English for Computer Science

Mohammad Farshi

Department of Computer Science,
Yazd University

1391-1392

Konkoor CS Arshad 92

Passage 1

The Pumping Lemma: If a language is regular, then every sufficiently long string in the language has a nonempty substring that can be "pumped," that is, repeated any number of times while the resulting strings are also in the language. This fact can be used to prove that many different languages are not regular.
Operations That Preserve the Property of Being a Regular Language: There are many operations that, when applied to regular languages, yield a regular language as a result. Among these are union, concatenation, closure, intersection, complementation, difference, reversal, homomorphism (replacement of each symbol by an associated string), and inverse homomorphism.

Passage 1

Testing Emptiness of Regular Languages: There is an algorithm that, given a representation of a regular language, such as an automaton or regular expression, tells whether or not the represented language is the empty set.
Testing Membership in a Regular Language: There is an algorithm that, given a string and a representation of a regular language, tells whether or not the string is in the language.
Testing Distinguishability of States: Two states of a DFA are distinguishable if there is an input string that takes exactly one of the two states to an accepting state. By starting with only the fact that pairs consisting of one accepting and one non-accepting state are distinguishable, and trying to discover additional pairs of distinguishable states by finding pairs whose successors on one input symbol are distinguishable, we can discover all pairs of distinguishable states.

16. The problem whether a given string is in a regular language · · · .

1) can be decided 2) is NP-Hard 3) is NP-complete 4) is undecidable

17- A regular language · · · .

1) has an empty set representation
2) can be represented by an automation
3) can always be represented by a regular expression
4) is always tested and shown to be an empty set

18- To prove that a language is not regular, · · · .

1) the pumping lemma can effectively be used
2) many other languages need to be proved not to be regular
3) both homomorphism and inverse homomorphism are needed
4) distinguishable states for the language being empty or nonempty are to be established in a DFA

16. The problem whether a given string is in a regular language · · · .

1) can be decided 2) is NP-Hard 3) is NP-complete 4) is undecidable

17- A regular language · · · .

1) has an empty set representation
2) can be represented by an automation
3) can always be represented by a regular expression
4) is always tested and shown to be an empty set

18- To prove that a language is not regular, · · · .

1) the pumping lemma can effectively be used
2) many other languages need to be proved not to be regular
3) both homomorphism and inverse homomorphism are needed
4) distinguishable states for the language being empty or nonempty are to be established in a DFA

16. The problem whether a given string is in a regular language · · · .

1) can be decided 2) is NP-Hard 3) is NP-complete 4) is undecidable

17- A regular language · · · .

1) has an empty set representation
2) can be represented by an automation
3) can always be represented by a regular expression
4) is always tested and shown to be an empty set

18- To prove that a language is not regular, · · · .

1) the pumping lemma can effectively be used
2) many other languages need to be proved not to be regular
3) both homomorphism and inverse homomorphism are needed
4) distinguishable states for the language being empty or nonempty are to be established in a DFA

19- Replacement of symbols by their associated things · · · results in a regular language.

1) in any language
2) in regular languages
3) being called concatenation
4) along with application of the Pumping Lemma

20- A regular language · · · .

1) is always nonempty
2) can be decided to be empty or not
3) is decided to be empty if the empty string is in the language
4) can be represented by a regular expression but not by an automation

19- Replacement of symbols by their associated things ··· results in a regular language.

1) in any language
2) in regular languages
3) being called concatenation
4) along with application of the Pumping Lemma

20- A regular language ··· .

1) is always nonempty
2) can be decided to be empty or not
3) is decided to be empty if the empty string is in the language
4) can be represented by a regular expression but not by an automation

Passage 2

Layering, or layered architecture, is a form of hierarchical modularity that is central to data network design. The concept of modularity (although perhaps not the name) is as old as engineering. In what follows, the word module is used to refer either to a device or to a process within some computer system. What is important is that the module performs a given function in support of the overall function of the system. Such a function is often called the service provided by the module. The designers of a module will be intensely aware of the internal details and operation of that module. Someone who uses that module as a component in a larger system, however, will treat the module as a "black box." That is, the user will be uninterested in the internal workings of the module and will be concerned only with the inputs, the outputs, and, most important, the functional relation of outputs to inputs (i.e. the service provided). Thus, a black box is a module viewed in terms of its input-output description. It can be used with other black boxes to construct a more complex module, which again will be viewed at higher levels as a bigger black box.

