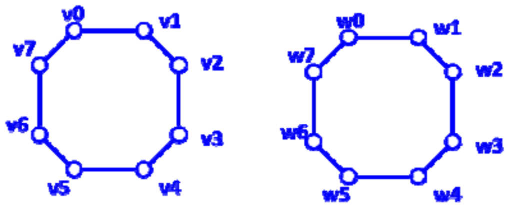
1. How many isomorphisms are there between two undirected cycles of length $k > 2$?

For each $j = 0, 1, ..., k - 1$ there is one isomorphism
φ_j(v_i) = w_((*i*+*j*) mod *k*).

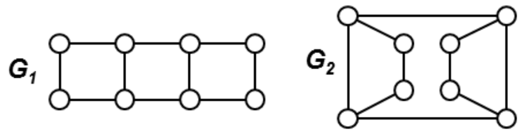
Each of these isomorphisms correspond to "rotation by *j* positions" in a fixed (clockwise or counterclockwise, but always the same) direction.



Another set of *k* isomorphisms corresponds to "flipping" or "displaying in a mirror" of one of the cycles before "rotation" is applied. The isomorphisms are
ψ_j(v_i) = w_((*k*-*i*+*j*) mod *k*), for each each $j = 0, 1, ..., k - 1$.

In total, there are $2k$ isomorphisms.

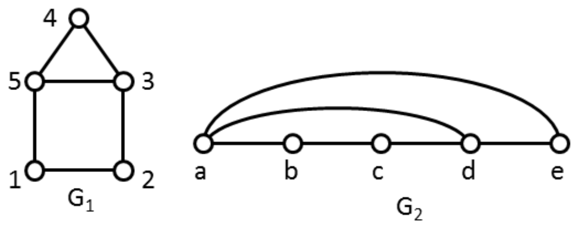
2. How many isomorphisms are there between G1 a G2?



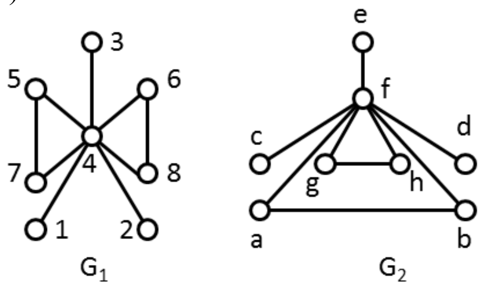
Label the nodes of the leftmost vertical edge in G1 by *a*, and *b*, label the two shortest vertical edges in G2 by pairs of labels (1, 2) and (3, 4). To maintain an isomorphism, edge (*a*, *b*) has to be mapped on either (1, 2) or (2, 1) or (3, 4) or (4, 3). This represent 4 different isomorphisms, which informally correspond to
1. flipping G1 around horizontal axis, 2. flipping G1 around vertical axis, 3. rotating G1 by 180 degrees, 4. not moving G1 at all.

3. In the picture bellow, what is the number of such bijections between the nodes of graph G1 and G2 which are not isomorphisms?

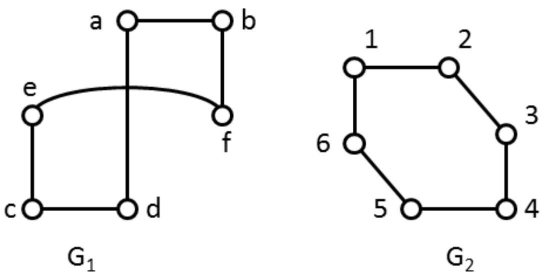
a)



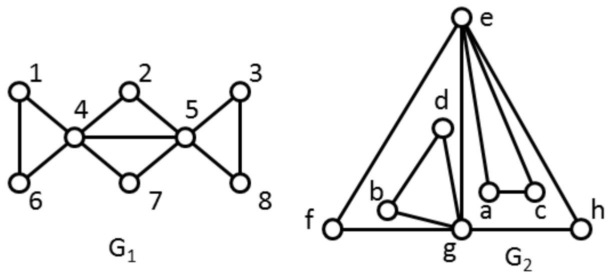
b)



c)



d)



