WiseTasks Graphs System

Zaikov D.G. and Pozdniakov S.N.

Abstract. The paper presents a learning resource WiseTasks Graphs, which allows you to create graph tasks from various modules. The possibilities of the presented system for various target groups

- of students studying mathematics and algorithms and creating implementations of system modules are illustrated;
- students who use the resource for research activities related to self-formulation and problem solving;
- teachers who are interested in creating constructive problems in graph theory, as well as monitoring the work of students on them;
- students and schoolchildren studying graph theory

The work was supported by the RFBR grant No. 19-29-14141

Introduction

The Wise Tasks Graphs system is designed to create and solve self-testable graph tasks. A self-testable task is such a task, the verification of the solution of which occurs not through reconciliation with a previously saved answer, but through reconciliation with a set condition. For example, if the task is "build a graph that is complete", then in order to understand whether such a problem has been solved correctly, the system needs to check the answer graph for completeness, and not compare it with the complete graphs that are stored in the database of answers.

The main formulation of tasks that can be set within the framework of the system: "Construct a graph with the specified properties...". Another formulation for which the system has verification mechanisms is "specify a subgraph of a given graph that has the properties ...". Other formulations are also allowed, related not only to the allocation of a subgraph, but also to the assignment of various types of marks to its elements, for example, the order of passing edges or vertices of the graph, edge weights, coloring of vertices or edges.

Modules that describe various graph properties and various algorithms on graphs are responsible for various graph properties. By combining different modules, you can create different tasks. This approach allows you to quickly build tasks

of a fairly wide class, since to create them you only need to combine different modules. This possibility is provided by the user interface of the “teacher” (note that the concept of “teacher” is conditional here, since the student or student himself has the opportunity to set tasks himself). To introduce numerical characteristics into the problem, in addition to checking for equality with a given number, you can use more and less relations, for example, “build a graph with more than 5 vertices ...”.

1. Task example

Here is an example of a new task:

1. Let’s choose the modules that we want to add to the task, namely “full graph” and “number of vertices" (Figure 1).

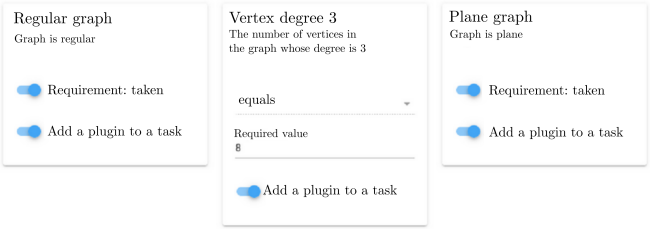

FIGURE 1. Choosing modules

2. Look at the condition that was created according to the selected modules (Figure 2)

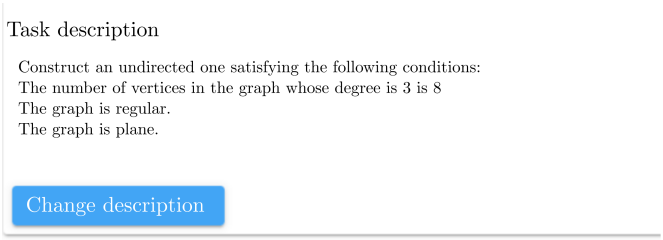

FIGURE 2. Task condition

3. Let's choose the name and category of the task (Figure 3)

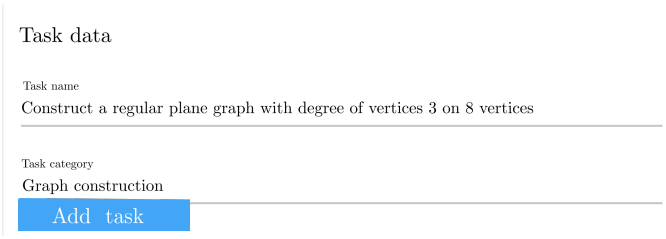

FIGURE 3. Creating task

4. Now let's solve this problem (Figure 4).

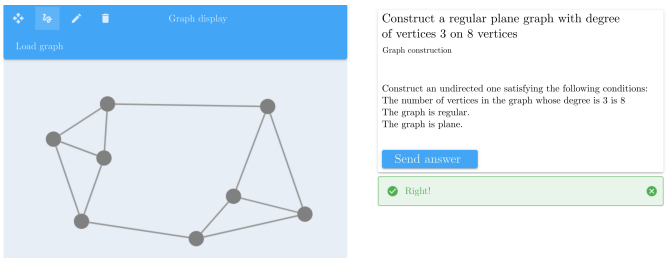

FIGURE 4. Solving task

5. If there are errors in the solution, the system will indicate exactly what the error was (Figure 5)

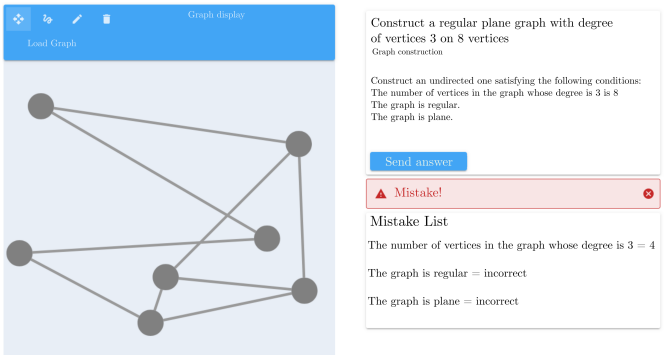

FIGURE 5. Creating mistakes

2. Technical aspects

To work with different types of tasks, the teacher sets the rights for the one who will solve it. In total, there are four types of rights available in the system - adding elements (edges and vertices), setting colors, deleting elements, and setting labels and weights for elements. For each task, you can select a specific set of rights (for example, prohibit deleting and adding elements in a graph, which can be useful in tasks where you want to color an existing graph)

As an example of modules, we will give several properties that are at the same time the names of the modules of the system (Table 1). We distinguish three types of properties: binary properties (whether the graph has the specified properties), numerical characteristics and properties of subgraphs of subgraphs selected in some way.

Binary properties	Numerical characteristics	Marked subgraphs
The constructed graph is a transitive relation graph	Sum of degrees of vertices of an undirected graph	Edges of an undirected graph labeled with natural numbers form an Euler path (traversal in ascending order of labels)
The constructed undirected graph is Euler graph	The clique number ϕ (density) of an undirected graph (the number of vertices in the largest clique)	The marked subset of vertices of an undirected graph is the maximum independent
The constructed undirected graph has a dominant vertex	Vertex connectivity number k (connectivity number) of an undirected graph	The order of traversing the edges of an undirected graph marked by natural numbers is a depth-first traversal (traversal in ascending order of marks)

Table 1. Examples of properties

The system assumes three main types of users:

1. A developer is a user who implements new modules in the system, thereby increasing the number of possible tasks to compile
2. A teacher is a user who creates tasks using modules
3. A student is a user who solves these tasks

3. Pedagogical aspects

Since the system is an open web resource, anyone can be a student, a teacher, and a developer, that is, anyone using the web interface can both create modules and

participate in filling out the task book by graphs, composing new tasks, and can also solve problems that were compiled by other users.

The “Wise Tasks Graphs” system has already been applied in practice among 2nd year students. 100 tasks were compiled for compiling modules (since the system is written in Java, the modules should have been written in it). The training stream of 100 people was divided into teams, each team had a captain (a person responsible for the correctness of the implementation and testing of modules in the team). Each student was asked to choose and write one module. When writing the module, the student learned to apply the algorithms studied in lectures in practice, which is a good way to consolidate knowledge in the course “Combinatorics and graph theory". Also, writing modules provided students with the opportunity to get acquainted with the Java programming language, which is new for students.

Conclusion

1. The Wise Tasks Graphs system provides verification of graph theory tasks without entering answers to the assigned tasks.
2. The Wise Tasks Graphs system provides conditions for educational and research work on the graph theory course: a student can set and solve problems independently.
3. The Wise Tasks Graphs system allows distributed filling not only with tasks, but also with tool modules describing graph properties and algorithms on graphs.
4. The main problem in filling the Wise Tasks Graphs system is the addition of new modules, since errors in the implemented algorithms will lead to incorrect reactions of the system to the tasks compiled in it. At the same time, the addition of new modules is carried out much less frequently than the addition of new tasks, so ensuring the correct operation of the system requires only the support of the module system, but not the task system.
5. The development of systems like Wise Tasks Graphs allows the organization of new forms of project work of students. So, in the process of filling the system with modules, the distributed work of second-year students of ’LETI’ was organized, who studied the graph theory course and added graph algorithm implementations written in Java to the system.

References

[1] V.A. Emelichev, O.I. Melnikov, V.I. Sarvanov, R.I. Tyshkevich - *Lectures on graph theory*: Nauka Gl. ed. physical and Mathematical lit.

[2] A. A. Voronenko, V. S. Fedorova. *Discrete mathematics. Tasks and exercises with solutions*: An educational and methodical manual.

[3] Pozdniakov S.N., Tolkacheva E.A., Zaikov D.G., Chukhnov A.S. - *Productive learning technologies in creating and using Wise Tasks type resources*

Zaikov D.G.
Saint Petersburg Electrotechnical University 'LETI'
St. Petersburg, Russia
e-mail: dima38091@gmail.com

Pozdniakov S.N.
Saint Petersburg Electrotechnical University 'LETI'
St. Petersburg, Russia
e-mail: pozdnkov@gmail.com