* **Problem :** [**https://cses.fi/problemset/task/1672/**](https://cses.fi/problemset/task/1672/)
* **Approach: DP approach**

**-> (**works with **negative edges and also with negative weight cycles** with a bit of modification)

-> (Works for both **Directed and Undirected** Graphs)

-> (Works for Graphs with multiple connected components(**disconnected graphs** also),as distance of a node of 1 connected component to a node in another component will anyways be INFINITY,so it won’t affect any distance in another component)

-> First we take 1 as intermediate node,so lets say we are checking 2->3.

So we enter min(2->3,2->1->3) in matrix,now off course the row and column of 1 remains unaffected, as e.g if we calculate 1->2 ,then 1->1->2 and 1->2 are same as we assume there are no self loops.

-> Let’s say we completed preparing matrix with 1 as Intermediate Node

Now lets take 2 as Intermediate

When we are calculating 3->4, We check min(3->4,3->2->4)

So this also includes 3->1->4, 3->1->2->4, 3->2->1->4, 3->1->2->1->4(if neg cycles allowed) (bcoz in previous iteration all distances with Intermediate node as “1” would have been covered) so basically it covers all possibilities.

* **FOR NEG- CYCLES:**

-> **Distance of any node from itself is always zero. But if there’s a Negative Weight Cycle , then distance of a node to itself,eventually becomes negative,so we detect a cycle.**

* **TIME COMPLEXITY:**

**-> So we process almost all values in matrix,that is V\*V values with each of V nodes as Intermediate nodes one by one,so Time=O(V^3)**

* **Code :** [**https://ideone.com/w5Br3K**](https://ideone.com/w5Br3K)

**Edge Case** :

Can be 2 or more roads with different costs from ‘u’ to ‘v’.

So store minimum of all.

Eg test case :

2 2 1

**1 2 1**

**2 1 2**

1 2