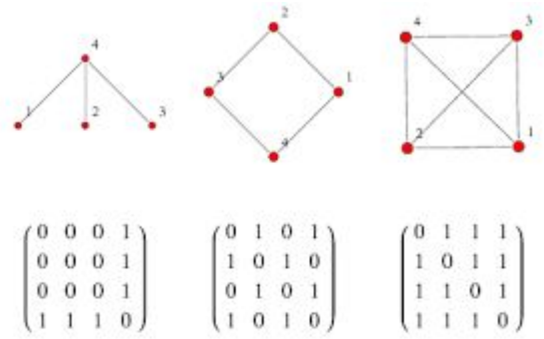
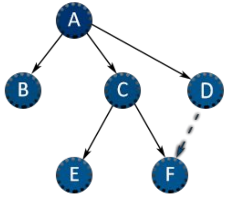
Recap of First Half of Winter Session

Week 5 AI Inspire Winter 2020



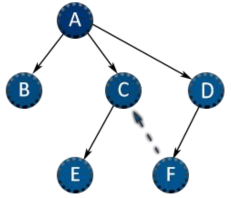
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

BFS



ABCDEF

DFS



ADFCEB

Week 1: Representing & Traversing Graphs

Define the following

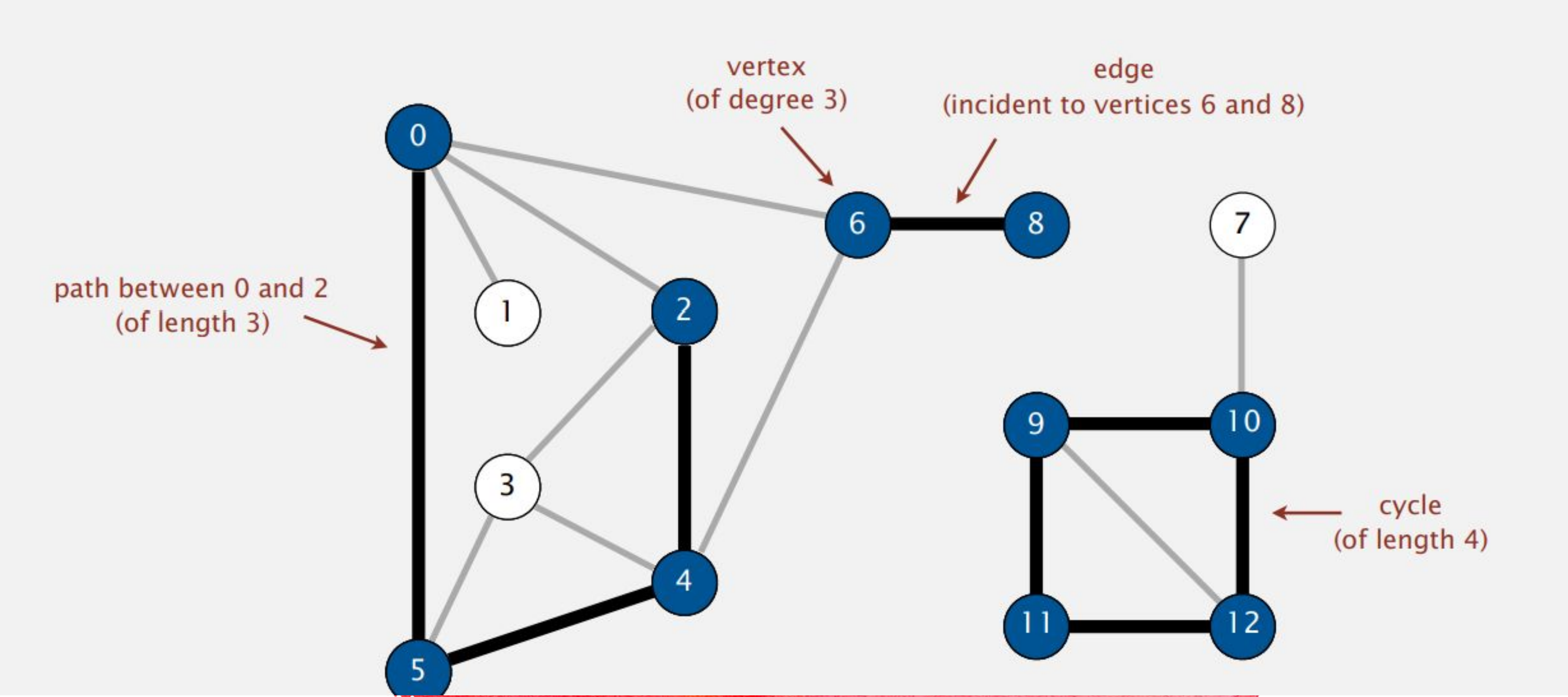
1) Graph

2) Path

3) Adjacent nodes

4) Cycle

5) Degree of node

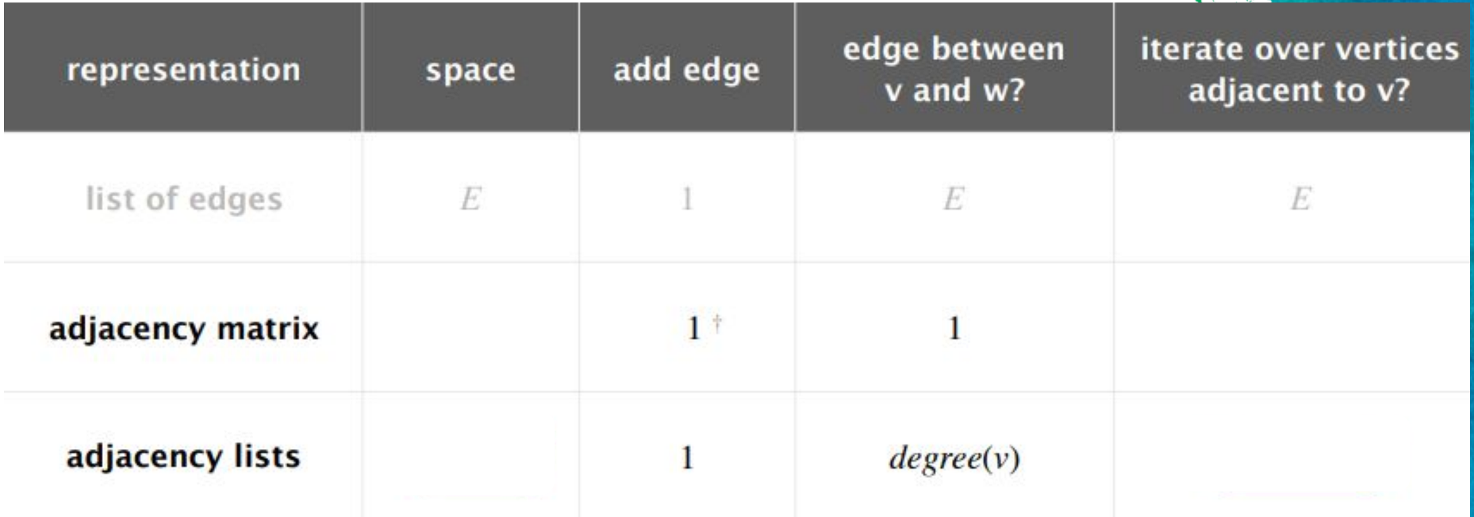

