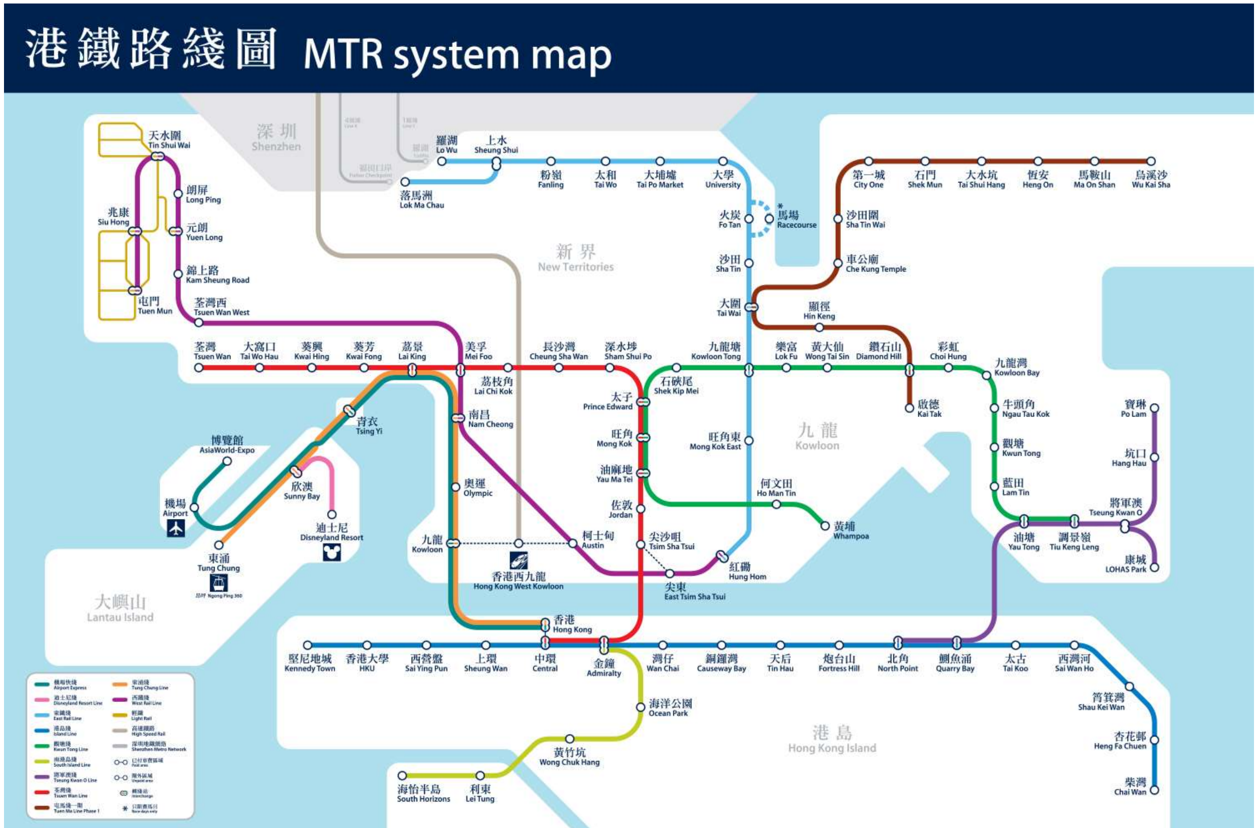
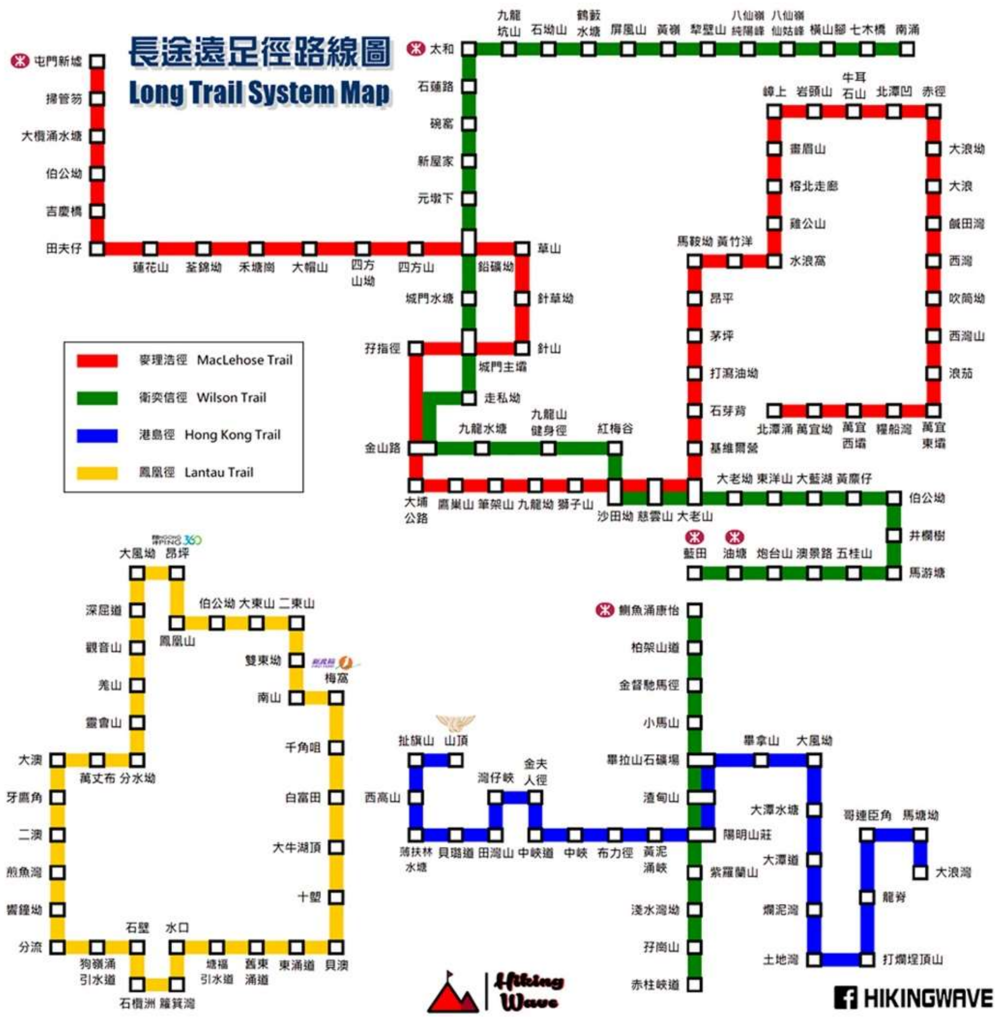
Graphs

