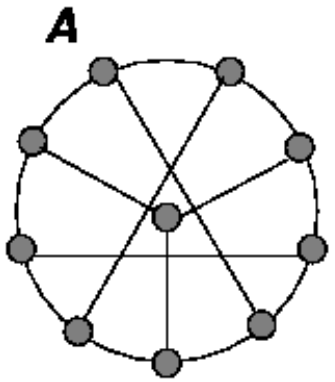
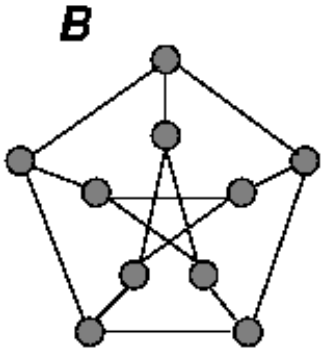
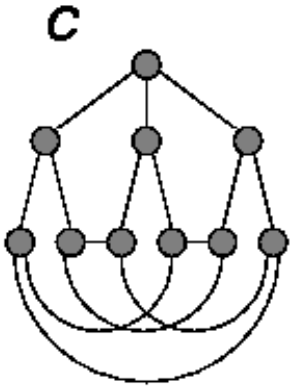
Which graphs are isomorphic to graph B?



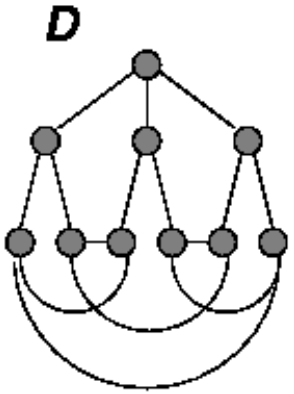
?



Petersen Graph



?



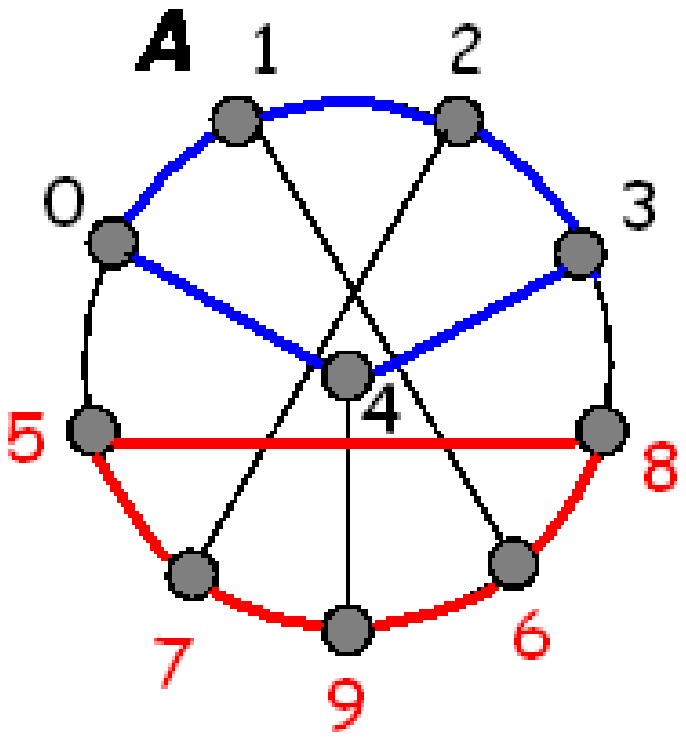
?

