What is a Computer?

Algorithms, Memory and Parallel Computers

Carlotta Pavese

10.31.13

Outline

Outline

Algorithms

Algorithms

Algorithms

Is it possible to write a program that inspects any other programs and determines whether or not they will eventually stop?

- An algorithm is a fail-safe procedure that, if executed correctly, guarantees the result.

Algorithms

Algorithms

Algorithms

Is it possible to write a program that inspects any other programs and determines whether or not they will eventually stop?

- An algorithm is a fail-safe procedure that, if executed correctly, guarantees the result.
- Programs are representations of algorithms in some programming language.

Algorithms

Algorithms

Algorithms

Is it possible to write a program that inspects any other programs and determines whether or not they will eventually stop?

- An algorithm is a fail-safe procedure that, if executed correctly, guarantees the result.
- Programs are representations of algorithms in some programming language.
- There are usually several ways of expressing in algorithms, depending on the programming language.

Algorithms

Comparing Algorithms

- Algorithms are compared on the basis of how fast they are.

Algorithms

Comparing Algorithms

- Algorithms are compared on the basis of how fast they are.
- But how algorithms' speed can be compared?

Algorithms

Comparing Algorithms

- Algorithms are compared on the basis of how fast they are.
- But how algorithms' speed can be compared?
  - Algorithms can be implemented in different ways!

Algorithms

Comparing Algorithms

- Algorithms are compared on the basis of how fast they are.
- But how algorithms' speed can be compared?
  - Algorithms can be implemented in different ways!
  - Algorithms can be applied to programs of different size!

Algorithms

Comparing Algorithms

- Algorithms are compared on the basis of how fast they are.
- But how algorithms' speed can be compared?
  - Algorithms can be implemented in different ways!
  - Algorithms can be applied to programs of different size!

Algorithms

Comparing Algorithms

- Algorithms are compared on the basis of how fast they are.
- But how algorithms' speed can be compared?
  - Algorithms can be implemented in different ways!
  - Algorithms can be applied to programs of different size!

Answer: We can describe the speed of an algorithm on the basis of how much the time required for performing the task grows along with the size of the problem.

Algorithms

Comparing Algorithms

First sock-matching algorithm

- Pull out a random sock out of the basket.
- Pull out a second sock out of the basket.
- Compare it with the first pulled out.
- If it matches, bundle them together.
- if it does not match, throw it back in the basket and restart the process.

Algorithms

Comparing Algorithms

Second sock-matching algorithm

- Pull out a random sock out of the basket on the table.
- Pull out a second sock out of the basket.
- Compare it with the first pulled out.
- If it matches with any sock on the table, bundle them together.
- if it does not match, set it next to the first.
- Repeat the process until all the socks are matched.

Algorithms

Comparing Algorithms

Compare the first and the second algorithm

- Assume that there are n socks in the basket.
- Using the first algorithm, finding two that match requires pulling out and pulling back an average of half of the remaining socks.
- So the number of sock removals is proportional to the square of number of socks.
- The algorithm is of order n^2, meaning that for large problem, the time of execution grows as the square of the problem size.
- The second algorithm, instead, is of order n for in it each

Algorithms

Comparing Algorithms

Compare the first and the second algorithm

- In other words, if there are m times as many socks in the basket, the time of execution of the second algorithm will only by m by n.
- The time of execution of the first algorithm will be instead m^2 by n.
- So if n=10, then using the second algorithm to match twice as many socks will take twice as much time.
- Instead, if n=10, then using the first algorithm to match twice as many socks will take four times as much time.

Outline

Heuristics

Heuristics

Introducing Heuristics

- Heuristics are procedures that guarantee the result only with high likelihood.
- Sometimes, it is more practical to use heuristics than algorithms.

Heuristics

Heuristics

An example

- Estimate the relative strength of each player's position by counting the number of places of each type remaining on the board.
- Move so as to put yourself in the strongest possible position a few moves in the future.
- Expect your opponent to employ a strategy similar to your own.