Passage 2

This approach to design leads naturally to a hierarchy of modules in which a module appears as a black box at one layer of the hierarchy, but appears as a system of lower-layer black boxes at the next lower layer of the hierarchy (see Fig. 1.6). At the overall system level (i.e., at the highest layer of the hierarchy), one sees a small collection of top-layer modules, each viewed as black boxes providing some clear-cut service. At the next layer down, each top layer module is viewed as a subsystem of lower-layer black boxes, and so forth, down to the lowest layer of the hierarchy. As shown in Fig. 1.6, each layer might contain not only black boxes made up of lower-layer modules but also simple modules that do not require division into yet simpler modules.

As an example of this hierarchical viewpoint, a computer system could be viewed as a set of processor modules, a set of memory modules, and a bus module. A processor module could, in tum, be viewed as a control unit, an arithmetic unit, an instruction fetching unit, and an input-output unit. Similarly, the arithmetic unit could be broken into adders, accumulators, and so on.

Passage 2

In most cases, a user of a black box does not need to know the detailed response of outputs to inputs. For example, precisely when an output changes in response to an input is not important as long as the output has changed by the time it is to be used. Thus, modules (i.e., black boxes) can be specified in terms of tolerances rather than exact descriptions. This leads to standardized modules, which leads, in tum, to the possibility of using many identical, previously designed (i.e., off-the-shelf) modules in the same system. In addition, such standardized modules can easily be replaced with new, functionally equivalent modules that are cheaper or more reliable.

21. Designers of modules · · · .

1) have no need to know the specific functions of the modules
2) treat the modules as "black boxes" having no specific function
3) make use of computers to build the layers of modules
4) are aware of the details of the service being intended

22. Once a module is constructed within a computer system, it can be replaced by a new one

1) only when it is disabled
2) that is cheaper even though not reliable
3) that is more dependable even though more expensive
4) only when a new definition of service is specified

23. The notion of language is · · · .

1) too old to be used for module construction
2) can not be effective in setting up a service
3) fundamental to construction of modules
4) not appropriately used for modules

21. Designers of modules · · · .

1) have no need to know the specific functions of the modules
2) treat the modules as "black boxes" having no specific function
3) make use of computers to build the layers of modules
4) are aware of the details of the service being intended

22. Once a module is constructed within a computer system, it can be replaced by a new one

1) only when it is disabled
2) that is cheaper even though not reliable
3) that is more dependable even though more expensive
4) only when a new definition of service is specified

23. The notion of language is · · · .

1) too old to be used for module construction
2) can not be effective in setting up a service
3) fundamental to construction of modules
4) not appropriately used for modules

21. Designers of modules · · · .

1) have no need to know the specific functions of the modules
2) treat the modules as "black boxes" having no specific function
3) make use of computers to build the layers of modules
4) are aware of the details of the service being intended

22. Once a module is constructed within a computer system, it can be replaced by a new one

1) only when it is disabled
2) that is cheaper even though not reliable
3) that is more dependable even though more expensive
4) only when a new definition of service is specified

23. The notion of language is · · · .

1) too old to be used for module construction
2) can not be effective in setting up a service
3) fundamental to construction of modules
4) not appropriately used for modules

24. The user of a module is · · · .

1) intensely interested in the outcome of modules application to the input
2) not interested to know the input-output relations
3) not interested in the outputs as much as in the inputs
4) interested to know the details of the constructive features of the module

25. A computer system is considered to be · · · .

1) a hierarchical module
2) composed of a set of single-layered modules
3) unjustifiably named a module
4) an inappropriate example of a module

24. The user of a module is · · · .

1) intensely interested in the outcome of modules application to the input
2) not interested to know the input-output relations
3) not interested in the outputs as much as in the inputs
4) interested to know the details of the constructive features of the module

25. A computer system is considered to be · · · .

1) a hierarchical module
2) composed of a set of single-layered modules
3) unjustifiably named a module
4) an inappropriate example of a module

Passage 3

GEOMETRIC ALGORITHMS are a collection of methods for solving problems involving points and lines (and other simple geometric objects) that have only recently come into use. We consider algorithms for finding the convex hull of a set of points, for finding intersections among geometric objects, for solving closest point problems, and for multidimensional searching. Many of these methods nicely complement more elementary sorting and searching methods.
GRAPH ALGORITHMS are useful for a variety of difficult and important problems. A general strategy for searching in graphs is developed and applied to fundamental connectivity problems, including shortest path, minimum spanning tree, network flow, and matching. A unified treatment of these algorithms shows that they are all based on the same procedure, and this procedure depends on a basic data structure developed earlier.

