Dr. WEI Qiang
Associate Professor
School of Economics and Management
Tsinghua University
Beijing 100084, China
weiq@sem.tsinghua.edu.cn

Data Structures and Algorithms

Contents

In the Big Data Context

Tsinghua-SEM

Physical World vs. Internet World

- Internet World
  - Search Queries: 100,000,000,000/Day
  - Email: 100,000,000,000/Day
  - SMS: 100,000,000,000/Day

- Physical World:
  - Hotel (U.S.A.): 1,000,000 People/Day
  - Aviation: 1,000,000 People/Day
  - Taxi (Shanghai): 1,000,000 People/Day

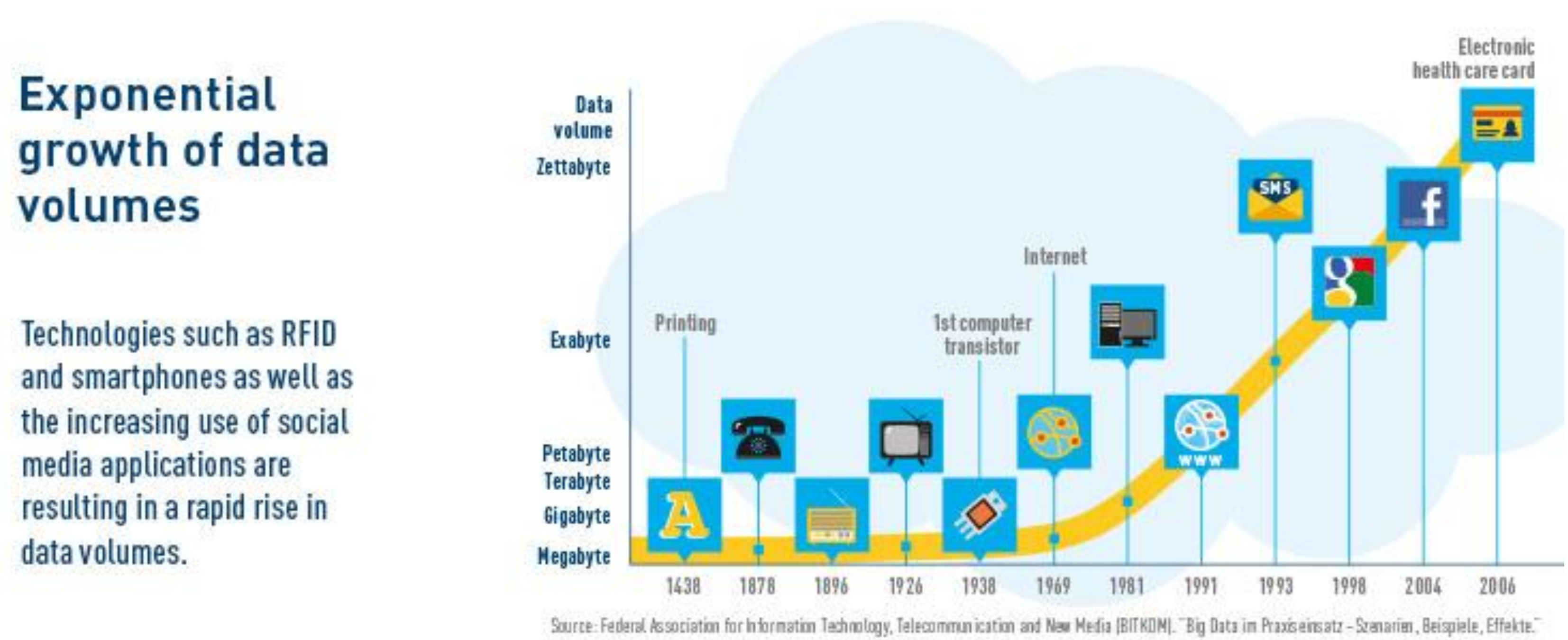
Tsinghua-SEM

Data Volumes

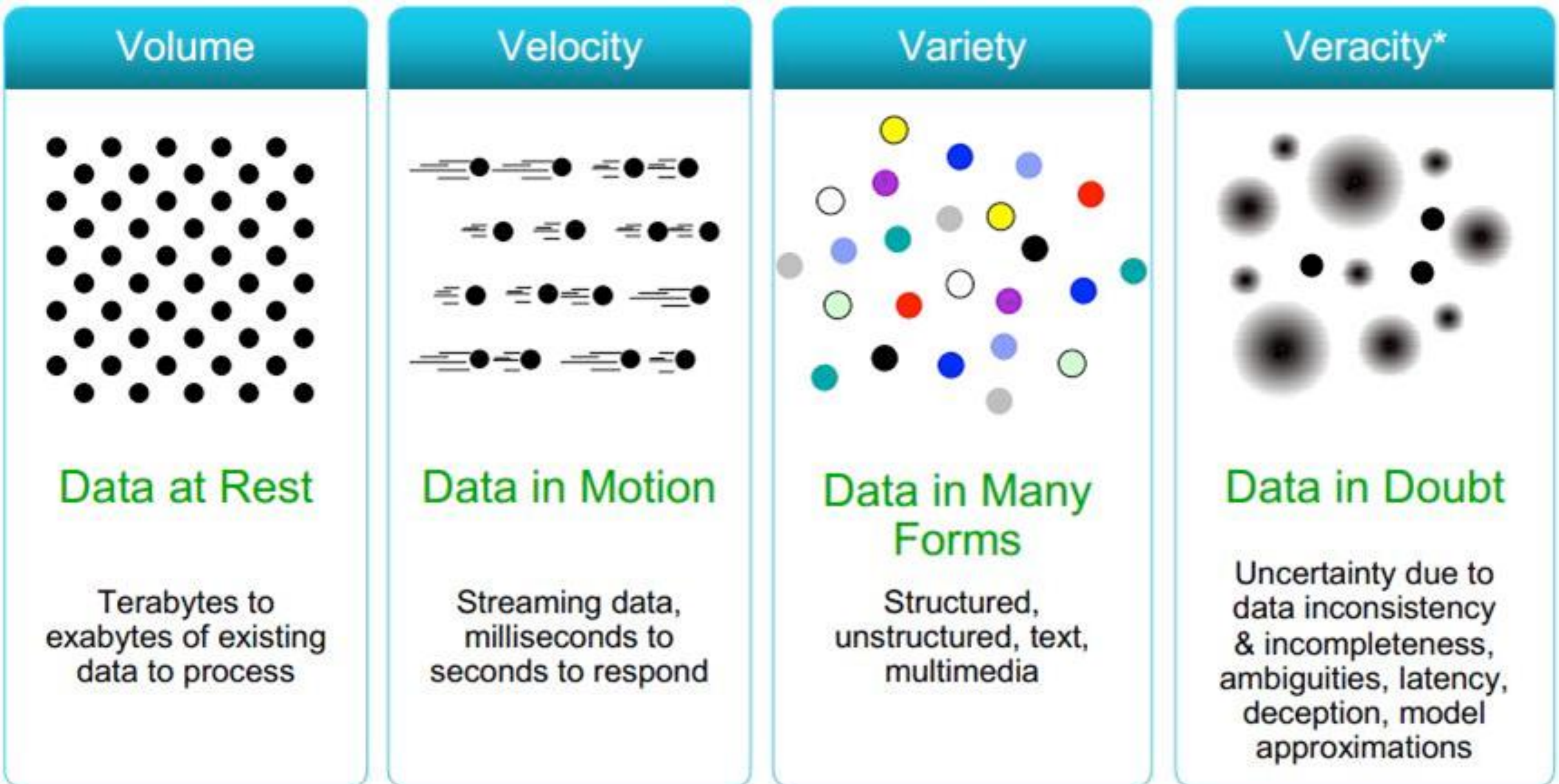
Volume	Meaning	Note
1 Bit	1 Bit	
1 Byte	8 Bit	1 character
1 Kilo Byte (KB)	1,024 Byte	1 plain text email
1 Mega Byte (MB)	1, 048,576 Byte	1 plain text novel, a picture
1 Giga Byte (GB)	1,073,741,824 Byte	30 minutes DVD video
1 Tera Byte (TB)	1,099,511,627,776 Byte	1~3 hard-disks
1 Peta Byte (PB)	1,024 TB	
1 Exa Byte (EB)	1,024 PB	
1 Zeta Byte (ZB)	1,024 EB	
1 YB	1,024 YB	
1 DB	1,024 YB	
1 NB	1,024 DB	1152921504606846976 1TB hard-disks, weighted 70 billons tons

Generation of Big Data

- Incrementally, the new information generated every 2 days on Internet, is equivalent to the information generated from human civilization till 2003. (Eric Schmidt, 2010)

