A1. Huawei Graph Mining Challenge

time limit per test: 15 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Given an undirected graph $G = (V, E)$, where V is the set of graph nodes, and E is the set of graph edges. *Non-overlapping community detection* for the graph $G = (V, E)$ is the task of the partitioning of the set V of all graph nodes into $n \geq 1$ mutually disjoint non-empty sets called communities,

$$C_1, C_2, \ldots, C_n \subseteq V,$$

whose union is V. In other words,

1. all communities must be non-empty: $C_k \neq \varnothing \quad (1 \leq k \leq n)$;
2. communities must be pairwise disjoint: $C_i \cap C_j = \varnothing \quad (i \neq j)$;
3. the set of all communities must cover all graph nodes: $C_1 \cup C_2 \cup \ldots \cup C_n = V$.

As a consequence, every graph node belongs to exactly one community.

Naturally, the partitioning can be done in many ways. We will evaluate the quality of partition based on the value of modularity measure.

Modularity measure

Given an undirected graph $G = (V, E)$ without loop edges and its partition into a set of communities, $C = \{C_1, C_2, \ldots, C_n\}$. The quality of the partition will be measured by the modularity Q, defined by the following formula:

$$Q = Modularity(C) + Regularization(C)$$

$$Modularity(C) = \sum_{c \in C} \left(\frac{|E_c^{in}|}{|E|} - \left(\frac{2|E_c^{in}| + |E_c^{out}|}{2|E|} \right)^2 \right) = \sum_{c \in C} \left(\frac{|E_c^{in}|}{|E|} - \left(\frac{\sum_{v \in c} deg(v)}{2|E|} \right)^2 \right),$$

$$Regularization(C) = 0.5 \cdot \left(\frac{1}{n} \sum_{c \in C} density(c) - \frac{1}{|V|/n} \right)$$

$$density(c) = \begin{cases} 1 & \text{if } |c| = 1; \\ \dfrac{|E_c^{in}|}{\frac{1}{2}\,|c|\,(|c|-1)} & \text{if } |c| \geq 2. \end{cases}$$

where:

- C is the set of all the communities, and c ranges over all communities in C;
- $n = |C|$ is the number of communities;
- $|E_c^{in}|$ is the number of edges between two nodes within the community c:
  $E_c^{in} = \left\{(v_i, v_j) \in E : v_i \in c,\ v_j \in c,\ i > j\right\}$;
- $|E_c^{out}|$ is the number of edges between nodes in the community c and nodes outside c:
  $E_c^{out} = \left\{(v_i, v_j) \in E : v_i \in c,\ v_j \notin c\right\}$;
- $|E|$ is the total number of edges in the graph;
- $deg(v)$ is the degree of a graph node v.

It is known that

- $-0.5 \leq Modularity(C) \leq 1$ for all graphs and all partitions;
- for a given graph, the maximum possible value of $Modularity(C)$ may not necessarily reach the value of 1 whatever partition into communities is considered.

Task statement & Competition rules

1. Given three input graphs with different structural properties. You need to provide partitions of these graphs into communities. You can download tests by the link: https://test1.codeforces.com/icpc-challenge-2020-tests.zip. The password is "`64e00d81811bbc463e1e636af`".
2. The contest contains three separate problems: A1, A2 and A3. You should submit solutions for the first given graph to A1, for the second given graph to A2 and for the third given graph to A3. The best (max scored) solution for each problem is counted in the standings.
3. The verification system will compute the scores based on modularity measure for three graphs independently. If modularity measure is Q, then the score is $(Q + 1.0) \cdot 10^5$ (rounded to 3 digits after the decimal point). The final combined score will be defined as sum of scores for three given graphs: $P_{comb} = P_1 + P_2 + P_3$.
4. The ranking will be made based on the value of P_{comb}, and the participant with the highest value of P_{comb} will be the winner of this challenge.
5. In case of a tie between two participants, the one who provides higher individual values of P for two graphs will be the winner. In case of a complex tie, the sponsor will break the tie.

On the complexity of community detection

The number of all possible partitions of an n-element set is known as the *Bell number* B_n. It is known that B_n grows exponentially with n (namely, $\log B_n \sim n \log n$). Thus, it is infeasible to find the best partition of a large graph by brute-force.

It is also known that, in the general case, the task of finding the set of communities with maximum modularity is NP-hard.

We hope that you can find an efficient approximate algorithm for this problem. Many strategies can be used to solve this problem, and an algorithm can have combinatorial, optimization, heuristical, or any other nature. The algorithm itself is not a deliverable; only the final value of modularity is counted.