vertex
(of degree 3)
edge
(incident to vertices 6 and 8)
path between 0 and 2
(of length 3)
cycle
(of length 4)
0
1
2
3
4
5
6
7
8
9
10
11
12

Graph API?

```
public class Graph

        ⋮                                                   ⋮
```

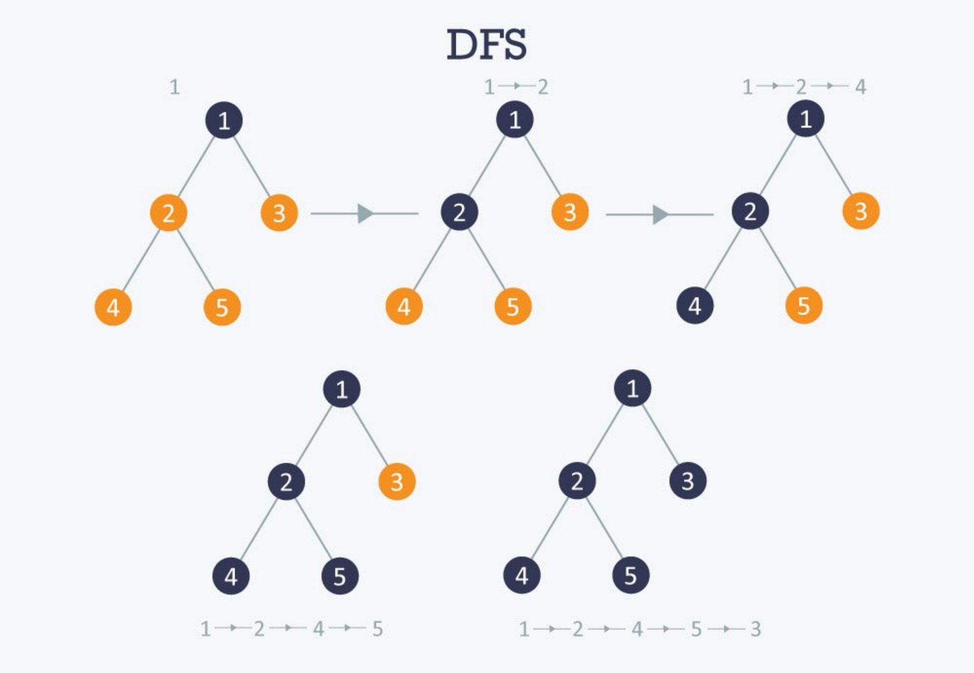
Representing Graphs



representation	space	add edge	edge between v and w?	iterate over vertices adjacent to v?
list of edges	E	1	E	E
adjacency matrix		1 †	1	
adjacency lists		1	$degree(v)$	

