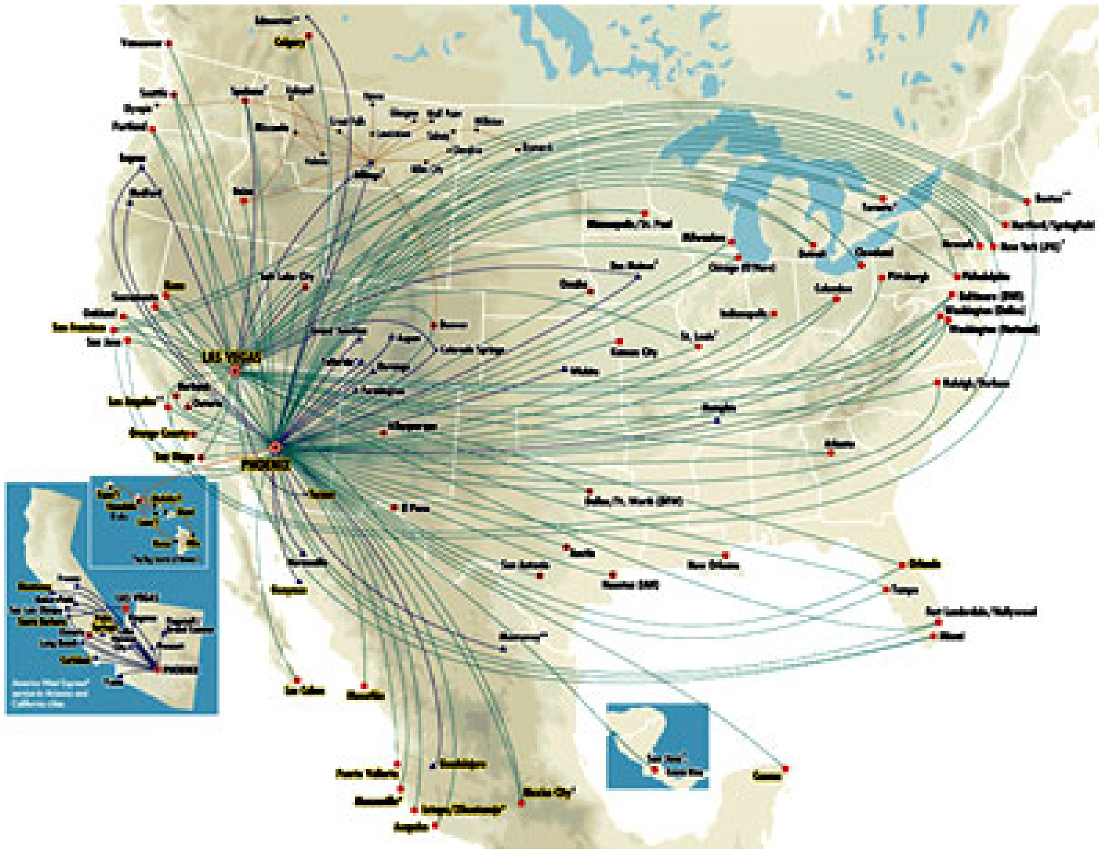
- **Topological Sort**
- **Spanning Tree**
- **Shortest Paths**

Data Structures

Shortest Paths

- Shortest path between two vertices in a **weighted** graph is ***the path that has the smallest sum of its edge weights***



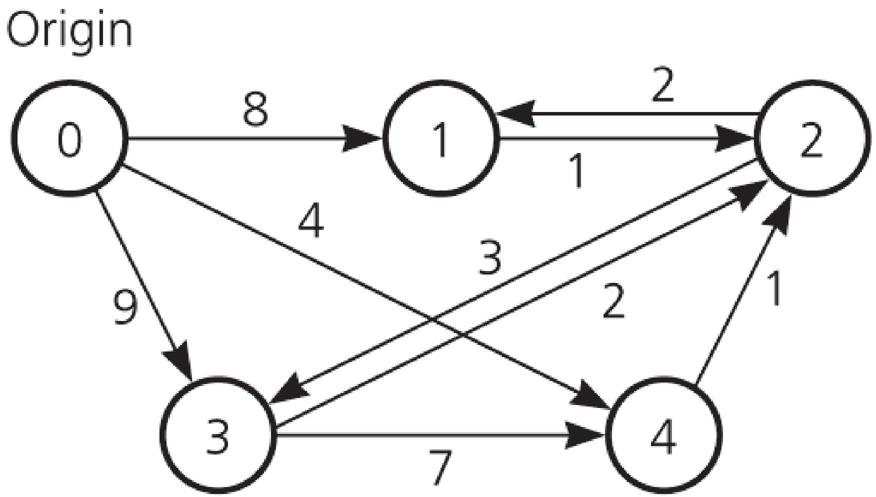


Shortest Paths: *Dijkstra's Algorithm*

- **Problem definition**
  - Find the shortest paths between a given origin and all other vertices
- **Basic idea**
  - A set *vertexSet* of selected vertices
  - An array *weight*, where *weight*[*v*] is the cheapest weight of the shortest path from vertex 0 (origin) to vertex *v* that *passes through only the vertices in vertexSet*

