* 1. We take an undirected cycle and we assign randomly direction to each edge. What is the relation between the number of roots and number of leaves in the resulting directed graph?

The sum of indegrees of all nodes is equal to the sum od outdegrees of all nodes. This is because each edge has the source and the target node and therefore contributes 1 to the sum of all outdegrees and also 1 to the sum of all indegrees.

Let us denote the number of all root, leaves and other nodes by R, L and N, respectively
The outdegree and indegree of each root is 2 and 0.
The outdegree and indegree of each leaf is 0 and 2.
The outdegree and indegree of each othe node is 1 and 1.

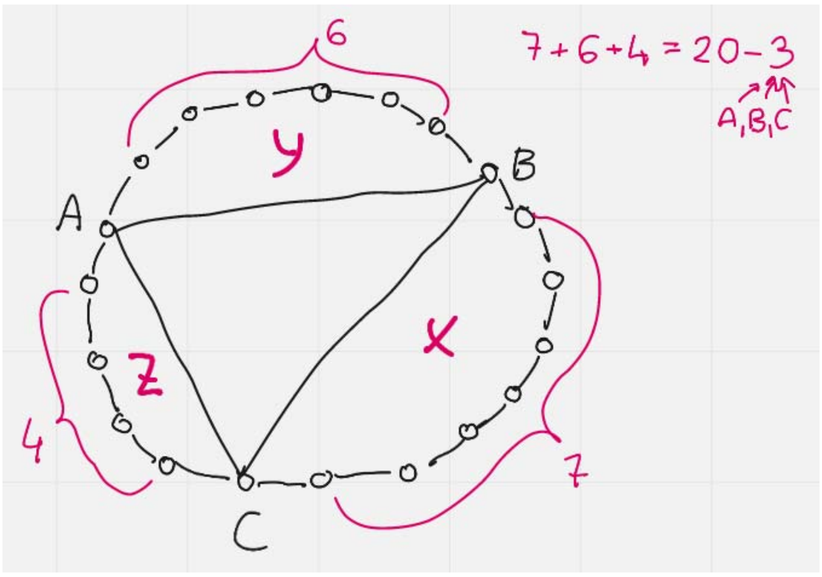
Knowing already that
sum of all outdegrees = sum of all indegrees
we can rewrite it as
$2R + 0L + 1N = 0R + 2L + 1N$.
Using elementary algebra, the equation simplifies to
$R = L$
which means that the number of roots is equal to the number of leaves, no matter how the individual edges are oriented.

* 2. A complete bipartite graph $K_{m,n}$ is Hamiltonian. What can be said about values *m* and *n*?

A. Hamilton cycle.
Denote the partitions in the bipartite graph by P and Q. $|P| = m$, $|Q| = n$..
There are no edges between any two nodes which are in the same partition in a bipartite graph.
In other words, each edge connects a node in P to a node in Q.
Therefore, the nodes in any cycle, have to alternate between the partitions, no two neightbour nodes in the cycle belong to the same partitions. Also, there are only two partitions in a bipartite graph. Thus, in the cycle, the number of nodes which belong to P is the same as the number on nodes which belong to Q.
When the cycle is Hamilton, that is, when it contains all nodes in the graph, the number of nodes in P and the number of nodes in Q are the same, $m = n$. The fact that the graph is a complete graph did not play any role in the reasoning.
Lastly, both *m* and *n* has to be bigger than 1. Otherwise, when $m = n = 1$, there is no cycle in the graph and the graph cannot contain a Hamilton cycle.

B. Hamilton path.
A reasoning analogous to case A can be applied here too. The nodes on any path in a bipartite graph alternate between partitions P and Q. Whenthe path contains all nodes in the graph, there are two possibilities.
The length of the path is even and then $|P| = |Q| = n = m$.
The lenght of the path is odd then either $|P| + 1 = |Q|$ or $|Q| + 1 = |P|$.
So, in case B, the values of *m* and *n* can differ by at most 1.
Lastly, $m = 1$ or $n = 1$ is also possible in case B.

* 3. We take an undirected cycle with 20 nodes and add another three edges in such way that the resulting graph contains an Euler cycle. We do not add any node. How many different (non-isomorphic) graphs can be produced this way?