Graph in daily life

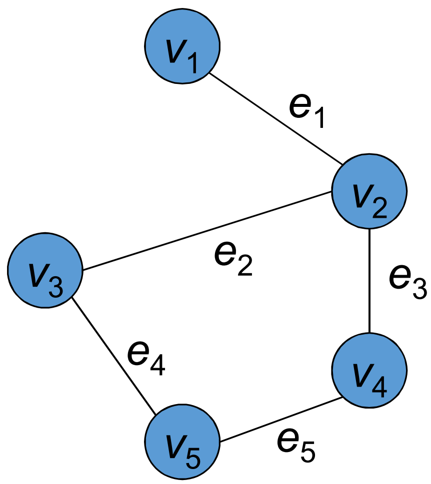




長途遠足徑路線圖
Long Trail System Map
麥理浩徑 MacLehose Trail
衛奕信徑 Wilson Trail
港島徑 Hong Kong Trail
鳳凰徑 Lantau Trail
屯門新墟
掃管笏
大欖涌水塘
伯公坳
吉慶橋
田夫仔
蓮花山
荃錦坳
禾塘崗
大帽山
四方山坳
四方山
草山
鉛礦坳
針草坳
針山
城門水塘
城門主壩
孖指徑
走私坳
金山路
九龍水塘
九龍山健身徑
紅梅谷
大埔公路
鷹巢山
筆架山
九龍坳
獅子山
沙田坳
慈雲山
大老山
太和
石礦路
碗窯
新屋家
元墩下
九龍坑山
石坳山
鶴藪水塘
屏風山
黃嶺
犁壁山
八仙嶺純陽峰
八仙嶺仙姑峰
橫山腳
七木橋
南涌
嶂上
岩頭山
牛耳石山
北潭凹
赤徑
大浪坳
大浪
鹹田灣
西灣
吹筒坳
西灣山
浪茄
萬宜東壩
耀船灣
萬宜西壩
萬宜坳
北潭涌
畫眉山
榕北走廊
雞公山
水浪窩
馬鞍坳
黃竹洋
昂平
茅坪
打爛油坳
石芽背
基維爾營
大老坳
東洋山
大藍湖
黃麖仔
伯公坳
井欄樹
馬游塘
五桂山
澳景路
炮台山
油塘
藍田
鰂魚涌康怡
柏架山道
金督馳馬徑
小馬山
畢拉山石礦場
渣甸山
陽明山莊
紫羅蘭山
淺水灣坳
孖崗山
赤柱峽道
畢拿山
大風坳
大潭水塘
大潭道
爛泥灣
土地灣
打爛埕頂山
龍脊
哥連臣角
馬塘坳
大浪灣
扯旗山
山頂
西高山
薄扶林水塘
貝璐道
田灣山
灣仔峽
金夫人徑
中峽道
中峽
布力徑
黃泥涌峽
大風坳
昂坪
深屈道
觀音山
羗山
靈會山
大澳
萬丈布
分水坳
牙鷹角
二澳
煎魚灣
響鐘坳
分流
狗嶺涌引水道
石壁
石欖洲
水口
籮箕灣
塘福引水道
舊東涌道
東涌道
貝澳
十塱
大牛湖頂
白富田
千角咀
梅窩
南山
雙東坳
伯公坳
大東山
二東山
鳳凰山
HIKINGWAVE
Hiking Wave

What is a Graph?

- A graph consists of some ***vertices*** (a.k.a. ***nodes***) and some ***edges*** (a.k.a. ***arcs***).



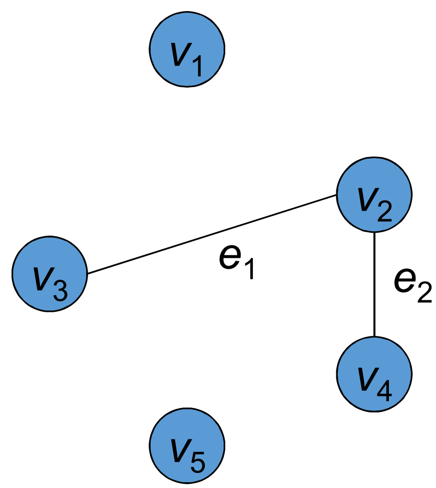
Vertices:
v_1, v_2, v_3, v_4, v_5

Edges:
$e_1 = (v_1, v_2)$
$e_2 = (v_2, v_3)$
$e_3 = (v_2, v_4)$
$e_4 = (v_3, v_5)$
$e_5 = (v_4, v_5)$

A graph with 5 *vertices* and 5 *edges*.

What is a Graph?

- A graph consists of some ***vertices*** and some ***edges***.



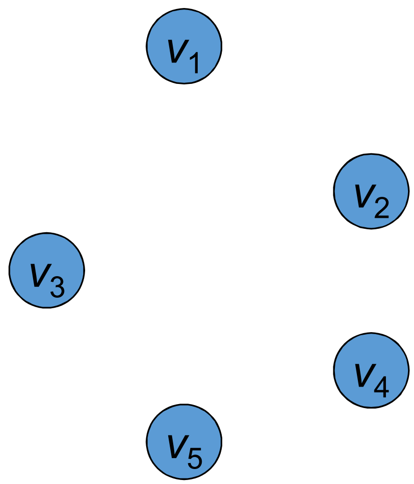
Vertices:
v_1, v_2, v_3, v_4, v_5

Edges:
$e_1 = (v_2, v_3)$
$e_2 = (v_2, v_4)$

A graph with 5 *vertices* and 2 *edges*.