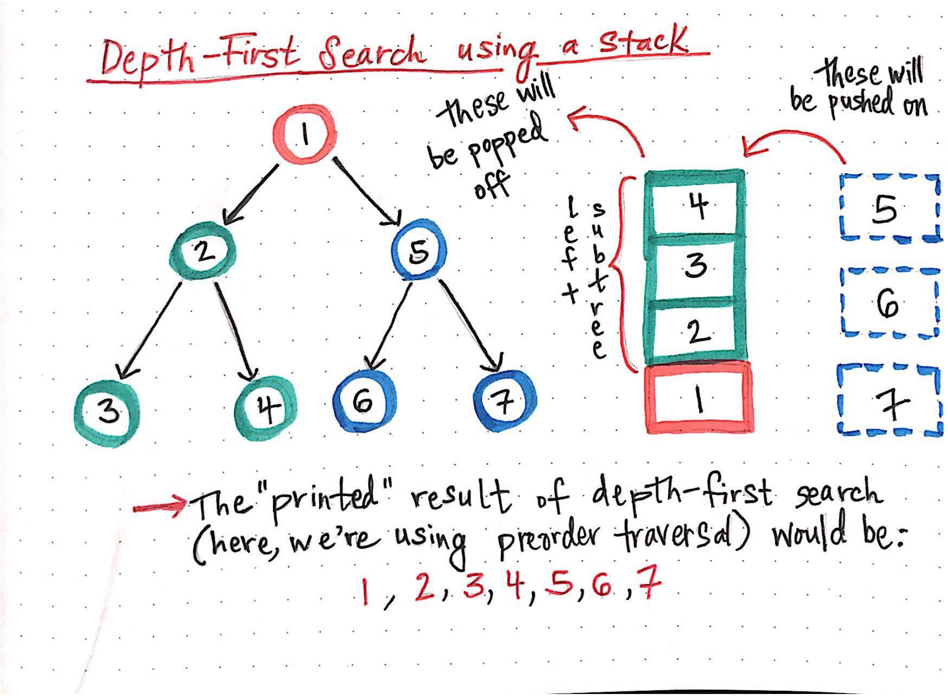
Credits - Princeton University COS 226 Lecture

Traversing Graphs



Traversing Graphs

Traversing Graphs



Depth-First Search using a Stack

→ The "printed" result of depth-first search (here, we're using preorder traversal) would be:

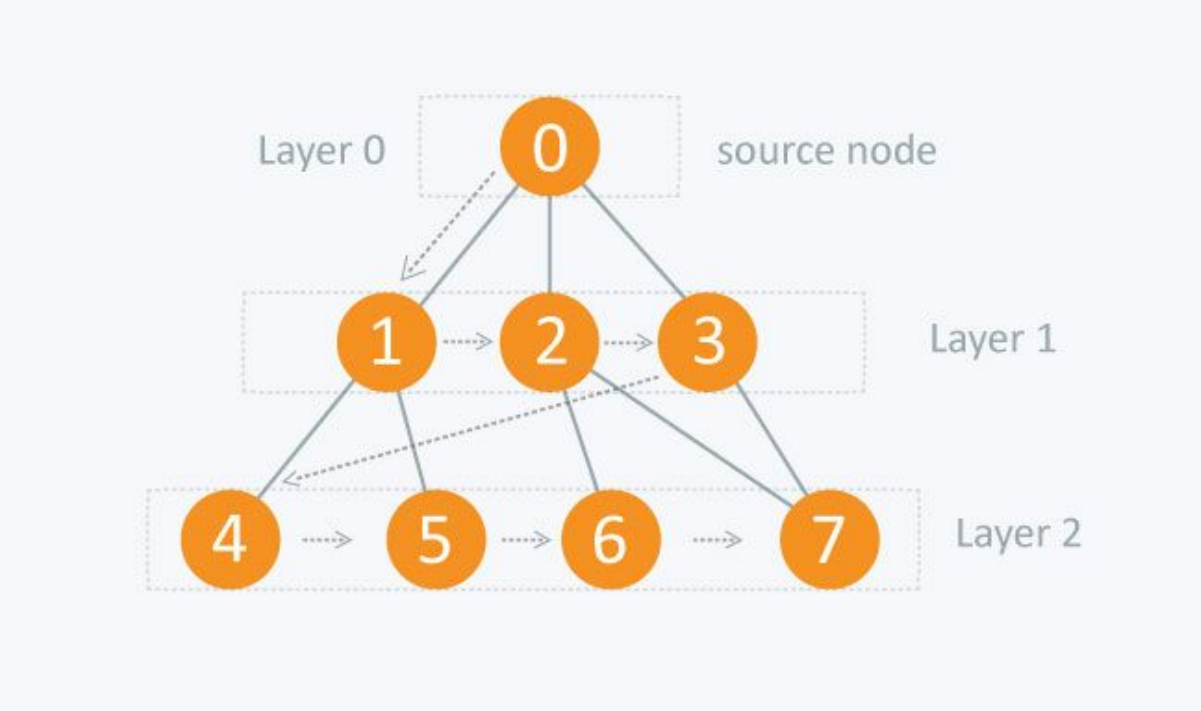
1, 2, 3, 4, 5, 6, 7

Pop off of stack once done exploring that node COMPLETELY

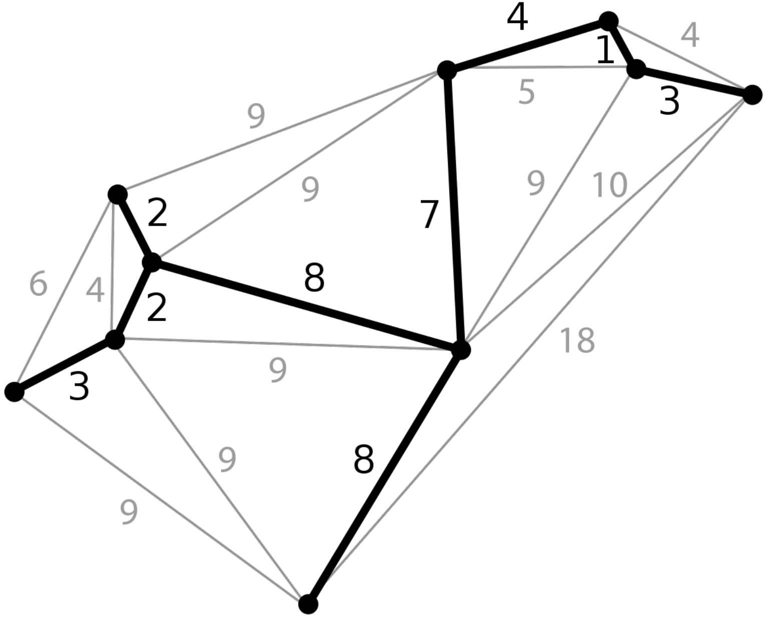
Push on adjacent nodes onto stack of fully unexplored node

Stack ⇒ Used for recursion

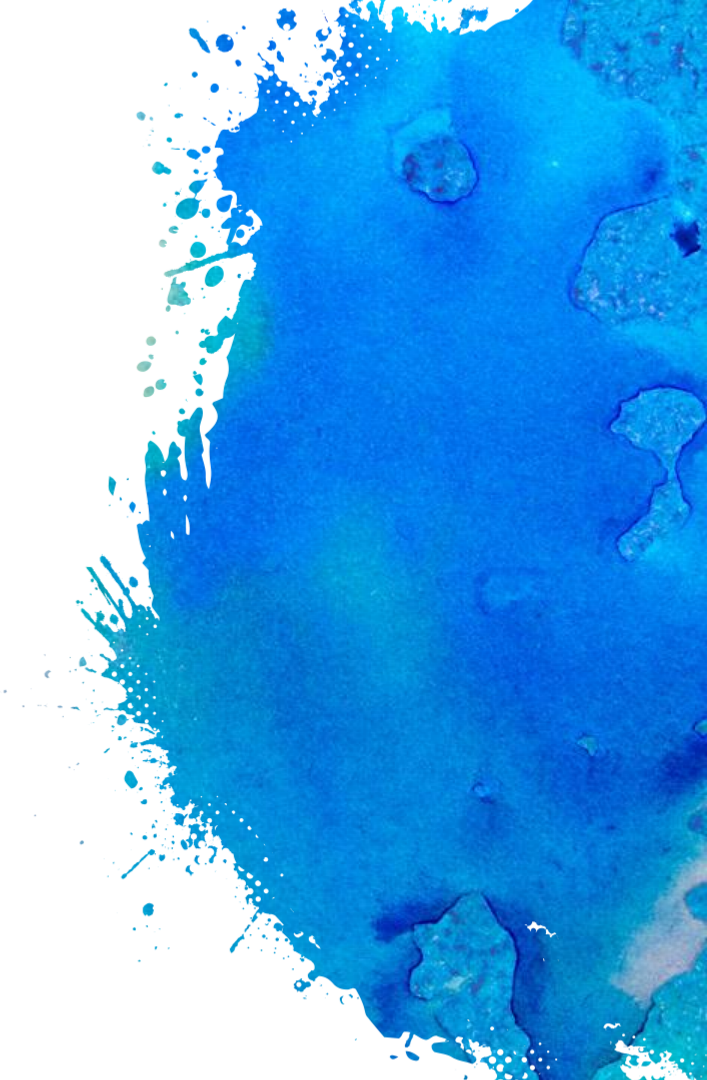
Traversing Graphs



Traversing Graphs



Week 2: MSTs



Some MST applications:

- *Laying cables across house network with least total cost (Ex: distance)*
- *Traveling salesman problem: visiting cities across US*

MST & Properties

1) What are they?

2) Is the graph directed?