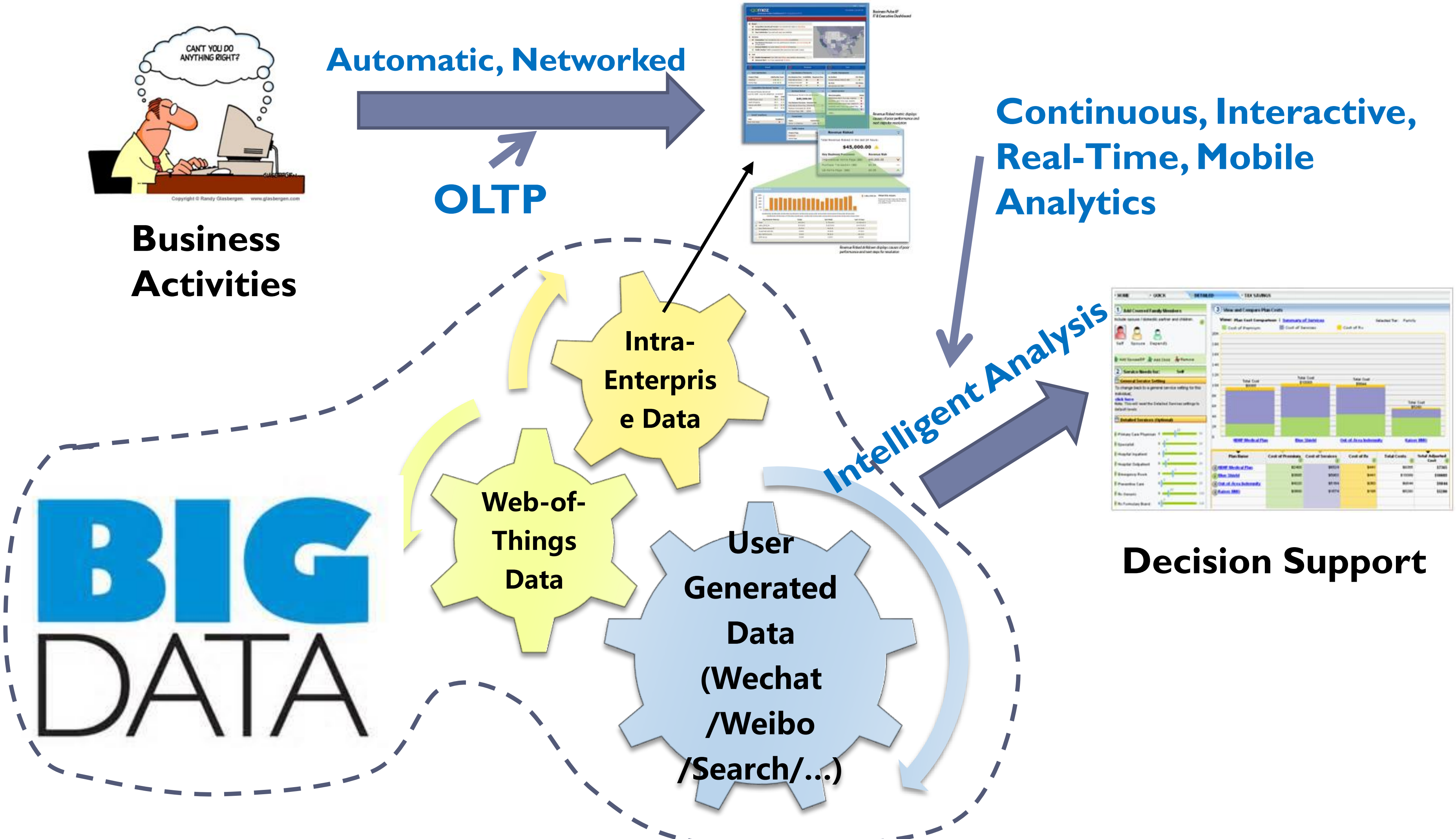


4V of Big Data



Management Decision in the Context of Big Data

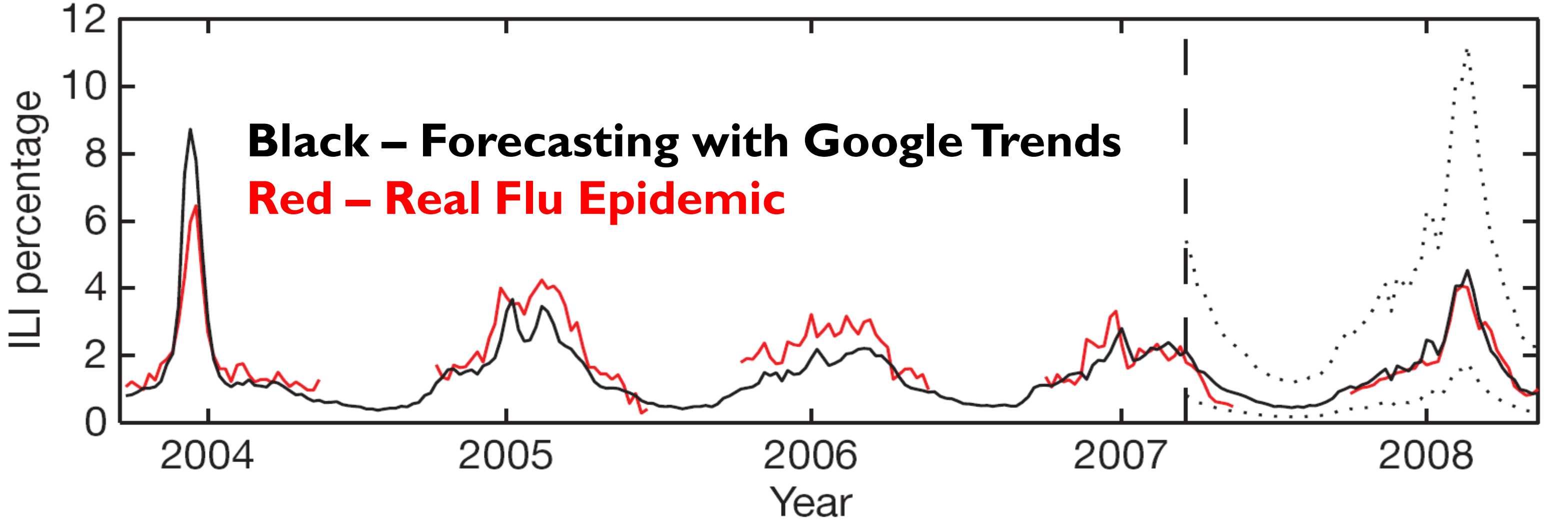
Tsinghua-SEM



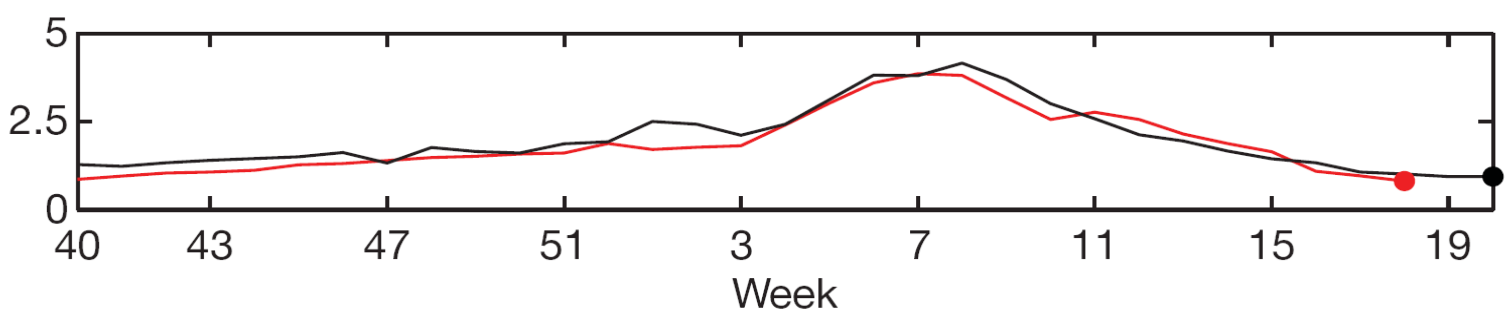
External Big Data

- **User Generated Data**
  - Search（Google Trends，百度指数）