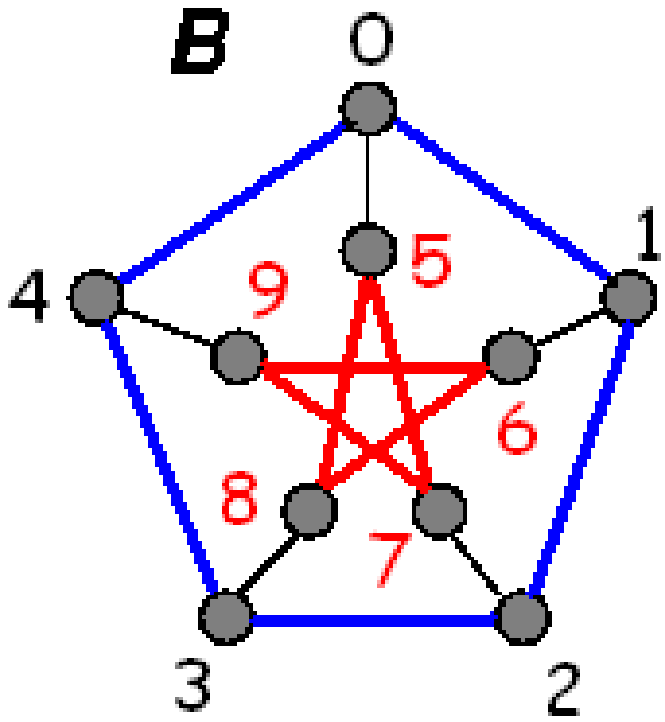
Graph Isomorphism

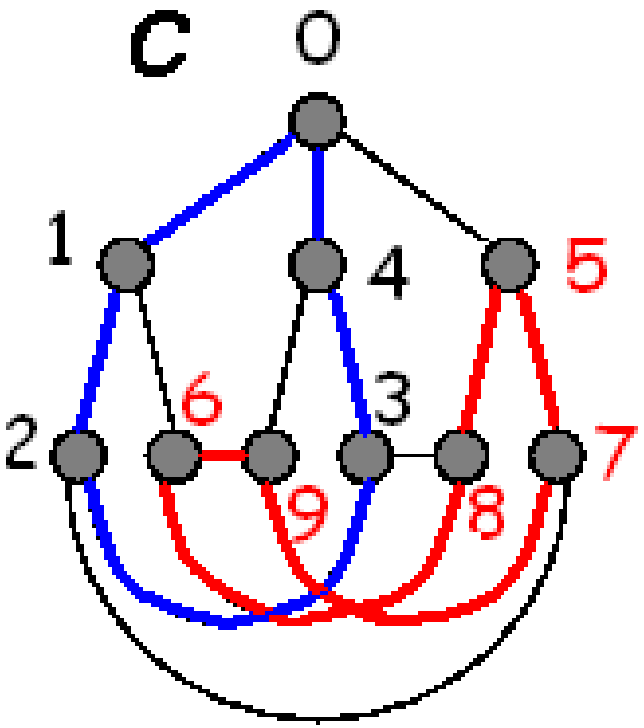
The graph isomorphism problem has no known polynomial time algorithm which works for an arbitrary graph.

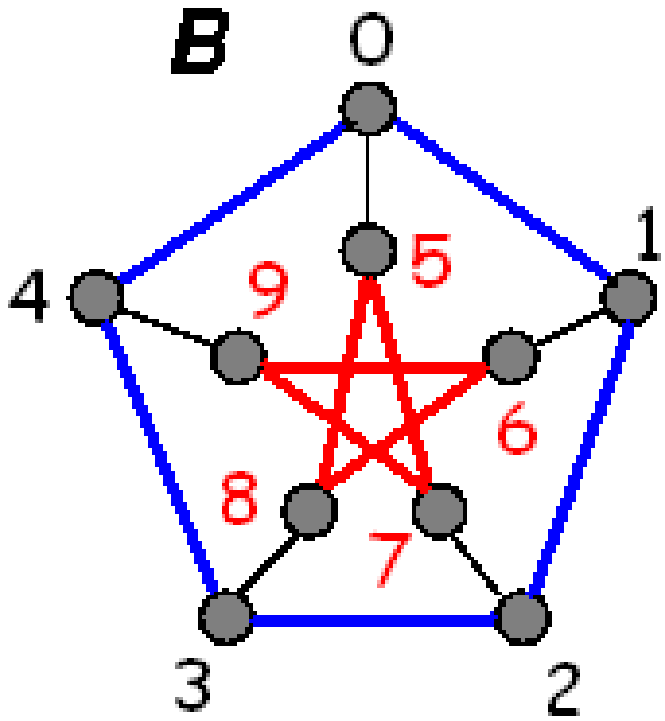
Canonical form: If two graphs are isomorphic, their canonical forms must be the same, otherwise, they must be different.

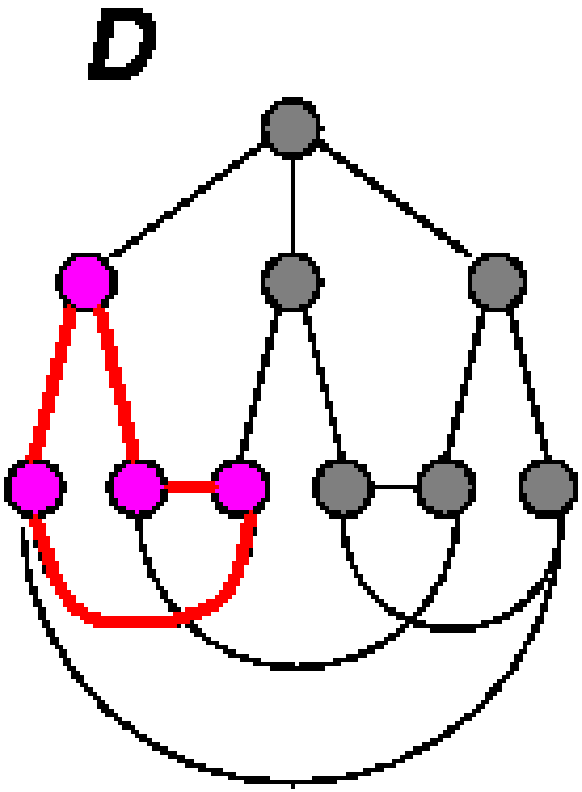
For trees and planar graphs, a canonical form can be computed in polynomial time.