Heuristics

Heuristics

An example

- It is possible to imagine a situation where those rules give unwanted results.
- For example, the strength of a player's position depends not just on the number of pieces, but their positions.
- However, it is true that as a general rule, having as many pieces as possible is better.
- Moreover, this method will trace out all possible lines of play for the next few move.
- However, considering the lines of play until the end of the game is highly impractical.

Heuristics

An example

- Of course, the heuristics will produce the right result only on the assumption that the other player will also reason in a way that is predicted by the third heuristics.
- On the assumption, in other words, that the other player also will calculate the relative strength of one's position in the same manner.

Heuristics

An example

- Most programs for chess incorporate other kinds of heuristics to abort searches of implausible lines of play and to search deeper in branches that involve exchange of pieces.
- There are also more elaborate systems for evaluating positions without searching—-for instance, systems that award points for keeping control of the center and for protecting the king.

Search spaces and heuristics

- A search space is a set of possibilities.
- For instance, in chess, the search space is the set of all possible play.
- Since search spaces may be too big to be searched exhaustively, heuristics are produced to narrow down the space.
- The reason why a search space may be very large is that it results from the combination of smaller elements—such as the individual moves in chess.
- This combining the elements gives rise to a combinatorial exposition of possibilities—an exposition that grows exponentially with the number of elements being combined.

- Combinations that share common elements are closer than combinations that do not.
- We can imagine possibilities as lying in a two-dimensional space known as fitness landscape.
- The desirability of a certain solution is thought of as altitude of the solution in the landscape.
- In this analogy, finding the best solution is like finding the top of a hill.

Outline

Memory

How does a Computer store information?

- Communication is exchange of information in space. (from one person to another person).
- Memory (or storage) is exchange of information in time.
- For example, you can send a letter to another person.
- Or you can send a letter to yourself, in which case your later self will receive it.
- An electronic computer stores information by constantly recirculating it, by continuously sending letters to itself.

Memory

How much memory is needed to store information?

- What are our unity of measure for information?
- Bits, of course!
- In general, n bits of memory are needed to store n bits of information.
- But how can we know how many bits are required to represent a given piece of information.
- For example, how many bits are in the words of this book?

Memory and Bits

How much memory is needed to store information?

- It turns out that the answer to this question depends on how the information is represented.
- Suppose we calculate it by the number of characters that "The pattern on the Stone" has.
- There are 250000 characters in it, and suppose that in our computers, 8 bits are needed to store each character.
- So we need 2 millions of bits to store the information contained in the book "The pattern on the stone."

Memory and Bits

How much memory is needed to store information?

- However, how many bits are required to store a character depends on the computer.
- So this method will not work in every case.
- Another method is the following.
- Use how many bits are necessary to code each different kind of characters.
- For example, if a book requires only 60 different kinds of characters, use 6 bits, so you get 6 by 250000 characters=1.5 million.

Compression

How can memory be compressed?

- The amount of memory required to store information can be compressed.
- One way to do so is to come up with a code that stores each character with fewer bits depending on how common that character is.
- So for instance, T and E are more common than Q and Z in most English text. So store the former with a shorter sequence of bits than for the latter.
- For example, in Morse code, the letter T is represented with a single dash and E with a single dot. Less common letters like Q are represented with 4-long sequences of

Compression

How can memory be compressed?

- Every compression will take advantage of regularities in the data.
- Another kind of regularity has to do with the frequency of certain pairs of letters.
- So for instance, Q is almost always followed by U, and Z almost never followed by K.
- There are also regularities in the grammar, sentence structure, and punctuation that allows for further compression of the text.

Compression

How can memory be compressed?

- Another kind of compression, especially used to store images, is one that records the process by which the image is generated, rather than the image itself.
- For instance, if an image is created by a list of lines, you can just store that list.
- You can do the same to store sounds and music.
- This leads us to another, more general measure of information.

Compression

How can memory be compressed?

- The amount of information in a pattern of bits is equal to the length of the smallest computer program capable of generating those bits.
- Note that once a string of information is maximally compressed, it will exhibit no further regularity.
- Because any further regularity would be an opportunity for further compression.
- So the strings of 1s and 0's representing optimally compressed text will look completely random.

Encryption

What is encryption?