What is a Graph?

- A graph consists of some ***vertices*** and some ***edges***.



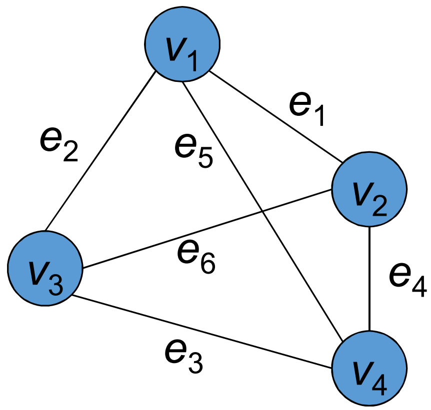
Vertices:
v_1, v_2, v_3, v_4, v_5

Edges:

A graph with 5 *vertices* and 0 *edges*.

What is a Graph?

- A graph consists of some ***vertices*** and some ***edges***.



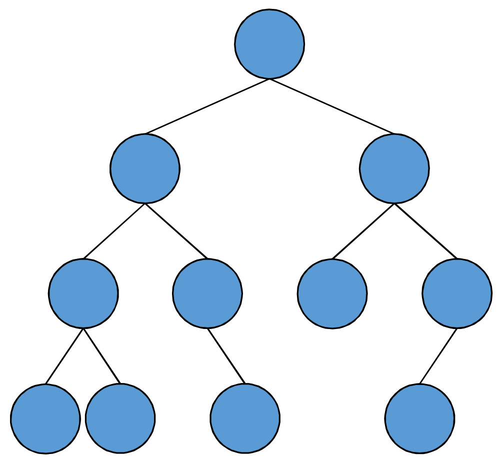
Vertices:
v_1, v_2, v_3, v_4

Edges:
$e_1 = (v_1, v_2)$
$e_2 = (v_1, v_3)$
$e_3 = (v_3, v_4)$
$e_4 = (v_2, v_4)$
$e_5 = (v_1, v_4)$
$e_6 = (v_2, v_3)$

A graph with 4 *vertices* and 6 *edges*.

Graph and Tree

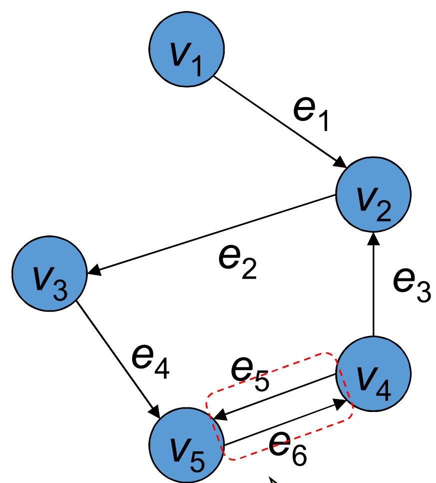
- A tree is also a graph (but not vice versa).



A graph with 11 *vertices* and 10 *edges*.

Directed Graph

- Edges in a graph can be ***directed***. A graph with directed edges is a ***directed graph*** (digraph).



Vertices:

v_1, v_2, v_3, v_4, v_5

Edges:

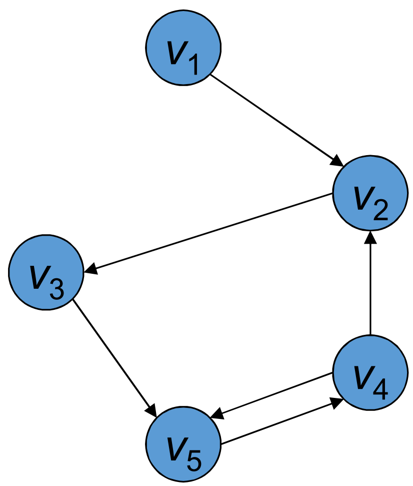
$e_1 = (v_1, v_2)$
$e_2 = (v_2, v_3)$
$e_3 = (v_4, v_2)$
$e_4 = (v_3, v_5)$
$e_5 = (v_4, v_5)$
$e_6 = (v_5, v_4)$

A directed graph with 5 *vertices* and 6 *directed edges*.

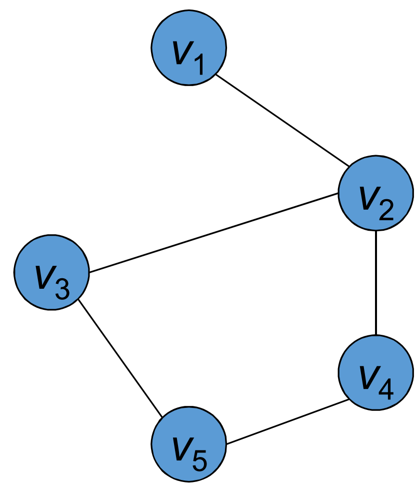
(v_4, v_5) and (v_5, v_4) are different edges.

head ***tail***

Directed and Undirected Graphs



Directed graph

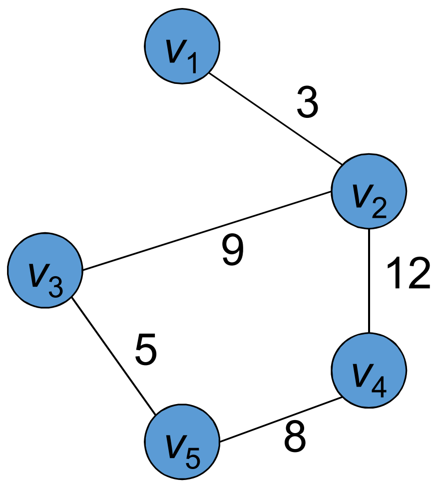


Undirected graph

These two graphs are different.

