Isomorphism of Graphs using Ordered Adjacency List

Dhananjay P. Mehendale
Sir Parashurambhau College, Tilak Road, Pune-411030,
India

Abstract

In this paper we develop a novel characterization for isomorphism of graphs. The characterization is obtained in terms of ordered adjacency lists to be associated with two given labeled graphs. We show that the two given labeled graphs are isomorphic if and only if their associated ordered adjacency lists can be made identical by applying the action of suitable transpositions on any one of these lists. We discuss in brief the complexity of the algorithm for deciding isomorphism of graphs and show that it is of the order of the cube of number of the number of edges.

1. **Introduction:** The two given graphs *G* and *H* are isomorphic when an adjacency preserving bijection exists between their vertices. To determine whether two given graphs are isomorphic is called the Graph Isomorphism Problem (GI). GI is of great interest to computer scientists and researchers in other fields such as chemistry, switching theory, information retrieval, and linguistics. In particular GI is of profound interest to complexity theorists because yet the graph isomorphism problem is neither proved P nor proved NP-complete.

   In this paper we develop an isomorphism criterion in terms of ordered adjacency lists that can be associated with given two labeled graphs to be tested for their isomorphism. We show that testing isomorphism of two given graphs is equivalent to checking whether there exist suitable transpositions to be applied in a sequence on the ordered adjacency list of any one graph to make it identical to the ordered adjacency list of the other graph.

2. **Graph Isomorphism (Characterization using transpositions and reordering of ordered adjacency list):** We now proceed to discuss our algorithm for Graph Isomorphism which in the worst case is of order $\sim O(e^3)$, where *e* stands for number of edges in the graph. In practice this algorithm works very efficiently and actually doesn't require the worst case complexity, namely, $\sim O(e^3)$ to produce the decision about the isomorphism of given two (*n*, *e*) graphs *G* and *H*. In this algorithm we may need to apply in the

worst case the action of e transpositions, and each such action of transposition further requires ordering of e edge labels using any standard algorithm of order $\sim O(e^2)$, thus, total order of our algorithm becomes $\sim O(e^3)$ in the worst case.

Let G be a labeled (p, q) graph whose vertices are labeled with labels $\{1, 2, 3, \ldots, p\}$. If there is an edge joining two vertex labels i and j say such that $i < j$ then we associate pair (i, j) as label with that edge.

Definition 2.1: The **ordered adjacency list** is the ordered list of edge labels presented in a row (or column) as follows: If vertex with label 1 is adjacent to vertices with labels $\{i_1, i_2, \cdots, i_{k_1}\}$ and $\{1 < i_1 < i_2 < \cdots < i_{k_1}\}$ then the first k_1 entries of adjacency list are $(1, i_1)(1, i_2)(1, i_3)\cdots(1, i_{k_1})$. If vertex with label 2 is adjacent to vertices with labels $\{j_1, j_2, \cdots, j_{k_2}\}$ and $\{2 < j_1 < j_2 < \cdots < j_{k_2}\}$ then the next k_2 entries of adjacency list are $(2, j_1)(2, j_2)(2, j_3)\cdots(2, j_{k_2})$. We then take in succession the vertices with the vertex labels 3, 4, …. and continue extending the list along same lines by the appending the next proper ordered pairs to the already constructed list till we reach the vertex with label (p-1) and finally by appending (or not appending) the vertex pair (p-1, p) when the vertex with label (p-1) is adjacent (nonadjacent) to vertex with label p we thus complete construction of ordered adjacency list for the graph.

It is well known to all that two graphs are isomorphic if and only if there exists one-one adjacency preserving map between their vertex sets, i.e. one graph can be transformed into other by relabeling the vertices of (any) one graph. More formally, let G and H be two (p, q) graphs and let A_G, A_H be the adjacency matrices associated with graphs G and H respectively then these graphs G and H are isomorphic if and only if there exists a permutation matrix P such that $A_G = PA_HP^{-1}$.

