```java
package edu.caltech.cs2.datastructures;

import edu.caltech.cs2.interfaces.IDictionary;
import edu.caltech.cs2.interfaces.IGraph;
import edu.caltech.cs2.interfaces.ISet;

public class Graph<V, E> implements IGraph<V, E> {
    private IDictionary<V, IDictionary<V,E>> graphDict = new ChainingHashDictionary<>(MoveToFrontDictionary :: new);

    @Override
    public boolean addVertex(V vertex) {
        if (graphDict.containsKey(vertex)){
            return false;
        }
        graphDict.put(vertex, new ChainingHashDictionary<>(MoveToFrontDictionary::new));
        return true;
    }

    @Override
    public boolean addEdge(V src, V dest, E e) {
        if (!graphDict.containsKey(src) || !graphDict.containsKey(dest)){
            throw new IllegalArgumentException();
        }
        if (graphDict.get(src).get(dest) != null){
            graphDict.get(src).put(dest, e);
            return false;
        }
        graphDict.get(src).put(dest, e);
        return true;
    }

    @Override
    public boolean addUndirectedEdge(V n1, V n2, E e) {
        if (!graphDict.containsKey(n1) || !graphDict.containsKey(n2)){
            throw new IllegalArgumentException();
        }
        boolean b1 = addEdge(n1, n2, e);
        boolean b2 = addEdge(n2, n1, e);
        return b1 && b2;
    }

    @Override
    public boolean removeEdge(V src, V dest) {
        if (!graphDict.containsKey(src) || !graphDict.containsKey(dest)){
            throw new IllegalArgumentException();
        }
        if (graphDict.get(src).get(dest) == null){
            return false;
        }
        graphDict.get(src).remove(dest);
        return true;
    }

    @Override
    public ISet<V> vertices() {
        return graphDict.keySet();
    }

    @Override
    public E adjacent(V i, V j) {
        if (!graphDict.containsKey(i) || !graphDict.containsKey(j)) {
            throw new IllegalArgumentException();
        }
        if (graphDict.get(i).get(j) != null){
            return graphDict.get(i).get(j);
        }
        return null;
    }

    @Override
    public ISet<V> neighbors(V vertex) {
        if (!graphDict.containsKey(vertex)){
            throw new IllegalArgumentException();
        }
        return graphDict.get(vertex).keySet();
    }
```

```
}
```