Lecture 1: Syllabus and introduction to algorithms

CS303: Algorithms

January 5, 2014

1 Syllabus

- Goal: principles of algorithm design and efficiency analysis (given an algorithm, is it correct? how efficient it is? can we do better?).
- Text: the design & analysis of algorithms. C++ programming is involved in programming assignments (talk about it later)
- Outline: cover 2/3 of the materials. Others are omitted due to time constraints and requirement on probability.
  - Ch 1-4: fundamentals
  - Ch 6-8: sorting
  - Ch 10-13: data structures used in important algorithms
  - Ch 15-16: DP and greedy algorithms
  - Ch 18-19: advanced data structures (B-tree and Fibonacci heap)
  - Ch 22-26: graph algorithms
  - Ch 29,32-34: selected topics
- Focus: It's a mind game. In order to design good strategies to solve new problems, you need to master a few very important algorithms. That's the primary purpose of this course. (given a problem, what principles should be applied in designing an algorithm to solve it? what is the efficiency of your algorithm? How can you improve an existing algorithm?)
- Other logistics on the syllabus

2 Introduction to algorithm

2.1 History of algorithms

- In ancient Europe, numbers are represented by Roman numbers: MDCCCCIIII (1448). Think about how MDCCCCIIII + DCCCXII. Or even multiplying those two numbers.
- Around AD 600, decimal numbers were invented by Indians.
- In the 9th century, one of the most influential mathematician from Baghdad named Al Khwarizmi, wrote a book about how to add, subtract, multiply, divide, calculate the square root and calculate π. $\rightarrow$ algorithm is named after him. These processes described in the book were precise, unambiguous, mechanical, efficient, and correct. $\rightarrow$ they are algorithms.

- Al Khwarizmi's book were adopted in Europe around AD1200. One mathematician played an important role in this process: Leonard Fibonacci. $\rightarrow$ better known for his series as Fibonacci series.

2.2 Why do we need efficient algorithms?

Q:/ Algorithms primarily deal with software efficiency (time and space). What are other attributes of a software package that are more important than speed?
A:/ user-interface, correctness, security, maintainability, portability, etc. $\rightarrow$ efficiency allows them to happen. It's like cash.
Q:/ Aren't all algorithms good algorithms?
A:/ No. e.g. Fibonacci sequence example
Fibonacci sequence comes from the following scenario:

If I give you a pair of new-born rabbits, one male and one female. These rabbits mature in a month so at the end of the month, the female rabbit can produce another pair of rabbits at the end of the second month. Our rabbit never dies and female rabbit always produce a pair of new rabbits (one male/one female).
Q:/ How many pair of rabbits will we have at end of the first year (12 month)
A:/ Mathematically, we like to add a 0 in the beginning of the series.

Month	1	2	3	4	5	6	7	8	9	10	11	12
Rabbits	1	1	2	3	5	8	13	21	34	55	89	144

General formula for Fibonacci sequence $F_n = \begin{cases} 0 & \text{if n=0} \\ 1 & \text{if n=1} \\ F_{n-1} + F_{n-2} & \text{if n} > 1 \end{cases}$
Widely used in biology, visual arts, music, simulation, etc
Q:/ How can we calculate the Fibonacci series Fn, such as F_{200}
A:/

```
Algorithm 0.1: $F_n$
if n=0
  then return 0
if n=1
  then return 1
return $F_{n-1} + F_{n-2}$
```

Q:/ Is this algorithm correct?
A:/ Yes, it's from the definition of the series

Q:/ How fast is this algorithm?
A:/ If we let T(n) be the number of computer steps to calculate F_n, then
$T(n) = T(n-1) + T(n-2) + 1 \approx 1.6^n$
In other words, $T(200) >= 2^{139}$

Q:/ What does it mean?
A:/ The world supercomputer right now can do $2.5 * 10^{15}$ steps per second. So it will take $2 * 10^{25}$ seconds for the fastest computer to calculate F_{200}! (about $8.5 * 10^8$ billion years. Note the sun will

deteriorate into a red star in 4 to 5 billion years!)

Q:/ Can we do better?
A:/ yes, we can because the previous algorithm repeats many calculations

```
Algorithm 0.2:
Set up an array of size n+1 A[0,1,..., n]
A[0]←0, A[1]← 1
for i=2,..., n
do
  A[i]=A[i-1]+A[i-2]
done
return A[n]
```

Q:/ How fast is this algorithm?
A:/ Linear with n. so we can use the super computer to calculate $F_{30,000,000}$ in a second!