- Encrypting a message is analogous to sending it in a locked box that can be opened only with a special key.
- Everyone that has a key is able to perform the conversion.
- The same key can be used for both decryption and encryption, or different keys for each.
- In a public encryption scheme, the keys for decryption and encryption are different and one eavesdropper who knows the encryption key will not thereby know the key needed to decrypt.

Encryption

What is encryption?

- Public key encryptions solve an important practical problem: many businesses that accept credit card numbers online publish their own public keys so that customers can encrypt their credit card numbers without fear that they will be intercepted and read.

Outline

Speed

Speed

- The speed of a computer is the amount of time required to send messages in and out of the memory.
- Two different kinds of computers, that differ in speed: sequential computers and parallel computers.
- Sequential computers operate on one word of data at a time.
- Parallel computers can process data in parallel.
- In particular, parallel computers can break down a memory into lots of little memories and give each its own processor.

Speed

Speed

- A long historical skepticism towards parallel computers.
- Two main reasons:
  - People tended to overestimate the complexity of parallel computers because they underestimate the rate in improvement in micro-electronic fabrication technology.
  - Inefficiency in breaking the computation into parallel parts. Less of a problem these days.
- The first parallel computers basically connected two or three large sequential computers.

Parallelism

Parallelism

- That was not very efficient.
- one inefficiency was that a task had to be divided into many stages, even hundred of states.
- But not every task lends itself to that kind of division.
- Imagine writing a newspaper article with one another person, or with hundreds of other reporters!
- Also all the computers would have access to a single memory, creating a lot of inefficiency.
- for example, imagine two processors access the data of empty seats to reserve one of them. If they have both access to it, they may both book it at the same time

Parallelism

Parallelism

All these kinds of inefficiency are summed up by Amdhal's law.
Amdhal's argument goes as follows:

- There always be part of the computation that is inherently sequential: work that can be done by only one processor at a time.
- If only 10 percent of the computation is sequential, no matter how much you speed up the rest 90 percent, the overall speed will never speed up more than a factor of 10.
- The processors working on the 90 per cent that can be done in parallel will have to wait for the processor working on the sequential part of the task.

The Brain is a parallel computer?

- Sequential computers see an image pixel by pixel.
- Brains see a picture at once, simultaneously.
- If the brains are parallel machines, it must be possible to build one that is as efficient. Hence, there must be something wrong with Amdhal's argument.

What is wrong with Amdhal's argument?

- The argument assumed that some part of the computation needed to be sequential.
- That does not need to be true.
- The crux of the issue is how the computation is divided into different processors.

What is wrong with Amdhal's argument?

A functional breaking down.

- One way of dividing the computation of a task into different processors is to give each processors different tasks, or functions.
- Like trying to divide painting a picture into a team of people.
- A functional breaking down gives each of them a different job: the job of opening the paint, the job of preparing the surface, the job of applying the paint, the job of cleaning the brushes.
- A functional breaking down of a task requires a lot of coordination among the people in the team.

What is wrong with Amdhal's argument?

Data parallel decomposition

- Another more efficient way of decomposing a task is called data parallel decomposition.
- In the analogy before, getting a fence painted by assigning different people different portions of the fence to paint.
- This does not require the same complexity of coordination among processors and really makes the computation more efficient.
  - In chess: have different processors search through different lines of play.
  - In imagine-processing tasks: assign a little patch of the image to each processor.

Seemingly sequential tasks that can be performed in parallel.

Chain following problem, Treasure Hunt

- A parallel computer with a million processors can find the last element in a chain of a million addresses in twenty steps.
- Assume every location corresponds to a processor and that processors can send each other messages.
- Each processor sends its own address to the next processor in the chain—i.e., the processor the address of which the first processor has stored in its memory.

Seemingly sequential tasks that can be performed in parallel.

...Continued.

- So now each processor knows the address of the processor that follows it as well as the address of the processor that precedes it.
- So now each processor uses this information to send the address of its successor to its predecessor.
- At each reduction step, the length of the chain is reduced by half.
- After a few reduction steps of this type, the chain following problem is executed in 20 steps.