



**MAHIDOL
UNIVERSITY**

Wisdom of the Land

[SCPY204]

Computer Programming for Physicists

Lecture 12: 20 April 2017

Content: Data structures, introduction to **Graph
Theory**

Instructor: Puwis Amatyakul



2017

**“Happy (Thai)
New Year”**

Today's Goals

Part I: (More) Data structure in Python

Part II: Graph Theory

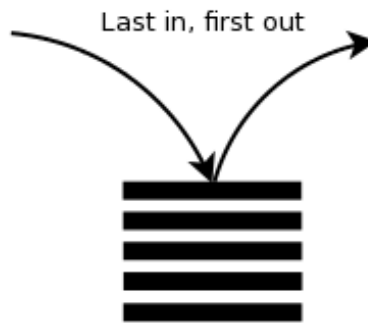
Part II: Exercises

Data Structures: stack and queue

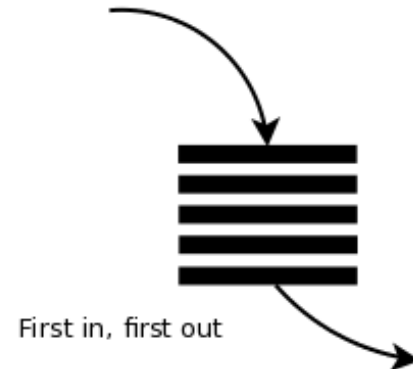
Derived Data Types

- List
- Array
- Stack
- Queue

Stack:



Queue:



The abstract data structures stack and queue.

Source: <http://cs.joensuu.fi/~appro/ics/05-03.php>

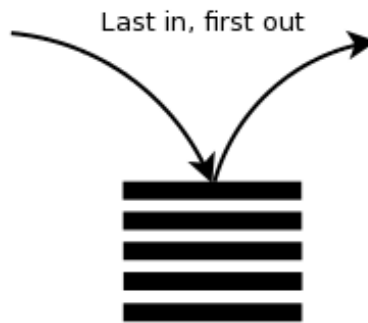
- ***Stack* and *queue*** are more restricted data structures than list.
- Element are arranged sequentially like in a list,
- but element additions and removals are possible to execute only to the ends of the collection.

Data Structures: stack and queue

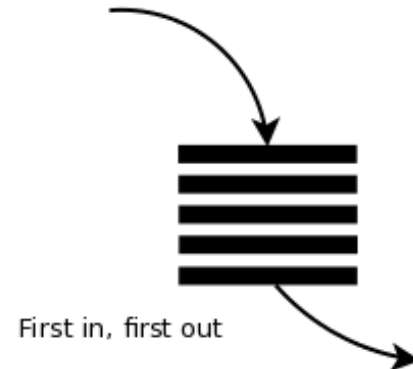
Derived Data Types

- List
- Array
- Stack
- Queue

Stack:



Queue:



The abstract data structures stack and queue.

Source: <http://cs.joensuu.fi/~appro/ics/05-03.php>

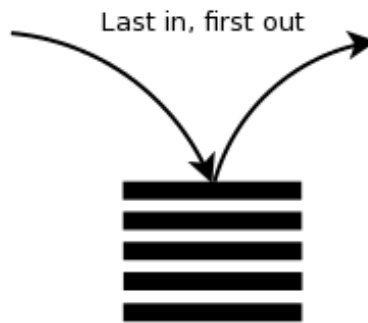
- ✓ In stack one of the collection ends (top) is used to element addition and removal,
- ✓ and thus the element added last is removed first.
- ✓ This is why stack is called as a **LIFO** structure (**Last In, First Out**).

Data Structures: stack and queue

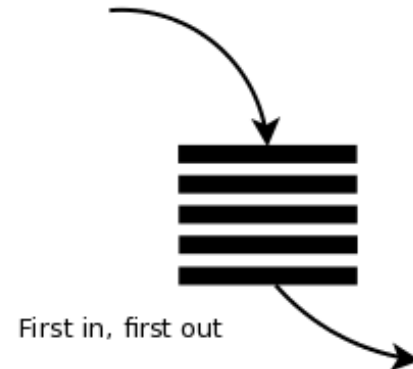
Derived Data Types

- List
- Array
- Stack
- Queue

Stack:



Queue:



The abstract data structures stack and queue.

Source: <http://cs.joensuu.fi/~appro/ics/05-03.php>

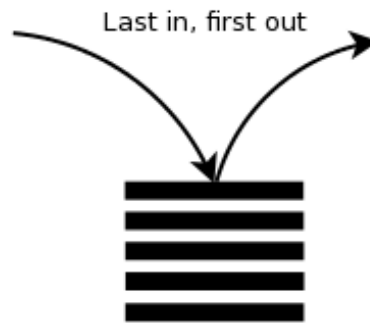
- ✓ In queue elements are added to one of the collection's ends (tail) and removed from the other end (head),
- ✓ and thus the element added first is also removed first.
- ✓ Hence, queue is called a **FIFO** structure (**First In, First Out**).