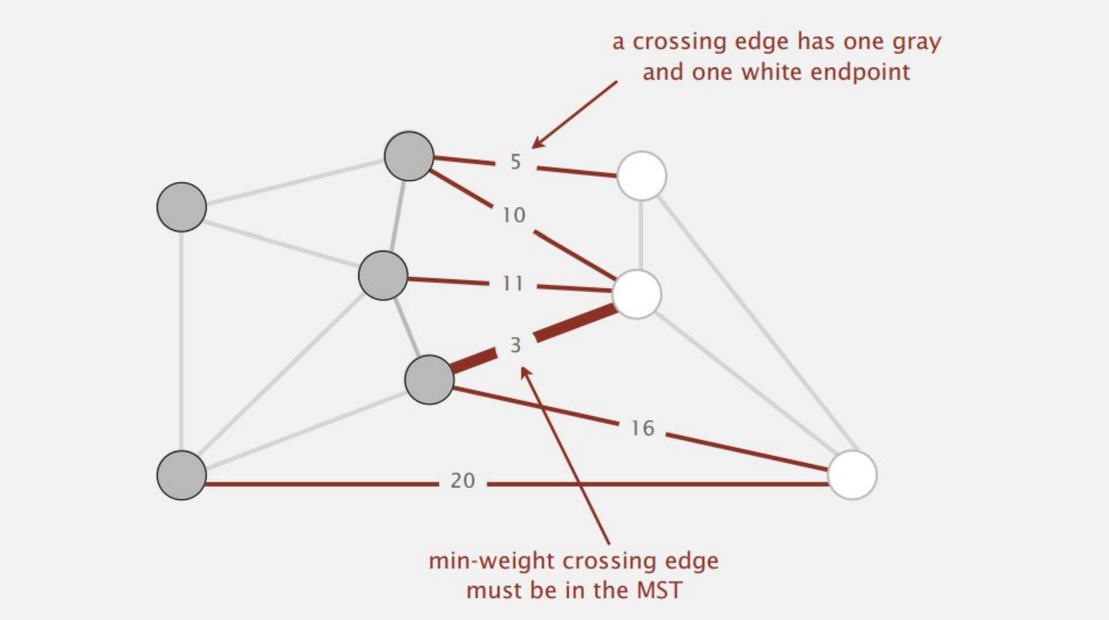
3) Do the edges have weights?

4) Will it always yield a shortest path

Cuts

1) What are they?

2) Cut property?

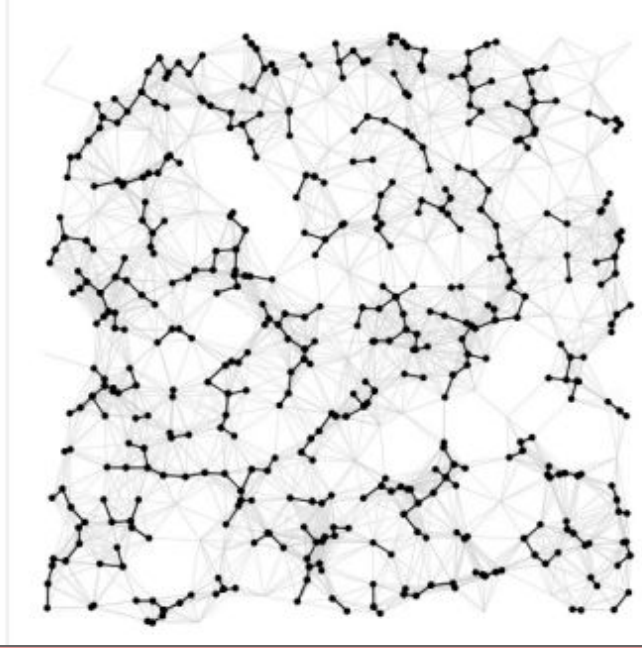

a crossing edge has one gray and one white endpoint
5
10
11
3
16
20
min-weight crossing edge must be in the MST

Credits = Princeton University (COS 226)

Kruskal’s Algorithm

1) Pseudocode

2) Data Type to Use for Implementation?

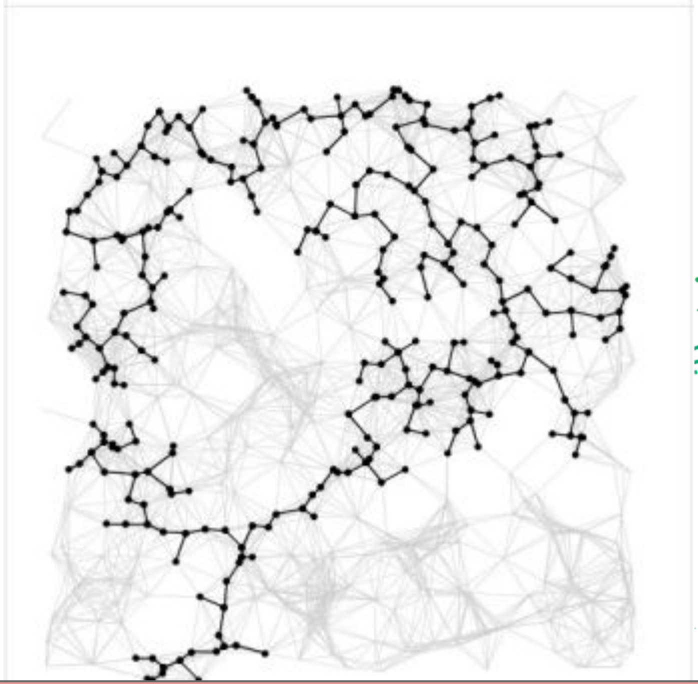


Credits = Princeton University (COS 226)