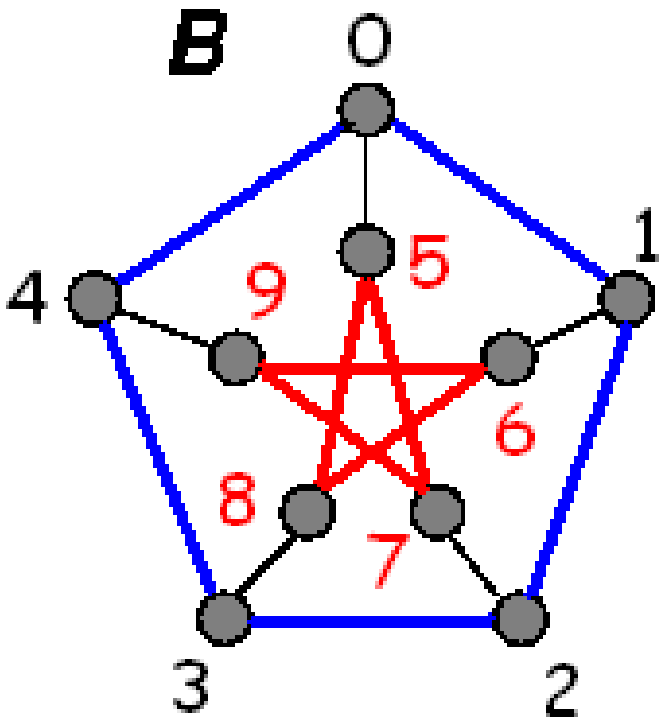

A
1
2
0
3
4
5
8
7
9
6


B
0
1
4
5
9
6
8
7
3
2

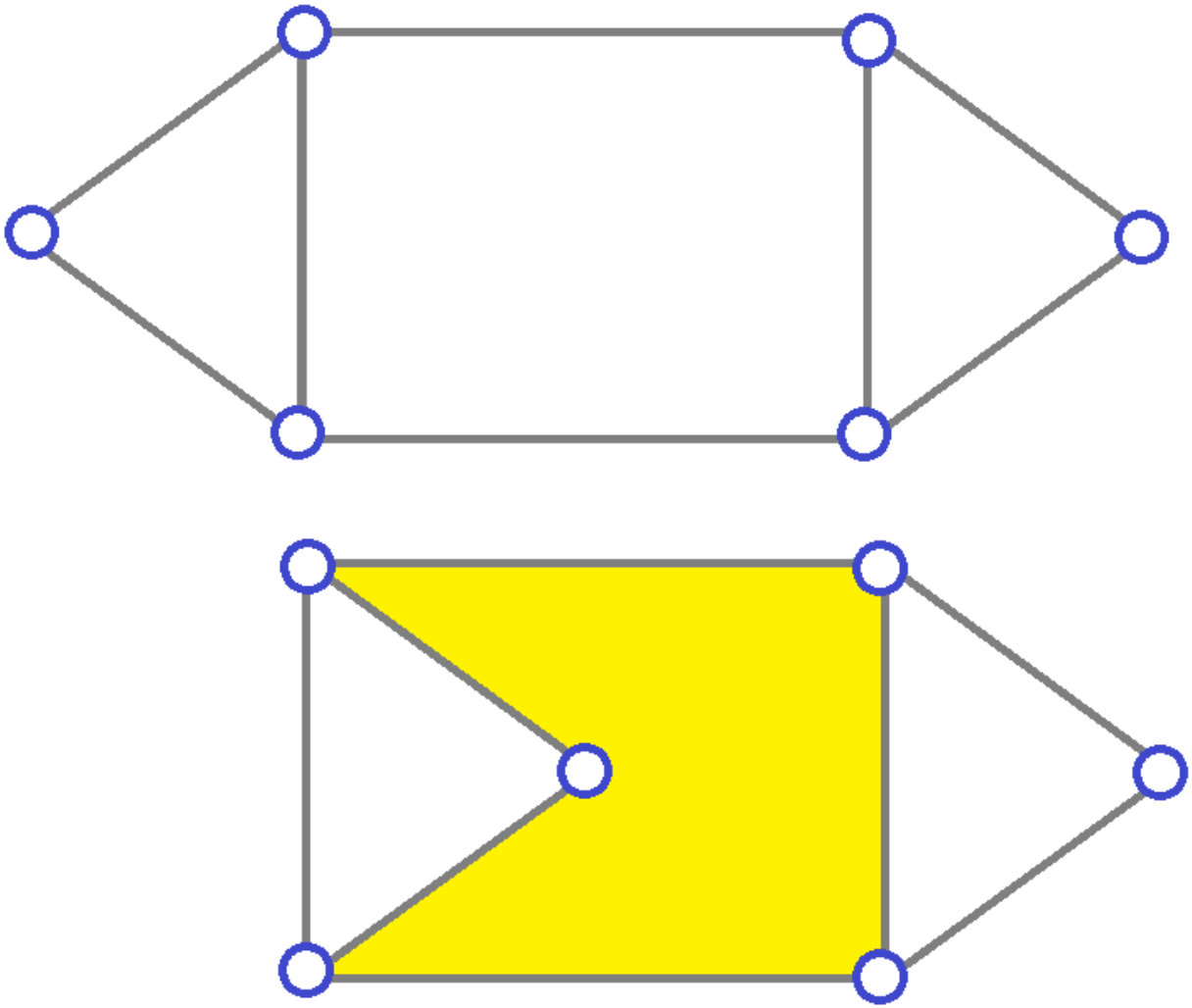

C
0
1
4
5
2
6
3
7
9
8


B
0
1
4
9
5
6
8
7
3
2

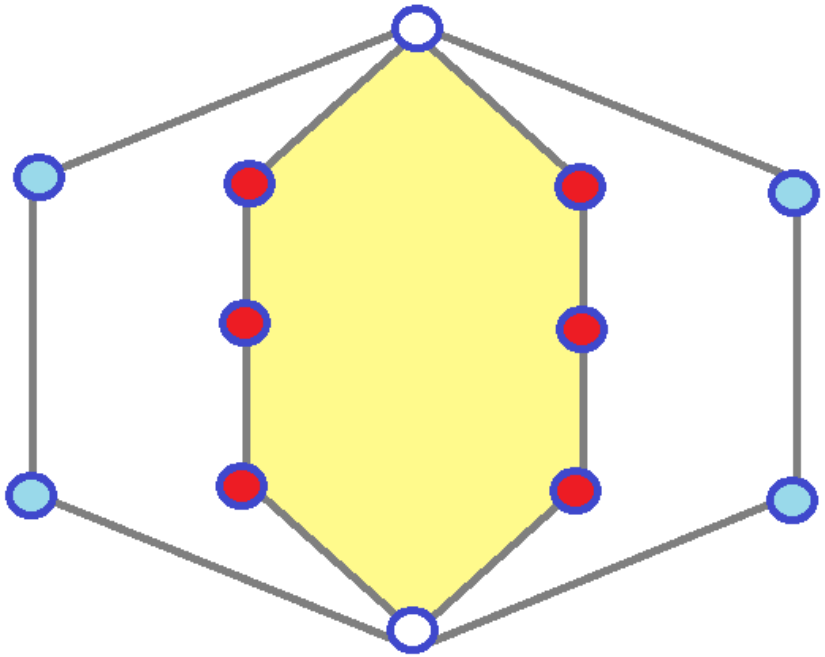

D

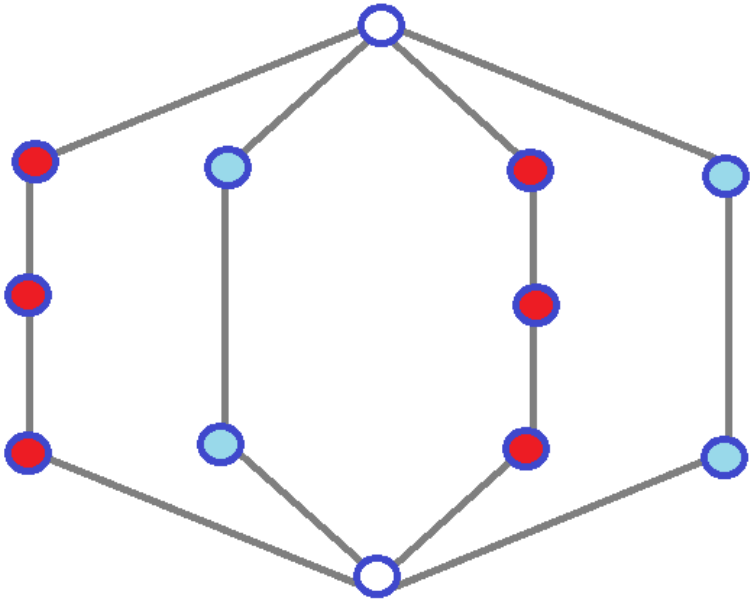

B
0
1
2