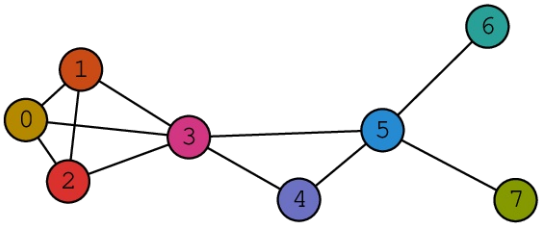
Examples

For a given partition of graph nodes into communities, modularity reflects the concentration of edges within communities compared with a random distribution of edges between all nodes. Thus, from the viewpoint of modularity maximization, it is beneficial to make a graph partition such that

1. communities are dense enough (meaning that the density of edges between graph nodes within the community is higher than the average density of edges in the graph);
2. communities are not too small (otherwise, it might be beneficial to enlarge the community keeping the high density).

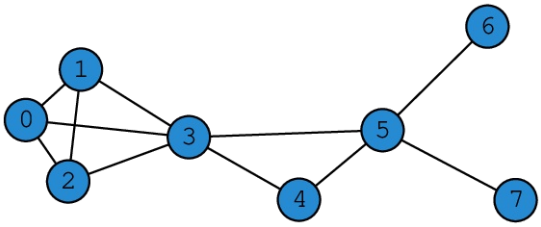
Example 1

In this picture, every graph node forms a singleton community. The modularity for this partition is negative, $Modularity \approx -0.153$.



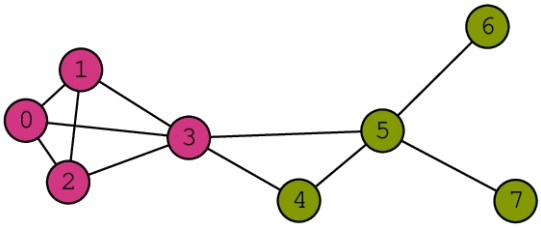
Example 2

In this picture, we have one community containing all graph nodes. The modularity for this partition is zero, $Modularity = 0$.



