Federal State Autonomous Educational Institution of Higher Education "Moscow Institute of Physics and Technology (National Research University)"

APPROVED
Head of the Phystech School of Applied Mathematics and Informatics
A.M. Raygorodskiy

Work program of the course (training module)

course: Data Structures and Algorithms II/Структуры данных и алгоритмы II
major: Applied Mathematics and Informatics
specialization: Computer Science/Информатика
Phystech School of Applied Mathematics and Informatics
Chair of Algorithms and Programming Technologies
term: 2
qualification: Bachelor

Semester, form of interim assessment: 3 (fall) - Grading test

Academic hours: 60 AH in total, including:
lectures: 30 AH.
seminars: 0 AH.
laboratory practical: 30 AH.

Independent work: 120 AH.

In total: 180 AH, credits in total: 4

Author of the program: V.V. Yakovlev, candidate of physics and mathematical sciences, head of chair

The program was discussed at the Chair of Algorithms and Programming Technologies 21.05.2020

Annotation

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output. Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language. The course studies classical algorithms on sorting, data access for various structures, and generic algorithms complexity.

1. Study objective

Purpose of the course

- form an understanding of the various computational problems in graph theory and the asymptotic complexity of their solutions;
- provide theoretical and practical knowledge about algorithms and data structures of graph theory with proof of the correctness of their work, about methods for assessing the complexity of algorithms.

Tasks of the course

- teach to formulate tasks in terms of theories studied, choose the appropriate algorithm for the task;
- teach you how to develop combinations of algorithms to solve problems, evaluate the complexity of algorithms, their modifications and combinations, including using depreciation analysis, select the appropriate data structures for the tasks, implement the algorithms in a generalized form in the C ++ programming language.

2. List of the planned results of the course (training module), correlated with the planned results of the mastering the educational program

Mastering the discipline is aimed at the formation of the following competencies:

Code and the name of the competence	Competency indicators
Gen.Pro.C-1 Apply fundamental knowledge of physics, mathematics, and/or natural sciences in professional settings	Gen.Pro.C-1.1 Analyze the task in hand, develop appproaches to complete it

3. List of the planned results of the course (training module)

As a result of studying the course the student should:

know:

- Graph algorithms and data structures associated with them;
- estimates of the complexity of standard algorithms;
- standard graph algorithms and data structures used, approaches to the modification of classical algorithms;
- a variety of classical problems in graph theory and the asymptotic complexity of their solutions.

be able to:

- Formulate tasks in terms of theories studied, choose the appropriate algorithm for the task;
- develop combinations of algorithms to solve the problem;
- evaluate the complexity of algorithms, their modifications and combinations, including using depreciation analysis;
- select the appropriate data structures for a specific task;
- implement the algorithm in a generalized form in the c ++ programming language;
- implement standard graph algorithms and data structures in the C ++ programming language.

master:

- Methods of decomposition of tasks in the field of information technology and the construction of a single solution using the studied algorithms;
- methods for assessing the complexity of algorithms, their modifications and combinations.

4. Content of the course (training module), structured by topics (sections), indicating the number of allocated academic hours and types of training sessions

4.1. The sections of the course (training module) and the complexity of the types of training sessions

№	Topic (section) of the course	Types of training sessions, including independent work			
		Lectures	Seminars	Laboratory practical	Independent work
1	Graph traversals	6		6	20
2	Shortest paths in a weighted graph	6		6	25
3	Spanning trees	6		6	25
4	Network streams	6		6	25
5	Segment Data Structures	6		6	25
AH in total		30		30	120
Exam preparation		0 AH.			
Total complexity		180 AH., credits in total 4			

4.2. Content of the course (training module), structured by topics (sections)

Semester: 3 (Fall)

1. Graph traversals

• Oriented graph, pseudograph. Undirected graph, pseudograph.
• Connectivity in neor. graph, connected components.
Weak and strong connectivity in op. column. Components of weak, strong connectivity.
• Walking in depth. The colors of the vertices. Entry and exit times. The white path lemma.
• Check connectivity of an undirected graph.
• Search for a loop in an undirected and oriented graph.
• Topological sorting.
• Finding components of strong connectivity. Kosarayu algorithm. Tarjan's algorithm.
• Components of rib biconnectedness. Bridges. Search for bridges.
• Components of vertex biconnectedness. Articulation points. Search for articulation points.
• Wave algorithm. Traversal in width (application of the queue in the wave algorithm).
• A criterion for the existence of the Euler path and cycle in an oriented and non-oriented graph. Search for the Euler path and cycle.

