## 281. Zigzag Iterator

Medium 581 30 Add to List Share

Given two vectors of integers `v1` and `v2`, implement an iterator to return their elements alternately.

Implement the `ZigzagIterator` class:

- `ZigzagIterator(List<int> v1, List<int> v2)` initializes the object with the two vectors `v1` and `v2`.
- `boolean hasNext()` returns `true` if the iterator still has elements, and `false` otherwise.
- `int next()` returns the current element of the iterator and moves the iterator to the next element.

**Example 1:**

```
Input: v1 = [1,2], v2 = [3,4,5,6]
Output: [1,3,2,4,5,6]
Explanation: By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1,3,2,4,5,6].
```

**Example 2:**

```
Input: v1 = [1], v2 = []
Output: [1]
```

**Example 3:**

```
Input: v1 = [], v2 = [1]
Output: [1]
```

**Constraints:**

- `0 <= v1.length, v2.length <= 1000`
- `1 <= v1.length + v2.length <= 2000`
- $-2^{31} <= v1[i], v2[i] <= 2^{31} - 1$

**Follow up:** What if you are given `k` vectors? How well can your code be extended to such cases?

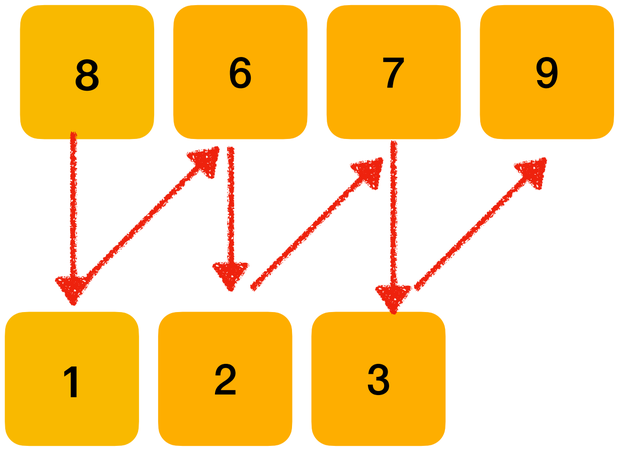
**Clarification for the follow-up question:**

The "Zigzag" order is not clearly defined and is ambiguous for `k > 2` cases. If "Zigzag" does not look right to you, replace "Zigzag" with "Cyclic".