a) Two isomorphisms : 1-- b, 2 -- c, 3 -- d, 4 -- e, 5 -- a; 1 -- c, 2 -- b, 3 -- a, 4 -- e, 5 -- d. Number of non-somorpism bijections is $5! - 2 = 118$.
b) Each isomorphism maps 4 -- f. There are 3 nodes of degree 1 in both graphs, these two sets can be mapped to each other in any of $3! = 6$ ways. Node 5 can be mapped to any of nodes *a*, *b*, *g*, *h*. After node 5 has been mapped, node 7 can be mapped in only one way to the neighbour of image of 5. Node 6 can be mapped to any of two nodes in the remaining triangle in G_2. In total, there are $6 \cdot 4 \cdot 2 = 48$ isomorphisms, the number of non-isomorphism bijections is $8! - 48 = 40272$.
c) The result of probem 1 yields $6! - 2 \cdot 6 = 108$ non-isomorphism bijections.

d) There are 4 possible mappings of node 1 to any of nodes *a*, *c*, *b*, *d*. Each of these mappings defines unambiguously the images of nodes 6, 4 and 5. Next, the pair of nodes (2, 7) can be mapped only to one of 2 pairs (*f*, *h*) or (*h*, *f*). After that, there are only two possibilities for node 3 to be mapped to one of the reaminig two nodes in G_2 to which nothing is mapped yet. In total, there are $4 \cdot 2 \cdot 2 = 16$ isomorphisms, the number of non-isomorphism bijections is $8! - 16 = 40304$.

4. There are two undirected graphs, both have *n* nodes and the degree sequence of both graphs is $(n-1, n-2, n-3, n-4, \ldots, n/2+1, n/2, n/2, n/2-1, n/2-2, \ldots, 3, 2, 1)$. As you can see, nearly all node degrees are different with the exception of two nodes which share the same degree $n/2$. The fact that node degrees are different may significantly speed up the process of deciding whether two graphs are isomorphic. What is the asymptotic complexity of the process you can think of in this case?

Obviously, there is just one node of any particular degree $1, 2, \ldots, n-1$ in each graph, except for the two nodes of degree $n/2$ in each graph. Apart from these four nodes of degree $n/2$, a node of degree *k* in one graph is mapped unambiguously the only node of degree *k* in the other graph. For the nodes of degree $n/2$ there are just 2 possible mappings. In total, there are only 2 different bijections between the nodes of the graphs which map a node of a particular degre in one graph to a node of the same degree in the other graph. Checking whether a bijection is an isomorphism takes $O(|E|)$ time. Therefore checking two bijections takes also just $O(|E|)$ time. Moreover, we know the number of edges in both graphs, it is (as in any simple graph) equal to the half of the sum of all node degrees which is in this case

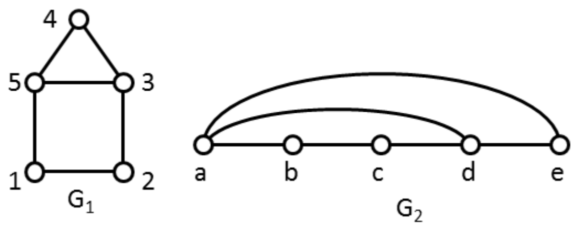
$1/2 \cdot ((n-1) + (n-2) + \ldots + (n/2+1) + n/2 + n/2 + (n/2-1) + \ldots + 2 + 1) = 1/2 \cdot (n(n-1)/2 + n/2) = n^2/4$.

The asymptotic complexity is thus $O(n^2)$.

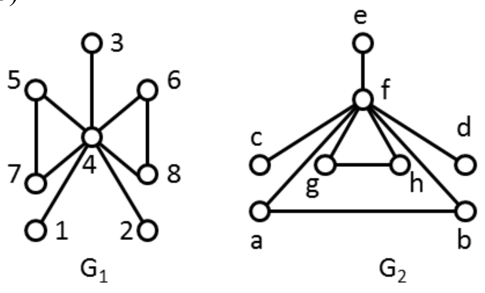
Additional remark: There is only one graph in sense of isomorphism with the given degree sequence for any even value of *n*. It can be easily shown by arranging the in the sorted order od their degrees nodes into a line and then constructing the edges starting in the nodes with highest degree. The process yields just one graph. It does not matter whether *n* is even or wheter *n* is odd and we consider $\lfloor n/2 \rfloor$ instead of $n/2$.