Data Structures: stack and queue

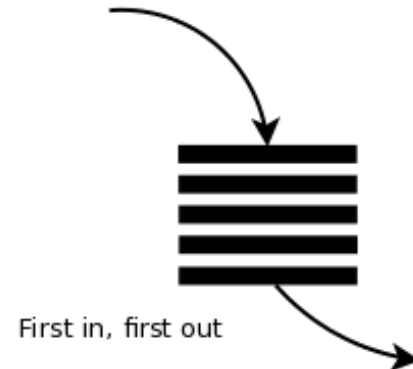
Derived Data Types

- List
- Array
- Stack
- Queue

Stack:



Queue:



The abstract data structures stack and queue.

Source: <http://cs.joensuu.fi/~appro/ics/05-03.php>

- ✓ In queue elements are added to one of the collection's ends (tail) and removed from the other end (head),
- ✓ and thus the element added first is also removed first.
- ✓ Hence, queue is called a **FIFO** structure (**First In, First Out**).

Data Structures: stack and queue in Python

Python's List

The list data type has some more methods. Here are all of the methods of list objects:

`list.append(x)`

Add an item to the end of the list; equivalent to `a[len(a):] = [x]`.

`list.extend(L)`

Extend the list by appending all the items in the given list; equivalent to `a[len(a):] = L`.

`list.insert(i, x)`

Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.remove(x)`

Remove the first item from the list whose value is `x`. It is an error if there is no such item.

Data Structures: stack and queue in Python

Python's List

`list.pop([i])`

Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list.

`list.index(x)`

Return the index in the list of the first item whose value is `x`. It is an error if there is no such item.

`list.count(x)`

Return the number of times `x` appears in the list.

`list.sort(cmp=None, key=None, reverse=False)`

Sort the items of the list in place (the arguments can be used for sort customization).

`list.reverse()`

Reverse the elements of the list, in place.

Data Structures: stack and queue in Python

Python's List: Using Lists as Stacks

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

Data Structures: stack and queue in Python

Python's List: Using Lists as Stacks

```
>>> from collections import deque
>>> queue = deque(["Eric", "John", "Michael"])
>>> queue.append("Terry") # Terry arrives
>>> queue.append("Graham") # Graham arrives
>>> queue.popleft() # The first to arrive now leaves
'Eric'
>>> queue.popleft() # The second to arrive now leaves
'John'
>>> queue # Remaining queue in order of arrival
deque(['Michael', 'Terry', 'Graham'])
```