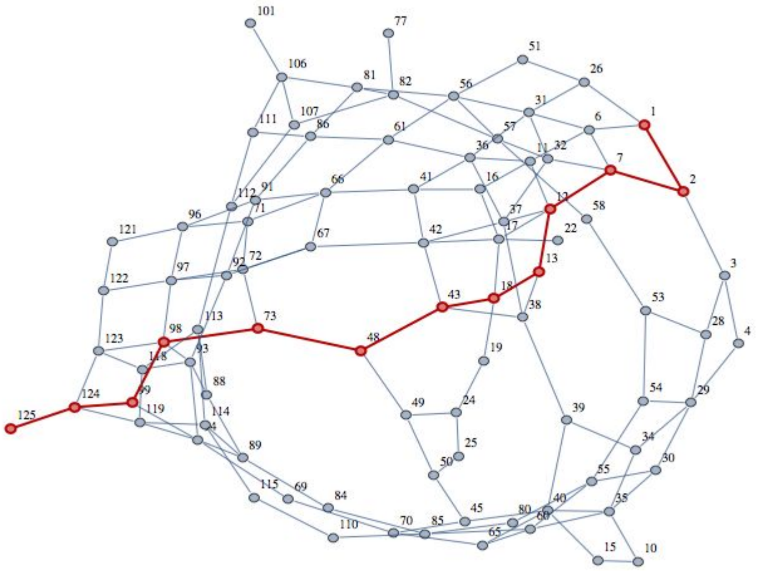
Prim’s Algorithm

1) Pseudocode

subgraph



Credits = Princeton University (COS 226)



Week 3: Shortest Paths

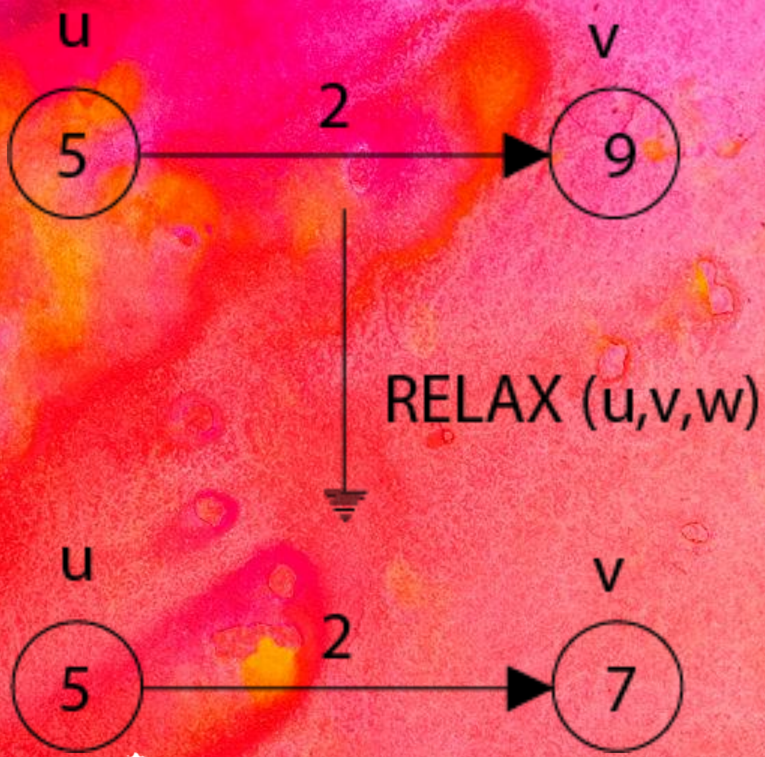
Define the following

1) Are there cycles in the graph?

2) Data structures needed?

3) What type of shortest paths does GPS use?

4) Define edge relaxation


u
v
5
2
9
RELAX (u,v,w)
u
v
5
2
7

Bellman Ford Algorithm

1) Pseudocode

2) Runtime analysis

3) How to improve worst case?

