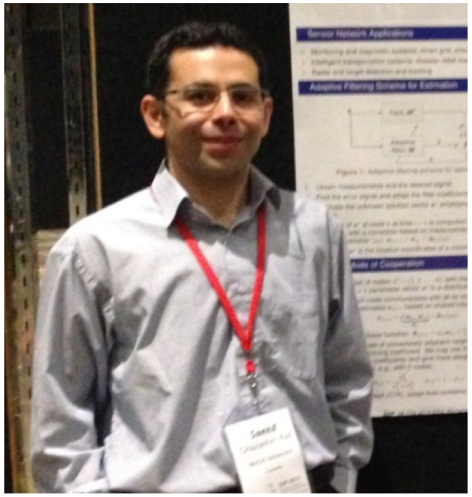
Introduction to Algorithms



Prof. Saeed Ghazanfari-Rad

Email: grad.saeed@gmail.com

Github: github.com/saeedgrad

References

- **Introduction to Algorithms**, Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, (Mit Press).
- **Algorithm Design**, Jon Kleinberg and Eva Tardos, (Pearson).
- **The Algorithm Design Manual,** Steven S S. Skiena, (Springer).
- **Algorithms,** R. Sedgewick, K. Wayne, (Addison-Wesley).

Basic definitions

- What is an algorithms?
- Informally, an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output in a finite amount of time.
- You can also view an algorithm as a tool for solving a well-specified computational problem. The statement of the problem specifies in general terms the desired input/output relationship for problem instances, typically of arbitrarily large size.

Basic definitions

- The algorithm describes a specific computational procedure for achieving that input/output relationship for all problem instances.

Features of algorithms

- Input
- Output
- Definiteness
- Correctness
- Finiteness
- Effectiveness
- Generality

The problem of sorting

Input: sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

Output: permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Example:

Input: 8 2 4 9 3 6

Output: 2 3 4 6 8 9

Problems solved by algorithms

- The internet enables people all around the world to quickly access and retrieve large amounts of information. With the aid of clever algorithms, sites on the internet are able to manage and manipulate this large volume of data.
- Examples of problems that make essential use of algorithms include finding good routes on which the data travels (techniques for solving such problems appear in Chapter 22), and using a search engine to quickly find pages on which particular information resides (related techniques are in Chapters 11 and 32).

Problems solved by algorithms

- Electronic commerce enables goods and services to be negotiated and exchanged electronically, and it depends on the privacy of personal information such as credit card numbers, passwords, and bank statements.
- The core technologies used in electronic commerce include public-key cryptography and digital signatures (covered in Chapter 31), which are based on numerical algorithms and number theory.

Problems solved by algorithms

- Manufacturing and other commercial enterprises often need to allocate scarce resources in the most beneficial way.
- An oil company might wish to know where to place its wells in order to maximize its expected profit.
- A political candidate might want to determine where to spend money buying campaign advertising in order to maximize the chances of winning an election.
- An airline might wish to assign crews to flights in the least expensive way possible, making sure that each flight is covered and that government regulations regarding crew scheduling are met.

Problems solved by algorithms

- A doctor needs to determine whether an image represents a cancerous tumor or a benign one. The doctor has available images of many other tumors, some of which are known to be cancerous and some of which are known to be benign.
- A cancerous tumor is likely to be more similar to other cancerous tumors than to benign tumors, and a benign tumor is more likely to be similar to other benign tumors.

Resources of time and space

- If computers were infinitely fast and computer memory were free, would you have any reason to study algorithms?
- If computers were infinitely fast, any correct method for solving a problem would do. You would probably want your implementation to be within the bounds of good software engineering practice.

Resources of time and space

- Of course, computers may be fast, but they are not infinitely fast. Computing time is therefore a bounded resource, which makes it precious.
- Memory may be inexpensive, but it is neither infinite nor free. You should choose algorithms that use the resources of time and space efficiently.

Efficiency

- Different algorithms devised to solve the same problem often differ dramatically in their efficiency. These differences can be much more significant than differences due to hardware and software.

Efficiency

As an example, Chapter 2 introduces two algorithms for sorting. The first, known as ***insertion sort***, takes time roughly equal to $c_1 n^2$ to sort n items, where c_1 is a constant that does not depend on n. That is, it takes time roughly proportional to n^2. The second, ***merge sort***, takes time roughly equal to $c_2 n \lg n$, where $\lg n$ stands for $\log_2 n$ and c_2 is another constant that also does not depend on n. Insertion sort typically has a smaller constant factor than merge sort, so that $c_1 < c_2$.

Efficiency

We'll see that the constant factors can have far less of an impact on the running time than the dependence on the input size n. Let's write insertion sort's running time as $c_1 n \cdot n$ and merge sort's running time as $c_2 n \cdot \lg n$. Then we see that where insertion sort has a factor of n in its running time, merge sort has a factor of $\lg n$, which is much smaller. For example, when n is 1000, $\lg n$ is approximately 10, and when n is 1,000,000, $\lg n$ is approximately only 20. Although insertion sort usually runs faster than merge sort for small input sizes, once the input size n becomes large enough, merge sort's advantage of $\lg n$ versus n more than compensates for the difference in constant factors. No matter how much smaller c_1 is than c_2, there is always a crossover point beyond which merge sort is faster.

Efficiency, Computer A vs. B ?

- A: faster computer (computer A) running insertion sort, computer A executes 10 billion instructions per second, the resulting code requires 2n^2 instructions to sort n numbers.
- B: slower computer (computer B) running merge sort, computer B executes only 10 million instructions per second, the resulting code taking 50 n lg n instructions.
- They each must sort an array of 10 million numbers.

Efficiency, Computer A vs. B ?

- A(faster)

$$\frac{2 \cdot (10^7)^2 \text{ instructions}}{10^{10} \text{ instructions/second}} = 20{,}000 \text{ seconds (more than 5.5 hours)},$$

- B(slower)

$$\frac{50 \cdot 10^7 \lg 10^7 \text{ instructions}}{10^7 \text{ instructions/second}} \approx 1163 \text{ seconds (under 20 minutes)}.$$

- By using an algorithm whose running time grows more slowly, even with a poor compiler, computer B runs more than 17 times faster than computer A!

Algorithms and other technologies

- Advanced computer architectures and fabrication technologies
- Easy-to-use, intuitive, graphical user interfaces (GUIs)
- Object-oriented systems
- Integrated web technologies
- Fast networking, both wired and wireless
- Machine learning
- Mobile devices