Today's Goals

Part I: (More) Data structure in Python

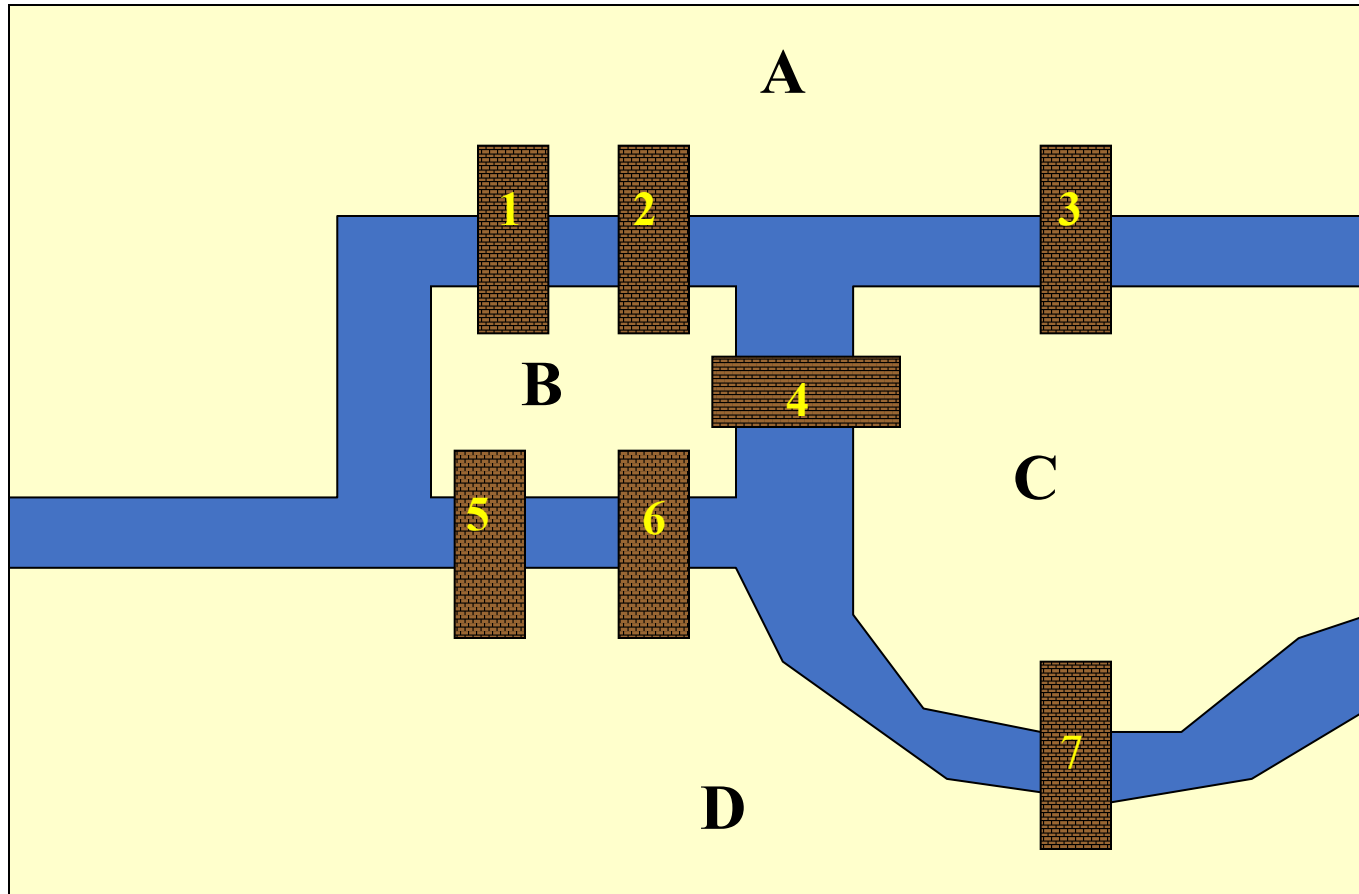
Part II: Graph Theory

Part II: Exercises

The Bridges of Königsberg: Euler 1736

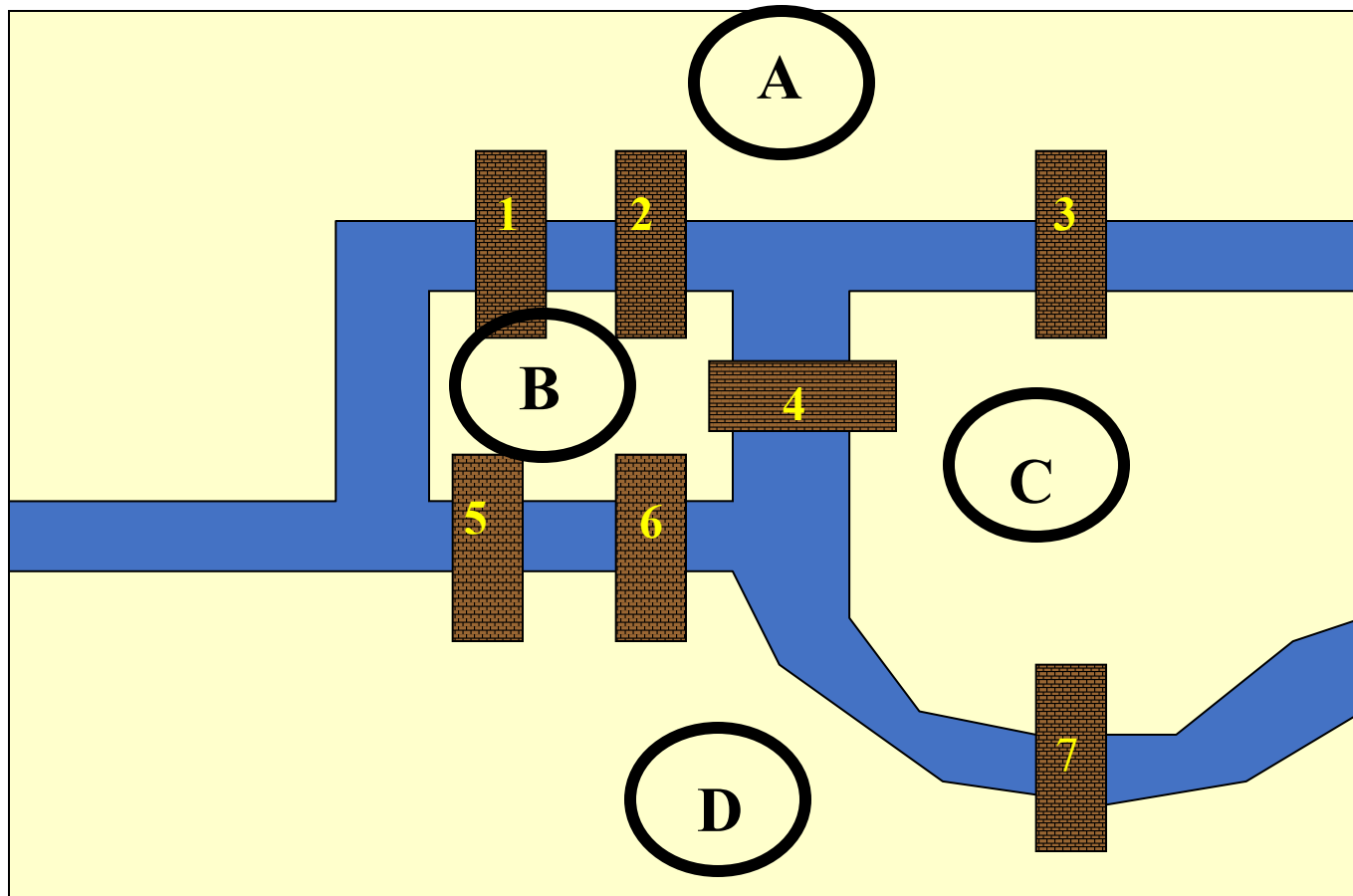
- “Graph Theory” began in 1736
- Leonhard Euler
 - Visited Königsberg
 - People wondered whether it is possible to take a walk, end up where you started from, and cross each bridge in Königsberg exactly once
 - Generally it was believed to be impossible