  - Online Review (我买网，京东)
  - Blogs（博客、微博、微信）
  - Forums/BBS（销售论坛）
  - Weixin/Weibo（QQ、微信、Skype）
  - Web 2.0（Tumblr, YouTube, Youku)
  - Transactions（股票，期货）
  - Emoticons（颜文字）
  - Crowdsourcing（维基百科，百度知道）
  - P2P（拍拍贷，众筹网）
  - ……
- **Web-of-Things Data**
  - GPS
  - Inertial Navigation System (惯性系统)
  - Cameras
  - Sensors
  - Bar codes
  - RFID
  - Wearable Devices
  - ……

Flu Forecasting/Sensing with Google Trends



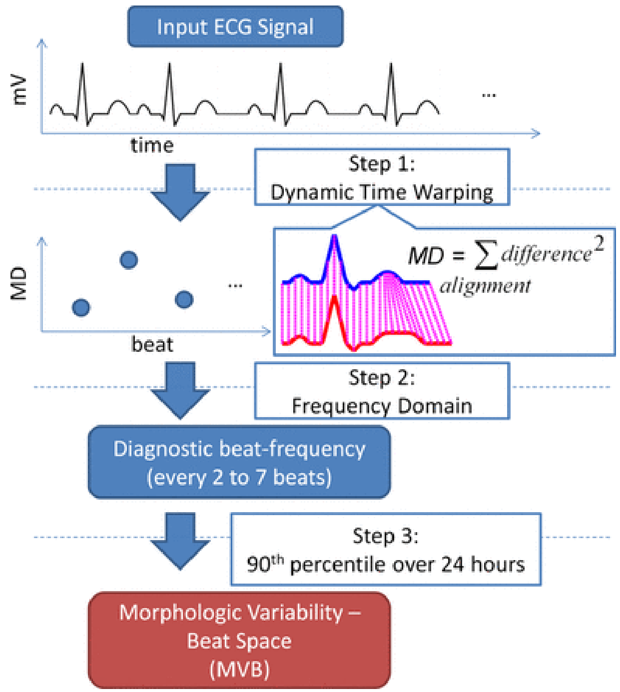
图B：2008年5月流感爆发前几周传播情况



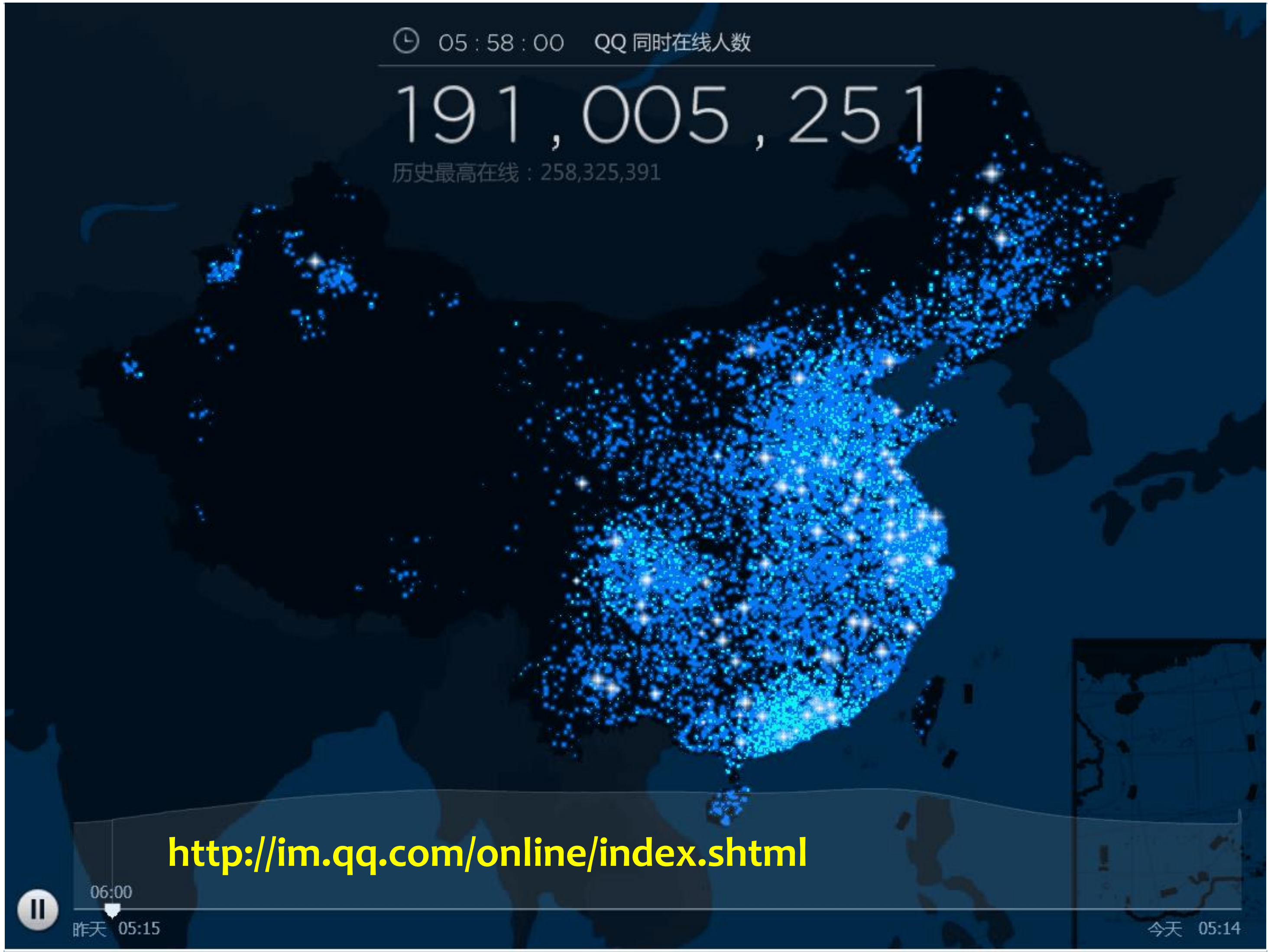
Forecasting iPhone sales by Google Trends