5. Repeat the process of determining the isomorphisms (lesson 07, slides 5.-8) for the pairs of graphs in problem 3.

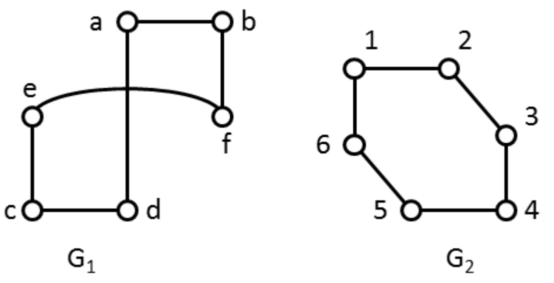
a)



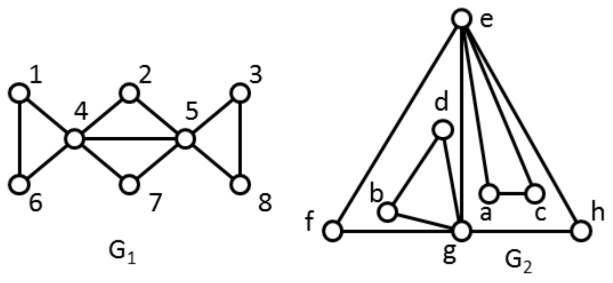
b)



c)



d)



a) 1. Node degrees

G_1 G_2

x	$D_1(G, x)$		x	$D_1(G, x)$
1	2		b	2
2	2		c	2
4	2		e	2
3	3		a	3
5	3		d	3

2. Number of nodes in distance 1, 2.

G_1 G_2

x	$D_2(G, x)$		x	$D_2(G, x)$
1	2, 2		b	2, 2
2	2, 2		c	2, 2
4	2, 2		e	2, 2
3	3, 1		a	3, 1
5	3, 1		d	3, 1

b) 1. Node degrees

G_1 G_2

x	$D_1(G, x)$		x	$D_1(G, x)$
1	1		c	1
2	1		e	1
3	1		d	1
4	7		f	7
5	2		a	2
6	2		b	2
7	2		g	2
8	2		h	2

2. Number of nodes in distance 1, 2.

G_1 G_2

x	$D_2(G, x)$		x	$D_2(G, x)$
1	1, 6		c	1, 6
2	1, 6		e	1, 6
3	1, 6		d	1, 6
4	7, 0		f	7, 0
5	2, 5		a	2, 5
6	2, 5		b	2, 5
7	2, 5		g	2, 5
8	2, 5		h	2, 5

c) 1. Node degrees

G_1 G_2

x	$D_1(G, x)$		x	$D_1(G, x)$
1	2		a	2
2	2		b	2
3	2		c	2
4	2		d	2
5	2		e	2
6	2		f	2

2. Number of nodes in distance 1, 2 ,3.

G_1 G_2

x	$D_2(G, x)$		x	$D_2(G, x)$
1	2, 2, 1		a	2, 2, 1
2	2, 2, 1		b	2, 2, 1
3	2, 2, 1		c	2, 2, 1
4	2, 2, 1		d	2, 2, 1
5	2, 2, 1		e	2, 2, 1
6	2, 2, 1		f	2, 2, 1

d) 1. Node degrees

G_1 G_2

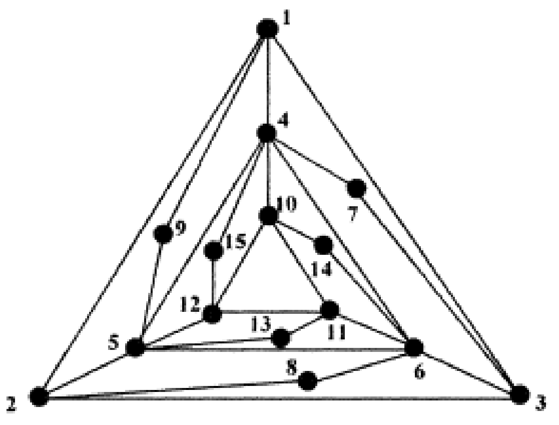
x	$D_1(G, x)$		x	$D_1(G, x)$
1	2		a	2
3	2		b	2
6	2		c	2
8	2		d	2
2	2		f	2
7	2		h	2
4	5		e	5
5	5		g	5