Weighted Graph

- Edges in a graph can have ***weights***. A graph with weighted edges is a ***weighted graph***.



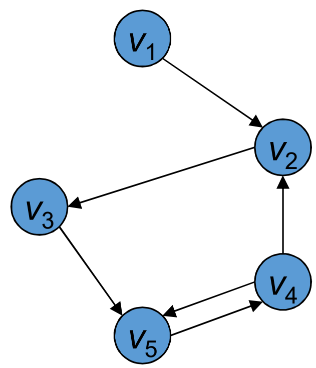
Vertices:

v_1, v_2, v_3, v_4, v_5

Edges:

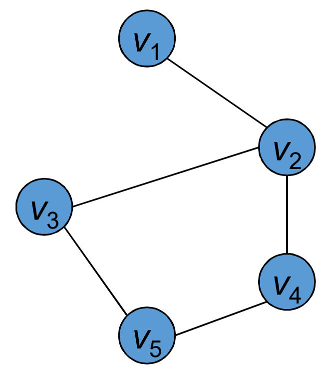
(v_1, v_2), weight = 3
(v_2, v_3), weight = 9
(v_2, v_4), weight = 12
(v_3, v_5), weight = 5
(v_4, v_5), weight = 8

(Un)directed (Un)weighted Graph

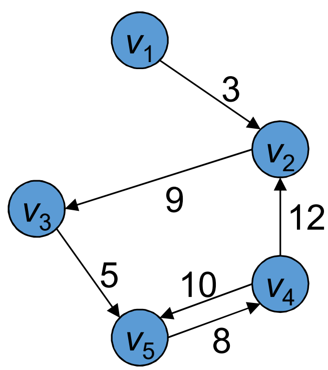


Directed unweighted graph

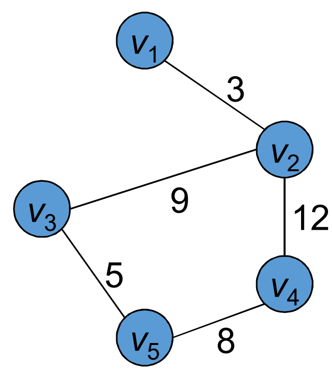
All four graphs are different.



Undirected unweighted graph

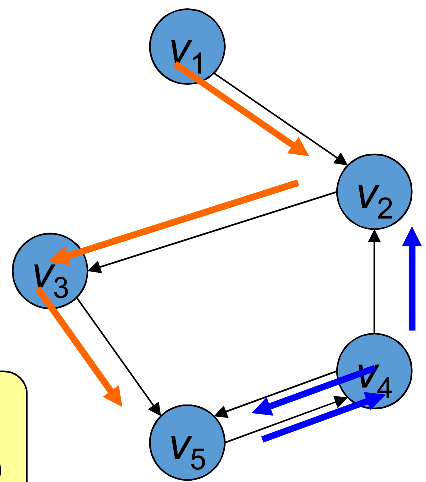


Directed weighted graph



Undirected weighted graph

Graph Terminologies



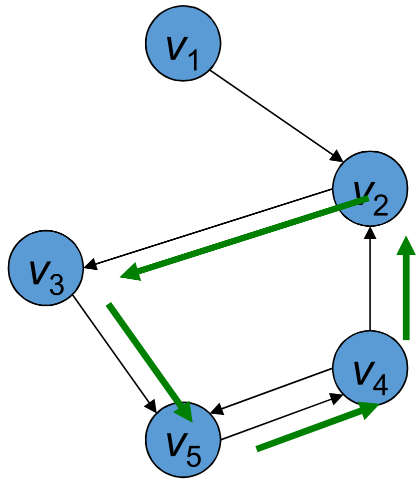
Vertex v_3 is ***adjacent*** to v_2.
(But v_2 is ***not*** adjacent to v_3.)

A ***path***: v_1, v_2, v_3, v_5
(***Length*** of path = 3)

Another ***path***: v_5, v_4, v_5 , v_4, v_2
(***Length*** of path = 4)

A ***simple*** path is a path with no duplicating vertices.
E.g., the path v_1, v_2, v_3, v_5 is simple, while v_5, v_4, v_5, v_4, v_2 is not simple.

Graph Terminologies



A ***cycle***: v_2, v_3, v_5, v_4, v_2

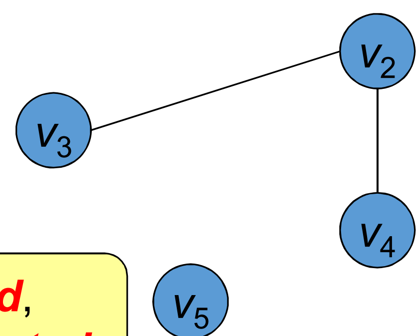
A ***simple*** cycle is a cycle with no duplicating vertices except the first and last vertex.

An ***acyclic*** graph is a graph with *no* cycles.

A ***d***irected ***a***cyclic ***g***raph is sometimes called a ***DAG***.

Graph Connectivity

- In an *undirected* graph, two vertices *u* and *v* are ***connected*** if there is a path from *u* to *v*.
- An undirected graph is ***connected*** if *all* pairs of vertices are connected.

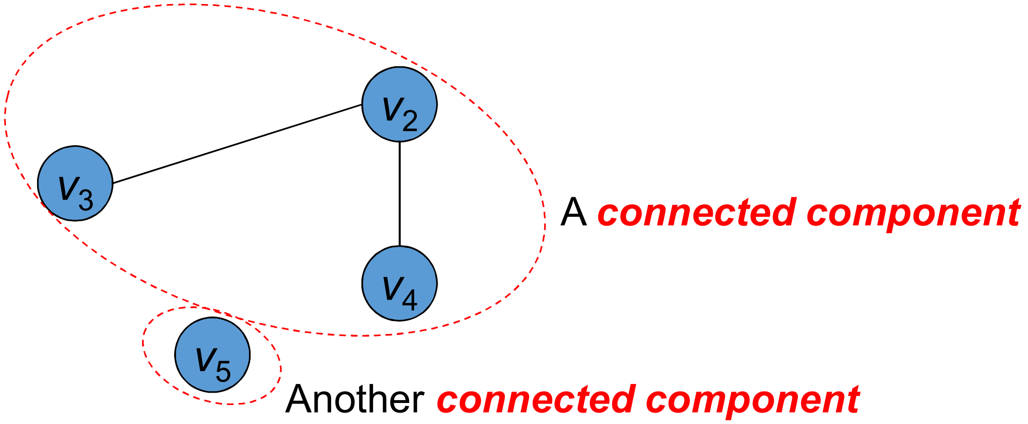


v_3 and v_4 are ***connected***, but v_3 and v_5 are ***disconnected***.

