**18(a).**

Given an undirected graph **G**=**(V,E)** in which each vertex has a label **L(v)** where the label is some integer from **{0,1,2,... 9}**. Suppose **s** and **t** are two given vertices. We want to design an algorithm that determines if there is a path from s to t whose labels follow the pattern **12312312312**.... (it need not end at 3). We call this the **RLV(G,s,t)** problem.

Reduce **G** to another directed graph **P**

**● How to construct the graph P?**

1. **What do vertices represent? How many vertices are there?**

Reduce the vertex set from vertices labeled **{0,1,2,... 9}** to vertices labeled **{1,2,3}** only.

Delete all vertices having labels other than **{1,2,3}.** Also delete the corresponding edges of the deleted vertices.

Hence, in **P,** each vertex represents a node in the path from **s** to **t**, if at all such a path exists which follows the above pattern.

Number of vertices in **P =** **|V| (i.e.** same as the number of vertices in **G,** asin the worst case **G** might contain vertices with labels **{1,2,3}** only.**)**

1. **What do edges represent (when would there be an edge from u -> v) ?**

Traverse the graph **G** & say there is an edge from **u** to **v**; then following 3 cases can occur:

mark a directed edge from **u** to **v** in **P** only in the following 3 cases:

* When **u** is of label **1** & **v** is of label **2 :** mark edge from **u** to **v.**
* When **u** is of label **2** & **v** is of label **3 :** mark edge from **u** to **v.**
* When **u** is of label **3** & **v** is of label **1 :** mark edge from **v** to **u.**

All remaining edges in **G** will be removed in **P**.

Hence, a path between 2 nodes in **P** will represent a pattern starting from **1** & the following sub-pattern would be **23123….** (it need not end at 3).

1. **How many edges are there?**

Number of edges in **P =** **|E| (i.e.** same as the number of edges in **G,** asin the worst case **G** might contain vertices with labels **{1,2,3}** only.**)**

1. **What is the time necessary to construct the graph P (in terms of the problem input size)?**

The time required to construct **P** equals the time required to traverse all vertices and edges of **G**, which is **O(V+E),** if we use any standard graph traversal algorithm i.e. **DFS.**

1. **What problem should be solved on the graph?**

The problem to be solved with the objective of finding a path from **s** to **t** in **P i.e** is **t** reachable from **s.**

Since, we have assumed the starting node **s** to be a vertex of label {1} only, & moreover the edges are only of the form 1->2, 2->3 or 3->1, hence a path to **t** ,if exists will always follow the given constraint pattern.

**What algorithm should be used?**

Standard **Depth First Search** can be used on root vertex **[s]** as the starting node.

If the node **[t]** is visited, then the search returns TRUE, else FALSE.

1. **Complexity analysis:**
2. **What is the complexity of the above algorithms in terms of P ?**

Constructing **P** takes **O( V+E )** time, as we need to traverse **G** in its entirety.