2. Shortest paths in a weighted graph

• Dijkstra's algorithm.
• The colors of the vertices. The tree of shortest paths.
• Potentials. The applicability condition for the Dijkstra algorithm for modified edge lengths. The potential $\pi(v) = \rho(v, t)$.
• Algorithm A *. Monotonicity condition on heuristics. Examples of heuristics.
• Two-way Dijkstra algorithm.
• Ford-Bellman Algorithm.
• Storage in the matrix: Dvk is equal to the length of the shortest path to the vertex v for exactly k edges (no more than k edges). Proof of correctness. Estimated work time.
• Restore the path.
• Detection of a negative weight cycle. Search for the cycle itself.
• Finding the shortest paths taking into account negative weight cycles.
• Floyd's algorithm. Evidence. Restoring the path.
• Finding a negative weight cycle.
• Johnson's algorithm. Adding a dummy root and dummy edges to run the Ford-Bellman algorithm.

3. Spanning trees

• Spanning tree. Build with a walk in depth and in width.
• Definition of a minimum spanning tree.
• Sectional theorem. Evidence.
• Prim's algorithm. Analogy with Dijkstra's algorithm.
• Proof using the cut theorem. Estimated runtime for various priority queue implementations: binary heap, Fibonacci heap (the latter without proof).
• Kruskal algorithm. Evidence. Estimated work time.
• A system of disjoint sets. Heuristic potentials with proof of an estimate of the run time.
• Heuristic compression paths without proof.
• Boruvka Algorithm. Evidence. Estimated work time.
• Approximation of the solution of the traveling salesman problem using a minimum spanning tree.

4. Network streams

• Network definition. Flow definition.
• Physical meaning. An analogy with the laws of Kirchhoff.
• Definition of section. The concepts of flow through a cut.
• Proof of the fact that the flow through any section is the same.
• The concept of a residual network. The concept of a complementary path.
• The need for a lack of a complementary path for maximum flow.
• Ford-Fulkerson theorem.
• Ford-Fulkerson Algorithm. Search for the minimum cut.
• An example of an integer network in which the algorithm runs for a long time.
• Edmonds-Karp algorithm.
• Proof that the shortest distance in a residual network does not decrease.
• General estimate of the running time of the Edmonds-Karp algorithm.
• Layered network. Algorithm Dinitsa.
• Matching. Maximum matching. The greatest matching. Perfect match.
• The task of finding the greatest matching. Examples of real problems.
• Alternating path. Berg's Lemma.
• The greatest matching in a bipartite graph. Rating.

5. Segment Data Structures

• RSQ and RMQ.
• Rarely-table.
• Segment tree.
• Processing requests from leaves.
• Processing requests from the root.
• Changing the values in the array, updating the tree of segments.
• Multiple operations.
• Fenwick tree.
• LCA. Binary lifting method.
• Reducing LCA to RMQ problem.
• Reducing RMQ to an LCA problem.
• Cartesian tree by implicit key.
• Multiple operations in a Cartesian tree using an implicit key.

5. Description of the material and technical facilities that are necessary for the implementation of the educational process of the course (training module)

- A projector with the ability to connect via HDMI and / or VGA);
- blackboard with chalk or whiteboard with felt-tip pens;
- computer class equipped with a PC.

6. List of the main and additional literature, that is necessary for the course (training module) mastering

Main literature

Additional literature

7. List of web resources that are necessary for the course (training module) mastering

1. SimonPeytonJones (editor), Haskell 98 LanguageandLibraries (TheRevisedReport)
2. JohnHughes, IntroductiontoProgramminginHaskell, www.cs.chalmers.se/~rjmh
3. PaulHudak, JohnPeterson, Joseph H. Fasel ,A GentleIntroductiontoHaskell 98
4. CordeliaHall, JohnHugs, ThelittleHaskeller
5. PhilipWadler, Monadsforfunctionalprogramming, DepartmentofComputingScience, UniversityofGlasgow
6. AllAboutMonads , http://www.nomaware.com/monads/html/
7. EmeryBerger , FP + OOP = Haskell, DepartmentofComputerScience, TheUniversityofTexasatAustin
8. RexPage, TwoDozenShortLessonsinHaskell, a participatorytextbookonfunctionalprogramming, SchoolofComputerScience, UniversityofOklahoma
9. DamirMedak, GerhardNavratil, Haskell-Tutorial, InstituteforGeoinformationTechnicalUniversityVienna
10. Haskell, http://haskell.org.

8. List of information technologies used for implementation of the educational process, including a list of software and information reference systems (if necessary)

Software for developing and debugging programs in the C ++ programming language.

9. Guidelines for students to master the course

A student studying a discipline must, on the one hand, master the general conceptual apparatus, and on the other hand, must learn to put theoretical knowledge into practice.