3
4
5
6
7
8
9

Planar graphs: Isomorphic graphs can have planar embeddings that are not isomorphic.



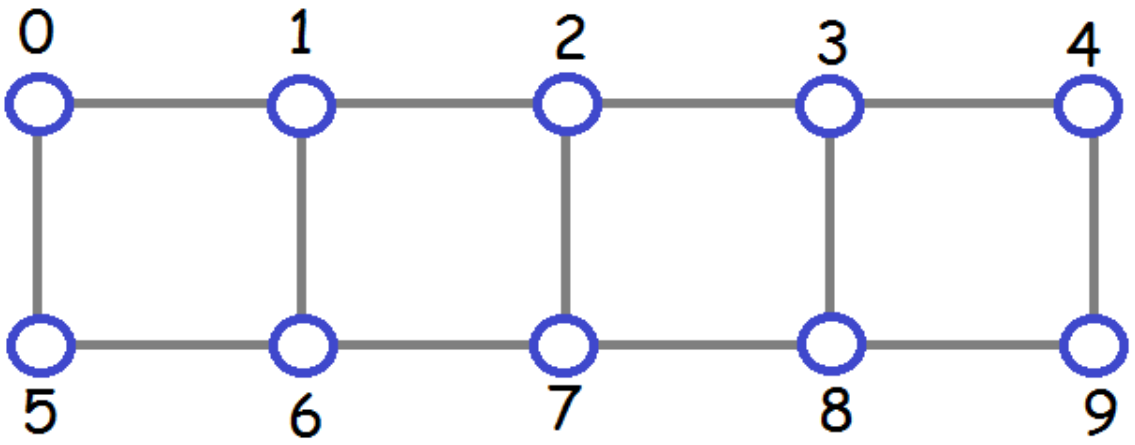
Two graphs that are isomorphic but their embeddings are not:





Isomorphism testing and finding automorphisms of embeddings.