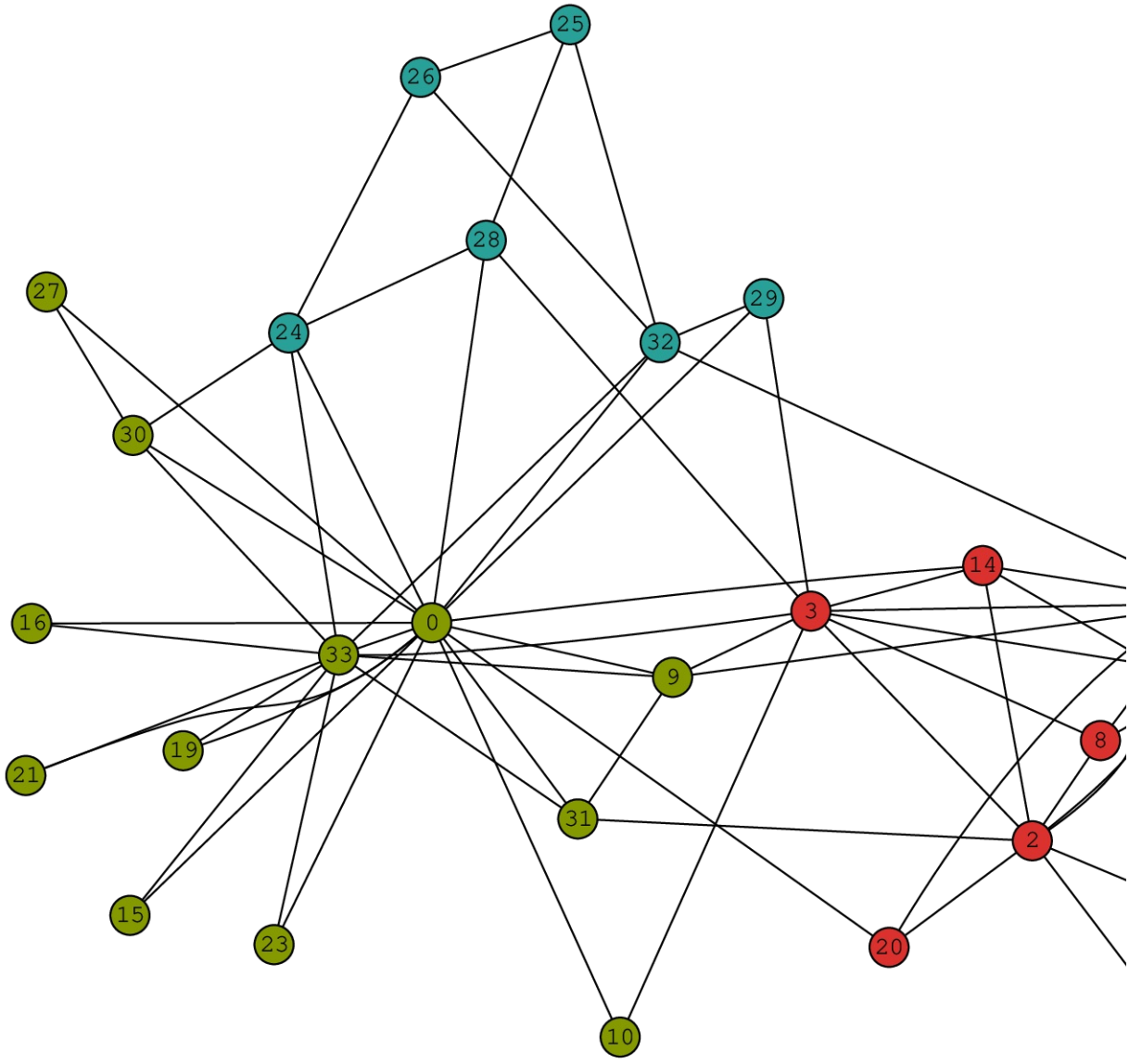
Example 3

In this picture, the graph is partitioned into two communities. The modularity for this partition is $Modularity \approx 0.281$.



Example 4

The picture illustrates a graph with an optimal partition of its nodes. The modularity value for this partition is $Modularity \approx 0.420$.
A higher modularity value is not reachable for this graph.



Input
Every input graph $G = (V, E)$ is defined by the set of its edges. It is given in text file format, where every line represents a graph edge, e.g. the following input defines a graph with 6 nodes and 6 edges:

```
0 1
0 2
0 3
1 2
1 4
3 5
```

We guarantee the following assumptions about the input format:

- The set of all graph nodes is a set of consecutive integers: $V = \{0, 1, 2, \ldots, |V| - 1\}$.
- Every graph node is connected to at least one edge; there are no nodes without incident edges.
- There are no loop edges (v_k, v_k).
- Every graph edge (v_1, v_2) is present in the text file only once, either as (v_1, v_2) or as (v_2, v_1), there are no duplicate edges.
- The graph edges can be listed in any order, i.e. they are not necessarily sorted.

Output

The output should be given in text file format where every line contains the list of all graph nodes of some community, e.g. the following output defines three communities:

```
0 1 2
4 5
3
```

The output will be verified by the verification system, and the modularity measure will be computed for it.

- The order of the communities and the order of graph nodes within a community is not relevant.
- The verification system will check that every graph node is assigned to exactly one community.
  In case some graph node is assigned to more than one community, or some graph node is not present in the output file, the result will be rejected by the verification system.

A2. Huawei Graph Mining Challenge

time limit per test: 15 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Given an undirected graph $G = (V, E)$, where V is the set of graph nodes, and E is the set of graph edges. *Non-overlapping community detection* for the graph $G = (V, E)$ is the task of the partitioning of the set V of all graph nodes into $n \geq 1$ mutually disjoint non-empty sets called communities,

$$C_1, C_2, \ldots, C_n \subseteq V,$$

whose union is V. In other words,

1. all communities must be non-empty: $C_k \neq \varnothing \quad (1 \leq k \leq n)$;
2. communities must be pairwise disjoint: $C_i \cap C_j = \varnothing \quad (i \neq j)$;
3. the set of all communities must cover all graph nodes: $C_1 \cup C_2 \cup \ldots \cup C_n = V$.

As a consequence, every graph node belongs to exactly one community.

Naturally, the partitioning can be done in many ways. We will evaluate the quality of partition based on the value of modularity measure.