Now the problem with using this result to test two given graphs for isomorphism is actually as follows: If given graphs are isomorphic how to discover such permutation σ whose associated permutation

matrix, P say, satisfies $A_G = PA_H P^{-1}$? Also, when given graphs will not be isomorphic there will not exists any such permutation σ whose associated permutation matrix, P say, achieves $A_G = PA_H P^{-1}$. How to conclusively arrive at the result that such permutation won't exist when two given graphs under consideration are nonisomorphic?

The answer to these questions can be extracted from the simple fact that any permutation (and so also the desired permutation) is product (composition) of transpositions. We will see that the desired permutation builds up from composition of suitable transpositions to be applied in succession on the ordered adjacency list of any one of the two given graphs. When the two given graphs are isomorphic one can write down their associated ordered adjacency lists. One then choose (any) one list and choose proper transpositions to be applied on this list for making it identical with the other ordered adjacency list. When the two given graphs are not isomorphic then one cannot make the lists identical by whatever transpositions one will choose to apply on (any) one ordered adjacency list to make it identical with the other ordered adjacency list, i.e. when one chooses the next transposition for further identification of two ordered adjacency lists, which have been made partially identical by earlier transpositions, then the new transposition disturbs the already obtained partial identification of the ordered adjacency lists.

Action of transposition on given ordered adjacency list: Let L be an ordered adjacency list associated with some labeled (p, q) graph G and its vertices are labeled with numbers $\{1, 2, 3, \ldots, p\}$. Let us denote the transposition by $[i, j]$. The action of this transposition on ordered adjacency list, expressed as $[i, j](L)$, is a two step procedure:

(i) It changes everywhere in the ordered adjacency list L the symbol i by symbol j and vice versa.

(ii) After the above step it reorders the (modified) adjacency list so that again it becomes ordered in the sense of definition 4.1.

Theorem 2.1: Let G and H be two (p, q) graphs. $G \cong H$ if and only if there exists a permutation made up of composition of transpositions which when applied successively on ordered adjacency list of G say, converts it successfully into ordered adjacency list of H.

Proof: Suppose *G* is isomorphic to *H* then by definition there exists permutation σ whose associated permutation matrix, *P* say, satisfies $A_G = PA_HP^{-1}$. If we break the permutation σ into transpositions and will apply these transpositions successively on ordered adjacency list of *H* we will straightway get the ordered adjacency list of *G*.
Now, suppose *G* is not isomorphic to *H* then there will not exist permutation σ whose associated permutation matrix, *P* say, satisfies $A_G = PA_HP^{-1}$ so if any σ will be taken and will be broken into transpositions and applied successively on ordered adjacency list of *H* it will certainly fail to produce the ordered adjacency list of *G*.

□

Algorithm 2.1 (Isomorphism using ordered adjacency list):

1) Take labeled copies of given (*p*, *q*) graphs *G* and *H* labeled with labels {1, 2, …., *p*} and prepare ordered adjacency lists say L_G and L_H and select L_H for applying transpositions and to see whether we can equalize L_H ultimately with L_G by action of successive suitable transpositions.

2) Let the first elements in the ordered adjacency lists L_G and L_H be $(1, i_1)$ and $(1, j_1)$ respectively. If $i_1 = j_1$ then proceed to next elements in the ordered adjacency lists L_G and L_H, namely, $(1, i_2)$ and $(1, j_2)$ to check whether $i_2 = j_2$. Else, if $i_1 \neq j_1$ then carry out action of transposition on L_H that replaces i_1 by j_1 and vice versa everywhere in L_H and then order the changed L_H in the sense of definition 4.1. This new L_H now will stand for L_H for further actions of transpositions. We represent this (two stepped) action, carried out when $i_1 \neq j_1$, symbolically as $[i_1, j_1](L_H)$. Note that this step has equalized the first element in the ordered adjacency lists L_G and (new) L_H which will now stand for L_H.