Input: rotation system.



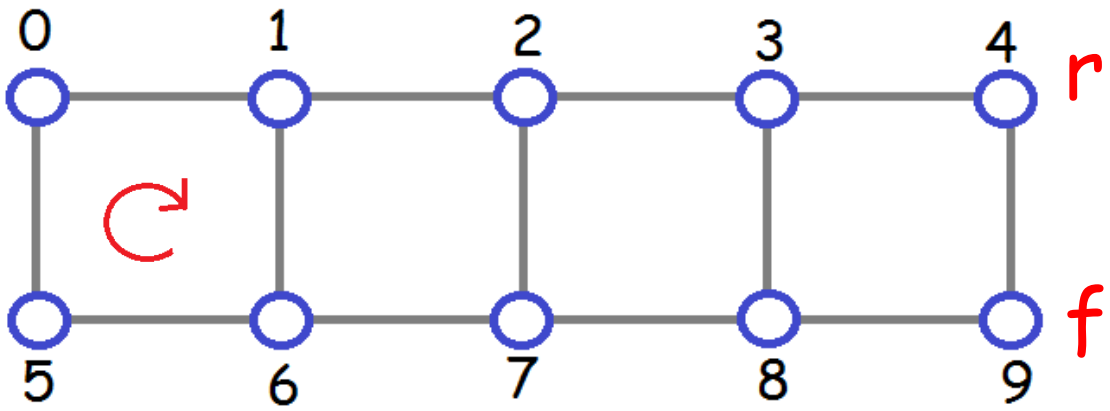
0: 1 5
1: 0 2 6
2: 1 3 7
3: 2 4 8
4: 3 9
5: 0 6
6: 1 7 5
7: 2 8 6
8: 3 9 7
9: 4 8

For each choice of a root vertex r, and a first child f that is a neighbour of r and for a direction d (clockwise or counterclockwise)

- Perform clockwise_BFS(r, f, d) and label vertices using the BFI's that result.

- Keep the smallest one (compare the rotation systems lexicographically).

This smallest rotation system $\widetilde{G}$ is the canonical form.

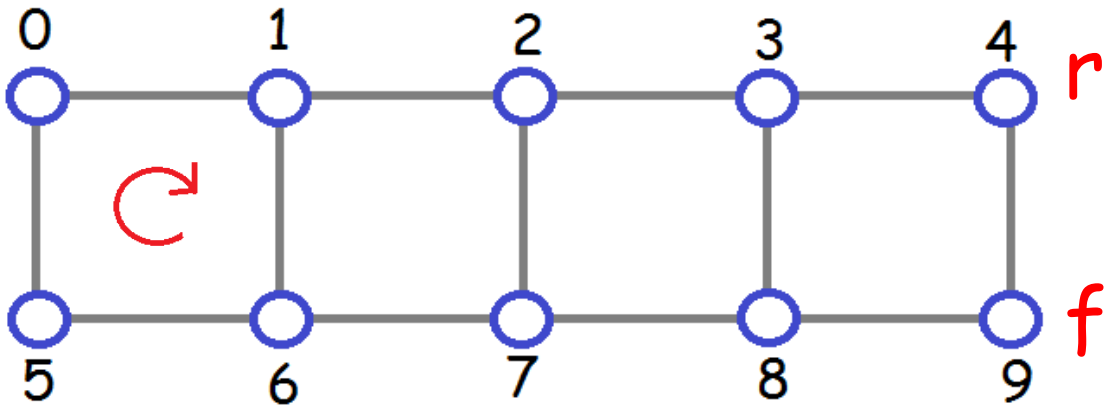


[0]: [1][2]

Neighbours of r=4 in cw order starting with f=9:

4: 3 9 ➡ 4: 9 3 ➡ 4 [0]: 9[1] 3[2]