Modularity measure

Given an undirected graph $G = (V, E)$ without loop edges and its partition into a set of communities, $C = \{C_1, C_2, \ldots, C_n\}$. The quality of the partition will be measured by the modularity Q, defined by the following formula:

$$Q = Modularity(C) + Regularization(C)$$

$$Modularity(C) = \sum_{c \in C} \left(\frac{|E_c^{in}|}{|E|} - \left(\frac{2|E_c^{in}| + |E_c^{out}|}{2|E|} \right)^2 \right) = \sum_{c \in C} \left(\frac{|E_c^{in}|}{|E|} - \left(\frac{\sum_{v \in c} deg(v)}{2|E|} \right)^2 \right),$$

$$Regularization(C) = 0.5 \cdot \left(\frac{1}{n} \sum_{c \in C} density(c) - \frac{1}{|V|/n} \right)$$

$$density(c) = \begin{cases} 1 & \text{if } |c| = 1; \\ \dfrac{|E_c^{in}|}{\frac{1}{2}\,|c|\,(|c| - 1)} & \text{if } |c| \geq 2. \end{cases}$$

where:

- C is the set of all the communities, and c ranges over all communities in C;
- $n = |C|$ is the number of communities;
- $|E_c^{in}|$ is the number of edges between two nodes within the community c:
  $E_c^{in} = \left\{ (v_i, v_j) \in E : v_i \in c,\ v_j \in c,\ i > j \right\}$;
- $|E_c^{out}|$ is the number of edges between nodes in the community c and nodes outside c:
  $E_c^{out} = \left\{ (v_i, v_j) \in E : v_i \in c,\ v_j \notin c \right\}$;
- $|E|$ is the total number of edges in the graph;
- $deg(v)$ is the degree of a graph node v.

It is known that

- $-0.5 \leq Modularity(C) \leq 1$ for all graphs and all partitions;
- for a given graph, the maximum possible value of $Modularity(C)$ may not necessarily reach the value of 1 whatever partition into communities is considered.

Task statement & Competition rules

1. Given three input graphs with different structural properties. You need to provide partitions of these graphs into communities. You can download tests by the link: https://test1.codeforces.com/icpc-challenge-2020-tests.zip. The password is "`64e00d81811bbc463e1e636af`".
2. The contest contains three separate problems: A1, A2 and A3. You should submit solutions for the first given graph to A1, for the second given graph to A2 and for the third given graph to A3. The best (max scored) solution for each problem is counted in the standings.
3. The verification system will compute the scores based on modularity measure for three graphs independently. If modularity measure is Q, then the score is $(Q + 1.0) \cdot 10^5$ (rounded to 3 digits after the decimal point). The final combined score will be defined as sum of scores for three given graphs: $P_{comb} = P_1 + P_2 + P_3$.
4. The ranking will be made based on the value of P_{comb}, and the participant with the highest value of P_{comb} will be the winner of this challenge.
5. In case of a tie between two participants, the one who provides higher individual values of P for two graphs will be the winner. In case of a complex tie, the sponsor will break the tie.

On the complexity of community detection

The number of all possible partitions of an n-element set is known as the *Bell number* B_n. It is known that B_n grows exponentially with n (namely, $\log B_n \sim n \log n$). Thus, it is infeasible to find the best partition of a large graph by brute-force.

It is also known that, in the general case, the task of finding the set of communities with maximum modularity is NP-hard.

We hope that you can find an efficient approximate algorithm for this problem. Many strategies can be used to solve this problem, and an algorithm can have combinatorial, optimization, heuristical, or any other nature. The algorithm itself is not a deliverable; only the final value of modularity is counted.

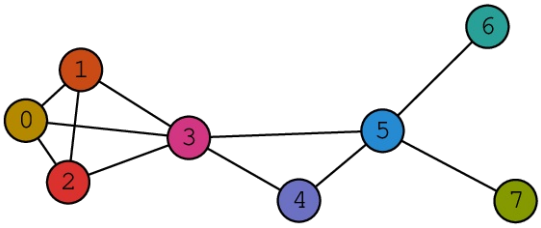
Examples

For a given partition of graph nodes into communities, modularity reflects the concentration of edges within communities compared with a random distribution of edges between all nodes. Thus, from the viewpoint of modularity maximization, it is beneficial to make a graph partition such that

1. communities are dense enough (meaning that the density of edges between graph nodes within the community is higher than the average density of edges in the graph);
2. communities are not too small (otherwise, it might be beneficial to enlarge the community keeping the high density).

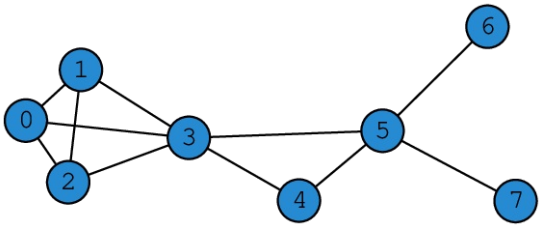
Example 1

In this picture, every graph node forms a singleton community. The modularity for this partition is negative, $Modularity \approx -0.153$.