3) Now, proceed to next elements in the ordered adjacency lists L_G and (new) L_H, namely, $(1, i_2)$ and $(1, j_2)$ and check whether $i_2 = j_2$. If

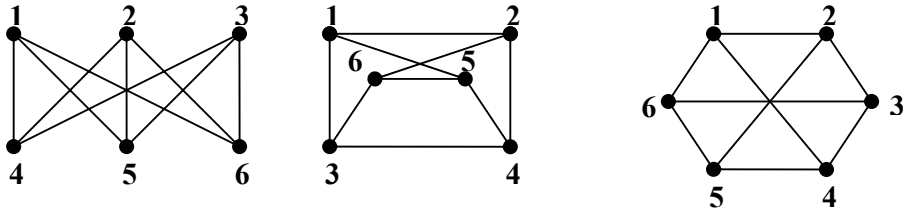
$i_2 = j_2$ then proceed to next elements in the ordered adjacency lists L_G and L_H, namely, $(1, i_3)$ and $(1, j_3)$ to check whether $i_3 = j_3$.. Else, if $i_2 \neq j_2$ then carry out action of transposition on L_H that replaces i_2 by j_2 and vice versa everywhere in L_H and then as previous order the changed L_H in the sense of definition 4.1. This new L_H now will stand for L_H for further actions of transpositions. We represent this (two stepped) action that carried out symbolically as $[i_2, j_2](L_H)$. Note that if action $[i_2, j_2](L_H)$ performed to equalize second elements in the ordered adjacency lists L_G and L_H does not disturbs the already achieved equality of first elements in the lists then up to this step one has equalized the first two elements in the ordered adjacency lists L_G and (new) L_H which will now stand for L_H. .

4) We continue in this way with equalizing next element in L_H with the element in L_G in the same place by the action of suitably chosen transpositions on (new) L_H which now stands for L_H till we can proceed along these lines without disturbing the equality of elements achieved earlier.

□

Remark 2.1: Note that when given (*p*, *q*) graphs *G* and *H* are isomorphic we should be able to achieve equality of L_G and L_H and when we are unable to achieve equality of L_G and L_H by algorithm 4.3 then *G* and *H* are not isomorphic.

Example: We now proceed to discuss one example to apply this algorithm. The graphs in this example are isomorphic. A similar example can be considered for the case of nonisomorphic graphs. There one will see that earlier obtained partial equality of ordered adjacency lists cannot be maintained in attempt of extending the equality further. Consider following labeled graphs *G*, *H*, *K* given below:



G ***H*** ***K***

We now proceed to record the ordered adjacency lists L_G, L_H, and L_K. We then carry out action of suitable transpositions, firstly on L_H and then on L_K, and show that we can identify L_H, L_K with L_G. This shows that graphs *G*, *H*, *K* are isomorphic.

$$L_G = (1,4)(1,5)(1,6)(2,4)(2,5)(2,6)(3,4)(3,5)(3,6)$$

$$L_H = (1,2)(1,3)(1,5)(2,4)(2,6)(3,4)(3,6)(4,5)(5,6)$$

$$L_K = (1,2)(1,4)(1,6)(2,3)(2,5)(3,4)(3,6)(4,5)(5,6)$$

We first check whether L_H can be made identical to L_G by the action of properly chosen transpositions: Consider

$$[2,4](L_H) = (1,3)(1,4)(1,5)(2,3)(2,4)(2,5)(3,6)(4,6)(5,6) = L_H^1$$

Here we now set $L_H^1 = L_H$. Proceeding with this (new) L_H we do

$$[3,6](L_H) = (1,4)(1,5)(1,6)(2,4)(2,5)(2,6)(3,4)(3,5)(3,6) = L_G$$

Let us now proceed to apply suitable transpositions on L_K. Consider

$$[2,5](L_K) = (1,4)(1,5)(1,6)(2,4)(2,5)(2,6)(3,4)(3,5)(3,6) = L_G$$

Therefore, graphs *G*, *H*, *K* are isomorphic.

Acknowledgements

I am thankful to Prof. M. R. Modak for useful discussions.

$$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$$$