CPSC-59700 Research in computer science
Spring 2017
January 28th , 2017
Ekta Patel

Inquiry 1

Algorithm analysis:

Analysis of Algorithms is as simple as the name suggests: Analysis of Algorithms. Proving theorems for algorithms such as time-complexity, space-complexity, etc., and theorems relating to the problems an algorithm is attempting to solve or approximate.

Algorithm analysis is an important part of a broader computational complexity theory , which provides theoretical estimates for the resources needed by any algorithm which solves a given computational problem These estimates provide an insight into reasonable directions of search for efficient algorithms.

In computer science , the analysis of algorithms is the determination of the amount of resources (such as time and storage) necessary to execute them. Most algorithms are designed to work with inputs of arbitrary length. Usually, the efficiency or running time of an algorithm is stated as a function relating the input length to the number of steps (time complexity) or storage locations (space complexity).

In theoretical analysis of algorithms it is common to estimate their complexity in the asymptotic sense, i.e., to estimate the complexity function for arbitrarily large input. Big O notation, Big-omega notation and Big-theta notation are used to this end.

For instance, binary search is said to run in a number of steps proportional to the logarithm of the length of the sorted list being searched, or in O(log(n)), colloquially "in logarithmic time". Usually asymptotic estimates are used because different implementations of the same algorithm may differ in efficiency.

However the efficiencies of any two "reasonable" implementations of a given algorithm are related by a constant multiplicative factor called a *hidden constant*.

A complete analysis of the running time of an algorithm involves the following steps:

- Implement the algorithm completely.
- Determine the time required for each basic operation.
- Identify unknown quantities that can be used to describe the frequency of execution of the basic operations.
- Develop a realistic model for the input to the program.
- Analyze the unknown quantities, assuming the modelled input.
- Calculate the total running time by multiplying the time by the frequency for each operation, then adding all the products.

- Time complexity:

the **time complexity** of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input. The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. When expressed this way, the time complexity is said to be described *asymptotically*, i.e., as the input size goes to infinity.

- Space complexity:

Space complexity is a measure of the amount of working storage an algorithm needs. That means how much memory, in the worst case, is needed at any point in the algorithm. As with **time complexity**, we're mostly concerned with how the space needs grow, in big-Oh terms, as the size N of the input problem grows.

algorithm correctness:

correctness of an algorithm is asserted when it is said that the algorithm is correct with respect to a specification. *Functional* correctness refers to the input-output behaviour of the algorithm (i.e., for each input it produces the expected output).

A distinction is made between total correctness, which additionally requires that the algorithm terminates, and partial correctness, which simply requires that *if* an answer is returned it will be correct.

Since there is no general solution to the halting problem, a total correctness assertion may lie much deeper.

A termination proof is a type of mathematical proof that plays a critical role in formal verification because total correctness of an algorithm depends on termination.

Onothere way to say likeThis means to verify if the algorithm leads to the solution of the problem (hopefully after a finite number of processing steps).

The main steps in the formal analysis of the correctness of an algorithm are:

• Identification of the properties of input data (the so-called problem's preconditions).

• Identification of the properties which must be satisfied by the output data (the so-called problem's postconditions).

• Proving that starting from the preconditions and executing the actions specified in the algorithms one obtains the postconditions. When we analyze the correctness of an algorithm a useful concept is that of state.

The algorithm's state is the set of the values corresponding to all variables used in the algorithm.

distributed algorithms:

Distributed algorithms are used in many practical systems, ranging from large computer networks to multiprocessor shared-memory systems. Distributed algorithms are algorithms designed to run on multiple processors, without tight centralized control. In general, they are harder to design and harder to understand than single-processor sequential algorithms.

A *distributed system* is a model in which components located on network computer communicate and coordinate their actions by passing message The components interact with each other in order to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. There are many alternatives for the message passing mechanism, including pure HTTP, RPC- like connectors and message queue.

Distributed algorithms are an established tool for designing protocols for sensor networks. There are a number of reasons why distributed algorithms are highly appealing for the design of sensor network protocols. First of all, we want to avoid that memory constrained sensor nodes have to store lots of information about far-away nodes. Thus, in fact it is an advantage if nodes compute the local part of a problem's solution only.

More importantly, restricting the distance from which information is collected implies that:

(i) algorithms are fast, as communication is typically the most time consuming part, and

(ii) (ii) changes in input or topology can be dealt with locally, i.e., merely a part of the nodes need to recompute their output. In other words, in a large system, faults or reconfigurations interfere with a small fraction of the system only.

This brings us to another impressive capability of distributed systems: If well-crafted, they can function correctly from a global perspective even if some of the nodes are crashing, show erroneous behaviour, or are even maliciously trying to make the system as a whole fail. Distributed algorithms are crucial in order to exploit this potential, as any part of an algorithm that runs on a single node will fail if the respective node crashes.

Even worse, if the node does not simply crash, it may corrupt the entire network with wrong information. Last but not least, as solutions are derived from as little knowledge on the overall state of the network as possible, many distributed algorithms turn out to be simple and elegant. While elegance is not necessarily a design issue for sensor networks, simplicity is, in particular considering the limited computational power, memory, and energy of sensor nodes.

