Devin Gonier
Lewis University

April 2, 2017
Assignment 1

Inquiry 1A: Algorithmic Analysis

Algorithmic analysis is a process for determining the quality of an algorithm and usually includes factors like: the amount of necessary time to operate relative to the size of an input (running time), the price of performance, and the efficiency for which it operates given the resources available. The crucial factor of running time can be further distinguished between algorithms which utilize multiple processors and sequential systems that only use one. In both types, the number of computational steps involved in running the algorithm is measured for a lower ($\Omega f(n)$) and upper bound ($O(f(n))$) of the worst-case-scenario. Computational steps refer to the amount of logical or arithmetic operations employed by the algorithm on data. For examples, algorithms that employ many nested *for loops* can quickly become exponentially more expensive than those that don't in terms of computational complexity. Parallel systems should also factor in the number of routing steps or communication channels involved in connecting processors together (t(n)). Distributing computational resources among multiple resources may reduce the number of computational steps, but if the communication between processors takes a significant amount of time, then any time advantage may be lost by employing such a system.

Certain inputs may result in extremely fast solutions by shear happenstance of the input, which is why worst-case-scenarios are used to determine performance. For example, in a search algorithm, by pure luck alone the first item evaluated could be the item being sought. However, this circumstance would not be representative of the efficiency of the search algorithm for other inputs. Therefore, any determination of running time for a given problem using an algorithm needs to consider the size of the data input, the number of computational steps involved, and in the case of parallel or distributed systems the number of communication channels between processors for the worst-case-scenario.

The price associated with an algorithm would be a function of the actual cost of building hardware associated with it. For example, does an algorithm require many cheap processors routed together, or one very expensive processor? Finally, the efficiency with which resources are used will depend on the problem at hand. For instance, in the case of sensors of a distributed system, how are wireless communications utilized, such that messages do not disrupt one another and how can more relevant processors be prioritized over less relevant ones? These and other factors are commonly used to determine if an algorithm is a good one or a bad one. Well written articles will evaluate proposed algorithms according to the specifics of the problem being addressed using these and other factors.

References

"A Gentle Introduction to Algorithm Complexity Analysis." A Gentle Introduction to Algorithm Complexity Analysis. N.p., n.d. Web. 30 Mar. 2017. (http://discrete.gr/complexity/)

Parallel & Distributed Computing Lecture Notes by Rashid Bin Muhammad, PhD.
http://www.personal.kent.edu/~rmuhamma/

Devin Gonier
Lewis University

Inquiry 1B: Distributed Algorithms

Distributed algorithms are algorithms which attempt to leverage the advantages associated with using multiple networked processors vs. a single processor to solve a problem. These algorithms can be used in very large systems spanning incredible distances like the internet, or small systems like a single computer in one room. Usually algorithms which span incredible distances (more than one room) are referred to as distributed systems, while those which are confined to a room are referred to as a multicomputer. Distributed algorithms are analyzed according to their influence on factors like content complexity of messages, the number of processors involved, when and how processors communicate and the number of connections between processors in the system.

Increased content complexity of a message can help map channels to define more direct routes, potentially reducing the amount of time necessary to solve the problem. However, with increased content complexity the amount of time needed for each node to process and store a message goes up and the technology requirements for each node may be expensive.

Another relevant factor for distributed algorithmic analysis is the effect of increasing the number of processors involved. This can help provide incremental growth in computing power for a well networked system by dividing the burden of certain processes amongst different processors. With certain types of problem this can also expand access to inputs because sensors are more geographically dispersed (such as with weather phenomena). However, increasing the number of processors also increases the size of the network and therefore the amount of time for messages to be communicated throughout the system.

The medium with which messages are communicated may restrict the amount of communication possible at any given point in time. Thus, determining when and how messages are delivered is also important. Synchronous systems work by having all nodes begin and end a round of communication at the same time, while asynchronous systems send and process messages according to the order they are received and the amount of time it takes to process them. Pros and cons of both relate to avoiding congestion while balancing the need for different processing times among nodes.

Perhaps the most important factor is the number and frequency of connections utilized for a given problem. Are messages ‘broadcast’, streamlined as a ‘unicast’, or somewhere in between as a ‘multicast’? Are messages initialized by a leader or from a central location? How are such roles among nodes determined? These and similar questions have significant effects on how quickly messages are communicated because they minimize the number of messages being sent. Successful research of distributed algorithms take all these (and other) factors into account when analyzing the quality of a distributed algorithm.

References:

Parallel & Distributed Computing Lecture Notes by Rashid Bin Muhammad, PhD.
http://www.personal.kent.edu/~rmuhamma/

Devin Gonier
Lewis University
April 2, 2017
Assignment 1

Inquiry 1C: Algorithm Correctness

Algorithm Correctness Proofs refer to strategies programmers use to determine if an algorithm will be successful for variable sized inputs. Many algorithms have an incredibly large amount of potential inputs, so programmers must formally analyze the steps using logical operators to determine if the algorithm will be successful with any arbitrary input (denoted 'n'). Most algorithmic proofs for example depend on the principle of mathematical induction, in which if one can demonstrate an algorithm is successful for an arbitrary value k, then if the algorithm is successful for k+1, then it will be successful for all integers up to n, assuming k+1 is less than or equal to n. The process of mathematical induction, like other proofs in logic, requires the use of logical operators on assumed facts to determine if the statement logically follows from assumptions.