Shortest Paths: *Dijkstra's Algorithm*

(a)



Step	v	vertexSet	weight [0]	[1]	[2]	[3]	[4]
1	–	0	0	8	∞	9	4
2	4	0, 4	0	8	5	9	4
3	2	0, 4, 2	0	7	5	8	4
4	1	0, 4, 2, 1	0	7	5	8	4
5	3	0, 4, 2, 1, 3	0	7	5	8	4

Single-Source All-Destination Shortest Paths

Data Structures

- **Dijkstra's algorithm** **[Edsger Wybe Dijkstra, 1930-2002]**

1. Initialize *vertexSet* & weight; $v = v_0$;
2. Update *weight* for each vertex *u* not in *vertexSet*, which is *adjacent* to *v*

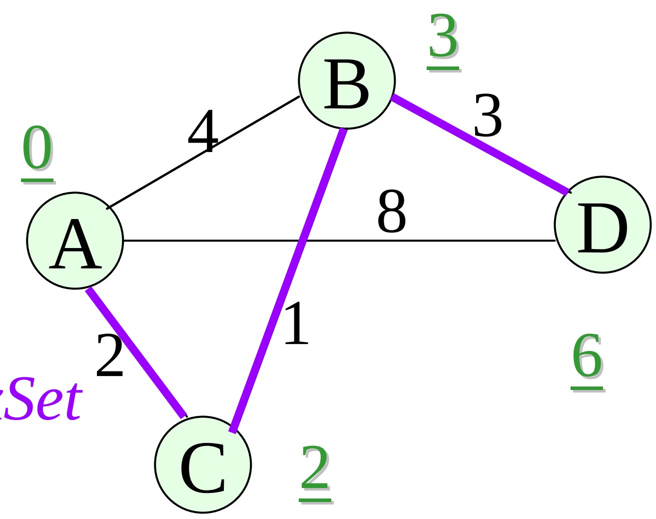
   weight[*u*] = *min*{weight[*u*], weight[*v*] + edgeWeight[*v*,*u*]}

3. Find the shortest path from *o* to *u* among every path that starts from *o*, passes vertices in *vertexSet*, and ends at a vertex not in *vertexSet*

   if (weight[*u*] is *minimum*) vertexSet = vertexSet + {*u*};

4. Repeat steps 2, 3 until *no more vertex can be added*

Single-Source All-Destination Shortest Paths



DijkstraAlgorithm(Vertex v_0)

  weight[0..*n*] = {0, ∞, …, ∞};

  vertexSet = ∅; $v = v_0$;

  do { *Add v into vertexSet;*

    for *edge (v,u) where u is not in vertexSet*

      weight[*u*] = ***min***{*weight*[*u*],

          weight[*v*]+*edgeWeight*[*v*,*u*]};

    cheapest = ∞**;**

    for *vertex u not in vertexSet*

      if (*weight*[*u*] < cheapest)

      {   *v* = *u*;   cheapest = *weight*[*u*];}

  } while (cheapest < ∞);

Dijkstra's Algorithm: *Adjacency Matrix*

$vertexSet_0 = \{\}$
$weight_0 = \{0, \infty, \infty, \infty\}$

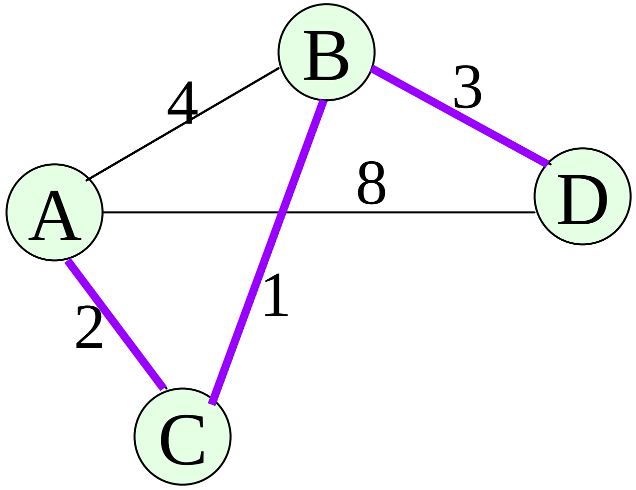
$vertexSet_1 = \{A\}$
$weight_1 = \{0, 4, 2, 8\}$

