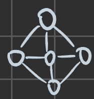
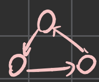
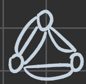
$\boxed{\text{Graph}}$

⇒ 數學定义：G(V,E) { V 頂點 / E 邊 ，且可以有cycle，邊可以是有向/無向

組成元件 {
1. node / node with label (○/ⓐ)
2. Edge / Edge with weight (权重) / Edge with direction / Edge with direction and weight / Edge with label (— / $\overset{8}{—}$ / → / $\overset{8}{\rightarrow}$ / $\overset{e2}{\rightarrow}$)

結構分類 {
1. Connect Graph ⇒ 所有點都連通 
2. Tree ⇒ 特殊連通且無環
3. Directed Graph (Digraph) ⇒ 邊有方向性 
4. Multigraph ⇒ 允許自环/多重边  (自环)  (多重边)

屬性分類 {
1. 同構 isomorphism) ⇒ 形狀看起來不同，結構上完全相同(关系同)的图形
2. 分度 Degree ⇒ 連接該頂點邊的數量 { In-degree 入度 ⇒ 指向該點的边數 / Out-degree 出度 ⇒ 該點指出的边數

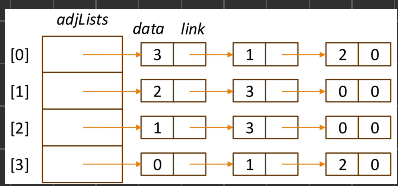
图形表示法 {
1. 鄰接矩陣 Adjacency Matrix

	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0

优 查詢边是否存在很快 ⇒ O(1)

缺 空間複雜度高 O(V²)，对于稀疏图(边很少的图)很浪費空間

2. 鄰接串列 Adjacency List



优 節省空間 O(V+E)，適合稀疏图，遍历效率高

缺 查詢边是否存在較慢 O(deg(v))

ADT ⇒ Create() 建立空图 / InsertVertex(v) 新增頂點 / InsertEdge(v1,v2) 新增边 / DeleteVertex / DeleteEdge 刪除頂點边 / Adjacent(v) 列出V的所有鄰居.

搜尋 {
Graph的traversal需要一個 visited[] 陣列來記錄已訪問過的點
1. BFS 廣度优先 ⇒ 使用 Queue → 按照 Degree 层级一层层向外扩散
2. DFS 深度优先 ⇒ 使用 Stack → 一條路走到底，遇到死路再回溯