Overall, we believe that distributed algorithms can offer a lot to designers of sensor networks. However, even though sensor networks seem to be out-of-the-book distributed systems in theory, in practice quite a few difficulties are between an abstract distributed algorithm and its implementation on a sensor node.

Inquiry 2:

I would like to discus on **Artificial intelligence and robotics :**

artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using the rules to reach approximate or definite conclusions), and self-correction. Particular applications of AI include expert systems, speech recognition and machine vision.

The motivation for developing these systems is not only the cost savings achieved by the reduction of 30 to 50 employees, but also a consistency in service provided to the end-users.

Intelligent systems are used to support decision-making and problem-solving applications.

- AI benefits include:

• Enhanced problem-solving.

• Improved decision quality.

• Ability to solve complex problems.

• Consistent decisions.

• Technology can be used to facilitate decision- making, but managers must be part of the process and work with it.

• It is good to capture expertise in a system so that if the "expert" leaves the knowledge does not go with him or her.

• There needs to be adaptability in any system that is built.

For the testing purpose we used th turing test for AI.The Turing test is designed to determine whether a computer exhibits intelligence. A computer can be considered smart only when a human interviewer cannot identify the computer while conversing with both an unseen human and a computer.

We also include infrencing

Inferencing is the reasoning process of AI. It takes place in the brain of an AI process.

So many AI technologies witch we are use in our company like . Expert systems, natural language processing, robotics, speech understanding, speech (voice) recognition, computer vision and scene recognition, intelligent computer-aided instruction, neural computing, intelligent agents, automatic programming, translation of languages, and summarizing news can all be considered AI technologies. The major technologies are expert systems, neural networks, intelligent agents, fuzzy logic, and genetic algorithms.

- I like to discus on **expert system from the AI**. which I can aply in the company

an expert system. An expert system is a computer program that attempts to mimic human experts by the system's capability to render advice, to teach and execute intelligent tasks.

The development environment for ES includes the activities and support that are necessary to acquire and represent the knowledge as well as to make inferences and provide explanations. The major players in this environment are the knowledge engineer and the domain expert who act as builders. Once the system is completed it is used for consultation by the nonexpert user via the consultation environment.

- The major components are:

* Knowledge base--the software that represents the knowledge.

* Inference engine--the reasoning mechanism.

* User interface--the hardware and software that provide the dialogue between people and the computer.

• Domain expert--the individual who is considered an expert.

• Knowledge engineer--the individual who acquires and represents the knowledge.

• Explanation facility--the software that answers questions such as "Why" and “How."

• Blackboard--a workplace for storing and working on intermediate information.

• Reasoning improvement--a facility (not available commercially) for improving the reasoning capabilities of an ES.

• User--the non-expert who uses the machine for consultation. • Hardware--the hardware that is needed to support the ES.

The brain of an expert system is the inference engine that provides a methodology for reasoning about information in the knowledge base. Inference can be performed using semantics networks, production rules, and logic statements.

Blackboard system activity on ES ,The blackboard records intermediate hypotheses and decisions, devises a plan of how to attack a problem, provides an agenda of actions awaiting execution, and lists the candidate solutions to be examined.

- Genetic categories of ES applications are:

Rule-based ES. Knowledge is represented by a series of rules.

Frame-based systems. Knowledge is represented as a series of frames (an object-oriented approach).

Hybrid systems. Involve several approaches such as fuzzy logic and neural networks.

Model-based systems. Structured around a model that simulates the structure and function of the system under study.

Ready-made systems. Utilize prepackaged software. Real-time systems. Systems designed to produce a just-in-time response.

- Success factors of ES are:

• Level of knowledge must be sufficiently high.

• Expertise must be available from at least one expert.

• The problem to be solved must by fuzzy. Chapter 10 Artificial Intelligence and Expert Systems: Knowledge- Based Systems 10-5

• The problem must be narrow in scope.

• The shell must be of high quality and naturally store and manipulate the knowledge.

• The user interface must be friendly to novice users.

• The problem to be solved must be difficult and important enough to justify the development of a system.

• Knowledgeable developers with good people skills are needed.

• The impact of the ES must be considered.

• The impact should be favorable.

• Management support is needed.

- Some of the limitations of ES are:

• Knowledge is not always readily available.

• It can be difficult to extract expertise from humans.

• There are frequently multiple correct assessments.

• Time pressures.

• Users have cognitive limits.

• ES works well only within a narrow domain of knowledge.

• Most experts do not have an independent means to validate results.

• Vocabulary is often limited and difficult to understand.

• Help from knowledge engineers is difficult to obtain and costly.

• Potential for lack of trust on the part of the end-users.

• Knowledge transfer is subject to biases.

In real-time ES the conclusions (recommendations) are derived fast so a process can be impacted immediately. They are used in quality control and robotics (e.g., to correct a malfunction).

- Problem of AI:

- The major difficulty in developing these systems is extracting the expertise needed to develop the knowledge base.
- It is difficult to extract an experts knowledge and codify it into a format that can be used in an automated application.
- Artificial intelligence is not creative , it is limited in the use of sensory devices .
- it cannot make use of a very wide context of experiences , and it does not use common sense .