$vertexSet_2 = \{A,C\}$
$weight_2 = \{0, 3, 2, 8\}$

$vertexSet_3 = \{A,C,B\}$
$weight_3 = \{0, 3, 2, 6\}$

A→B: 4
A→C→B: 2+1=3

A→D: 8
A→C→B→D: 3+3=6



adjacency matrix

→	A	B	C	D
A	0	4	2	8
B	4	0	1	3
C	2	1	0	∞
D	8	3	∞	0

Dijkstra's Algorithm: *Adjacency List*

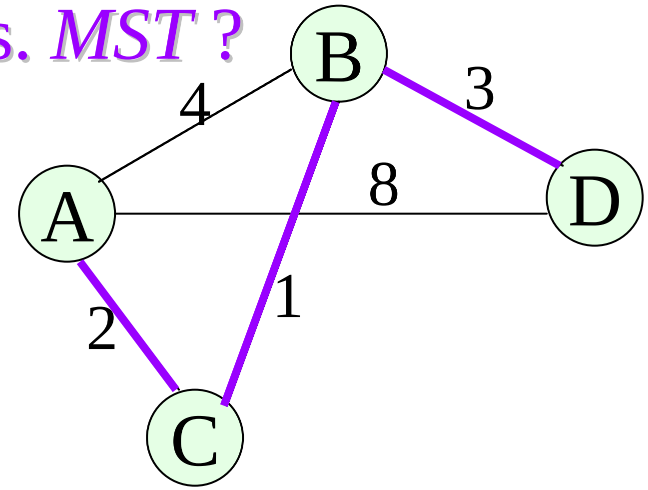
$vertexSet_0=\{\}$
$weight_0=\{0, \infty, \infty, \infty\}$

$vertexSet_1=\{A\}$
$weight_1=\{0, 4, 2, 8\}$

$vertexSet_2=\{A,C\}$
$weight_2=\{0, 3, 2, 8\}$

$vertexSet_3=\{A,C,B\}$
$weight_3=\{0, 3, 2, 6\}$

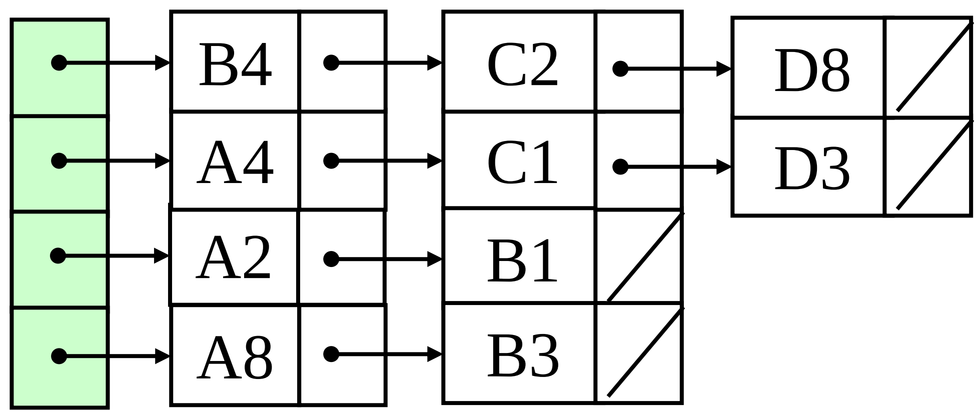
Shortest path tree vs. MST ?