The Bridges of Koenigsberg: Euler 1736



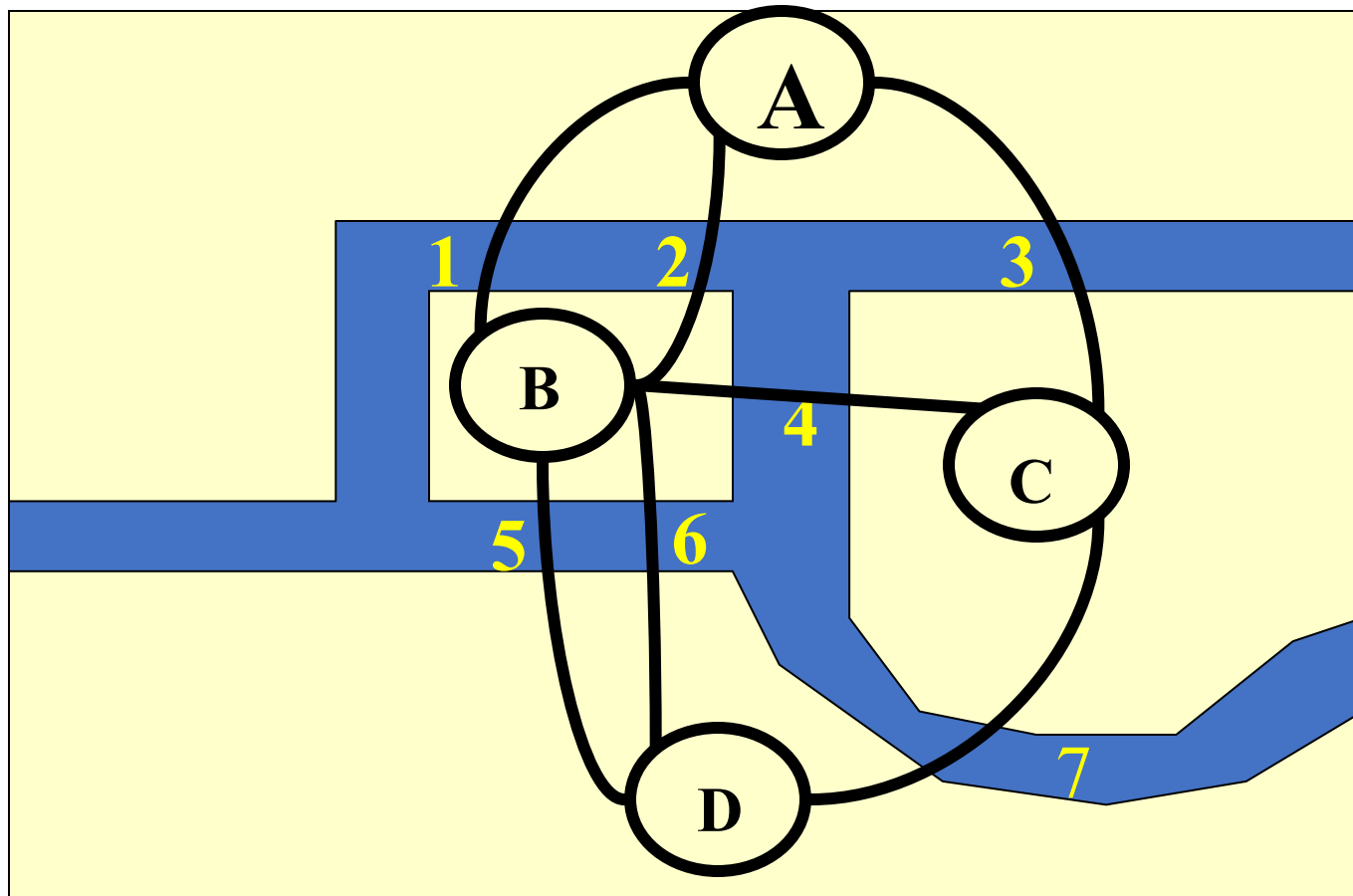
Is it possible to start in A, cross over each bridge exactly once, and end up back in A?