The resulting graph contains an Euleri cycle, therefore the degree of each of its nodes is even.
Originally the graph was a mere cycle and the degree ofeach node was 2.
When the first edge was added, between nodes A and B, the degree of A and B became 3.
To restore the even parity of the nodes A and B each of the remaining two edges which were added had to have one of the nodes A or B as its end node, thus increasing the degrees of A nad B to even value 4.
These two remaining edges should meet in one node C, so that parity of the degree of C is again even, equal to 4.
Consequently, the updated graph looks like the one in the picture, with three smaller loops marked X, Y and Z.
Each of the loop is characterized by the number of nodes of degree 2 it contains, in the example those numbers are 7,6,4. The sum of these numbers equals 20 – 3 = 17.
To produce a graph which is not isomorphic to the presented one, the set of its anlalogously constructed three numbers must be different from the set {7,6,4}.
In general, each such graph is characterized by a set of positive numbers r, s, t, which sum is 17.
Note that none of r,s,t can be 0, that would result a loop of length 2 in the modified graph, which would be made of two parallel edges between A and B, or between A and C, or between B and C. Typically, we do not accept parallel edges in most graph problems in computer science, so let us avoid them here as well. (If you are especially fond of parallel edges you may repeat the following callculation with 0's as well.)

So, let us list all sets {r, s, t}, wehere the elements are positive integers with sum equal to 17.
This can be done either by hand or by a program if we apply/develop a suitable algorithm.
By hand, it is fast for these small values:

```
for r in [15 .. 6]
  for s in [ 17– r – 1 .. 17]
     t = 17– r – s
     if t > s: break
     write( r, s, t )
```

The method produces the desired triples of values r,s,t. Each triple contains values in non-increasing order, all 24 triples are produced and listed below in in descending lexicographical order. This organization helps us to keep track of the generated values and it allows for manual/optical verification of the correctness of the result.

15,1,1	14,2,1	13,3,1 13,2,2	12,4,1 12,3,2	11,5,1 11,4,2 11,3,3	10,6,1 10,5,2 10,4,3	9,7,1 9,6,2 9,5,3 9,4,4	8,8,1 8,7,2 8,6,3 8,5,4	7,7,3 7,6,4, 7,5,5	6,6,5

There are 24 different (non-isomorphic) graphs which satisfy the conditions of the problem

* 4. We say that two directed graphs are weakly equivalent iff their respective condensations contain equal number of nodes. What is the best possible asymptotic complexity of verification whether two graphs are weakly equivalent?

A condensation of the graph can be created in $\Theta(|V|+|E|)$ time using a modification of Tarjan's Algorithm which will return the number of strongly connected components it detected in the graph. It suffices to add one counter variable to the algorithm and increase it by 1 each time a component is detected.The modified algorithm will return the counter value.
Let the given graphs be $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The complexity of running the modified algorithm on both G_1 and G_2 and comparing the returned values is clearly $\Theta(|V_1|+|E_1|) + \Theta(|V_2|+|E_2|) = \Theta(|V_1|+|V_2|+|E_1|+|E_2|)$.

* 5. You have to find and write out all paths of length 3 in a simple (no parallel edges) directed acyclic graph. What is the maximum possible number of these paths relatively to the number of nodes in the graph? What is the asymptotic complexity of your algorithm?

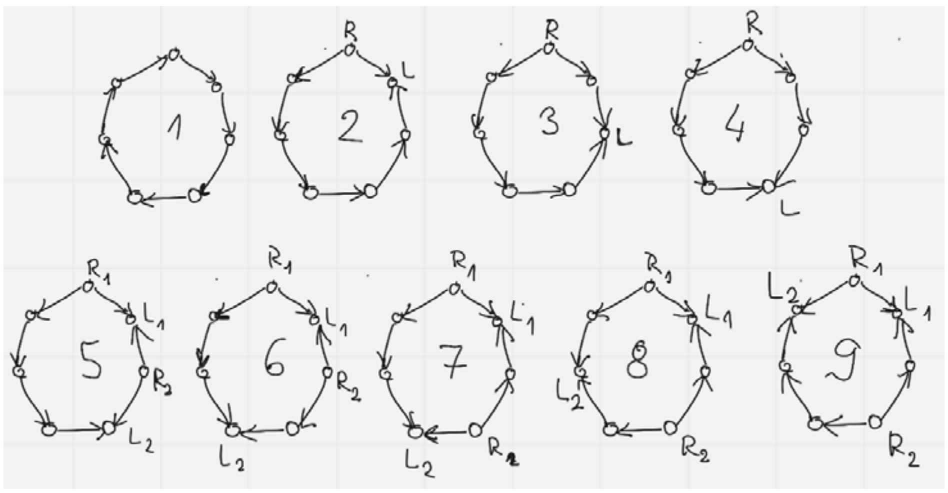
The maximum number of paths will appear in the graph which contains maximum possible number of edges.
The graph is directed acyclic, so let us consider its topological order and create edges from all nodes x, to all nodes y, whenever $x < y$.
Each pair of nodes {x, y} would then specify one edge. The number of pairs and also the number of edges, denoted by $|E|$, would be then equal to Comb(N,2), where N is the number of nodes in the graph.
Now, having an edge between each pair of the nodes, let us consider all paths of length 3, Each such path contains 3 edges and consequently 4 nodes.
When we choose any 4-tuple of nodes, it specifies exactly one path of length 3. Denote the nodes of the 4-tuple by a,b,c,d, in such way that $a < b < c < d$ in the lexicographical order of the nodes of the graph. The edges in the path would be then (a,b), (b,c), (c, d).
The number of all paths of length3 is equal to the number of all 4-tuples of the nodes, their number is exactly $\text{Comb}(N,4) = N(N-1)(N-2)(N-3)/24 \in \Theta(N^4)$.