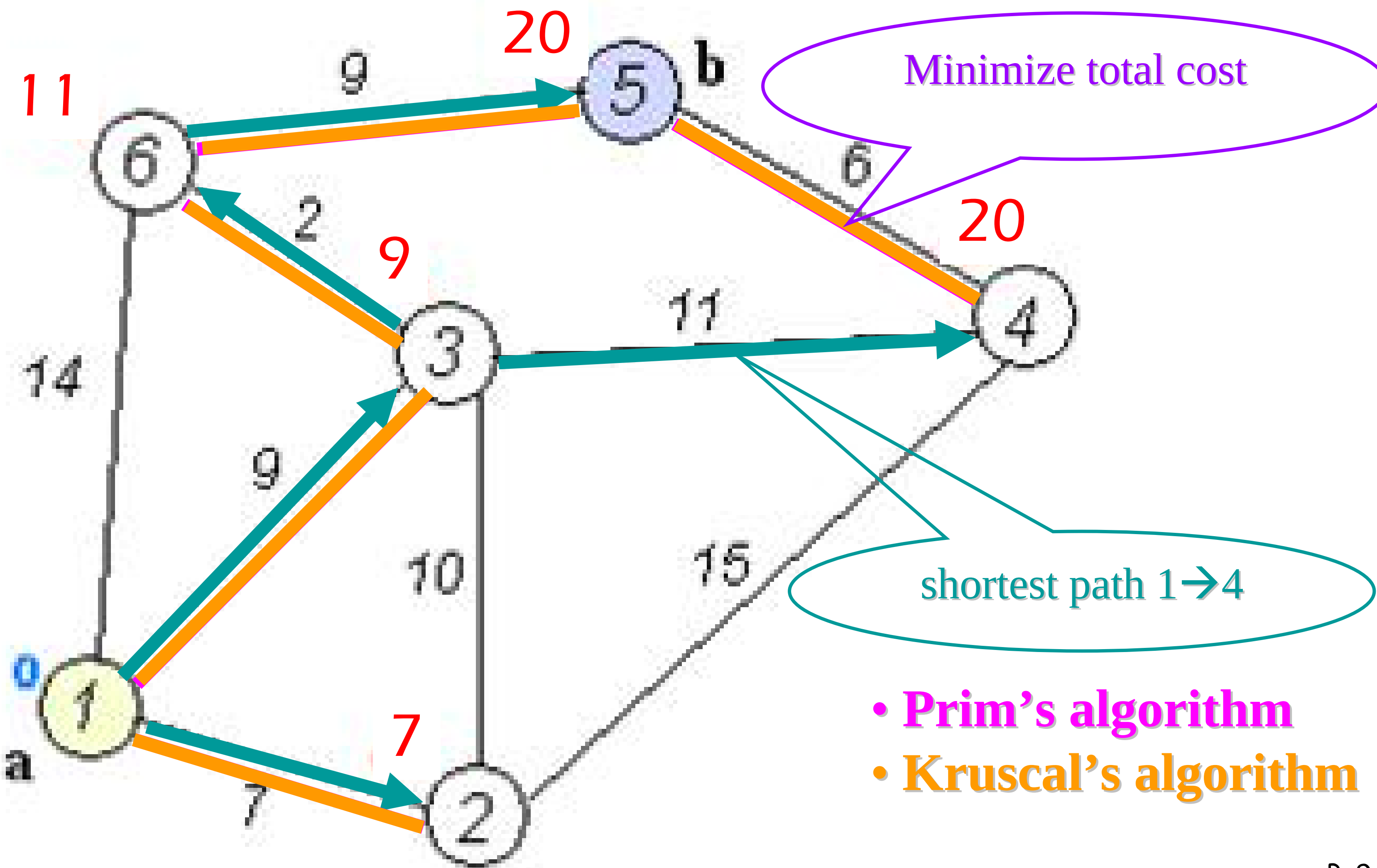
vertexSet

T	A
T	B
T	C
T	D

adjacency lists

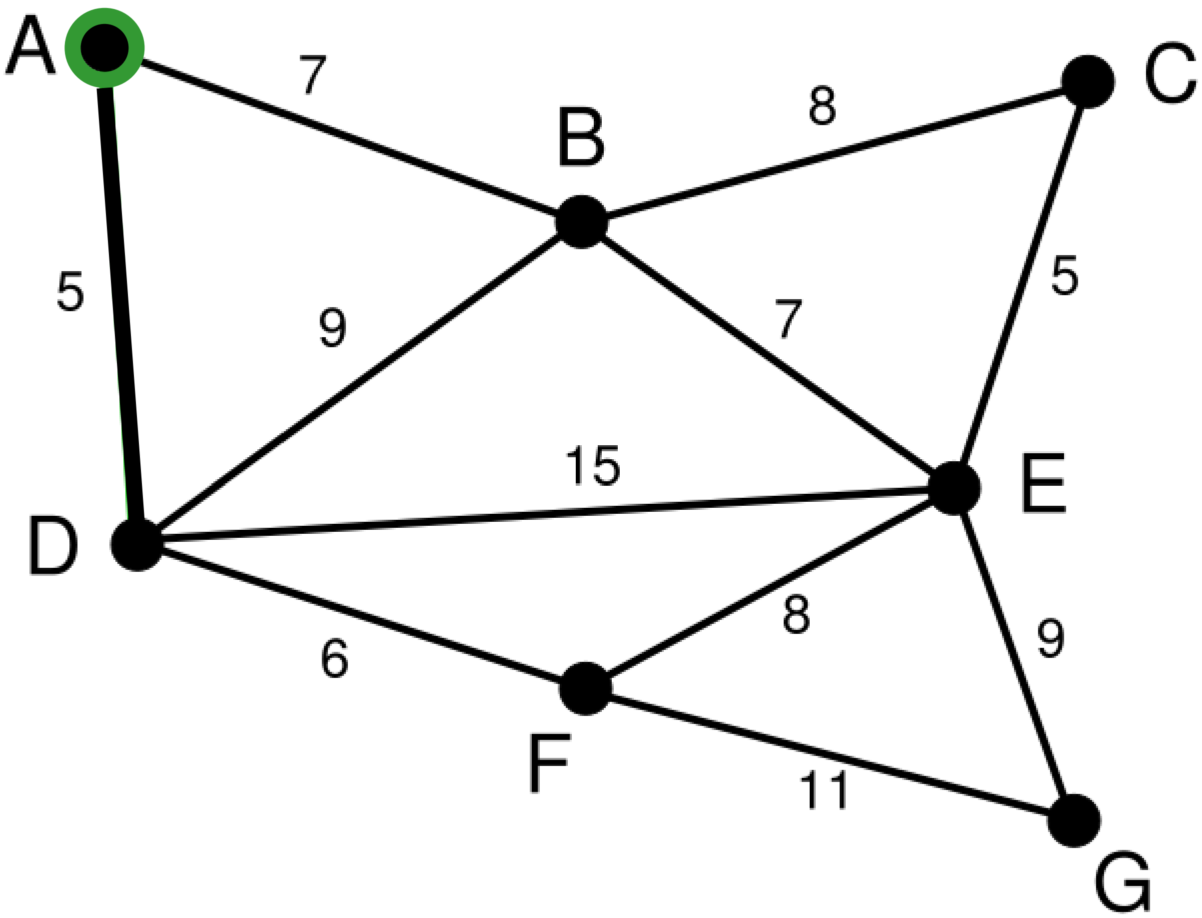


Shortest Path Tree vs. *Minimum Spanning Tree*