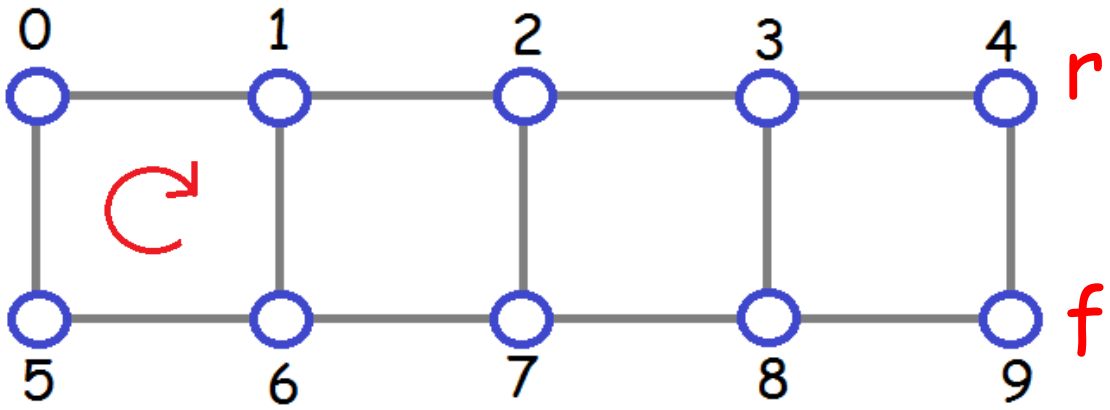
	0	1	2	3	4	5	6	7	8	9
Queue	4	9	3							
Parent				4	4					4
BFI				2	0					1



[0]: [1][2]
[1]: [0][3]

Neighbours of 9 in cw order starting with 4:
9: 4 8 → 9: 4 8 → 9[1]: 4[0] 8[3]

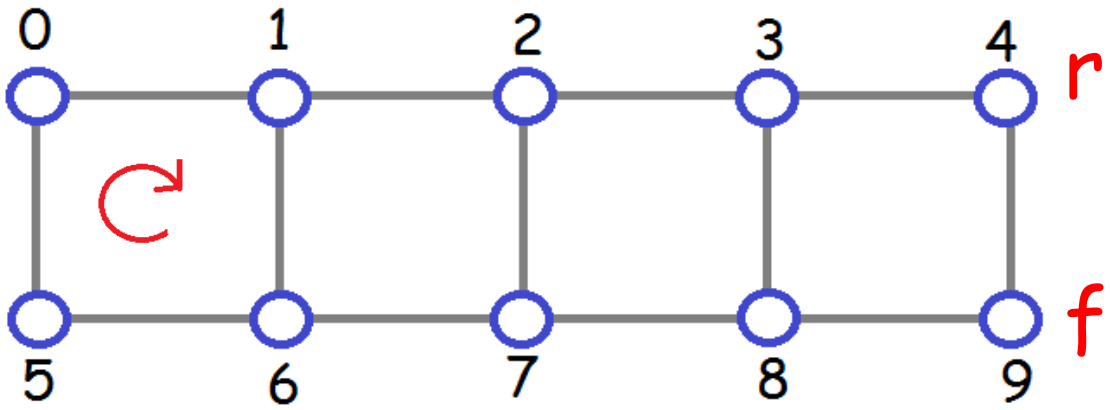
	0	1	2	3	4	5	6	7	8	9
Queue	4	9	3	8						
Parent				4	4				9	4
BFI				2	0				3	1



[0]: [1][2]
[1]: [0][3]
[2]: [0][3][4]

Neighbours of 3 in cw order starting with 4:
3: 2 4 8 → 3: 4 8 2 → 3[2]: 4[0]8[3]2[4]

	0	1	2	3	4	5	6	7	8	9
Queue	4	9	3	8	2					
Parent			3	4	4				9	4
BFI			4	2	0				3	1

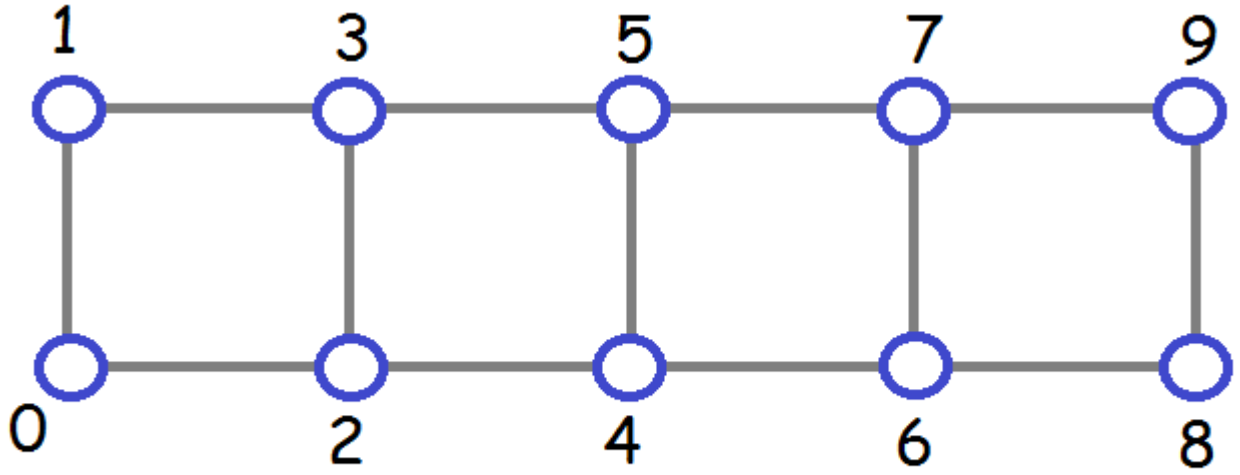


[0]: [1][2]
[1]: [0][3]
[2]: [0][3][4]
[3]: [0][5][2]