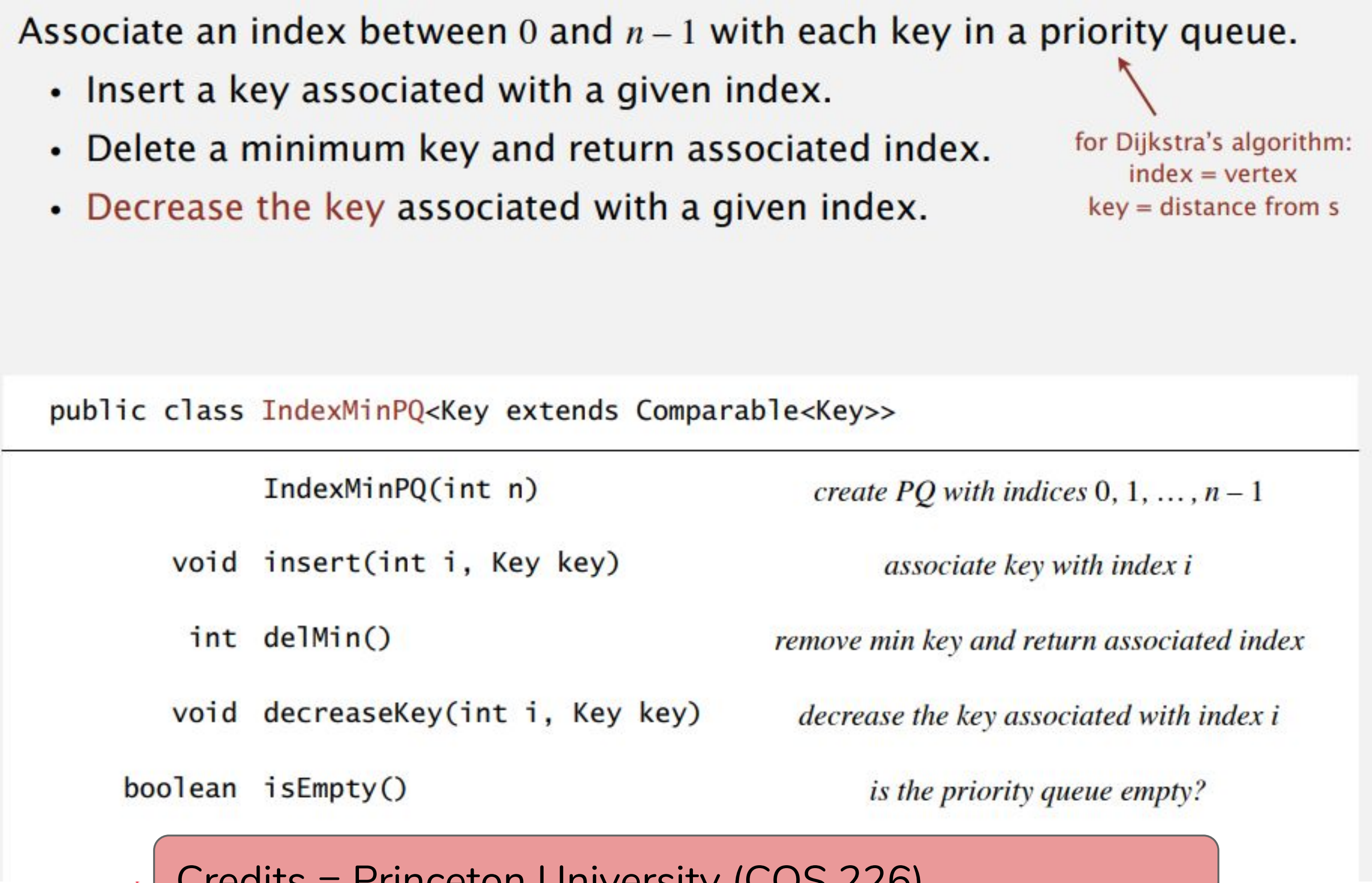
Dijkstra’s Algorithm

1) Pseudocode

2) Directed cycles and negative weights?

Dijkstra’s Algorithm

1) Worst case runtime & implementation



Associate an index between 0 and $n - 1$ with each key in a priority queue.

