Part II

Course Goals and Overview

High-Level Questions

1. Computation, formally.
    1. Is there a formal definition of a computer?
    2. Is there a "universal" computer?
2. Algorithms
    1. What is an algorithm?
    2. What is an efficient algorithm?
    3. Some fundamental algorithms for basic problems
    4. Broadly applicable techniques in algorithm design
3. Limits of computation.
    1. Are there tasks that our computers cannot do?
    2. How do we prove lower bounds?
    3. Some canonical hard problems.

Course Structure

Course divided into three parts:

1. Basic automata theory: finite state machines, regular languages, hint of context free languages/grammars, Turing Machines
2. Algorithms and algorithm design techniques
3. Undecidability and NP-Completeness, reductions to prove intractability of problems

Goals

1. # Algorithmic thinking
2. Learn/remember some basic tricks, algorithms, problems, ideas
3. Understand/appreciate limits of computation (intractability)
4. Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)

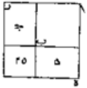
History

History

Muhammad ibn Musa al-Khwarizmi (c.780–c.850)

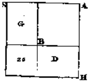
Text on Algebra

على تسعة وثلثين ليتم السطح الاعظم الذي هو سطح ره فبلغ
ذلك كله اربعة وستين فاخذنا جذرها وهو ثمانية وهو احد
اضلاع السطح الاعظم فاذا نقصنا منه مثل ما زدنا عليه وهو
خمسة بقي ثلثة وهو ضلع سطح اب الذي هو المال وهو جذره
والمال تسعة وهذه صورته



واما مال واحد وعشرون درهما يعدل عشرة اجذاره فانا
نجعل المال سطحا مربعا مجهول الاضلاع وهو سطح اد ثم نضم
اليه سطحا متوازي الاضلاع عرضه مثل احد اضلاع سطح اد وهو
ضلع ون والسطح هب فصار طول السطحين جميعا ضلع جه
وقد علمنا ان طوله عشرة من العدد لان كل سطح مربع
متساوي الاضلاع والزوايا فان احد اضلاعه مضروبا في واحد جذر
ذلك السطح وفي اثنين جذراه فلما قال مال واحد وعشرون
يعدل عشرة اجذاره علمنا ان طول ضلع هج عشرة اعداد لان
ضلع جد جذر المال فقسمنا ضلع جه بنصفين على نقطة

the first quadrate, which is the square, and the two quadrangles on its sides, which are the ten roots, make together thirty-nine. In order to complete the great quadrate, there wants only a square of five multiplied by five, or twenty-five. This we add to thirty-nine, in order to complete the great square S H. The sum is sixty-four. We extract its root, eight, which is one of the sides of the great quadrangle. By subtracting from this the same quantity which we have before added, namely five, we obtain three as the remainder. This is the side of the quadrangle A B, which represents the square; it is the root of this square, and the square itself is nine. This is the figure:—



*Demonstration of the Case: "a Square and twenty-one Dirhems are equal to ten Roots."**

We represent the square by a quadrate A D, the length of whose side we do not know. To this we join a parallelogram, the breadth of which is equal to one of the sides of the quadrate A D, such as the side H N. This paralellogram is H B. The length of the two

Algorithm Description

If some one says: "You divide ten into two parts: multiply the one by itself; it will be equal to the other taken eighty-one times." Computation: You say, ten less a thing, multiplied by itself, is a hundred plus a square less twenty things, and this is equal to eighty-one things. Separate the twenty things from a hundred and a square, and add them to eighty-one. It will then be a hundred plus a square, which is equal to a hundred and one roots.