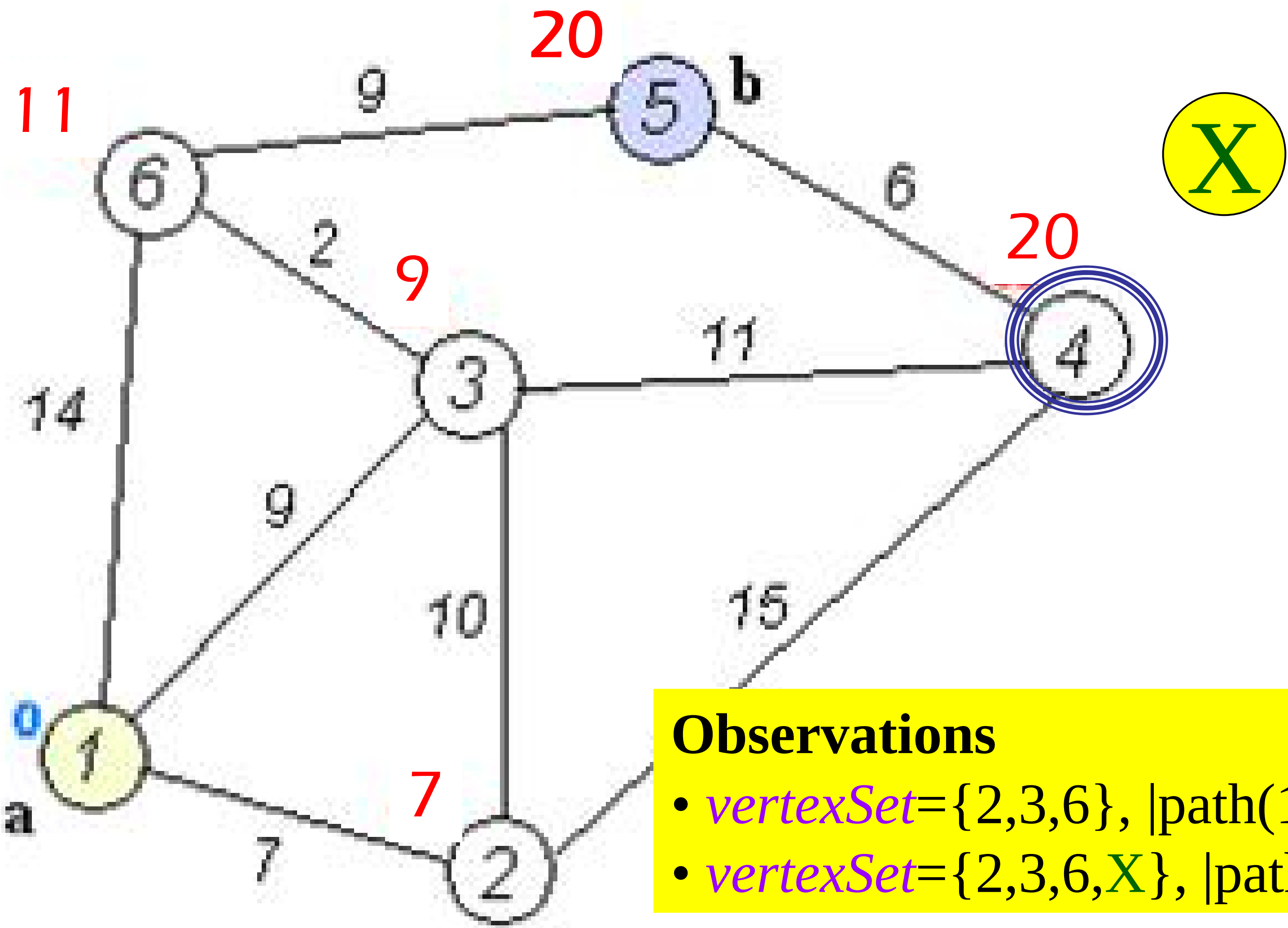


- **Prim's algorithm**
- **Kruscal's algorithm**

Practice 9: *Dijkstra's Algorithm*



Q: Is the shortest path (20) correct?