**Follow-up Example:**

```
Input: v1 = [1,2,3], v2 = [4,5,6,7], v3 = [8,9]
Output: [1,4,8,2,5,9,3,6,7]
```

# Zigzag Iterator

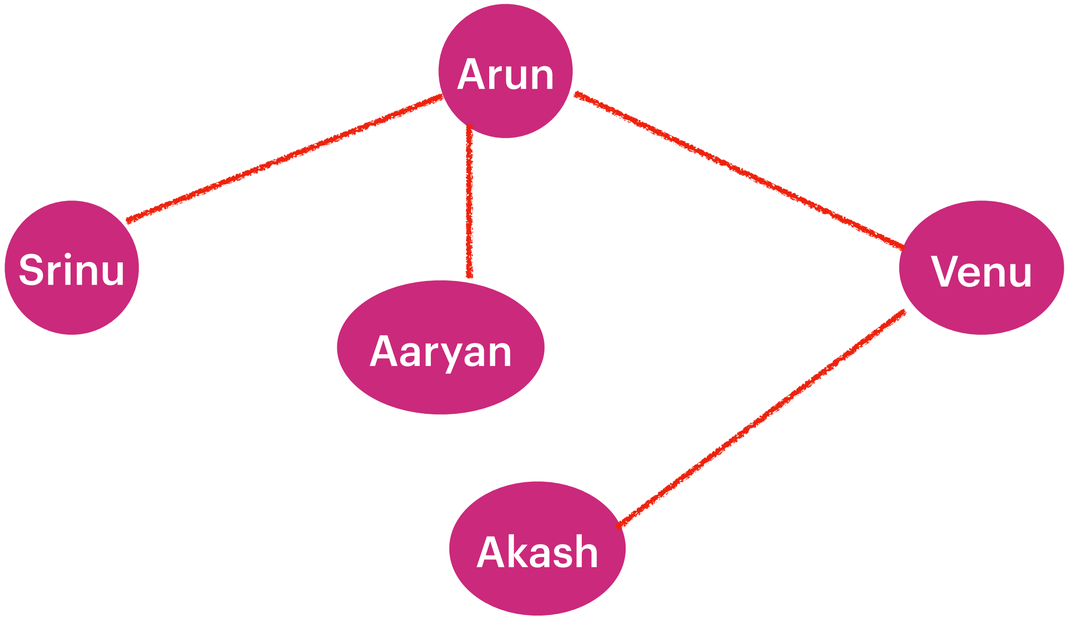


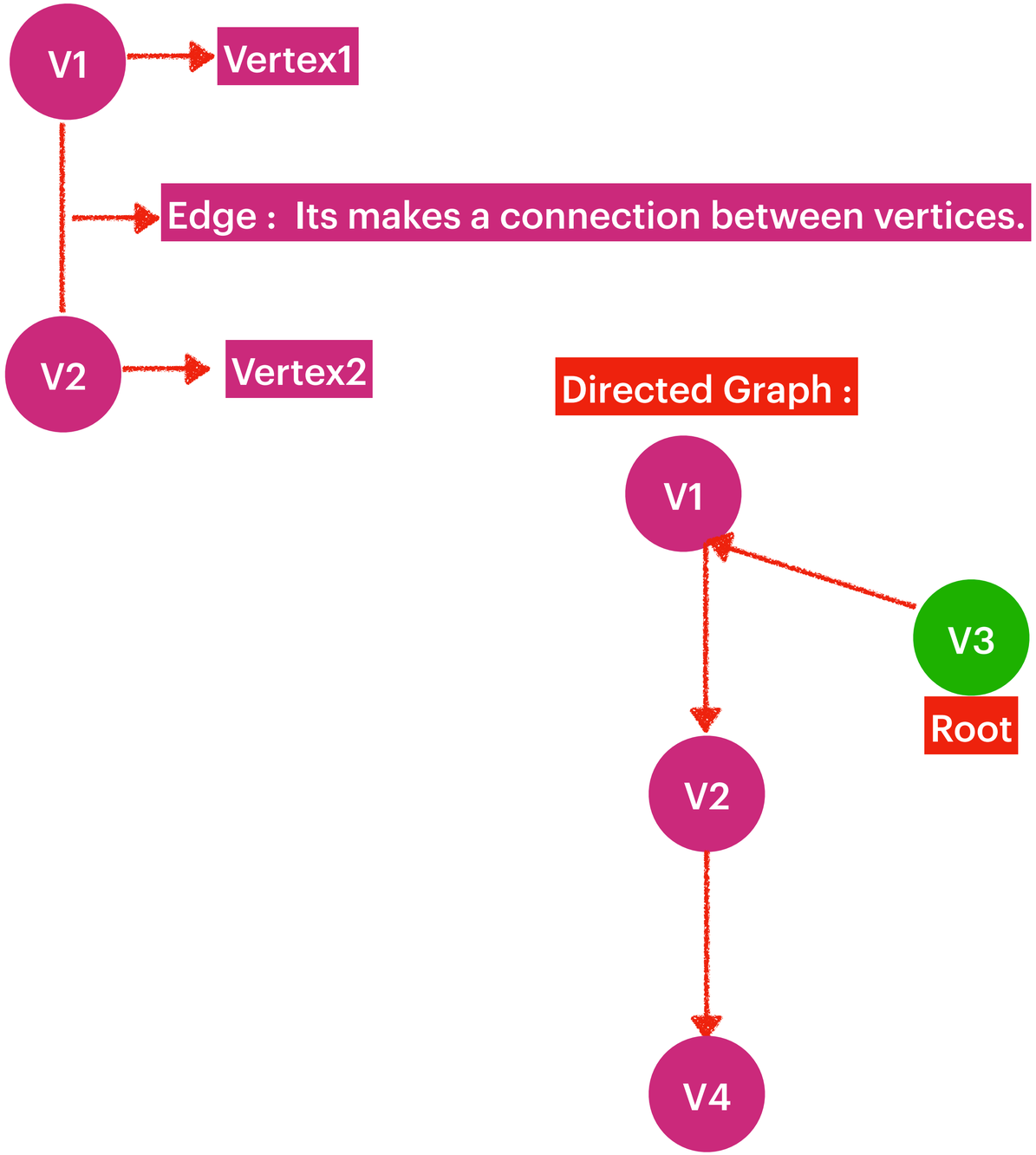

Output : {8,1,6,2,7,3,9}

# Graph <——> Terminology

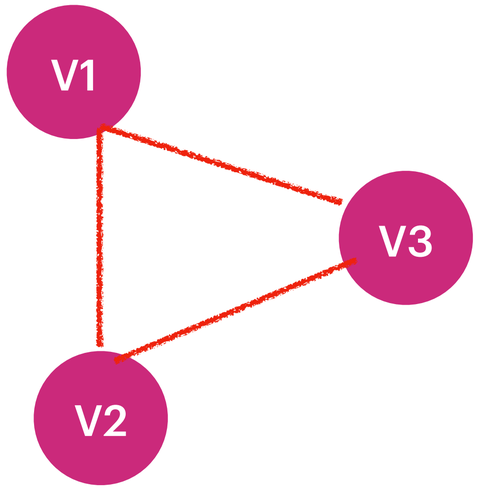
Graph is a data structure, which does not follow any structure .