1. **What is the complexity of the overall problem in terms of G ?**

Total time complexity, **T(n)** = time complexity to construct the graph **P**

+

time taken to find the path/sequence from **[s]** to node **[t]**

**T(n)** = **O(V+E)** + **O(V+E)** **=> T(n)=O(V+E)**

**Space Complexity = O(|V|) as well - since in the worst case you need to hold all vertices in the stack.**

**18(b).**

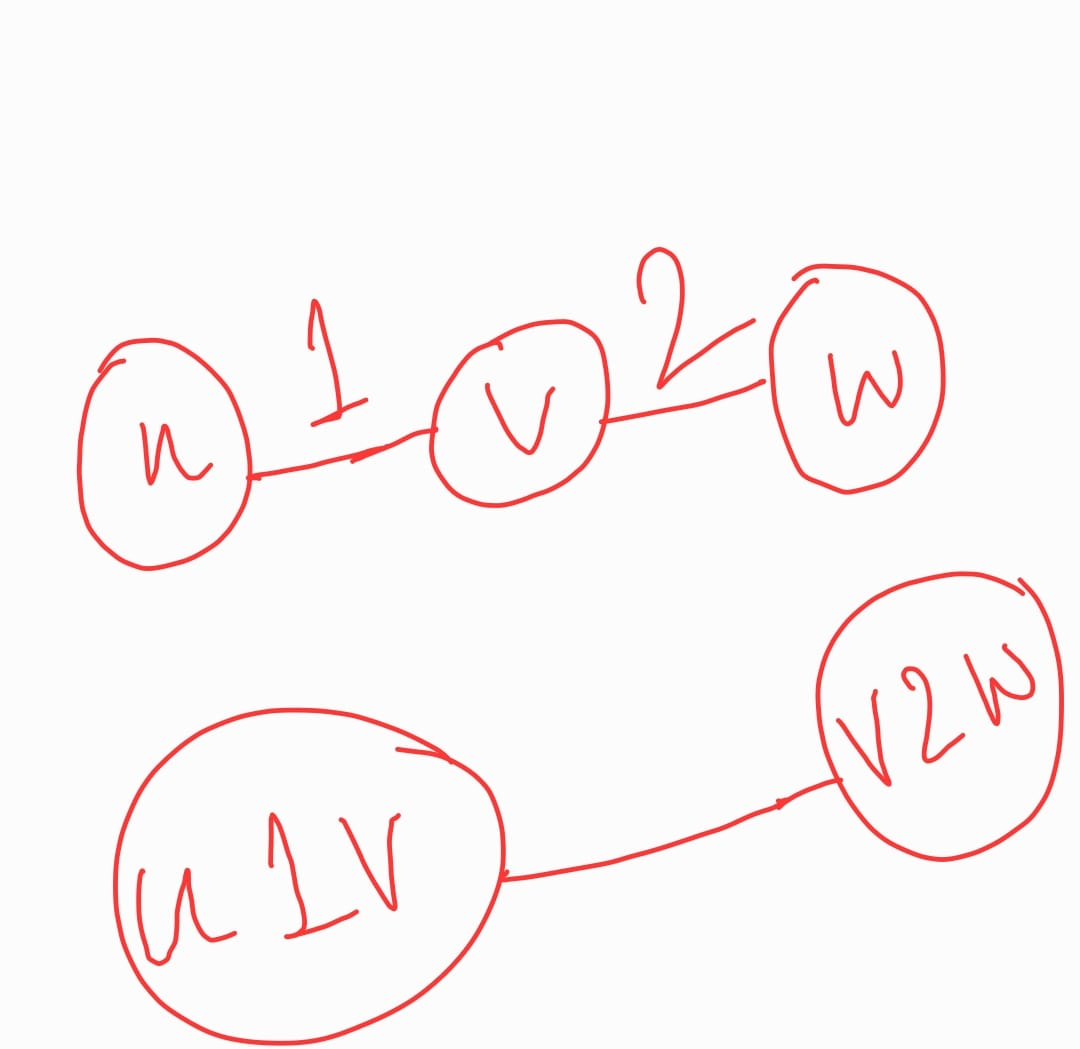
Given an undirected graph **G**=**(V,E)** in which every edge is labeled, similarly as before. Each edge has a label **L(v)** where the label is some integer from **{0,1,2,... 9}**. Suppose **s** and **t** are two given vertices. We want to design an algorithm that determines if there is a path from s to t whose edges follow the pattern **12312312312**.... (it need not end at 3). We call this the **RLE(G,s,t)** problem.

Reduce **G** to another undirected graph **P** & then input it to above **RLV** which will find out if there exists a path satisfying the constraint.