The algorithm has to list each path of lenght 3, thus its complexity is also at least $\Theta(N^4)$.
It will not be bigger than $\Theta(N^4)$, though, because the straightforward four_nested_loops approach does the job:

```
for node x in G
   for all edges (x, y)
      for all edges(y, z)
         for all edges(z, w)
            print (x,y,z,w)
```

The acyclicity of the graph quarantees that the sequence of nodes x,y,z,w, does not contain any cycle, therefore all output 4-tuples x,y,z,t really specify a path (with no loops in it).

6. Assign direction to each edge of undirected cycle with 7 nodes. How many non-isomorfic directed graphs can be produced this way?



There is one graph with no root (1), three graphs with one root (2,3,4) and five graphs with two roots (5,6,7,8,9).
Obviously, the graph with no root can be only one.
The three different graphs with one root differ from each other by the distance of the leaf from the root: the distance is either 1 or 2 or 3, it cannot be bigger.
When there are wo roots in the graph, we distinguish two cases: The distance of the roots is either 2 or 3.
(Verify yourself, that the distance of the roots cannot be 1 or bigger than 3.)

The case of root distance equal to 2 is in the graphs 5 and 6.
The case of root distance equal to 3 is in the graphs 7,8,9.

7. We are given a directed graph G with *n* nodes and *m* edges. The task is to produce a strongly connected graph by adding some edges (no nodes) into G. The number of added edges should be minimal. Suggest an effective algorithm which solves the task and find its asymptotic complexity.

The problem presented here is not a trivial one, the references can be found in
J. Bang-Jensen, G. Z. Gutin: Digraphs: Theory, Algorithms and Applications, Springer 2009.
It is a mistake that the problem appeared here, we apologize.

8. An undirected graph is directionaly homogenous when the distance from a root to a leaf is the same for each possible pair (root, leaf). Formulate an effective algorithm which decides whether a graph is directionally homogenous. Assess the asymptotic complexity of your algorithm. Additionally, suppose that we know that the graph is acyclic. Can the algorithm be improved to handle acyclic graphs more effectively?

Slow method
Run BFS from each root and assign the distance form that root too each of the leaves.
All distances assigned to all roots for all runs of BFS must be the same for the graph to be directionaly homogenous.
If any of the leaf distances differ from other distances (of the same leaf or of other leafs) the the graph is not directionaly homogenous.
The number of roots may be quite high, in the order of O(N), thus the complexity
would be O(N) * O(N+M), where M is the number of edges. M itself may be in irder O(N^2), so the total complexity would be O(N) * O(N^2) = O(N^3).

Faster method
Run BFS from each root as in the previous variant. During a search, assign the distance from the current root to each node which was not yet visited by this search or any of the previous searches.
The first BFS will be terminated when it visits and assign the distances to all nodes accesible from the first root.
However, each of the subsequent BFS'es will not expand a node x which already has a distance assigned to it.
If the current BFS would assign a different distance to x than that already registered in x, that would indicate that the
distance from the current root to x and to all subsequent nodes on a path to a leaf would be different from the previously assigned distance and consequently the algorithm would stop at this moment, indicating that the graph is not directionaly homogenous.
If, on the other hand, the current BFS would assign distance to x equal to that already registered in x, it would also assign the same distance to all tsubsequent nodes on a path to a leaf and therfore it woud not generate any new information. Thus, it has not continue the search from x further.
In this variant, each edge will be traversed only once, reducing the complexity of the algorithm to the same complexity as that of a single search -- Θ(N+M).

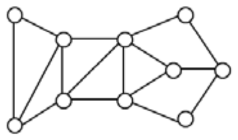
Concluding remark: To make the whole scheme work, it additionally has to neglect and not process any such nodes from which no leaves are accessible. These nodes can be discovered (and marked as such) by running BFS in reverse direction starting in all leaves simultaneously (and thus keeping the standard complexity Θ(N+M)).

9. Find a directed graph in which the indegree and outdegree of each node is at least 1 and also there is a node in the graph which is not part of any cycle.