Observations

- *vertexSet*={2,3,6}, |path(1→X)|≥20
- *vertexSet*={2,3,6,X}, |path(1→X)|≥20

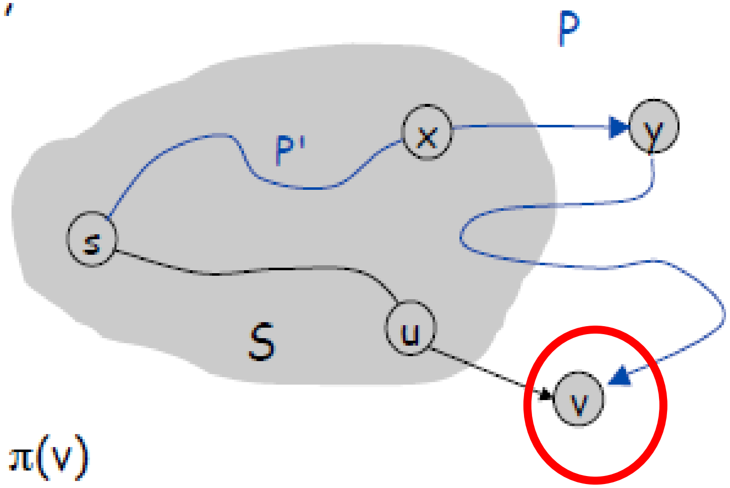
Invariant. For each node $u \in S$, $d(u)$ is the length of the shortest s-u path.

Pf. (by induction on $|S|$)

Base case: $|S| = 1$ is trivial.

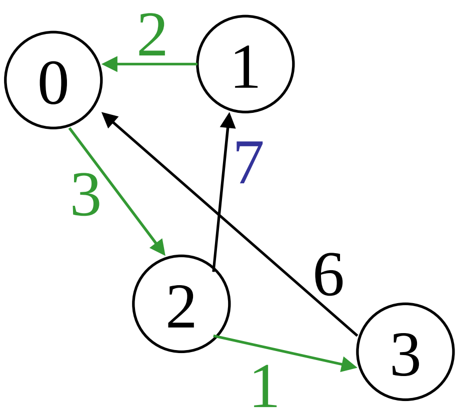
Inductive hypothesis: Assume true for $|S| = k \geq 1$.

- Let v be next node added to S, and let u-v be the chosen edge.
- The shortest s-u path plus (u, v) is an s-v path of length $\pi(v)$.
- Consider any s-v path P. We'll see that it's no shorter than $\pi(v)$.
- Let x-y be the first edge in P that leaves S, and let P' be the subpath to x.
- P is already too long as soon as it leaves S.



$$\ell(P) \underset{\text{nonnegative weights}}{\geq} \ell(P') + \ell(x,y) \underset{\text{inductive hypothesis}}{\geq} d(x) + \ell(x,y) \underset{\text{defn of } \pi(y)}{\geq} \pi(y) \underset{\text{Dijkstra chose v instead of y}}{\geq} \pi(v)$$

All-Pairs Shortest Paths



0,1	0→2→1	10
0,2	0→2	3
0,3	0→2→3	4
1,0	1→0	2
1,2	1→0→2	5
1,3	1→0→2→3	6
2,0	2→3→0	7
2,1	2→1	7
2,3	2→3	1
3,0	3→0	6
3,1	3→0→2→1	16
3,2	3→0→2	9

All-Pairs Shortest Paths: *Floyd's Algorithm*

Floyd–Warshall algorithm **[Robert Floyd, 1962][S. Warshall, 1962]**

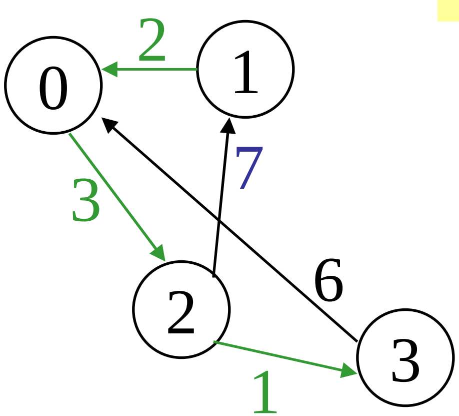
1. Initialize *distance matrix* D^{-1} = *adjacency matrix*;
2. **For** k = 0 to |V|-1

   $D^k \leftarrow D^{k-1}$; // *Add vertex k into vertexSet*

For *i* = *0* to |*V*|-1

For *j* = *0* to |*V*|-1

$D^0[1,2] = \min \{D^{-1}[1,2], D^{-1}[1,0]+D^{-1}[0,2]\}$



D^{-1}	0	1	2	3
0	0	∞	$\boxed{3}$	∞
1	$\boxed{2}$	0	∞	∞
2	∞	7	0	1
3	6	∞	∞	0

D^0	0	1	2	3
0	0	∞	3	∞
1	2	0	**5**	∞
2	∞	7	0	1
3	6	∞	∞	0

Floyd's Algorithm: *Directed Graph*

Data Structures

D^{-1}: *all-pairs* shortest paths with no intermediate vertex

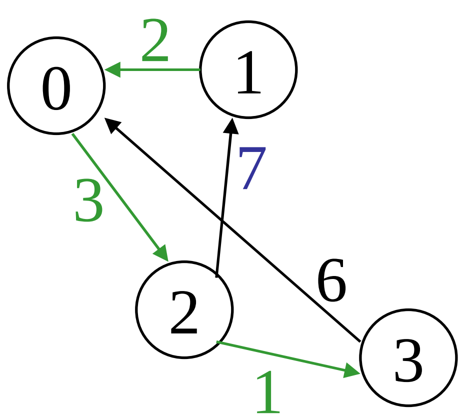
D^{0}: *all-pairs* shortest paths with intermediate vertex *0*