Wasted ECGs (心电图)

- Guttag J. & Stultz C. collected wasted ECGs.
- Image processing and mining
- 3 novel patterns of heart diseases.



QQ Census



05 : 58 : 00 QQ 同时在线人数
191,005,251
历史最高在线：258,325,391
http://im.qq.com/online/index.shtml
06:00
昨天 05:15
今天 05:14

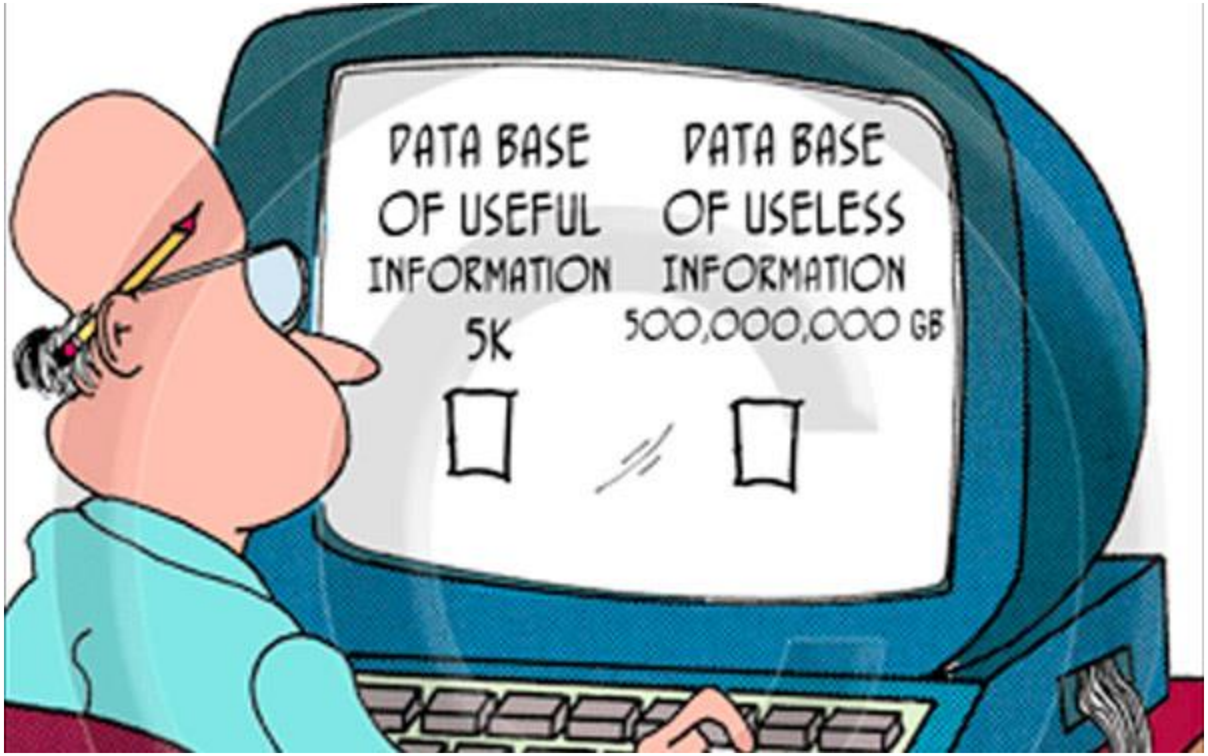
Baidu Migration

Baidu is automatically tracking:

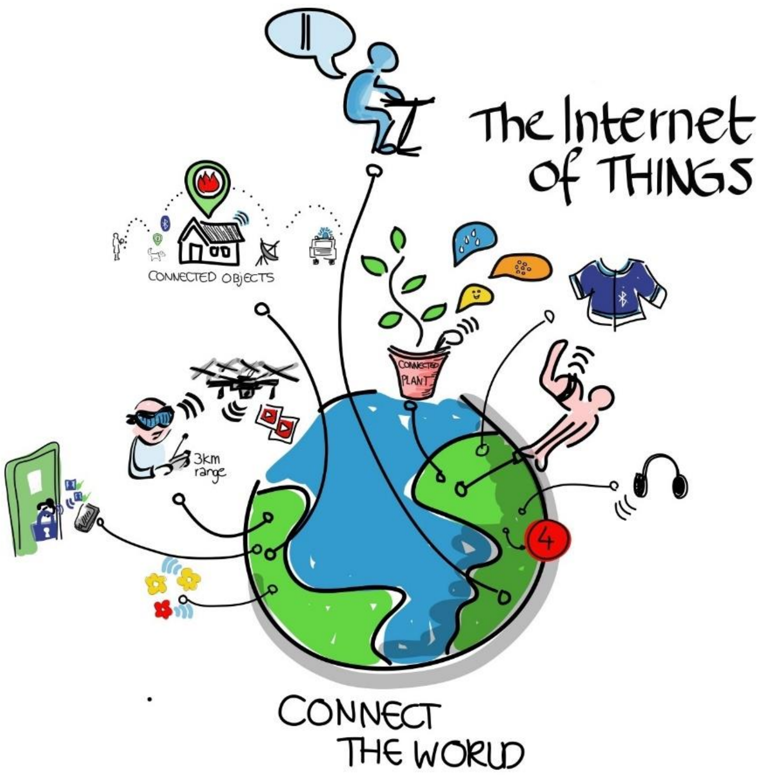
- 370m active users
- 337 cities
- 5m kms roads
- 95% highways
- 50 cities' real-time traffic
- 70,000 bus routes
- 1.8m terminals
- 0.5m partners
- 10 billions requests

http://qianxi.baidu.com

Why Big Data Now ?!



Big Data Collectable!



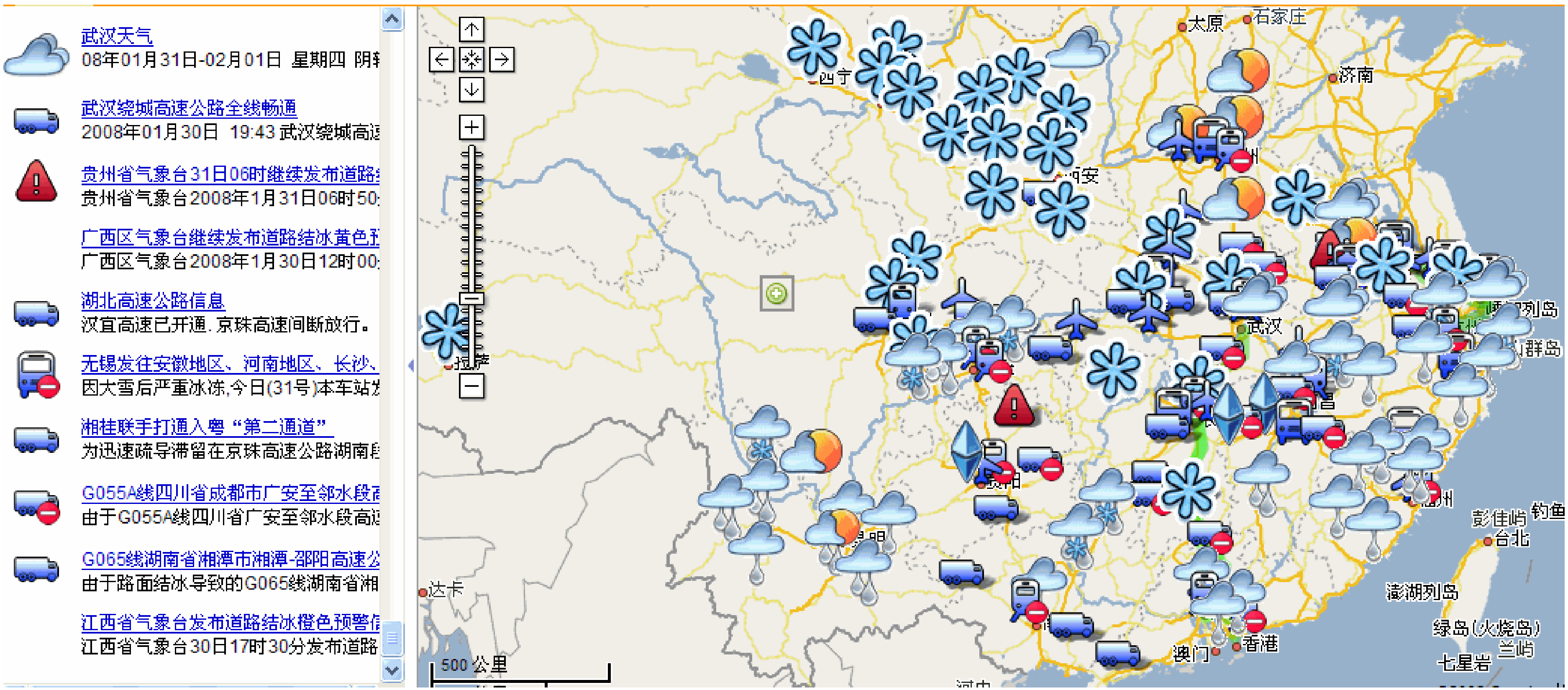
User Generated Contents **Web-of-Things Data**

Big Data Tractable!

- 【国际刑警组织否认其护照数据库使用缓慢】马来西亚内政部长扎希德·哈米迪于本周三向议会表示，国际刑警组织数据库所承受的繁重负担自然放缓了出入境检查。扎希德说， 国际刑警组织的4020万本丢失护照数据库实在是“太大”了 ，会使马来西亚的数据库管理系统陷于瘫痪。总部设在法国里昂的国际刑警组织表示，它的数据库仅需要0.2秒便可以向当局透露一本护照是否已被列为被盗。它补充说，还没有一个成员国抱怨过该过程过于缓慢。尽管其他几个国家每年使用该数据库数百万次，在马航370客机3月8日失踪前，马来西亚的移民部门在今年一次也没有使用其数据库检查飞机乘客的护照。

Google Data Center

About 2008 Google 春运 （Spring Festival Transportation）



Google Family

Core Competence?