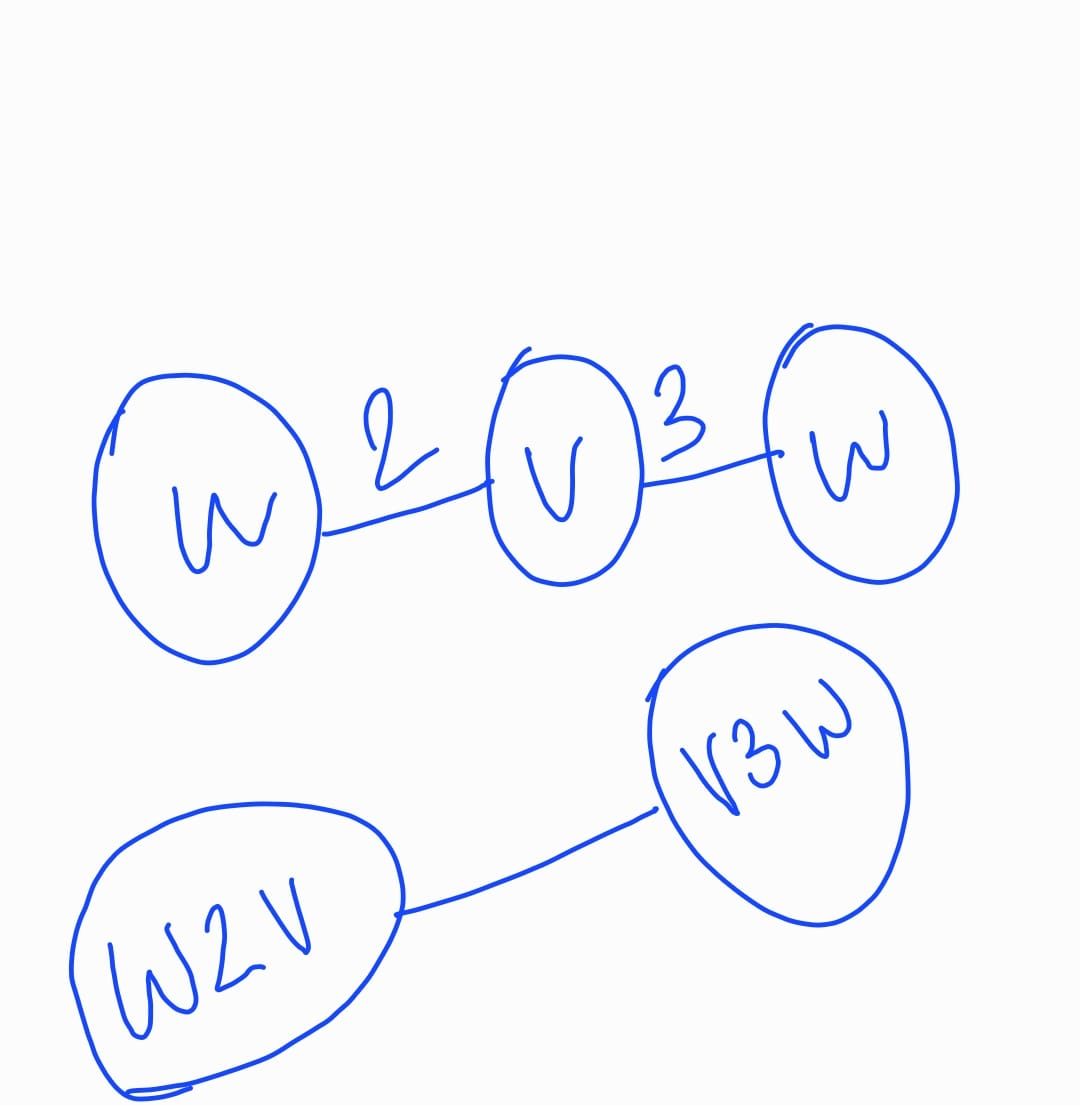
**● How to construct the graph P?**

1. **What do vertices represent? How many vertices are there?**