D^{1}: *all-pairs* shortest paths with intermediate vertices *0, 1*

D^{2}: *all-pairs* shortest paths with intermediate vertices *0, 1, 2*

…

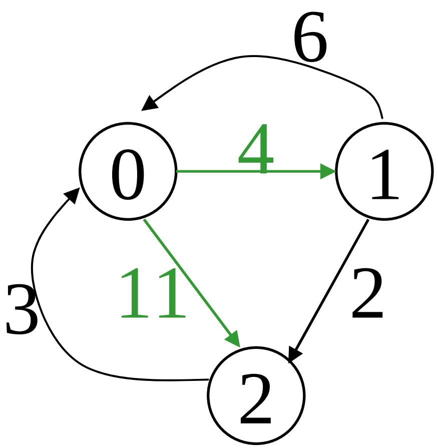
$D^2[3,1] = \min \{ D^1[3,1], D^1[3,2]+D^1[2,1] \}$



D^1	0	1	2	3
0	0	∞	3	∞
1	2	0	5	∞
2	9	7	0	1
3	6	∞	9	0

D^2	0	1	2	3
0	0	10	3	4
1	2	0	5	?
2	9	7	0	1
3	6	16	9	0

Floyd's Algorithm: Another Example



D^{-1}	0	1	2
0	0	4	11
1	6	0	2
2	3	∞	0

D^{0}	0	1	2
0	0	4	11
1	6	0	2
2	3	7	0

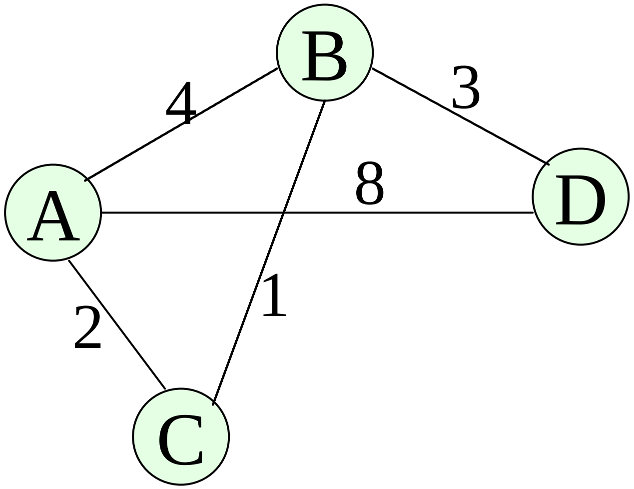
D^{1}	0	1	2
0	0	4	6
1	6	0	2
2	3	7	0

D^{2}	0	1	2
0	0	4	6
1	5	0	2
2	3	7	0

Floyd's Algorithm: *Undirected Graph*

D^0	A	B	C	D
A	0	4	2	8
B	4	0	1	3
C	2	1	0	10
D	8	3	10	0

D^3	A	B	C	D
A	0	3	2	6
B	3	0	1	3
C	2	1	0	4
D	6	3	4	0



D^1	A	B	C	D
A	0	4	2	7
B	4	0	1	3
C	2	1	0	4
D	7	3	4	0

D^2	A	B	C	D
A	0	3	2	6
B	3	0	1	3
C	2	1	0	4
D	6	3	4	0

adjacency matrix

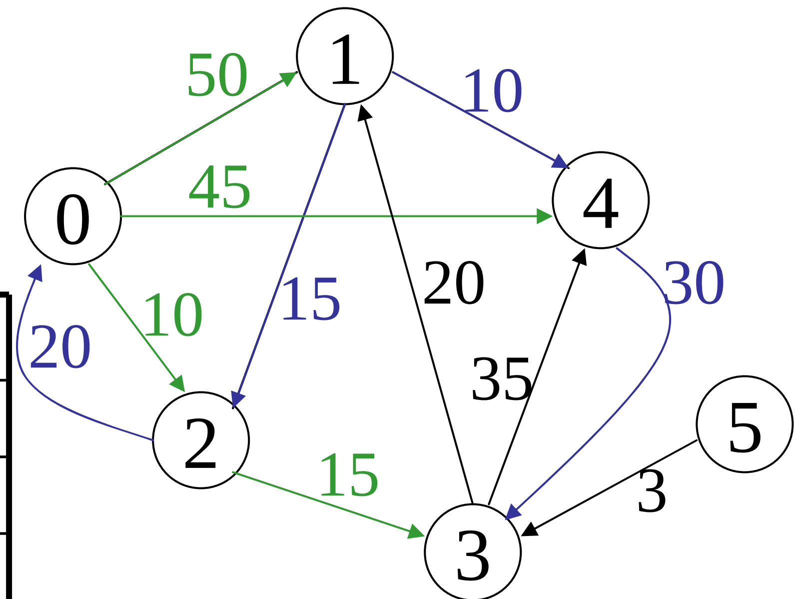
D^{-1}	A	B	C	D
A	0	4	2	8
B	4	0	1	3
C	2	1	0	∞
D	8	3	∞	0