Tsinghua-SEM

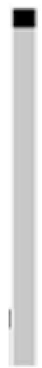
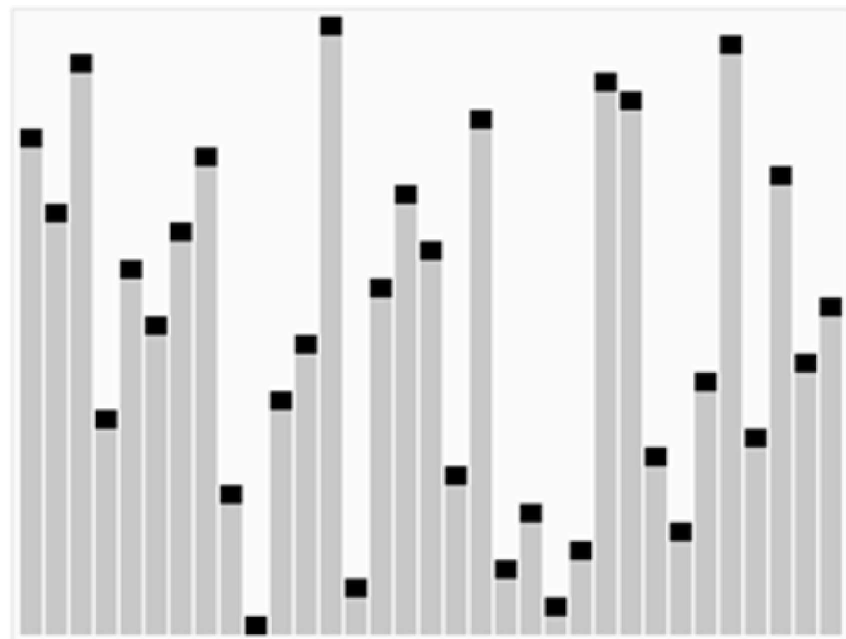
Core Competence of Google

SEARCH

TIME EFFICIENCY

SPACE EFFICIENCY

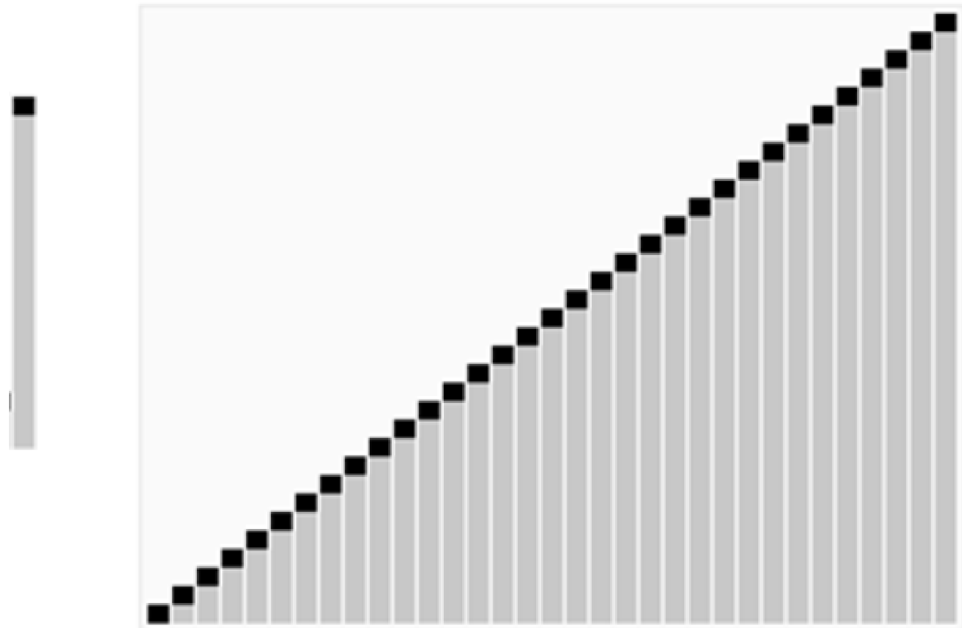
How to Search Efficiently?

How to efficiently search  *in*  *?*

Compare one by one ?!