Algorithm Description

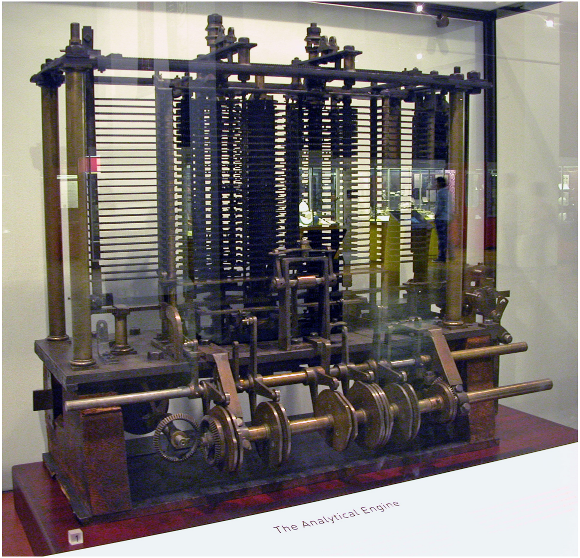
If some one says: "You divide ten into two parts: multiply the one by itself; it will be equal to the other taken eighty-one times." Computation: You say, ten less a thing, multiplied by itself, is a hundred plus a square less twenty things, and this is equal to eighty-one things. Separate the twenty things from a hundred and a square, and add them to eighty-one. It will then be a hundred plus a square, which is equal to a hundred and one roots.

$$\begin{aligned}(10 - x)^2 &= 81x \\ x^2 - 20x + 100 &= 81x \\ x^2 + 100 &= 101x\end{aligned}$$

Models of Computation vs Computers

1. Model of Computation: an "idealized mathematical construct" that describes the primitive instructions and other details
2. Computer: an actual "physical device" that implements a very specific model of computation

First Computer



Babbage's analytical engine—designed in 1837, never built.

First Program

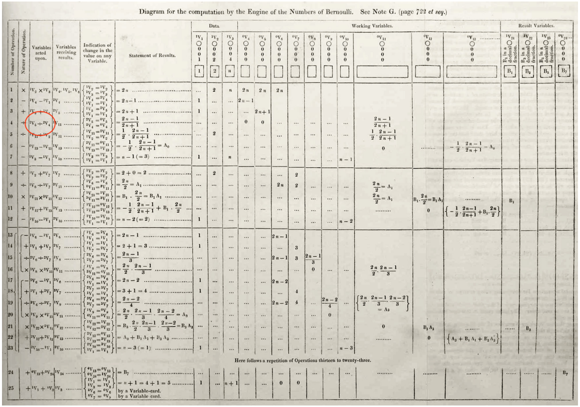
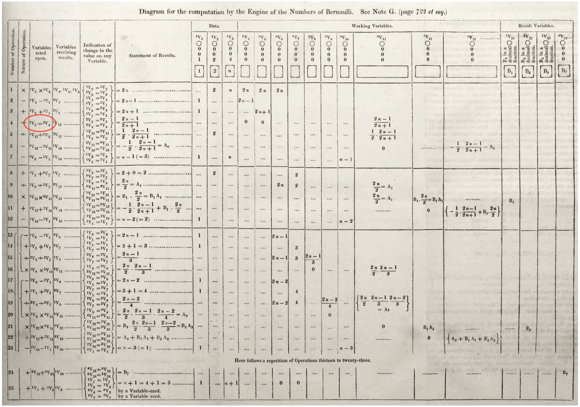