The Bridges of Koenigsberg: Euler 1736



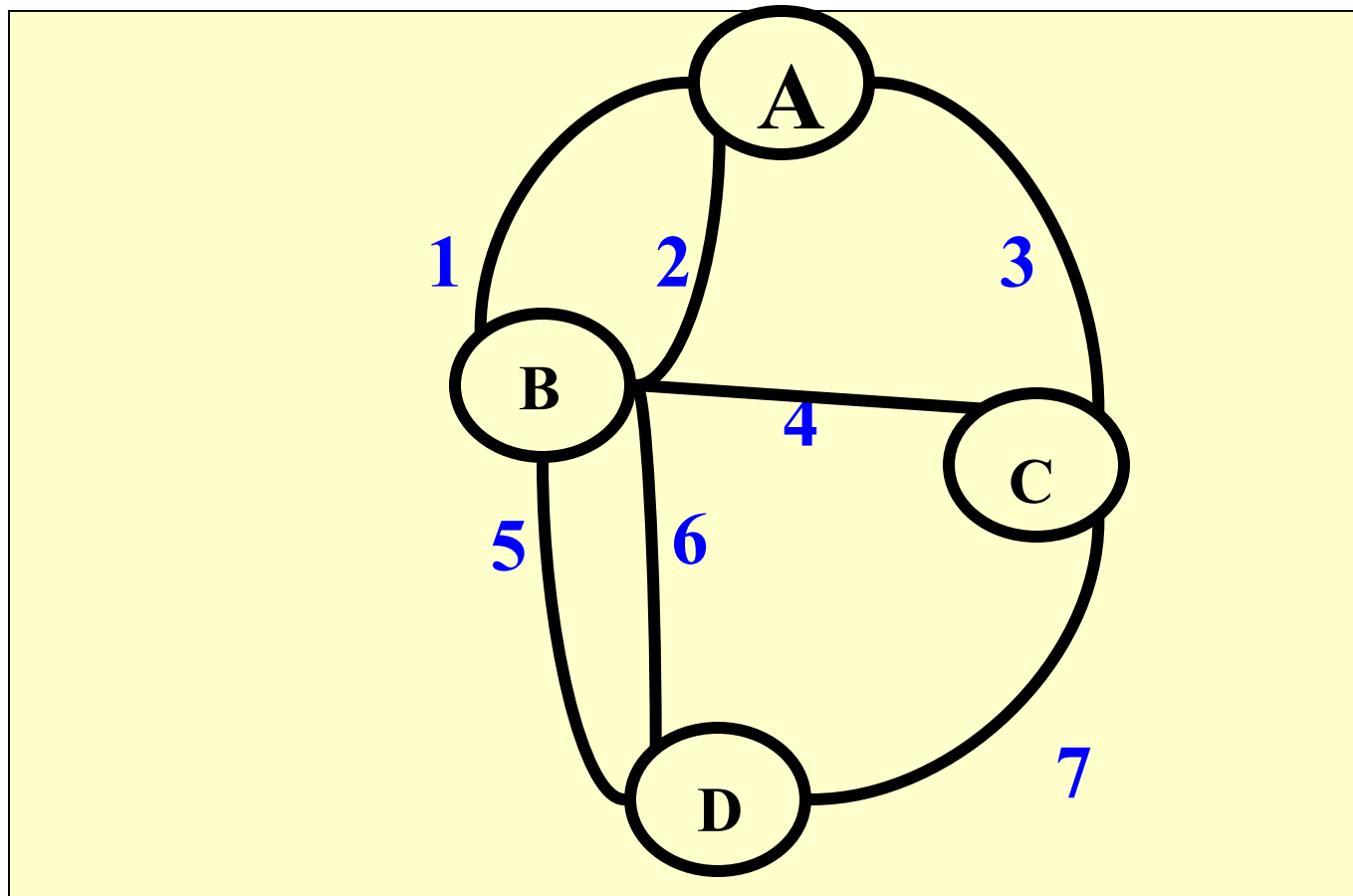
Conceptualization: Land masses are “nodes”.