Since proofs of complex algorithms can become tedious and often are susceptible to human error, software like HOL Light Theorem Prover, PVS, and ACL2 have been designed to analyze algorithmic correctness via formal proof methods. Usually these programs are designed on a logical core set of primitive statements (such as transitivity or reflexivity) and can accommodate higher-order logical approaches like the lambda-calculus. As a continuation of the work by logicians such as Frege or Russell, programmers today are creating proofs of more complex mathematical statements. In other words, Frege was able to develop and use second-order logic to provide proofs for mathematical expressions like addition and subtraction using basic assumptions and definitions (such as defining natural numbers in relation to zero). However, more complex mathematical expression such as the identity relation for tan(b+x) require very complex proofs with very many primitive steps, and as a result would be more successfully accomplished by computers running algorithmic correctness proofs.

More advanced software used in algorithmic correctness proofs is increasingly helpful for establishing formal verification of complex algorithms that may take very large datasets as input. In today's society, this is increasingly important as algorithmic correctness has significant effects on financial gains or even life and death.

References:

Harrison, John. Formal Verification of Mathematical Algorithms. Intel Corporation, June 3rd, 2002.

Devin Gonier
Lewis University

Inquiry 2: Computer Science Problems in my Profession

To best answer questions about the relevance of certain topics in Computer Science for my profession, it's probably best to distinguish between my current profession and my intended profession, since both involve tasks that relate to some of the "hottest topics in Computer Science." Currently, I work as a service-quality performance consultant for large companies like AAA. Much of this entails data analysis, database management, and strategic advising on behaviors that best improve service quality and how best to motivate those behaviors in employees. While most of the general analysis I do is straightforward work in programs like Excel, I hope to develop more complex modeling systems for AAA that would require machine learning algorithms. For example, when someone has a roadside emergency and calls AAA for assistance, telephone operators use very basic systems to determine the amount of time it will take for a driver to arrive at the location of the incident (called 'PTA': Promised Time of Arrival). Having accurate PTAs has proven to be incredibly important for keeping members satisfied, but there are many variables that can slow and speed up the time it takes for a driver to arrive and assist a member. I intend to design a machine learning algorithm that will account for these factors (e.g. driver technician history, weather, overall current volume, and location of the breakdown) to vastly improve the accuracy of current predictions.

Hopefully as I begin researching different machine learning algorithms such as logistic regression, neural networks, and k-means clustering I will figure out a strategy that works best for this problem. Therefore, it will be essential I use research based approaches to determine which of these algorithms is most appropriate to the needs of AAA. For example, while logistic regression may be ideal for determining if a member is satisfied or for evaluating the propensity for membership renewal, it may not be ideal for calculating a specific time estimate. Furthermore, since many variables involved in this type of modeling can be understood as categorical (i.e. it's snowing or it isn't), while others are scalable (i.e. the distance from a driver to a member) it may be good to incorporate multiple approaches. I will research the various advantages and disadvantages of algorithms to determine the best approach.

While these and other problems in my current profession interest me and relate to hot topics in computer science, I am more interested in the problems which relate to my intended profession. My goal is to eventually to do research on Artificial Intelligence for an organization like Machine Intelligence Research Institute or Berkeley's Human Compatible Artificial Intelligence Program. Value-Alignment (devising strategies for aligning human and machine values) is a rapidly growing field in the Artificial Intelligence research community. I believe my background in philosophy as an undergraduate combined with skills I develop at Lewis University and afterwards would position me well to contribute to solutions of these problems. Value-Alignment is not only important because it can help avoid King Midas doomsday type scenarios, but also because it would vastly improve the ways in which machines can predict and interpret human wishes and intentions without them being explicitly defined.

Solutions to the problem of Value-Alignment in the field of Artificial Intelligence depend on research-based approaches in computer science, cognitive science, psychology, and philosophical domains (e.g. what is a good ethical system for calculating moral consequences of behaviors). Specifically, computer scientists like Stuart Russell at Berkeley are doing work on inverse reinforcement learning that could be really promising. Such approaches make maximizing the computers objectives contingent on maximizing human objectives, such that it is essential human and computer systems work together to accomplish goals. Computers are more successful when they anticipate the needs and goals of their human counterpart. In such scenarios, computers are initially unaware of the human's goal, but must interpret it as the scenario (usually a game) continues. Researching approaches like this will help solve

the problem of value-alignment. We must evaluate these approaches according to their accuracy relative to intended goals, their efficiency (would such a program produce results in a reasonable amount of time?), and whether or not the precise goals themselves are sufficient to accomplish the overall goals of value alignment. Thus, I foresee many opportunities to address computer science problems utilizing research based approaches in both my current profession and my intended one.