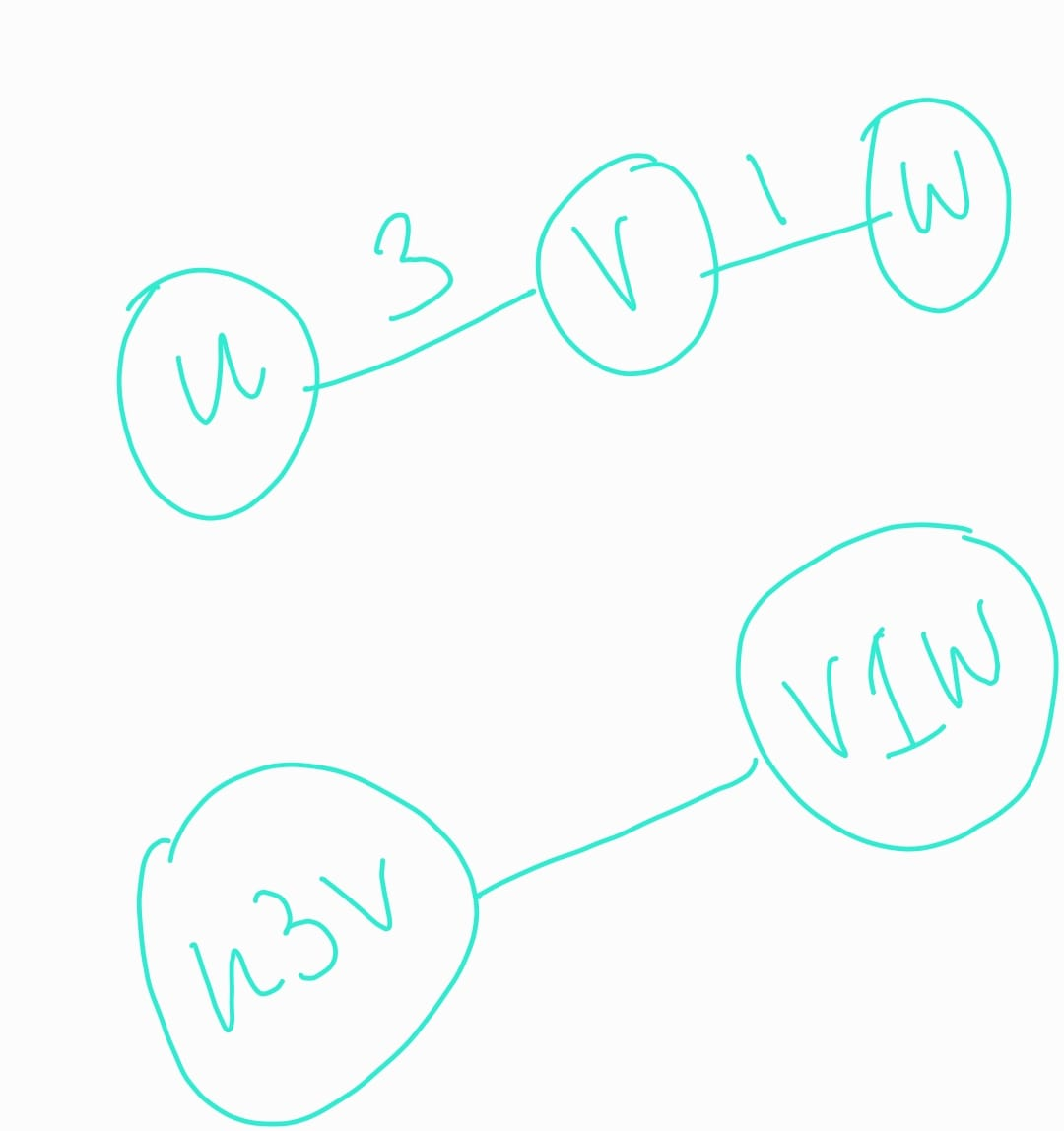
For every 2 edges **u--(1)--v** & **v--(2)--w** in **G,**  we put 2 vertices **(u1v) & (v2w)** in **P** & put an edge between them.