The Bridges of Koenigsberg: Euler 1736



Conceptualization: Bridges are “arcs.”

The Bridges of Königsberg: Euler 1736



Is there a “walk” starting at A and ending at A and passing through each arc exactly once?

Graph Theory: Basic terminology

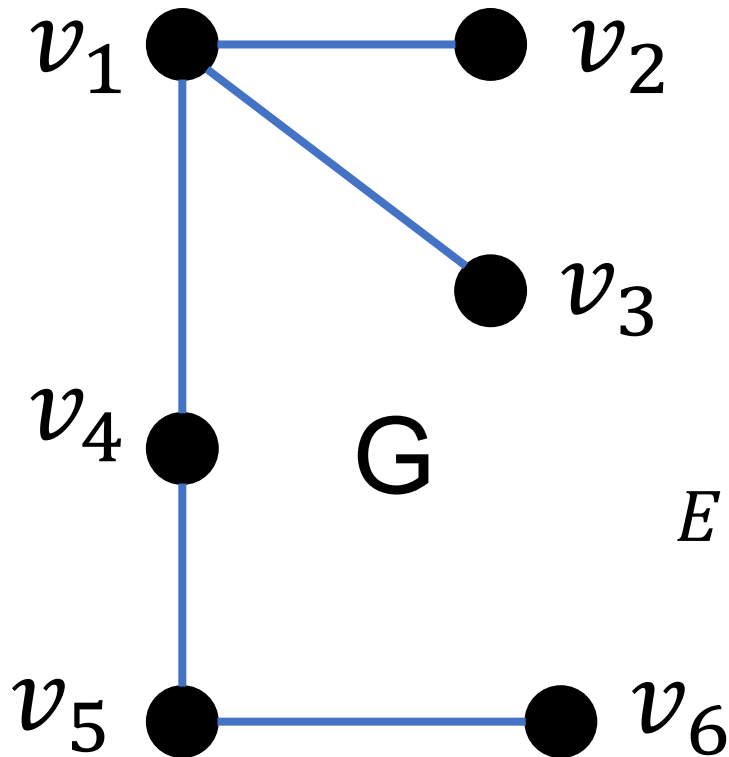
Basic terminology and ideas

Graph = finite set of vertices, V , along with a set of edges, E .
 E is a 2-element subset of V .

Graph Theory:

Basic terminology and ideas

Graph = finite set of vertices, V , along with a set of edges, E .
 E is a 2-element subset of V .



Vertices and Edges

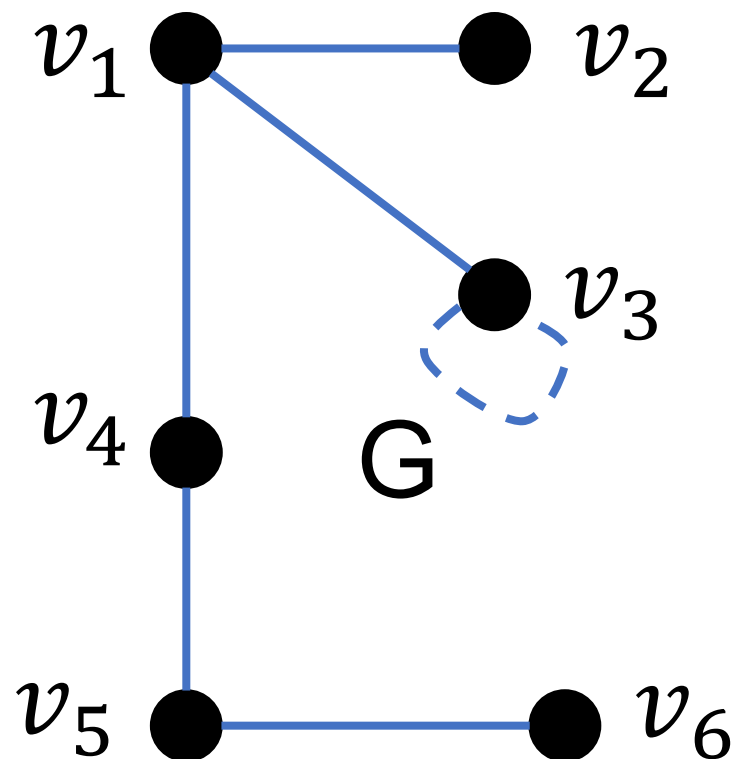
$$V = \{v_1, v_2, \dots, v_6\}$$

$$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \dots, \{v_5, v_6\}\}$$

Graph Theory:

Basic terminology and ideas

Graph = finite set of vertices, V , along with a set of edges, E .
 E is a 2-element subset of V .