Practice 10: *Floyd's Algorithm*

- **Find *all-pairs shortest paths***
  - **Show the distance matrices in each step**

D^{-1}	0	1	2	3	4	5
0	0	50	10	∞	45	∞
1	∞	0	15	∞	10	∞
2	20	∞	0	15	∞	
3	∞	20	∞	0	35	
4	∞	∞	∞	30	0	
5	∞	∞	∞	3	∞	

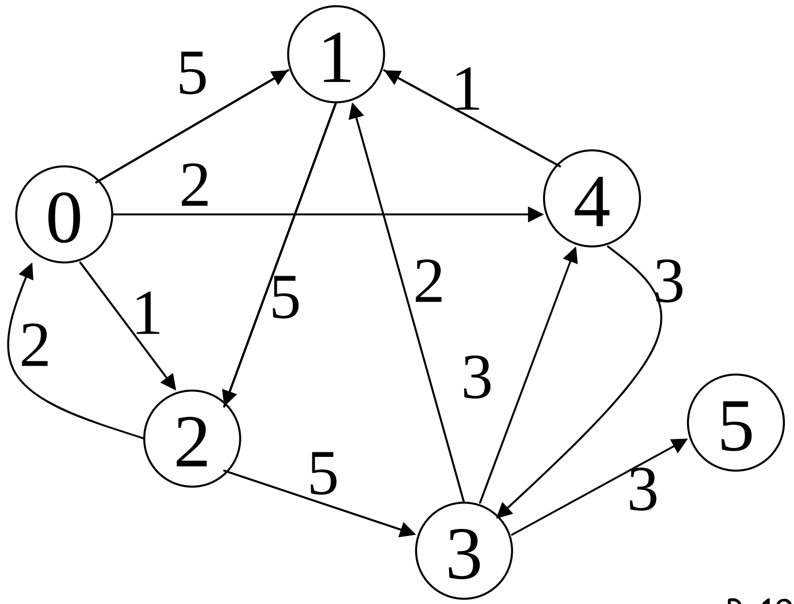
D^{5}	0	1	2	3	4	5
0	0	45	10	25	45	∞
1	35	0	15	30	10	∞
2	20	35	0	15	45	∞
3	55	20	35	0	30	∞
4	85	50	65	30	0	∞
5	58	23	38	3	33	0



Self-exercise 6

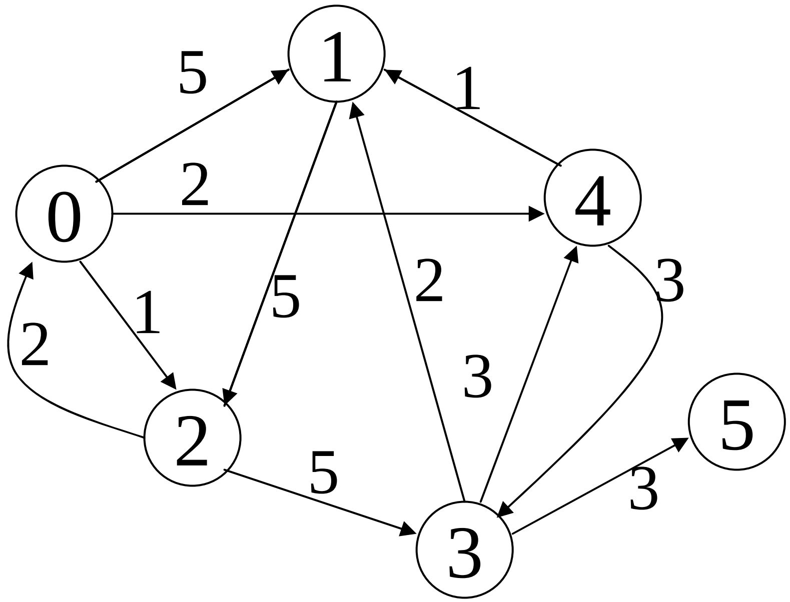
1. Use ***Dijkstra's*** **algorithm** to find the *shortest paths* from vertex 0 to any other vertex. Show the content of vertexSet and weight obtained at the end of each round.

- Choose the *smallest* label first if two or more vertices have the minimum weights.



Self-exercise 6

2. Use ***Floyd's* algorithm** to find *all-pairs shortest paths*. Show the content of distance matrix as the final result.



Summary

- **Topological sorting produces a linear order of the vertices in a directed graph without cycles**
- **Trees are connected undirected graphs without cycles**
- **A spanning tree of a connected undirected graph is**
  - A subgraph that contains all the graph's vertices and enough of its edges to form a tree

Summary

Data Structures

- **A minimum spanning tree for a weighted undirected graph is**
  - A spanning tree whose edge-weight sum is minimal
- **The shortest path between two vertices in a weighted directed graph is**
  - The path that has the smallest sum of its edge weights