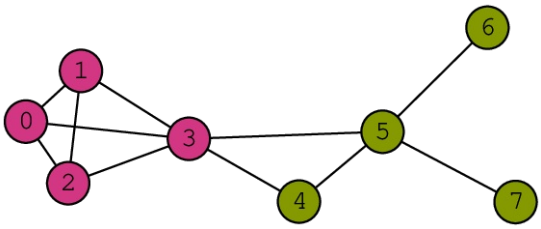
Example 2

In this picture, we have one community containing all graph nodes. The modularity for this partition is zero, $Modularity = 0$.



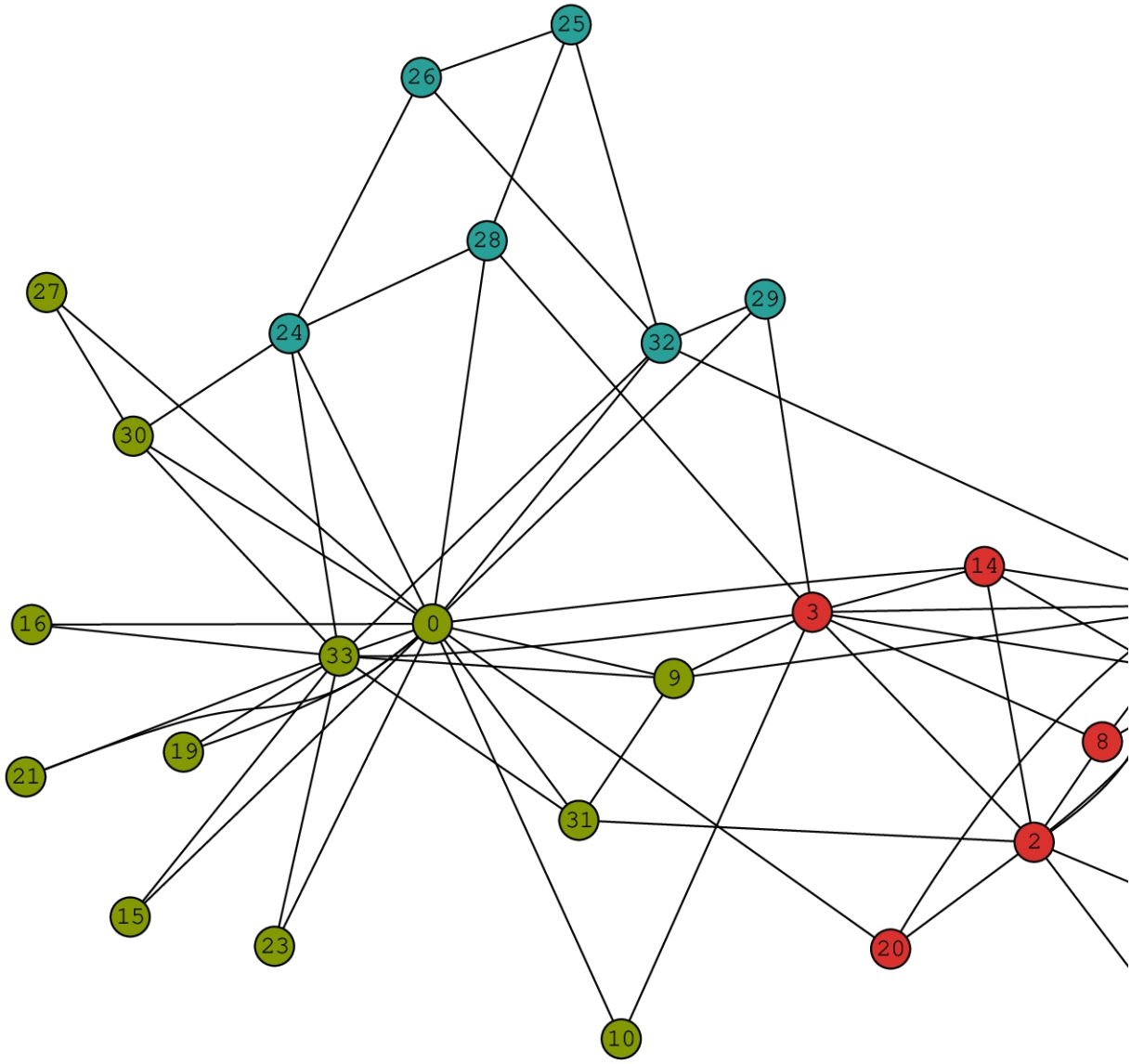
Example 3

In this picture, the graph is partitioned into two communities. The modularity for this partition is $Modularity \approx 0.281$.