Thus, this is a ***disconnected*** graph.

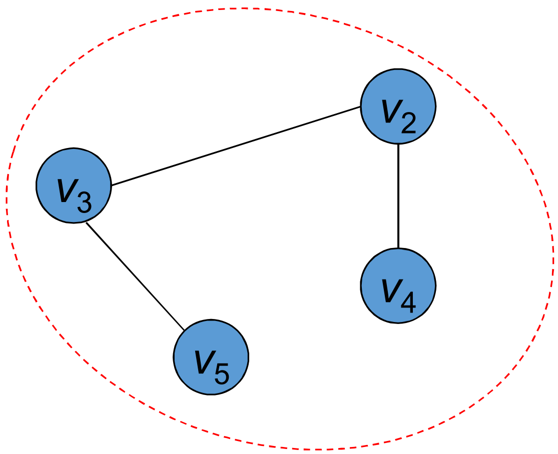
Graph Connectivity

This graph has 2 ***connected components***.



Graph Connectivity

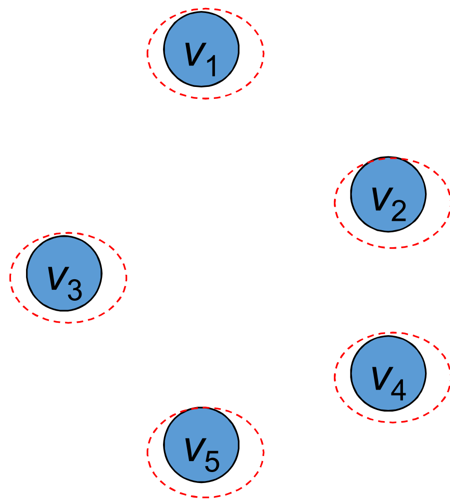
This graph has 1 ***connected component***.



A ***connected*** graph is a graph that has only ***one connected component***.

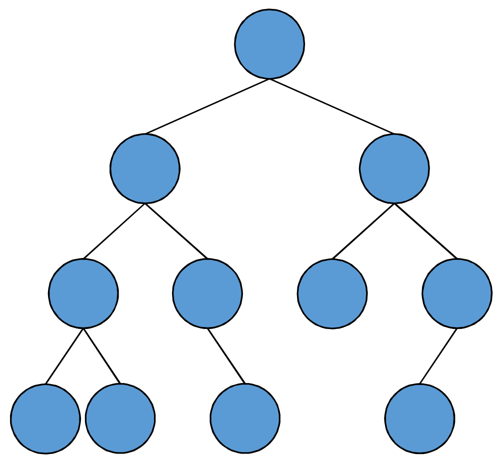
Graph Connectivity

This graph has 5 ***connected components***.



Graph and Tree

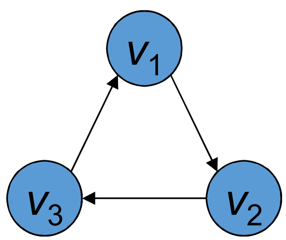
- A tree is a *connected acyclic undirected* graph.



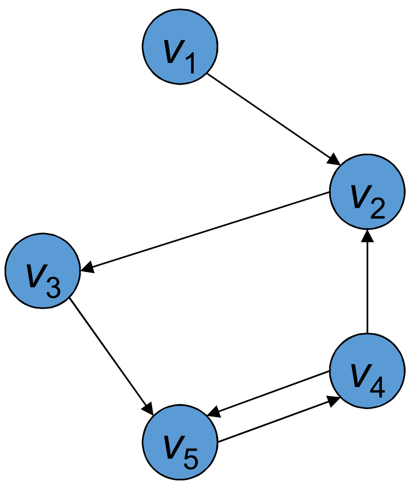
A graph with 11 *vertices* and 10 *edges*.

Graph Connectivity

- A *directed* graph is ***strongly connected*** if for every pair of vertices, *u* and *v*, it contains a directed path from *u* to *v*.



A ***strongly connected*** graph

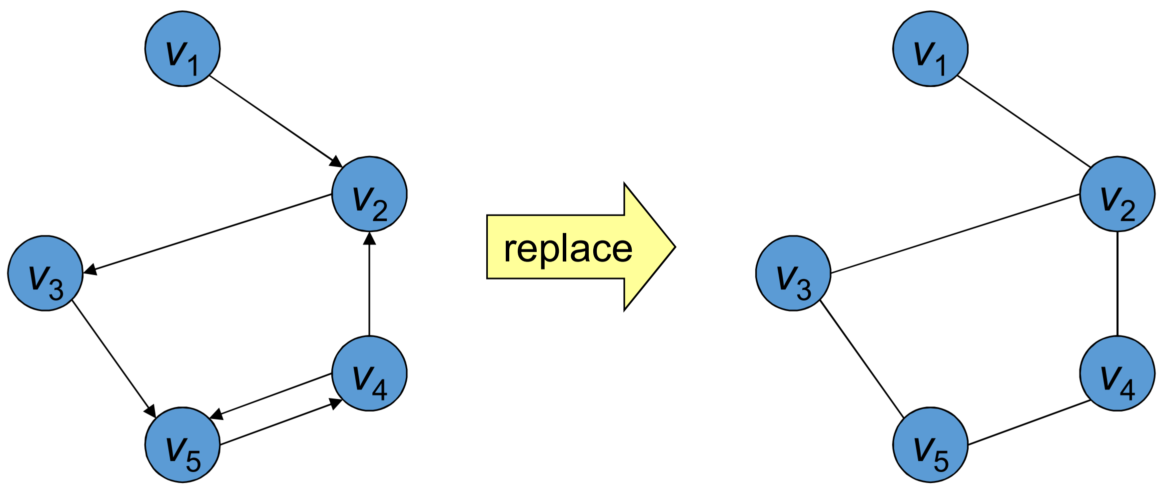


Not a ***strongly connected*** graph

Graph Connectivity

- A *directed* graph is ***weakly connected*** if replacing all directed edges with undirected edges produces a connected (undirected) graph.

