Write a Union-Find data structure with path compression. Use this data structure to detect a cycle in an undirected graph.

```
public class DisJointUnionSetDemo {

	int parent[];
	int n;
	int rank[];
	public DisJointUnionSetDemo(int n) {
		rank=new int[n];
		parent=new int[n];
		this.n=n;
		makeSet();
	}//end of constructor

	void makeSet() {
		for(int i=0;i<n;i++) {
			parent[i]=i;
		}//for
	}//makeset

	int find(int x) {
		while(parent[x]!=x) {
			x=parent[x];
		}//if
		return parent[x];
	}

	void union(int x,int y) {
		int xRoot=find(x),yRoot=find(y);
		if(xRoot==yRoot)
```

```
				return;
			if(parent[xRoot]<parent[yRoot])
				parent[xRoot]=yRoot;
			else if(parent[yRoot]<parent[xRoot])
				parent[yRoot]=xRoot;
			else {
				parent[yRoot]=xRoot;
				rank[xRoot]=rank[xRoot]+1;
			}//else
		}//union

		public static void main(String[] args) {

			int n=5;
			DisJointUnionSetDemo obj=new DisJointUnionSetDemo(n);
			obj.union(0, 2);
			obj.union(4, 2);
			obj.union(4, 1);

			if(obj.find(4)==obj.find(0)) {
				System.out.println("Yes");
			}
			else
				System.out.println("No");
			if(obj.find(1)==obj.find(0))
				System.out.println("Yes");
			else
				System.out.println("No");
		}//end of main

	}//end of class
```