Example 4

The picture illustrates a graph with an optimal partition of its nodes. The modularity value for this partition is $Modularity \approx 0.420$. A higher modularity value is not reachable for this graph.



Input
Every input graph $G = (V, E)$ is defined by the set of its edges. It is given in text file format, where every line represents a graph edge, e.g. the following input defines a graph with 6 nodes and 6 edges:

```
0 1
0 2
0 3
1 2
1 4
3 5
```

We guarantee the following assumptions about the input format:

- The set of all graph nodes is a set of consecutive integers: $V = \{0, 1, 2, \ldots, |V| - 1\}$.
- Every graph node is connected to at least one edge; there are no nodes without incident edges.
- There are no loop edges (v_k, v_k).
- Every graph edge (v_1, v_2) is present in the text file only once, either as (v_1, v_2) or as (v_2, v_1), there are no duplicate edges.
- The graph edges can be listed in any order, i.e. they are not necessarily sorted.

Output
The output should be given in text file format where every line contains the list of all graph nodes of some community, e.g. the following output defines three communities:

```
0 1 2
4 5
3
```

The output will be verified by the verification system, and the modularity measure will be computed for it.

- The order of the communities and the order of graph nodes within a community is not relevant.
- The verification system will check that every graph node is assigned to exactly one community.
  In case some graph node is assigned to more than one community, or some graph node is not present in the output file, the result will be rejected by the verification system.

A3. Huawei Graph Mining Challenge

time limit per test: 15 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Given an undirected graph $G = (V, E)$, where V is the set of graph nodes, and E is the set of graph edges. *Non-overlapping community detection* for the graph $G = (V, E)$ is the task of the partitioning of the set V of all graph nodes into $n \geq 1$ mutually disjoint non-empty sets called communities,

$$C_1, C_2, \ldots, C_n \subseteq V,$$

whose union is V. In other words,

1. all communities must be non-empty: $C_k \neq \varnothing \quad (1 \leq k \leq n)$;
2. communities must be pairwise disjoint: $C_i \cap C_j = \varnothing \quad (i \neq j)$;
3. the set of all communities must cover all graph nodes: $C_1 \cup C_2 \cup \ldots \cup C_n = V$.

As a consequence, every graph node belongs to exactly one community.

Naturally, the partitioning can be done in many ways. We will evaluate the quality of partition based on the value of modularity measure.

Modularity measure

Given an undirected graph $G = (V, E)$ without loop edges and its partition into a set of communities, $C = \{C_1, C_2, \ldots, C_n\}$. The quality of the partition will be measured by the modularity Q, defined by the following formula:

$$Q = Modularity(C) + Regularization(C)$$

$$Modularity(C) = \sum_{c \in C} \left(\frac{|E_c^{in}|}{|E|} - \left(\frac{2|E_c^{in}| + |E_c^{out}|}{2|E|} \right)^2 \right) = \sum_{c \in C} \left(\frac{|E_c^{in}|}{|E|} - \left(\frac{\sum_{v \in c} deg(v)}{2|E|} \right)^2 \right),$$

$$Regularization(C) = 0.5 \cdot \left(\frac{1}{n} \sum_{c \in C} density(c) - \frac{1}{|V|/n} \right)$$

$$density(c) = \begin{cases} 1 & \text{if } |c| = 1; \\ \dfrac{|E_c^{in}|}{\frac{1}{2}\,|c|\,(|c|-1)} & \text{if } |c| \geq 2. \end{cases}$$

where:

- C is the set of all the communities, and c ranges over all communities in C;
- $n = |C|$ is the number of communities;
- $|E_c^{in}|$ is the number of edges between two nodes within the community c:
  $E_c^{in} = \left\{(v_i, v_j) \in E : v_i \in c,\ v_j \in c,\ i > j\right\}$;
- $|E_c^{out}|$ is the number of edges between nodes in the community c and nodes outside c:
  $E_c^{out} = \left\{(v_i, v_j) \in E : v_i \in c,\ v_j \notin c\right\}$;
- $|E|$ is the total number of edges in the graph;
- $deg(v)$ is the degree of a graph node v.

It is known that

- $-0.5 \leq Modularity(C) \leq 1$ for all graphs and all partitions;
- for a given graph, the maximum possible value of $Modularity(C)$ may not necessarily reach the value of 1 whatever partition into communities is considered.