Vertices and Edges

$$V = \{v_1, v_2, \dots, v_6\}$$

$$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \dots, \{v_5, v_6\}\}$$

Cardinality of G

$$|G| = 6$$

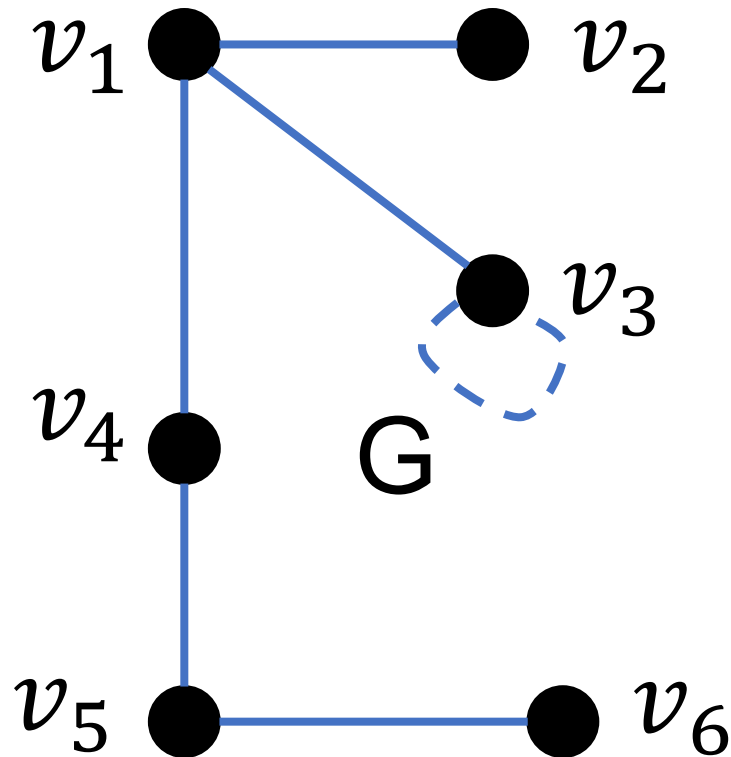
Degree of v

$$\deg(v_1) = 3$$

Graph Theory:

Basic terminology and ideas

Graph = finite set of vertices, V , along with a set of edges, E .
 E is a 2-element subset of V .



Adjacency List

$v_1: v_2, v_3, v_4$

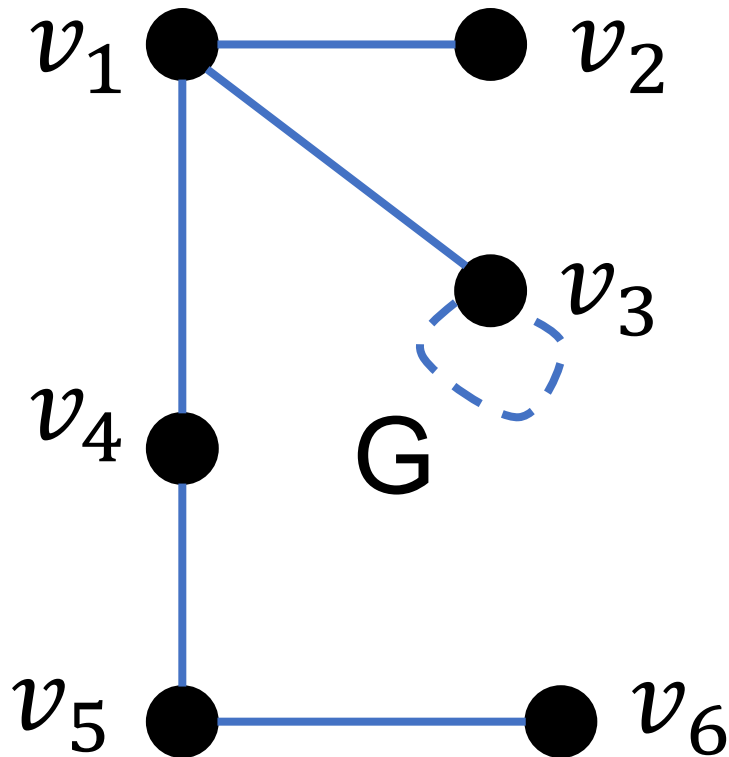
$v_2: v_1$

...