Neighbours of 8 in cw order starting with 9:
8: 3 9 7 → 8: 9 7 3 → 8[3]: 9[0]7[5]3[2]

	0	1	2	3	4	5	6	7	8	9
Queue	4	9	3	8	2	7				
Parent			3	4	4			9	9	4
BFI			4	2	0			5	3	1

Renumber rotation system so it is the lexicographic minimum (canonical form):



Two rotation systems are isomorphic to each other if and only if they have the same canonical form. A 3-connected planar graph has a unique embedding (if an embedding and its "flip" are the same) so this solves isomorphism for these graphs.

To get all automorphisms of the canonically labelled graph:

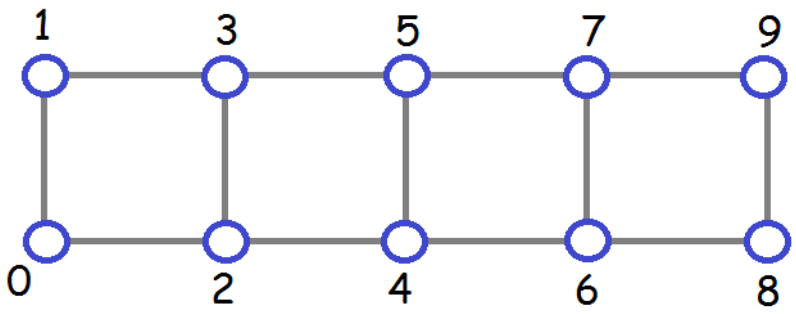
For each choice of a root vertex r, and a first child f that is a neighbour of r and for a direction d (clockwise or counterclockwise)

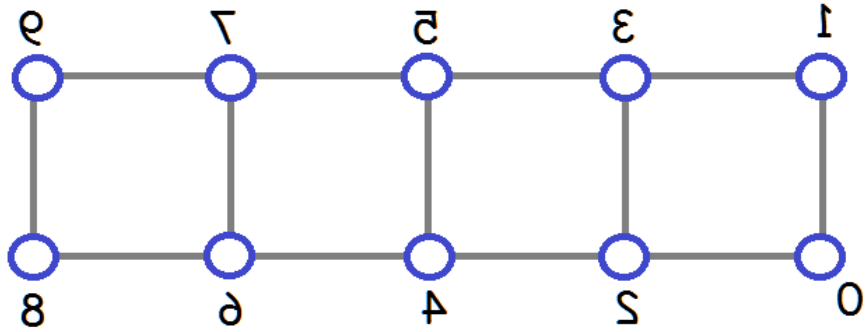
- Perform clockwise_BFS(r, f, d) and label vertices using the BFI's that result.
- The BFI permutation is an automorphism if the resulting rotation system is identical to the canonical one.

You don't need to write code for both clockwise and counterclockwise BFS if you apply the following trick:

For direction counterclockwise, first flip the rotation system (reverse the cyclic order of the neighbours of each vertex).

Then use your clockwise BFS routine on this flipped embedding.





flipped embedding

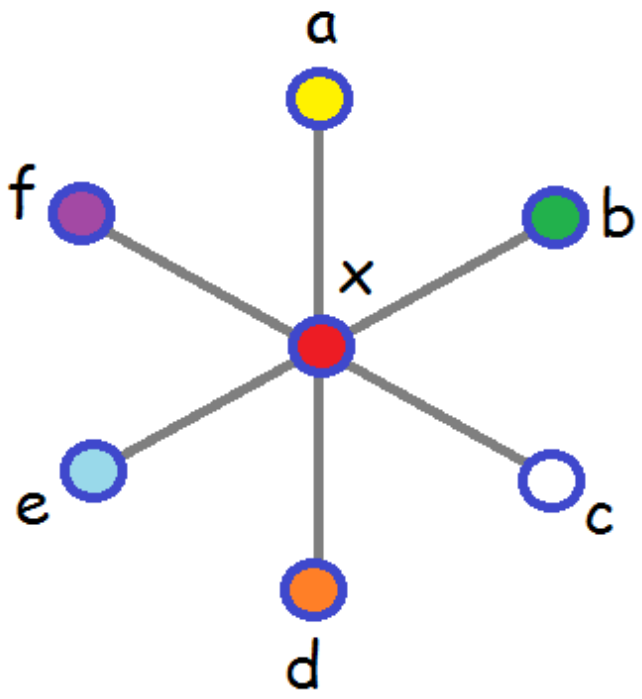
0: 1 2
1: 0 3
2: 0 3 4
3: 1 5 2
4: 2 5 6
5: 3 7 4
6: 4 7 8
7: 5 9 6
8: 6 9
9: 7 8