Similarly, for every 2 edges **u--(2)--v** & **v--(3)--w** in **G,**  we put 2 vertices **(u2v) & (v3w)** in **P** & put an edge between them.



Similarly, for every 2 edges **u--(3)--v** & **v--(1)--w** in **G,**  we put 2 vertices **(u3v) & (v1w)** in **P** & put an edge between them.



Hence, in **P** each vertex represents an edge in the path from **s** to **t**  having the label either of **{1,2,3},** such that there is another edge after it which follows the given pattern. ***[Ref. above images]***

Number of vertices in **P = |V’| =** **2\*|E| (i.e.** same as the number of edges in **G,** asin the worst case **G** might contain all edges with labels **{1,2,3}** only.**)**

Each vertex will be counted twice, as the undirected edges are considered twice for both directions.

1. **What do edges represent (when would there be an edge from u -> v) ?**

Traverse the graph **G** & say there are 2 edges from **(u to v) & (v to w)**; then following 3 cases can occur:

mark an edge from (**u(i)v** to **v(j)w)** in **P [ (i,j) ∈ {1,2,3} ]** only in the following 3 cases :

* When **u--(1)--v** & **v--(2)--w :** mark edge between **(u1v) & (v2w).**
* When **u--(2)--v** & **v--(3)--w :** mark edge between **(u2v) & (v3w)**.
* When **u--(3)--v** & **v--(1)--w :** mark edge between **(u3v) & (v1w)**.

All remaining edges in **G** will be removed in **P**.

Hence, a path between 2 nodes in **P** will represent a pattern of edges in **G** starting from **1** & the following sub-pattern would be **23123….** (it need not end at 3).

1. **How many edges are there?**

Number of edges in **P = |E’| =** |**V**|**C3 ;** which is **O(V3)**, as in the worst case **G** might contain all edges with labels **{1,2,3}** only, all possible edges might be relevant for **P**.

1. **What is the time necessary to construct the graph P (in terms of the problem input size)?**

The time required to construct **P** equals the time required to traverse all vertices and edges of **G**, which is **O(V+E),** if we use any standard graph traversal algorithm i.e. **DFS.**

1. **What problem should be solved on the graph?**

The problem to be solved with the objective of finding a path from **if** (**\_ \_ t)** is reachable from (**s1u),** where **u,v** are some edges in **G & s,t** are starting & ending nodes in **G.**

(**\_ \_ t) is a node in P such that there exists an edge in G (labeled 1/2/3) between any other node to t in G.**

**Similarly,** (s **1 \_) is a node in P such that there exists an edge in G (labeled 1 only) between s to any other node G.**

Since, we have assumed the starting node **s** to be a vertex of label {1} only, & moreover the edges in **P** are only of the form 1->2, 2->3 or 3->1, hence a path to **t** ,if exists will always follow the given constraint pattern.

Also, we need to run the **RLV** algorithm **[**as defined above in part **(a).],** on all **(s1x)** nodes in **P** corresponding to **s** in **G,** as there might be multiple edges in **G** labeled **1** originating from **s**.

Root vertex **[s 1 \_]** is the starting node.

If the node **[\_ \_ t]** is visited, then the search returns TRUE, else FALSE.

1. **Complexity analysis:**
2. **What is the complexity of the above algorithms in terms of P ?**

Constructing **P** takes **O( V3+E )** time, as we need to mark **V3** edges in **P**.

1. **What is the complexity of the overall problem in terms of G ?**

Total time complexity, **T(n)** = time complexity to construct the graph **P**

+

time taken to run **RLV** from **[s 1 \_]** to node **[\_ \_t]**

**T(n)** = **O(V3+E)** + **O(|V’|+|E’|) =O(V3+E) + O(V3+E)** **=> T(n)=O(V3)**

**Space Complexity = O(|V’|) as well - since in the worst case you need to hold all vertices in the stack. = Space Complexity = O(|V3|)**