Task statement & Competition rules

1. Given three input graphs with different structural properties. You need to provide partitions of these graphs into communities. You can download tests by the link: https://test1.codeforces.com/icpc-challenge-2020-tests.zip. The password is "`64e00d81811bbc463e1e636af`".
2. The contest contains three separate problems: A1, A2 and A3. You should submit solutions for the first given graph to A1, for the second given graph to A2 and for the third given graph to A3. The best (max scored) solution for each problem is counted in the standings.
3. The verification system will compute the scores based on modularity measure for three graphs independently. If modularity measure is Q, then the score is $(Q + 1.0) \cdot 10^5$ (rounded to 3 digits after the decimal point). The final combined score will be defined as sum of scores for three given graphs: $P_{comb} = P_1 + P_2 + P_3$.
4. The ranking will be made based on the value of P_{comb}, and the participant with the highest value of P_{comb} will be the winner of this challenge.
5. In case of a tie between two participants, the one who provides higher individual values of P for two graphs will be the winner. In case of a complex tie, the sponsor will break the tie.

On the complexity of community detection

The number of all possible partitions of an n-element set is known as the *Bell number* B_n. It is known that B_n grows exponentially with n (namely, $\log B_n \sim n \log n$). Thus, it is infeasible to find the best partition of a large graph by brute-force.

It is also known that, in the general case, the task of finding the set of communities with maximum modularity is NP-hard.

We hope that you can find an efficient approximate algorithm for this problem. Many strategies can be used to solve this problem, and an algorithm can have combinatorial, optimization, heuristical, or any other nature. The algorithm itself is not a deliverable; only the final value of modularity is counted.

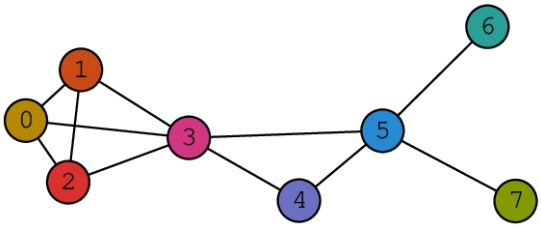
Examples

For a given partition of graph nodes into communities, modularity reflects the concentration of edges within communities compared with a random distribution of edges between all nodes. Thus, from the viewpoint of modularity maximization, it is beneficial to make a graph partition such that

1. communities are dense enough (meaning that the density of edges between graph nodes within the community is higher than the average density of edges in the graph);
2. communities are not too small (otherwise, it might be beneficial to enlarge the community keeping the high density).

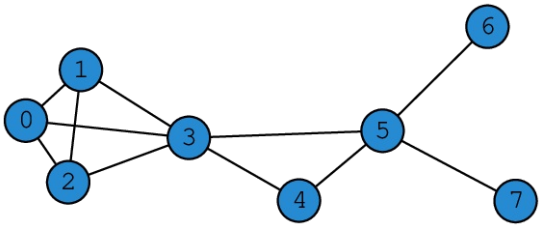
Example 1

In this picture, every graph node forms a singleton community. The modularity for this partition is negative, $Modularity \approx -0.153$.



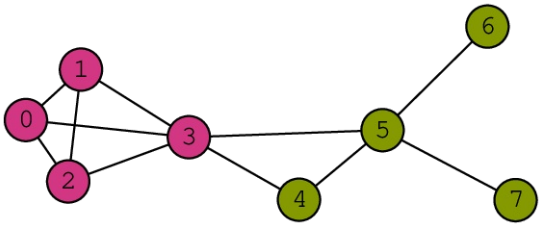
Example 2

In this picture, we have one community containing all graph nodes. The modularity for this partition is zero, $Modularity = 0$.