2. Number of nodes in distance 1, 2 ,3.

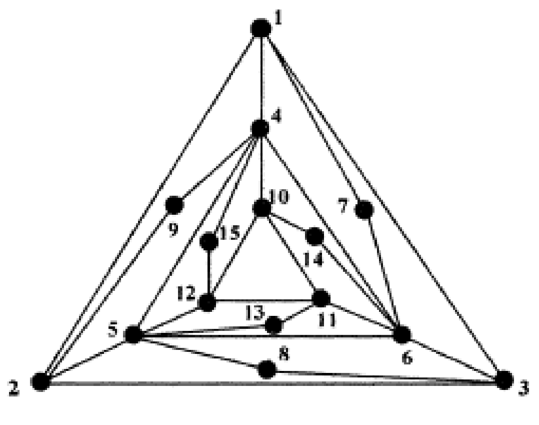
G_1 G_2

x	$D_2(G, x)$		x	$D_2(G, x)$
1	2, 3, 2		a	2, 3, 2
3	2, 3, 2		b	2, 3, 2
6	2, 3, 2		c	2, 3, 2
8	2, 3, 2		d	2, 3, 2
2	2, 5, 0		f	2, 5, 0
7	2, 5, 0		h	2, 5, 0
4	5, 2, 0		e	5, 2, 0
5	5, 2, 0		g	5, 2, 0

6. Decide whether the two given graphs are isomorphic. Describe the process of decision, remember, it shoud be effective.



(a)



(b)

Divide the nodes into maximum possible families of nodes. Node in one family should be characterized by the same value of some invariant, the value of invariant should be different for nodes in different families. Denote the graphs by L (left) and R (Right), respectively. Let us follow the lecture method:

1. Node degrees

2. Number of nodes in distance 1, 2 ,3.

L			R	
x	$D_1(G, x)$		x	$D_1(G, x)$
1	4		1	4
2	4		2	4
3	4		3	4
4	6		4	6
5	6		5	6
6	6		6	6
7	2		7	2
8	2		8	2
9	2		9	2
10	4		10	4
11	4		11	4
12	4		12	4
13	2		13	2
14	2		14	2
15	2		15	2

L			R	
x	$D_2(G, x)$		x	$D_2(G, x)$
1	4, 6, 4		1	4, 6, 4
2	4, 6, 4		2	4, 6, 4
3	4, 6, 4		3	4, 6, 4
4	6, 8, 0		4	6, 8, 0
5	6, 8, 0		5	6, 8, 0
6	6, 8, 0		6	6, 8, 0
7	2, 6, 6		7	2, 6, 6
8	2, 6, 6		8	2, 6, 6
9	2, 6, 6		9	2, 6, 6
10	4, 6, 4		10	4, 6, 4
11	4, 6, 4		11	4, 6, 4
12	4, 6, 4		12	4, 6, 4
13	2, 6, 6		13	2, 6, 6
14	2, 6, 6		14	2, 6, 6
15	2, 6, 6		15	2, 6, 6

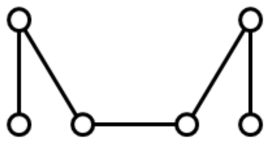
We can se that the partition induced by D_1 (namely {1,2,3,10,11,12}, {4,5,6}, {7,8,9,13,14,15}) is the same as partition induced by D_2, so D_2 brings no improvement, that is, the number of families of nodes is still the same, equal to 3. The number of all node bijections between L and R, which respect the current partitioning is $6! \cdot 3! \cdot 6! = 720 * 6 * 720 = 3110400$.

Now we can try some other invariant inducing functions like e.g. degrees of nodes in particular distance from x, eccentricity of x, eccentricity of neighbours of x, number of triangles incident with x, number of cycles of lenght 4 incident to x, etc. None of those would significantly increase the number of different node families in the graphs. The example was chosen to demonstrate the difficulty of of isomorphism assessment.