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.
1	×	${}^1V_2 \times {}^1V_3$	${}^1V_4, {}^1V_5, {}^1V_6$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_3 = {}^1V_3 \right\}$	$= 2n$
2	−	${}^1V_4 - {}^1V_1$	${}^2V_4$	$\left\{ {}^1V_4 = {}^2V_4 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n - 1$
3	+	${}^1V_5 + {}^1V_1$	${}^2V_5$	$\left\{ {}^1V_5 = {}^2V_5 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n + 1$
4	÷	${}^2V_5 \div {}^2V_4$	${}^1V_{11}$	$\left\{ {}^2V_5 = {}^0V_5 \atop {}^2V_4 = {}^0V_4 \right\}$	$= \frac{2n-1}{2n+1}$
5	÷	${}^1V_{11} \div {}^1V_2$	${}^2V_{11}$	$\left\{ {}^1V_{11} = {}^2V_{11} \atop {}^1V_2 = {}^1V_2 \right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$
6	−	${}^0V_{13} - {}^2V_{11}$	${}^1V_{13}$	$\left\{ {}^2V_{11} = {}^0V_{11} \atop {}^0V_{13} = {}^1V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$
7	−	${}^1V_3 - {}^1V_1$	${}^1V_{10}$	$\left\{ {}^1V_3 = {}^1V_3 \atop {}^1V_1 = {}^1V_1 \right\}$	$= n - 1\ (= 3)$
8	+	${}^1V_2 + {}^0V_7$	${}^1V_7$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^0V_7 = {}^1V_7 \right\}$	$= 2 + 0 = 2$
9	÷	${}^1V_6 \div {}^1V_7$	${}^3V_{11}$	$\left\{ {}^1V_6 = {}^1V_6 \atop {}^0V_{11} = {}^3V_{11} \right\}$	$= \frac{2n}{2} = A_1$
10	×	${}^1V_{21} \times {}^3V_{11}$	${}^1V_{12}$	$\left\{ {}^1V_{21} = {}^1V_{21} \atop {}^3V_{11} = {}^3V_{11} \right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$
11	+	${}^1V_{12} + {}^1V_{13}$	${}^2V_{13}$	$\left\{ {}^1V_{12} = {}^0V_{12} \atop {}^1V_{13} = {}^2V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$
12	−	${}^1V_{10} - {}^1V_1$	${}^2V_{10}$	$\left\{ {}^1V_{10} = {}^2V_{10} \atop {}^1V_1 = {}^1V_1 \right\}$	$= n - 2\ (= 2)$
13	−	${}^1V_6 - {}^1V_1$	${}^2V_6$	$\left\{ {}^1V_6 = {}^2V_6 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n - 1$
14	+	${}^1V_1 + {}^1V_7$	${}^2V_7$	$\left\{ {}^1V_1 = {}^1V_1 \atop {}^1V_7 = {}^2V_7 \right\}$	$= 2 + 1 = 3$
15	÷	${}^2V_6 \div {}^2V_7$	${}^1V_8$	$\left\{ {}^2V_6 = {}^2V_6 \atop {}^2V_7 = {}^2V_7 \right\}$	$= \frac{2n-1}{3}$
16	×	${}^1V_8 \times {}^3V_{11}$	${}^4V_{11}$	$\left\{ {}^1V_8 = {}^0V_8 \atop {}^3V_{11} = {}^4V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$
17	−	${}^2V_6 - {}^1V_1$	${}^3V_6$	$\left\{ {}^2V_6 = {}^3V_6 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n - 2$
18	+	${}^1V_1 + {}^2V_7$	${}^3V_7$	$\left\{ {}^2V_7 = {}^3V_7 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 3 + 1 = 4$
19	÷	${}^3V_6 \div {}^3V_7$	${}^1V_9$	$\left\{ {}^3V_6 = {}^3V_6 \atop {}^3V_7 = {}^3V_7 \right\}$	$= \frac{2n-2}{4}$
20	×	${}^1V_9 \times {}^4V_{11}$	${}^5V_{11}$	$\left\{ {}^1V_9 = {}^0V_9 \atop {}^4V_{11} = {}^5V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$
21	×	${}^1V_{22} \times {}^5V_{11}$	${}^0V_{12}$	$\left\{ {}^1V_{22} = {}^1V_{22} \atop {}^0V_{12} = {}^2V_{12} \right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$
22	+	${}^2V_{12} + {}^2V_{13}$	${}^3V_{13}$	$\left\{ {}^2V_{12} = {}^0V_{12} \atop {}^2V_{13} = {}^3V_{13} \right\}$	$= A_0 + B_1 A_1 + B_3 A_3$
23	−	${}^2V_{10} - {}^1V_1$	${}^3V_{10}$	$\left\{ {}^2V_{10} = {}^3V_{10} \atop {}^1V_1 = {}^1V_1 \right\}$	$= n - 3\ (= 1)$
					Here follows a repetition of Operations thirteen to twenty-three.
24	+	${}^4V_{13} + {}^0V_{24}$	${}^1V_{24}$	$\left\{ {}^4V_{13} = {}^0V_{13} \atop {}^0V_{24} = {}^1V_{24} \right\}$	$= B_7$
25	+	${}^1V_1 + {}^1V_3$	${}^1V_3$	${}^1V_1 = {}^1V_1$; ${}^1V_3 = {}^1V_3$; ${}^5V_6 = {}^0V_6$; ${}^5V_7 = {}^0V_7$	$= n + 1 = 4 + 1 = 5$; by a Variable-card.; by a Variable card.