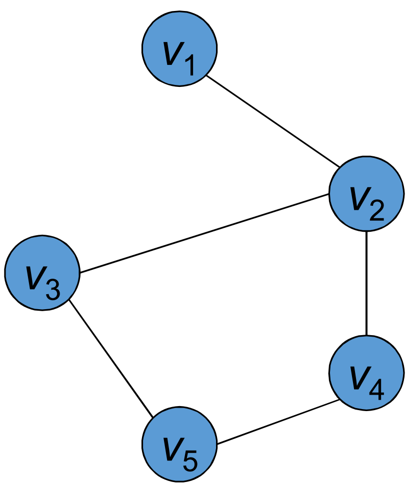
Note that a ***strongly connected*** graph is also a ***weakly connected*** graph, but ***not*** vice versa.



Thus, this graph is ***weakly connected***.

Degree of Graph

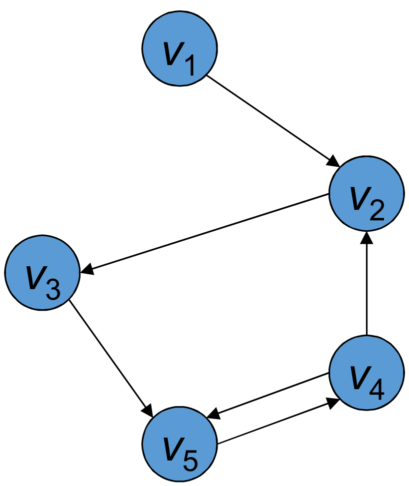
- In an *undirected* graph, the ***degree*** of a vertex v is the number of adjacent vertices to v.



Vertex	Degree
v_1	1
v_2	3
v_3	2
v_4	2
v_5	2

Degree of Graph

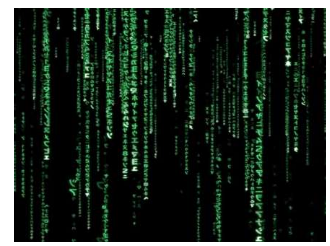
- In a *directed* graph,
  - the ***out-degree*** of a vertex *v* is the number of edges whose *head* is *v*.
  - the ***in-degree*** of a vertex *v* is the number of edges whose *tail* is *v*.

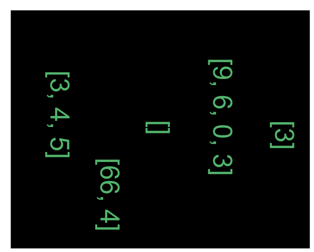


Vertex	Out-degree	In-degree
v_1	1	0
v_2	1	2
v_3	1	1
v_4	2	1
v_5	1	2

Graph Representations

- Two most common and popular representations for graphs as a data structure
  - ***Adjacency matrix***
  - ***Adjacency list***

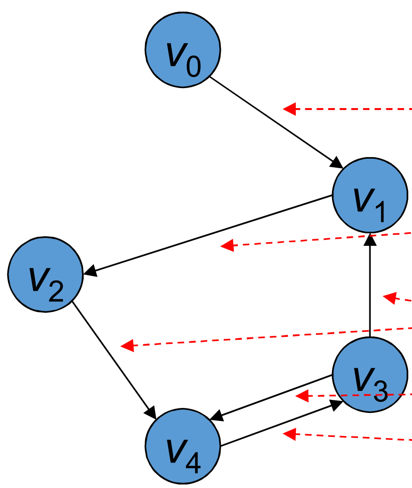




Adjacency Matrix : Unweighted Graph

- Suppose a graph has n vertices v_0, …, v_{n-1}.
- The ***adjacency matrix*** of an *unweighted* graph is a two-dimensional $n \times n$ array M such that
  - M[i][j] = 1 if the graph contains the edge (v_i, v_j);
  - M[i][j] = 0 otherwise.

Adjacency Matrix: Example (Directed Graph)

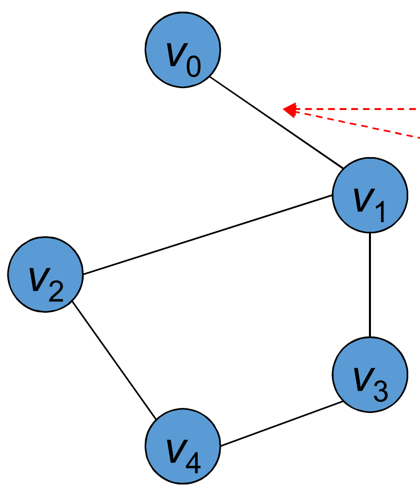


i \ j	0	1	2	3	4
0	0	1	0	0	0
1	0	0	1	0	0
2	0	0	0	0	1
3	0	1	0	0	1
4	0	0	0	1	0

Directed unweighted graph

Adjacency matrix M

Adjacency Matrix: Example (Undirected Graph)



Undirected unweighted graph

i \ j	0	1	2	3	4
0	0	1	0	0	0
1	1	0	1	1	0
2	0	1	0	0	1
3	0	1	0	0	1
4	0	0	1	1	0

Adjacency matrix M

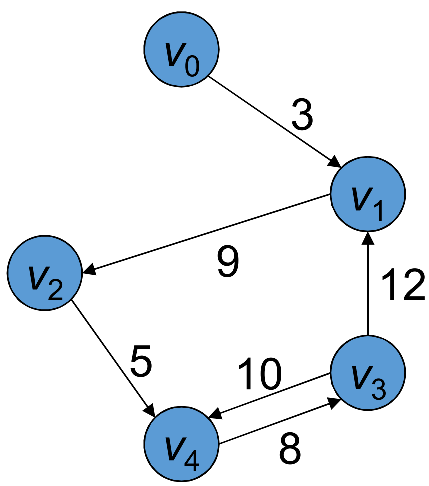
symmetrical

Adjacency Matrix: Weighted Graph

- In a *weighted* graph, we can store the weights in the adjacency matrix.
  - M[i][j] = *weight*(v_i, v_j) if the graph contains the edge (v_i, v_j);
  - M[i][j] = ∞ otherwise.