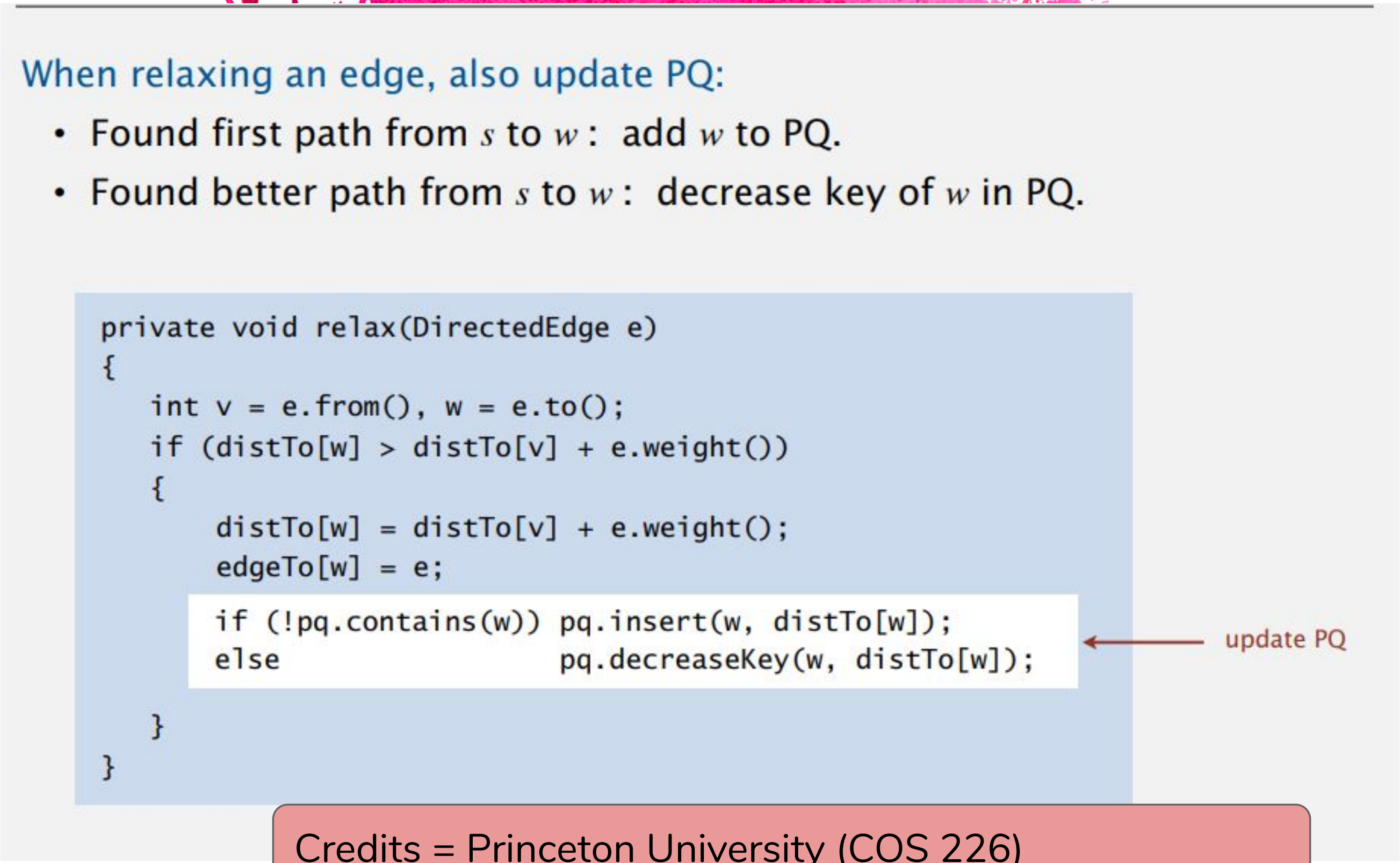
- Insert a key associated with a given index.
- Delete a minimum key and return associated index.
- Decrease the key associated with a given index.

for Dijkstra's algorithm:
index = vertex
key = distance from s

```
public class IndexMinPQ<Key extends Comparable<Key>>
```

	`IndexMinPQ(int n)`	*create PQ with indices* $0, 1, \ldots, n - 1$
`void`	`insert(int i, Key key)`	*associate key with index i*
`int`	`delMin()`	*remove min key and return associated index*
`void`	`decreaseKey(int i, Key key)`	*decrease the key associated with index i*
`boolean`	`isEmpty()`	*is the priority queue empty?*

Credits = Princeton University (COS 226)



When relaxing an edge, also update PQ:

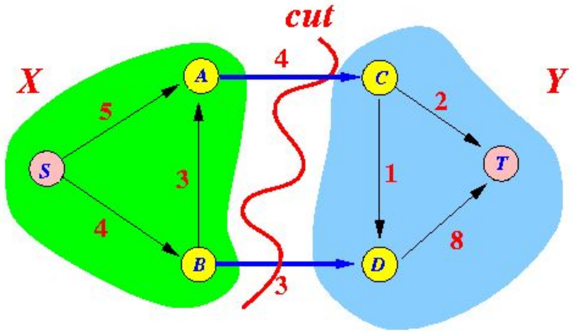
- Found first path from s to w: add w to PQ.
- Found better path from s to w: decrease key of w in PQ.

```
private void relax(DirectedEdge e)
{
   int v = e.from(), w = e.to();
   if (distTo[w] > distTo[v] + e.weight())
   {
       distTo[w] = distTo[v] + e.weight();
       edgeTo[w] = e;
       if (!pq.contains(w)) pq.insert(w, distTo[w]);
       else                 pq.decreaseKey(w, distTo[w]);

   }
}
```

update PQ

Credits = Princeton University (COS 226)



Cut = { (A,C), (B,D) }

Week 4: Maxflows & Mincuts

Some MST applications:

- *Want to get maxflow over water pipes over city network across source to sink*
- *Bipartite matching problem*
  - *People to tasks*
  - *Donating blood types*
  - *Stable marriage prob*

Define the following

1) Maxflow problem?

2) Mincut problem?

Ford Fulkerson Algorithm

1) Augmenting path

2) Pseudocode

Maxflow Mincut Theorem

1) Augmenting path theorem

2) Mincut maxflow theorem

3) How to get mincut from maxflow

Bipartite Matching Problem

1) N people
2) N tasks
3) Assign people to tasks (Every person has 1 task and has a qualified person = edge goes in between)
4) Construct flow network by adding source, sink, edges, and capacities
5) Maxflow problem and use FF algo ⇒ get maxflow ⇒ yields in perfect matching

We are halfway done with winter 2020 AI Inspire session! :)

https://www.youtube.com/watch?reload=9&v=QvyTEx1wyOY