See the example below, the node between the too loops is not a part of any of the cycles:

```
o----->o---->o---->o---->o
^      |           ^     |
|      |           |     |
|      V           |     V
o<-----o           o<----o
```

10. The graph in the picture does not contain a Hamiltonian cycle. Present a complete and exhaustive explanation of this fact. Decide whether the graph contains a Hamiltonian path. Can you add a single edge to the graph so that the result will contain a Hamiltonian cycle?



```
A-----B-----C--------D          A-----B-----C########D
|    /|    /|\       |          |    /|    /|#       #
|   / |   / | \      |          |   / |   / | #      #
|  /  |  /  |  E-----F          |  /  |  /  |  E-----F
| /   | /   | /      |          | /   | /   | #      #
|/    |/    |/       |          |/    |/    |#       #
G-----H-----I--------J          G-----H-----I########J

A) The original graph                        B)
```

Suppose there is Hamilton cycle in the graph. Consider the image B). The cycle must go through D, using highlighted edges {C,D} and {D,F}. Also, cycle must go through J, using highlighted edges {I,J} and {J,F}. Ths means that the cycle also goes through F via edges {D,F} and {F,E} and it cannot contain the edge {E.F}. As the cycle goes also through node E, it has to contain the edges {C,E} and {I,E}, there is no other pair of edges evailable at E. So far, we have shown that any Hamilton cycle must contain edges {C,D}, {D,F}, {F, J}. {J,I}, {I,E}, {E,C}. However this set of edges already forms a cycle, it is clear from the image. The cycle does not contain all nodes of the graph, it is not a Hamilton cycle (and, of course, it cannot be extended to a Hamilton cycle by adding more edges, that is clearly impossible).

```
                                      ..........
                                      :        :
A#####B-----C########D          A#####B-----C########D
#    /|    #|\       #          #    /|    #|\ :     #
#   / |   # | \      #          #   / |   # | \:     #
#  /  |  #  |  E-----F          #  /  |  #  |  E-----F
# /   | #   | #      #          # /   | #   | #      #
#/    |#    |#       #          #/    |#    |#       #
G#####H-----I########J          G#####H-----I########J

        C)                                  D)
```

There is a Hamilton path in the graph shown in the picture C). Adding a single edge between B and E, shown as the dotted line between B and E, closes the path in the picture C and makes a Hamilton cycle in the updated graph.

11. Undirected graph of type (r) is produced as follows. Choose two sets of nodes A = {a_1, a_2, ... a_r}, B = {b_1, b_2, ... b_r}. Create a complete graph G1 which node set is A and another complete graph G2 which node set is B. Next, create a final graph by joinig G1 and G2 by edges (a_1, b_1), (a_2, b_2) ..., (a_r, b_r). You have to decide for which values of r the resulting graph of type (r) will be an Euler graph.

All the node degrees in the resulting graph must be even for the graph to be an Euler graph.

According to the definition, the degrees of all in the given graph will be the same, namely r. For each node x, there are r–1 edges going from x to the other nodes of the same set and one edge going form x to its counterpart in the other set.
Thus the graph of type (r) will be Euler graph only for positive even values of r.

12. We are given an acyclic weighted graph and the task is to find the most expensive path from some root to some leaf both of which we can choose. Formulate the algorithm, explain its asymptotical complexity and decide which graph representation is best suited for the task.

Add an artifical root R and an edge weighted by 0 to all roots in the original graph. Also add an artificial leaf L and add edges weighted by 0 form all leaves in the original graph to L.
In the resulting graph, find the longest (=most expensive) path from R to L using the standard longest path in DAG algorithm based on Dynamic programming. The complexity is also standard, $\Theta(|V|+|E|)$. Also, the standard list representation of the graph would be sufficient, as it is sufficient (and suitable) for most DP algorithms on DAG.

13. A distance from a strong component A to another strong component B in a directed graph is defined as a minimum number of such strong components not equal to A or B which intersetion with the shortest path from A to B is unempty. If no path from A to B exists then the distance from A to B is defined as positive infinity. Find an algorithm which input is a pair of nodes x, y and the output is the distance from component A to the component B, where $x \in A$, $y \in B$. What is the asymptotic complexity of the algorithm?

14. Suppose that a directed graph is represented by an unordered list of edges and no other representation is available and no other representation can be created additionally. We have to produce a new graph which represents the condensation of the original one. You may freely choose the representation of the new graph. What would be the asymptotic complexity of your algorithm?

15. A weakly connected directed graph G with *n* nodes and *m* edges contains *c*1 roots and *c*2 leaves. The constants *n*, *c*1 and *c*2 are fixed, the number of edges *m* may vary. How can you choose the value of *m* to be certain that G is acyclic? What is the maximum possible value of *m* depending on *n*, *c*1 and *c*2?