Passage 3

MATHEMATICAL ALGORITHMS include fundamental methods from arithmetic and numerical analysis. We study methods for arithmetic with integers, polynomials, and matrices as well as algorithms for solving a variety of mathematical problems that arise in many contexts: random number generation, solution of simultaneous equations, data fitting, and integration. The emphasis is on algorithmic aspects of the methods, not the mathematical basis.
ADVANCED TOPICS are discussed for the purpose of relating the material in the book to several other advanced fields of study. Special-purpose hardware, dynamic programming, linear programming, exhaustive search, and NP-completeness are surveyed from an elementary viewpoint to give the reader some appreciation for the interesting advanced fields of study suggested by the elementary problems confronted in this book.

Passage 3

The study of algorithms is interesting because it is a new field (almost all of the algorithms we will study are less than twenty-five years old) with a rich tradition (a few algorithms have been known for thousands of years). New discoveries are constantly being made, and few algorithms are completely understood. In this book we will consider intricate, complicated, and difficult algorithms as well as elegant, simple, and easy algorithms. Our challenge is to understand the former and appreciate the latter in the context of many different potential applications. In doing so, we will explore a variety of useful tools and develop a way of "algorithmic thinking" that will serve us well in computational challenges to come.

26. Geometric algorithms · · · sorting and searching.

1) are developed to eliminate the need for
2) are not used for complicated
3) lead to simple methods of
4) may be useful in

27. Mathematical algorithms are concerned with · · · .

1) problems involving only integer data
2) diverse problems
3) basic mathematical arithmetic only
4) proofs for mathematical results

28. A procedure based on a fundamental data structure · · · .

1) serves to unify all graph algorithms
2) is inappropriate for various graph algorithms
3) is the only method of use in graph algorithms
4) is used to study various graph algorithms coherently

26. Geometric algorithms · · · sorting and searching.

1) are developed to eliminate the need for
2) are not used for complicated
3) lead to simple methods of
4) may be useful in

27. Mathematical algorithms are concerned with · · · .

1) problems involving only integer data
2) diverse problems
3) basic mathematical arithmetic only
4) proofs for mathematical results

28. A procedure based on a fundamental data structure · · · .

1) serves to unify all graph algorithms
2) is inappropriate for various graph algorithms
3) is the only method of use in graph algorithms
4) is used to study various graph algorithms coherently

26. Geometric algorithms · · · sorting and searching.

1) are developed to eliminate the need for
2) are not used for complicated
3) lead to simple methods of
4) may be useful in

27. Mathematical algorithms are concerned with · · · .

1) problems involving only integer data
2) diverse problems
3) basic mathematical arithmetic only
4) proofs for mathematical results

28. A procedure based on a fundamental data structure · · · .

1) serves to unify all graph algorithms
2) is inappropriate for various graph algorithms
3) is the only method of use in graph algorithms
4) is used to study various graph algorithms coherently

29. Advanced topics · · · problems.

1) include only theoretical
2) include useful but not interesting
3) are treated as elementary but interesting
4) are shown to be merely elementary

30. Algorithmic thinking is developed by · · · .

1) appreciation of complicated algorithms
2) the study of elegant algorithms
3) understanding the mathematical basics of the problems being concerned
4) learning complicated algorithms as well as appreciating simple algorithms

29. Advanced topics · · · problems.

1) include only theoretical
2) include useful but not interesting
3) are treated as elementary but interesting
4) are shown to be merely elementary

30. Algorithmic thinking is developed by · · · .

1) appreciation of complicated algorithms
2) the study of elegant algorithms
3) understanding the mathematical basics of the problems being concerned
4) learning complicated algorithms as well as appreciating simple algorithms

END.

16	1
17	2
18	1
19	2
20	2
21	4
22	3
23	3
24	1
25	1
26	4
27	2
28	4
29	3
30	4