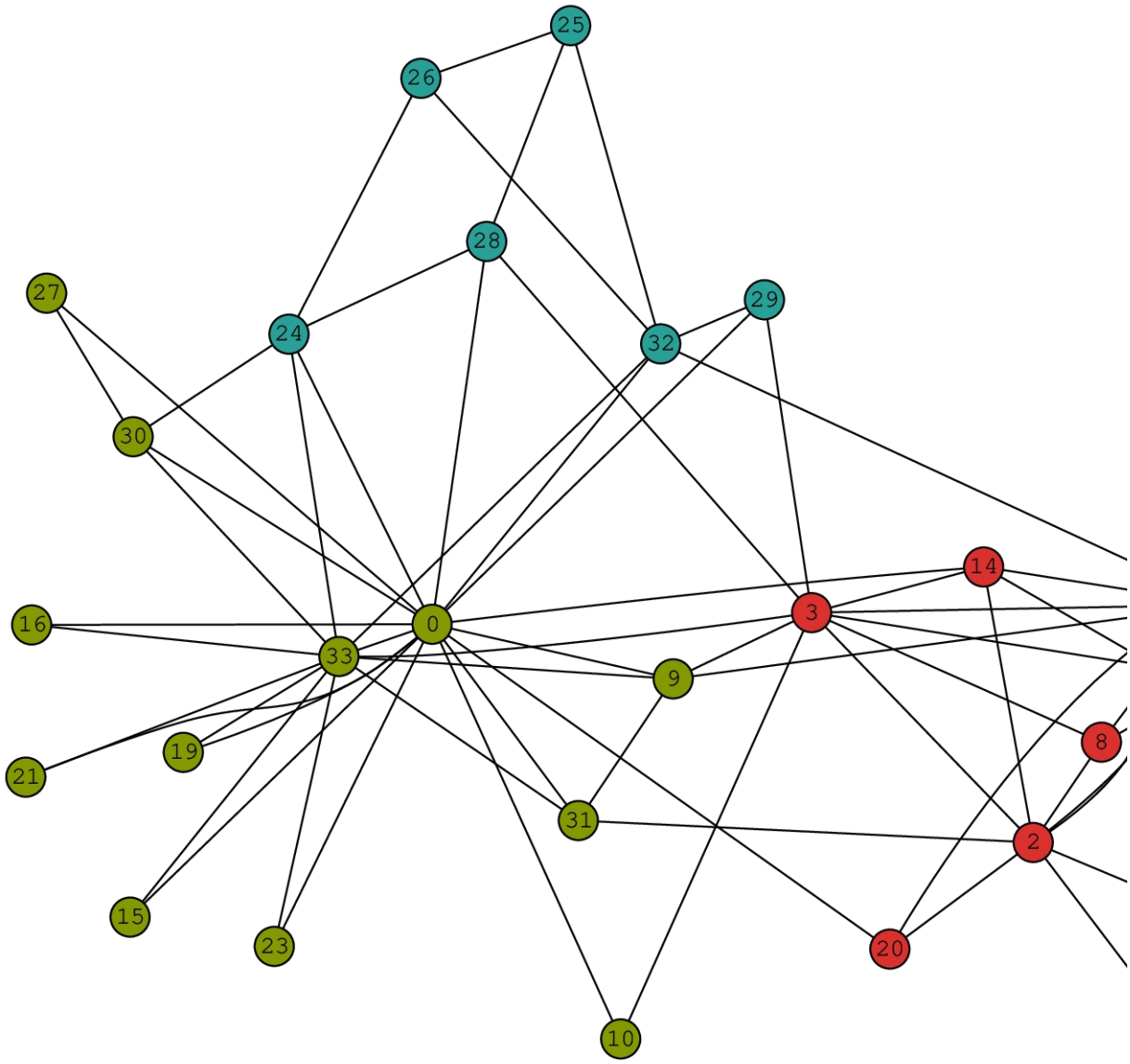
Example 3

In this picture, the graph is partitioned into two communities. The modularity for this partition is $Modularity \approx 0.281$.



Example 4

The picture illustrates a graph with an optimal partition of its nodes. The modularity value for this partition is $Modularity \approx 0.420$. A higher modularity value is not reachable for this graph.



Input

Every input graph $G = (V, E)$ is defined by the set of its edges. It is given in text file format, where every line represents a graph edge, e.g. the following input defines a graph with 6 nodes and 6 edges:

```
0 1
0 2
0 3
1 2
1 4
3 5
```

We guarantee the following assumptions about the input format:

- The set of all graph nodes is a set of consecutive integers: $V = \{0, 1, 2, \ldots, |V| - 1\}$.
- Every graph node is connected to at least one edge; there are no nodes without incident edges.
- There are no loop edges (v_k, v_k).
- Every graph edge (v_1, v_2) is present in the text file only once, either as (v_1, v_2) or as (v_2, v_1), there are no duplicate edges.
- The graph edges can be listed in any order, i.e. they are not necessarily sorted.

Output

The output should be given in text file format where every line contains the list of all graph nodes of some community, e.g. the following output defines three communities:

```
0 1 2
4 5
3
```

The output will be verified by the verification system, and the modularity measure will be computed for it.

- The order of the communities and the order of graph nodes within a community is not relevant.
- The verification system will check that every graph node is assigned to exactly one community.
  In case some graph node is assigned to more than one community, or some graph node is not present in the output file, the result will be rejected by the verification system.

The only programming contests Web 2.0 platform