Graph Theory:

Basic terminology and ideas

Graph = finite set of vertices, V , along with a set of edges, E .
 E is a 2-element subset of V .



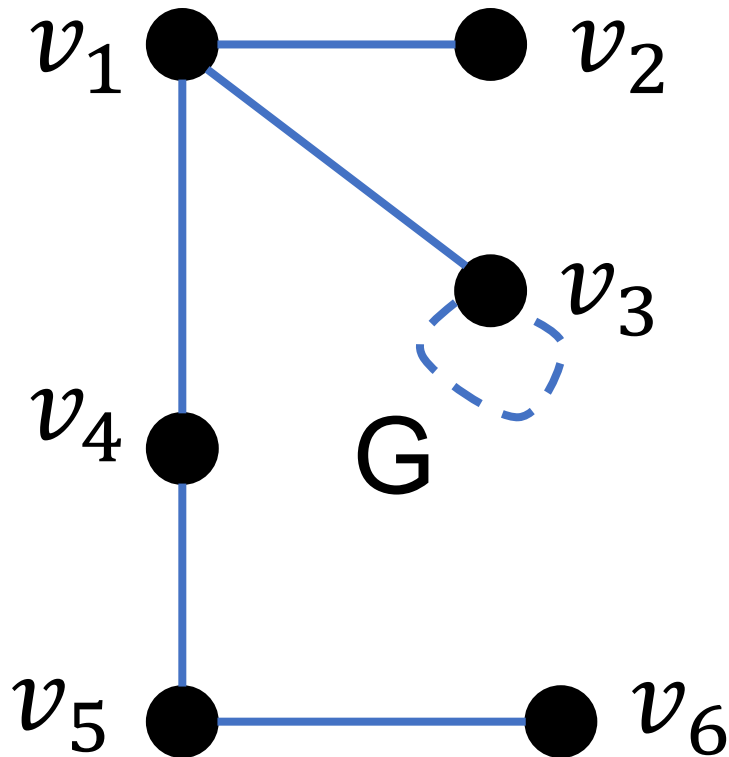
Adjacency Matrix

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	1	1	0	0
v_2	1	0	0	0	0	0
v_3	1	0	0	0	0	0
v_4	1	0	0	0	1	0
v_5	0	0	0	1	1	0
v_6	0	0	0	0	1	0

Graph Theory:

Basic terminology and ideas

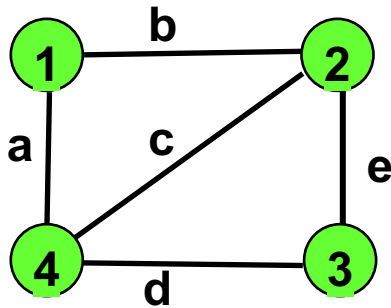
Graph = finite set of vertices, V , along with a set of edges, E .
 E is a 2-element subset of V .



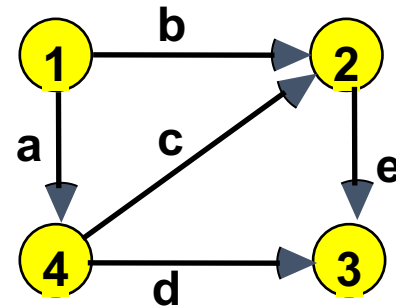
Traversal

Graph Theory:

Basic terminology and ideas



An Undirected Graph or
Undirected Network



A Directed Graph or
Directed Network

More Jargons

Edges: Loops, Multiple edges, Directed edges

Traversal: Walk, Trail, Path

Today's Goals

Part I: (More) Data structure in Python

Part II: Graph Theory

Part II: Exercises

Graph Theory: exercise

Problem: Bangkok train network

1. Can you draw a graph representative of currently available Bangkok's train network?
The graph should include only terminal station and the interconnection station
(Only sky and subway, no regional train)
2. How many vertices (station) that have a degree > 2 ?
3. Path to **Phloen Chit station** from **Lad Phrao station**
 1. List three possible paths.
 2. Use any approximation on distance (or time) between relevant stations.
 3. Find the shortest distance (or time) path.
 4. Find the shortest distance (or time) path.
 5. [Challenge] Estimate the cost between stations based on the actual rate.
 6. [Challenge] Find the path with the minimum cost.
 7. [Challenge] Find the optimum path between cost vs distance (time).

Instruction

- Try to do it on the given paper first, then let do these exercises in Python.
- You may try inventing your own algorithm first.
- After that, compare you algorithm with **Dijkstra's Algorithm**.
(https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