    Other specific values may be used to denote non-existent edges, such as 0 or $-\infty$.

Adjacency Matrix: Example (Directed Graph)

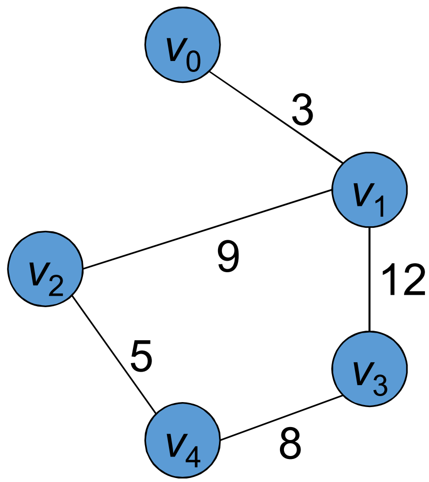


Directed weighted graph

i \ j	0	1	2	3	4
0	∞	3	∞	∞	∞
1	∞	∞	9	∞	∞
2	∞	∞	∞	∞	5
3	∞	12	∞	∞	10
4	∞	∞	∞	8	∞

Adjacency matrix M

Adjacency Matrix: Example (Undirected Graph)



Undirected weighted graph

i \ j	0	1	2	3	4
0	∞	3	∞	∞	∞
1	3	∞	9	12	∞
2	∞	9	∞	∞	5
3	∞	12	∞	∞	8
4	∞	∞	5	8	∞

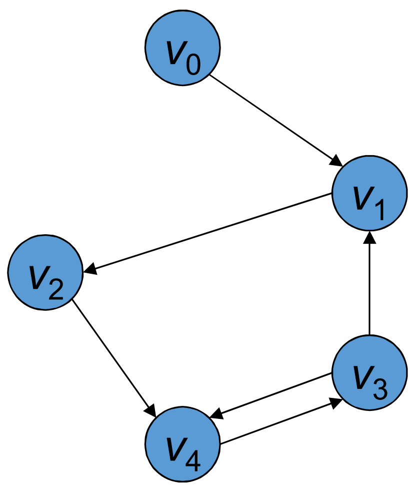
Adjacency matrix M

symmetrical

Adjacency List: Unweighted Graph

- Suppose a graph has n vertices v_0, …, v_{n-1}.
- The ***adjacency list*** of an *unweighted* graph is an array L of n lists such that
  - the elements of list L[i] contain the vertices that are adjacent to v_i.

Adjacency List: Example (Directed Graph)

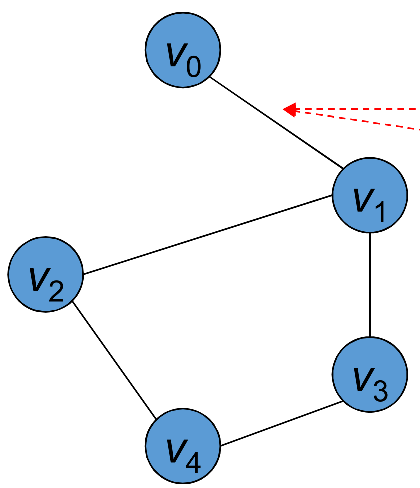


Directed unweighted graph

i	L[i]	
0	[1]	Denotes (v_0, v_1)
1	[2]	Denotes (v_1, v_2)
2	[4]	Denotes (v_2, v_4)
3	[1, 4]	Denotes (v_3, v_1) and (v_3, v_4)
4	[3]	Denotes (v_4, v_3)

Adjacency list L

Adjacency List: Example (Undirected Graph)



Undirected unweighted graph

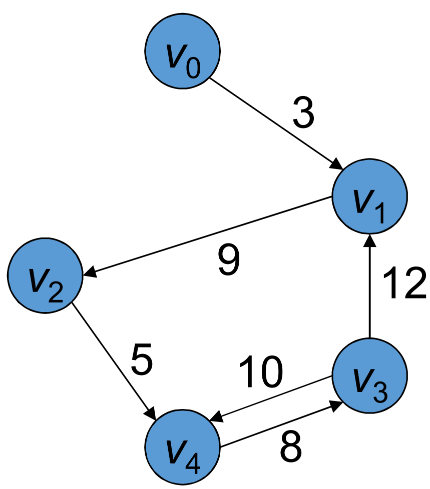
i	L[i]
0	[1]
1	[0, 2, 3]
2	[1, 4]
3	[1, 4]
4	[2, 3]

Adjacency list L

Adjacency List: Weighted Graph

- In a *weighted* graph, the elements of list L[i] contains the vertices v_j that are adjacent to v_i *as well as* the weights $weight(v_i, v_j)$.