Ada Lovelace's "Note G" describing how to calculate Bernouilli numbers using the analytical engine.

First Bug!



Ada Lovelace's "Note G" describing how to calculate Bernouilli numbers using the analytical engine.
This version contains a bug!

Models of Computation vs. Computers

Models and devices:

1. Algorithms: usually at a high level in a model
2. Device construction: usually at a low level
3. Intermediaries: compilers
4. How precise? Depends on the problem!
5. Physics helps implement a model of computer
6. Physics also inspires models of computation

Adding Numbers

Problem Given two n-digit numbers x and y, compute their sum.

Basic addition

$$\begin{array}{r} 3141 \\ +7798 \\ \hline 10939 \end{array}$$

Adding Numbers

```
c = 0
for i = 1 to n do
    z = x_i + y_i
    z = z + c
    If (z > 10)
        c = 1
        z = z − 10        (equivalently the last digit of z)
    Else c = 0
    print z
End For
If (c == 1) print c
```

Adding Numbers

```
c = 0
for i = 1 to n do
    z = x_i + y_i
    z = z + c
    If (z > 10)
        c = 1
        z = z − 10      (equivalently the last digit of z)
    Else c = 0
    print z
End For
If (c == 1) print c
```

1. Primitive instruction is addition of two digits
2. Algorithm requires $O(n)$ primitive instructions

Multiplying Numbers

Problem Given two n-digit numbers x and y, compute their product.

Grade School Multiplication

Compute "partial product" by multiplying each digit of y with x and adding the partial products.

$$
\begin{array}{r}
3141 \\
\times\ 2718 \\
\hline
25128 \\
3141 \\
21987 \\
6282 \\
\hline
8537238
\end{array}
$$

Time analysis of grade school multiplication

1. Each partial product: $\Theta(n)$ time
2. Number of partial products: $\leq n$
3. Adding partial products: n additions each $\Theta(n)$ (Why?)
4. Total time: $\Theta(n^2)$
5. Is there a faster way?

Fast Multiplication

Best known algorithm: $O\left(n \log n \cdot 4^{\log^* n}\right)$ by Harvey and van der Hoeven, published in 2018!

Conjecture: there exists an $O(n \log n)$ time algorithm

Fast Multiplication

Best known algorithm: $O\left(n \log n \cdot 4^{\log^* n}\right)$ by Harvey and van der Hoeven, published in 2018!
Conjecture: there exists an $O(n \log n)$ time algorithm

We don't fully understand multiplication!
Computation and algorithm design is non-trivial!

Aside about O-notation

Some previous versions of multiplication are still widely used:

- Karatsuba algorithm $O(n^{\log_2 3})$ [1962]
- Schönhage-Strassen (FFT) $O(n \log n \log \log n)$ [1971]

Why?

Aside about O-notation

Some previous versions of multiplication are still widely used:

- Karatsuba algorithm $O(n^{\log_2 3})$ [1962]
- Schönhage-Strassen (FFT) $O(n \log n \log \log n)$ [1971]

Why? Fürer's algorithm (2007) $O(n 2^{O(\log^* n)})$

Aside about O-notation

Some previous versions of multiplication are still widely used:

- Karatsuba algorithm $O(n^{\log_2 3})$ [1962]
- Schönhage-Strassen (FFT) $O(n \log n \log \log n)$ [1971]

Why? Fürer's algorithm (2007) $O(n 2^{O(\log^* n)})$

…beats Schönhage-Strassen for numbers greater than $2^{2^{64}}$.