Example : LinkedIn Connections

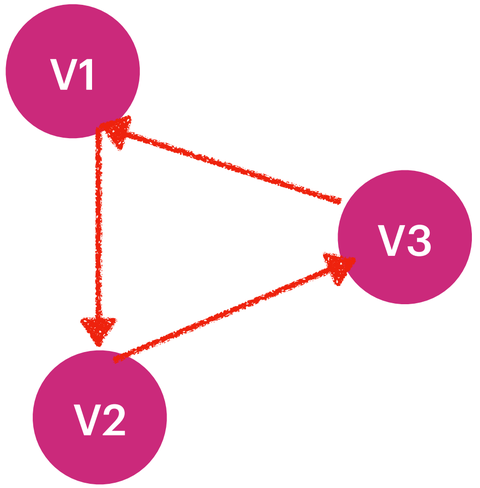



V1
Vertex1
Edge : Its makes a connection between vertices.
V2
Vertex2
Directed Graph :
V1
V3
Root
V2
V4
Undirected Graph :
V1
V3
V2
Here any vertex can be root
V4

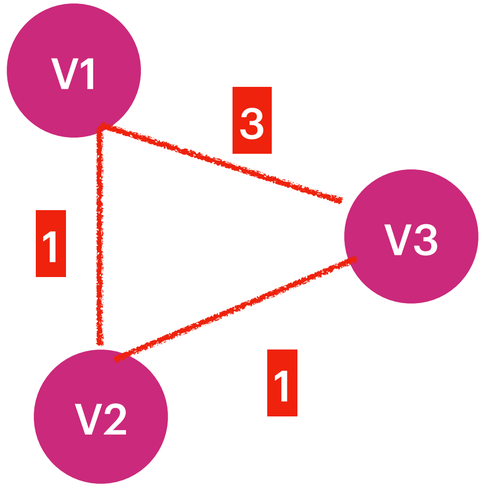
Cyclic Graph :




Cyclic Graph :

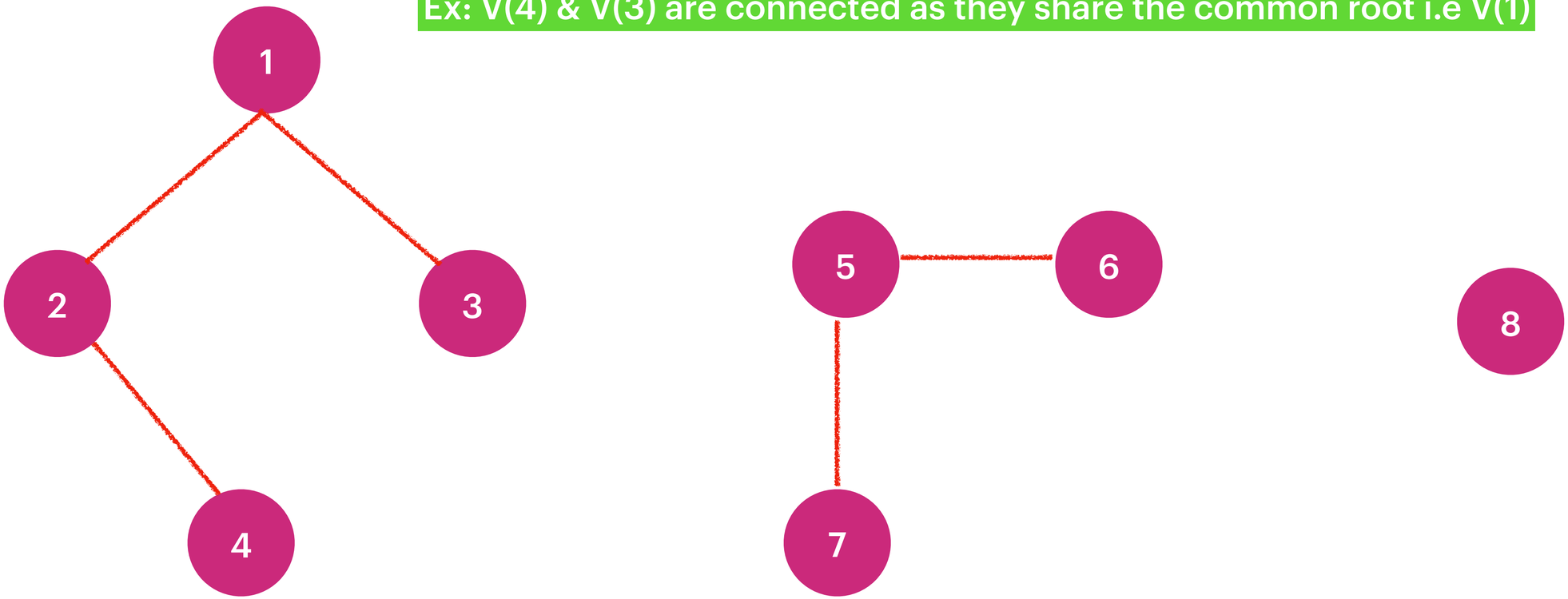



Weighted Graph



Every Vertex, Can act as a root to itself.

If two vertices are connected means , they shares the common root.
Ex: V(4) & V(3) are connected as they share the common root i.e V(1)




isConnect(1 & 4) —> true

isConnect(1 & 7) —> False

isConnect(4 & 3) —> true