Adjacency list for directed unweights graph	
i	L[i]
0	[1]
1	[2]
2	[4]
3	[1, 4]
4	[3]



Directed weighted graph

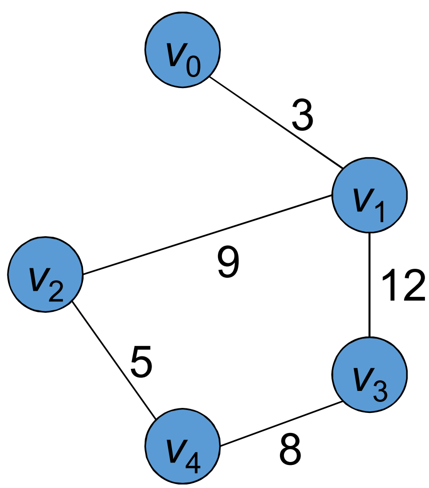
i	L[i]
0	[⟨1, 3⟩]
1	[⟨2, 9⟩]
2	[⟨4, 5⟩]
3	[⟨1, 12⟩, ⟨4, 10⟩]
4	[⟨3, 8⟩]

vertex

weight

Adjacency list L

Adjacency List: Example (Undirected Graph)



Undirected weighted graph

i	L[i]
0	[⟨1, 3⟩]
1	[⟨0, 3⟩, ⟨2, 9⟩, ⟨3, 12⟩]
2	[⟨1, 9⟩, ⟨4, 5⟩]
3	[⟨1, 12⟩, ⟨4, 8⟩]
4	[⟨2, 5⟩, ⟨3, 8⟩]

Adjacency list L

Adjacency Matrix vs Adjacency List

n: number of vertices
e: number of edges
d: degree/out-degree of vertex

- Space complexities of different operations

Adjacency matrix	Adjacency list
M[i][j], i, j = 0, …, n-1 $O(n^2)$	L[i], i = 0, …n-1 $O(n+e)$

Number of lists and total number of nodes

Adjacency Matrix vs Adjacency List

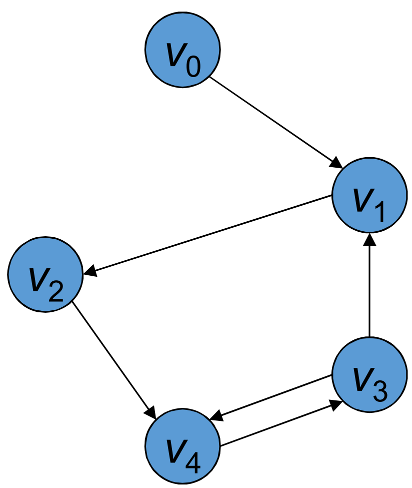
n: number of vertices
e: number of edges
d: degree/out-degree of vertex

- Time complexities of different operations

	Adjacency matrix	Adjacency list
Is there an edge from v_i to v_j?	M[i][j] == 1? $O(1)$	Traverse L[i] $O(d)$
Find all vertices adjacent to v_i.	Traverse row, M[i][?] == 1? $O(n)$	Traverse L[i] $O(d)$
How many edges are there in a graph?	Traverse M, M[?][?] == 1 ? $O(n^2)$	Traverse L $O(n + e)$

Number of lists and total number of nodes

Another Graph Representation: Edge List



Directed unweighted graph

(0,1),
(1,2),
(2,4),
(3,1),
(3,4),
(4,3)

Edge list L

Adjacency list	
i	L[i]
0	[1]
1	[2]
2	[4]
3	[1, 4]
4	[3]

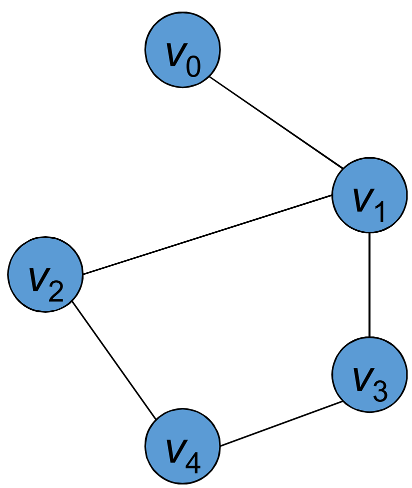
- Edge List
  - One list with elements denoting the edges

[(0,1), (1,2), (2,4), (3,1), (3,4), (4,3)]

- Adjacency List
  - n lists with elements denoting the adjacent nodes

L[0]	1
L[1]	2
L[2]	4
L[3]	1, 4
L[4]	3

Another Graph Representation: Edge List

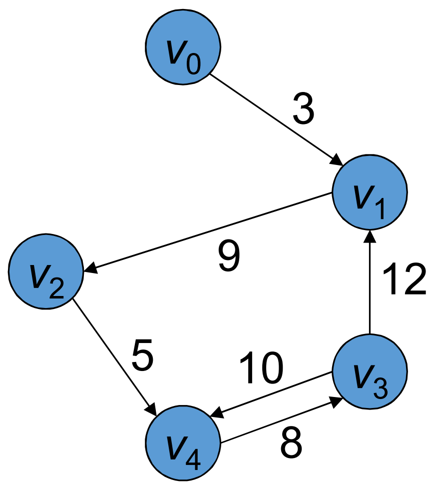


Undirected unweighted graph

(0,1),
(1,2),
(1,3),
(3,4),
(2,4)

Edge list L

Another Graph Representation: Edge List

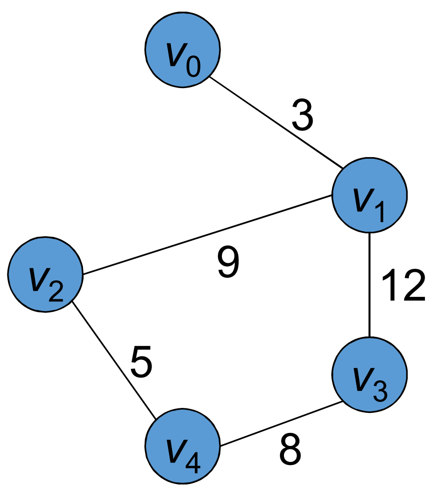


Directed weighted graph

(0,1,3), weight
(1,2,9),
(2,4,5),
(3,1,12),
(3,4,10),
(4,3,8)

Edge list L

Another Graph Representation: Edge List



Undirected weighted graph

(0,1,3),
(1,2,9),
(2,4,5),
(3,1,12),
(4,3,8)

Edge list L