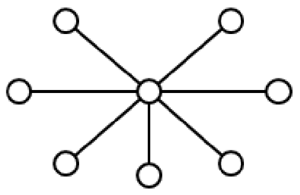
The graphs are not isomorphic. R contains a circle of length 6, which contains two nodes of degree 2 and the distance of those two nodes is 2. For example, cycle 5 -- 8 -- 3 -- 6 -- 11 -- 13. L contains no such cycle.

7. Suppose a certificate of a tree is given. Each node of the tree is represented in the certificate as a (continuous) substring. Create certificates for the trees in cases a) - d) and determine which substrings of the certificates correspond to which nodes.

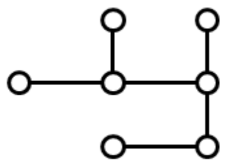
a)



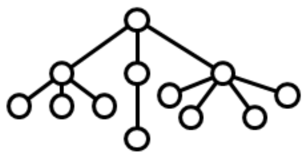
b)



c)



d)



a) 000111000111
b) 0010101010101011
c) 00010110011011
d) 000101010110010101100111

8. Reconstruct the tree from the given certificate.
a) 0011
b) 0001011001010111 (*note that the last "1" migh be missing in the original assignment copy, due to an error*)
c) 000101100101100101111
d) 0000110001011110010001011111

a) This certificate is incorrect. There is only one tree with just two nodes and its certificate is 0101.
b) The tree can be reconstructed, but the certificate is also incorrect. The two subtrees of the center node are coded by labels 001011 and 00101011 and the second is lexicographically smaller than the first, the certificate should be 0 00101011 001011 1 (spaces added for optical clarity). The shape of the tree is a root with two children x and y, x has two children, y has 3 children.
c) Add spaces for optical clarity: 0 001011 001011 001011 1. The tree consists of a root and three identical subtrees, each has its own root and two children.
d) The tree can be described as follows: Draw path of 9 nodes 1 -- 2 -- 3 -- ... -- 9. Append paths of length 1 (leaves, in fact) to nodes 2, 6 and 8. Append path of lenght 2 to node 4. The center of the tree (the root, if you will) is in node 5.

9. We know the certificate of a particular tree. Describe how to find the number of leaves in the tree using only the certificate without reconstructing the tree itself.

According to the algorithm of certificate construction, all leaves are labeled by string 01. A label of an internal node X Internal node labels consist of at least 4 symbols and this label contains as substrings all labels of subtrees rooted in X. Therefore the correspondence between substrings 01 in the tree certificate and the tree leaves is a bijection. The number of substrings 01 in the certificate is equal to the number of leaves in the tree.

10. We know the certificate of a particular tree. Describe how to find the maximum degree of all nodes in the tree using only the certificate without reconstructing the tree itself.

The sequence od 0's and 1's in the tree certificate corresponds to the sequence of visited edges during a DFS traversal of the tree. DFS starts in the center of the tree and always chooses the child of the current node (the node to expand the search to) as the one with the laxicographically smallest label. The order of visited children is important during encoding (= certificate creation); during certificate decoding it is enough just to draw a next edge whenever 0 is encountered in the certificate and to move back along one edge in the direction of the center whenever 1 is encountered. The number of returns to a node denotes the number of the children of the node drawn in the process. We implement a stack which is originally empty and which contains the nodes drawn or visited during the process. When a node is left for the last time and DFS moves back to the parent of the node (node is closed) the node is removed from the stack. All we need is to register the number of visits to the node while the node is in the stack. When the node is closed the number of visits is equal to the degree of that node.

To assess the maximum it is enough to update the maximum degree in a separate variable during the DFS.

11. A tree is said to be of type T(1,3) when its node degrees are only 1 or 3. Describe informally how the certificate of such tree looks like and design an algorithm based on the tree certificate which will decide whether the corresponding tree is of type T(1,3).

Use the discussion of the previous problem to assess the degrees of all nodes in the tree. Then (or during the process) just check if each internal node degree is 3.