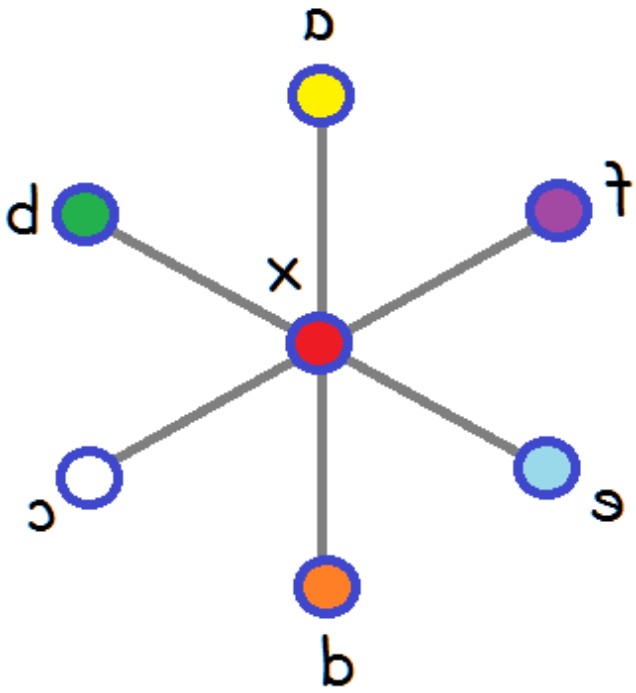
0: 1 2
1: 0 3
2: 0 4 3
3: 1 2 5
4: 2 6 5
5: 3 4 7
6: 4 8 7
7: 5 6 9
8: 6 9
9: 7 8

A generic flip: Note that we standardize the rotation systems by listing the smallest numbered neighbour first.

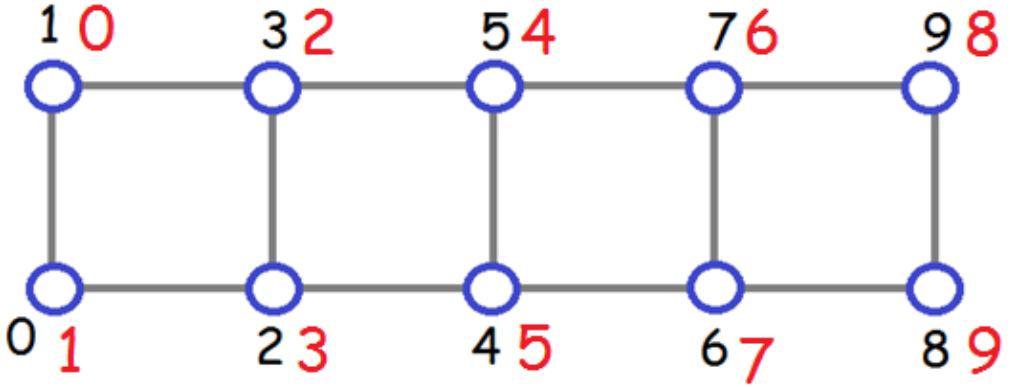
So when we flip:

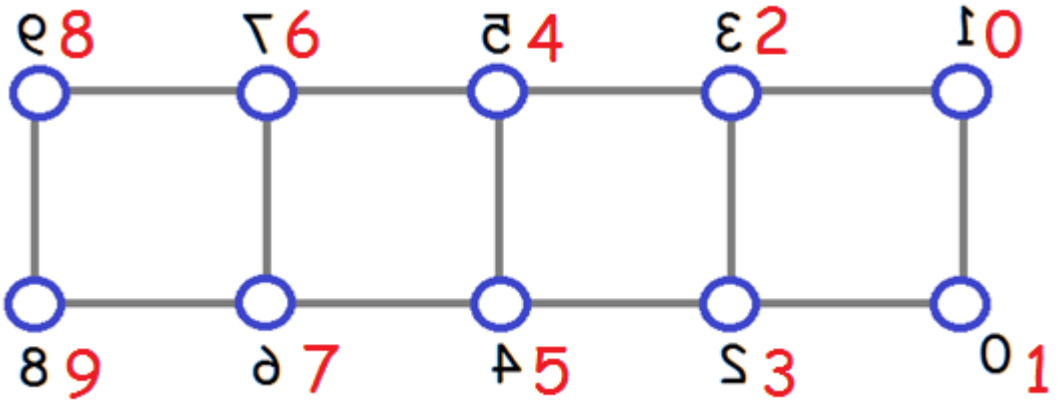
x: a b c d e f we get x: a f e d c b





IMPORTANT: to do ccw directly on a worksheet, when making new rotation system, you must traverse neighbours of each vertex in ccw order to write it down (starting with smallest neighbour).





Top one read ccw is same as flipped one read cw.

How much time does this take for a planar graph?

Planar graphs have at most 3n-6 edges.

Number of cw BFS's required in worst case: for each edge (u, v):

u= r, v= f, d= cw

u= r, v= f, d= ccw

v=r, u=f, d=cw

v=r, u=f, d= ccw

So 4[3n-6] which is in $O(n)$. Each takes $O(n)$ time for a total of $O(n^2)$.