As a result of studying the discipline, the student must know the basic definitions, concepts, axioms.

Successful development of the course requires intense independent work of the student. The course program provides the minimum necessary time for the student to work on the topic. Independent work includes:

- reading and taking notes of recommended literature;

- study of educational material (according to lecture notes, educational and scientific literature), preparation of answers to questions intended for independent study, proof of individual statements, properties.

The management and control of the student's independent work is carried out in the form of individual consultations.

It is important to gain an understanding of the material being studied, and not its mechanical memorization. If it is difficult to study individual topics, questions, you should consult a lecturer.

SUPPLEMENT

Assessment funds for course (training module)

major:	Applied Mathematics and Informatics
specialization:	Computer Science/Информатика
	Phystech School of Applied Mathematics and Informatics
	Chair of Algorithms and Programming Technologies
term:	2
qualification:	Bachelor

Semester, form of interim assessment: 3 (fall) - Grading test

Author: V.V. Yakovlev, candidate of physics and mathematical sciences, head of chair

1. Competencies formed during the process of studying the course

Code and the name of the competence	Competency indicators
Gen.Pro.C-1 Apply fundamental knowledge of physics, mathematics, and/or natural sciences in professional settings	Gen.Pro.C-1.1 Analyze the task in hand, develop appproaches to complete it

2. Competency assessment indicators

As a result of studying the course the student should:

know:

- Graph algorithms and data structures associated with them;
- estimates of the complexity of standard algorithms;
- standard graph algorithms and data structures used, approaches to the modification of classical algorithms;
- a variety of classical problems in graph theory and the asymptotic complexity of their solutions.

be able to:

- Formulate tasks in terms of theories studied, choose the appropriate algorithm for the task;
- develop combinations of algorithms to solve the problem;
- evaluate the complexity of algorithms, their modifications and combinations, including using depreciation analysis;
- select the appropriate data structures for a specific task;
- implement the algorithm in a generalized form in the c ++ programming language;
- implement standard graph algorithms and data structures in the C ++ programming language.

master:

- Methods of decomposition of tasks in the field of information technology and the construction of a single solution using the studied algorithms;
- methods for assessing the complexity of algorithms, their modifications and combinations.

3. List of typical control tasks used to evaluate knowledge and skills

Not provided.

4. Evaluation criteria

1. What an exception. What does the term try with resources.
2. Describe the CSV format.
3. What is a generic? When is it worth using?
4. Tell us about the work of GarbageCollector.
5. List the methods of the Object class.
6. Tell us in detail about the method of the public final native class getClass ().
7. Tell us more about the method public native int hashCode ().
8. Tell us more about the public boolean equals (Object obj) method.
9. Tell us more about the protected native Object clone () method that throws a CloneNotSupportedException.
10. Tell us more about the public String toString () method.
11. Tell us in detail about the protected void finalize () method throws Throwable.
12. Expand the concepts of hash and hash functions.
13. Describe the main ideas of the Model-View-Controller.

excellent

10 comprehensive, systematized, deep knowledge of the curriculum of the discipline and the ability to confidently apply them in practice when solving specific problems, free and correct justification of decisions made;

9 systematic, deep knowledge of the curriculum of the discipline and the ability to confidently apply them in practice when solving specific problems, the correct justification of decisions made;

8 deep knowledge of the curriculum of the discipline and the ability to apply them in practice when solving specific problems, the correct justification of decisions made;

good

7 firmly knows the material, correctly and essentially sets out it, knows how to apply the knowledge gained in practice, but admits some inaccuracies in the answer or in solving problems;

6 knows the material, correctly presents it, knows how to apply the acquired knowledge in practice, but admits some inaccuracies in the answer or in solving problems;

5 knows the basic material, correctly presents it, knows how to apply the knowledge gained in practice, but admits inaccuracy in the answer or in solving problems;

satisfactorily

4 fragmented, fragmented nature of knowledge, insufficiently correct wording of basic concepts, violation of logical sequence in the presentation of program material, but at the same time he owns the main sections of the curriculum necessary for further training and can apply the acquired knowledge in the standard situation;

3 the nature of knowledge is sufficient for further training and can apply the acquired knowledge on the model in a standard situation;

unsatisfactory

2 does not know most of the main content of the curriculum of the discipline, makes gross errors in the wording of the basic concepts of the discipline and does not know how to correctly use the knowledge gained in solving typical practical problems.

1 does not know the wording of the basic concepts of the discipline and does not know how to use the knowledge gained in solving typical practical problems.

5. Methodological materials defining the procedures for the assessment of knowledge, skills, abilities and/or experience

The test time is 2 academic hours. During the test, students can use the discipline program and the source code.