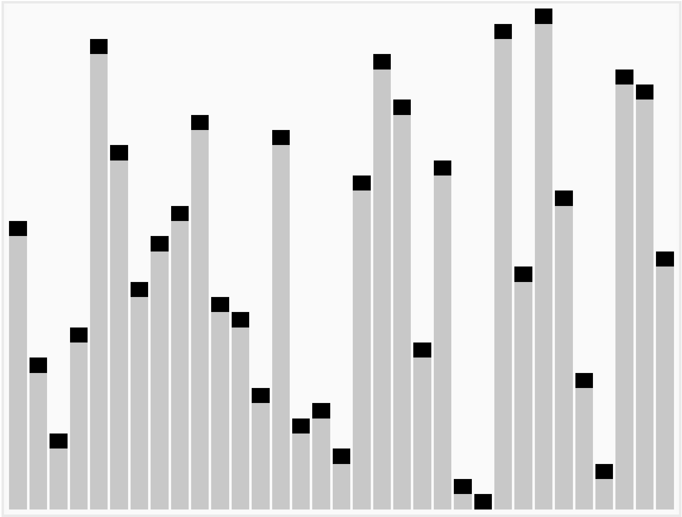
Search based on Sorted Results

- After sorting

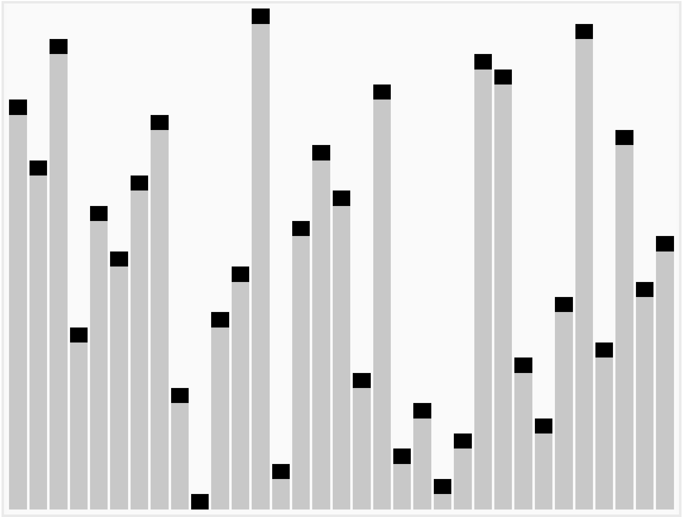


Binary Search (折半查找)

How to Sort Efficiently?

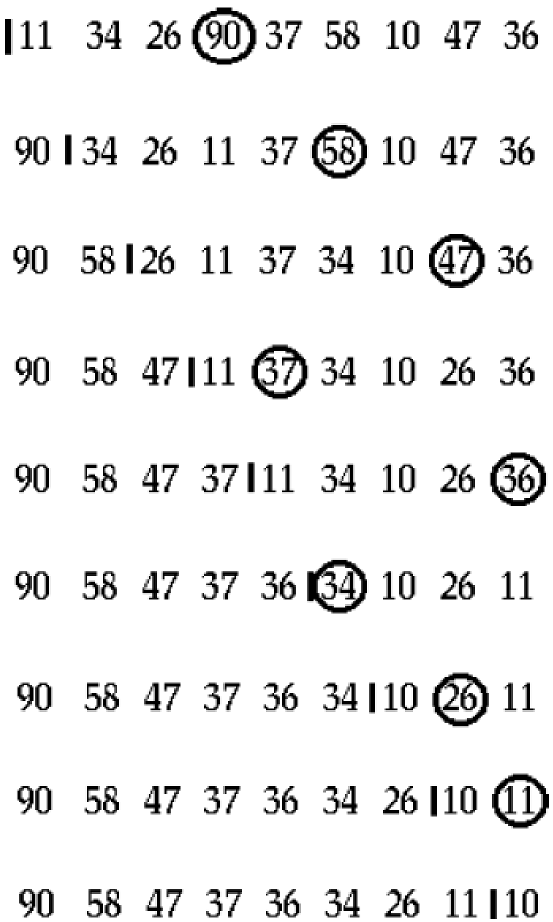


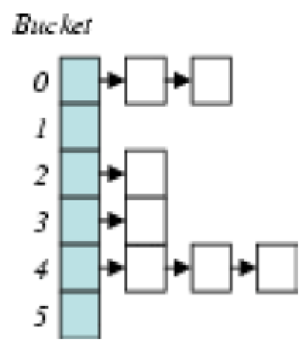
Quick Sort Algorithm

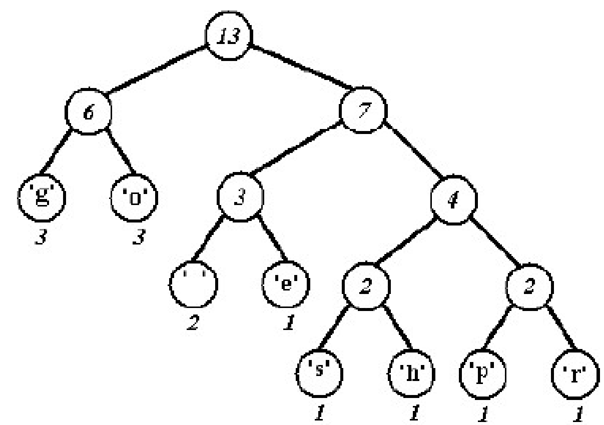


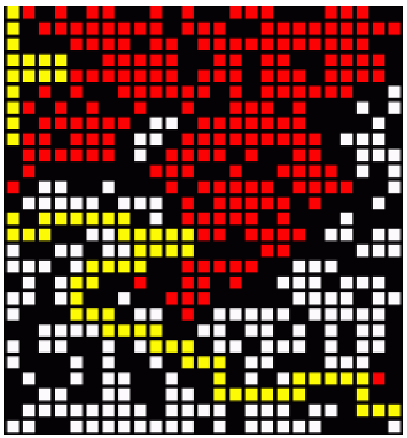
Heap Sort Algorithm

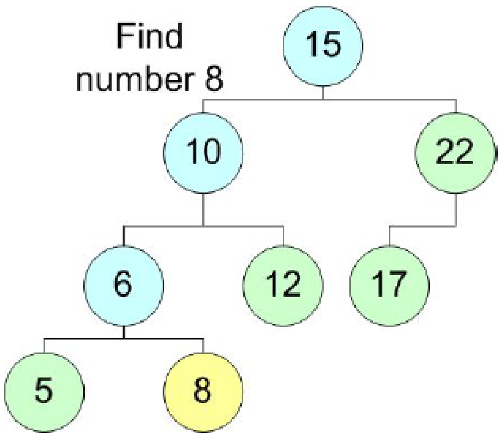
Data Structures (Narrow Definition)





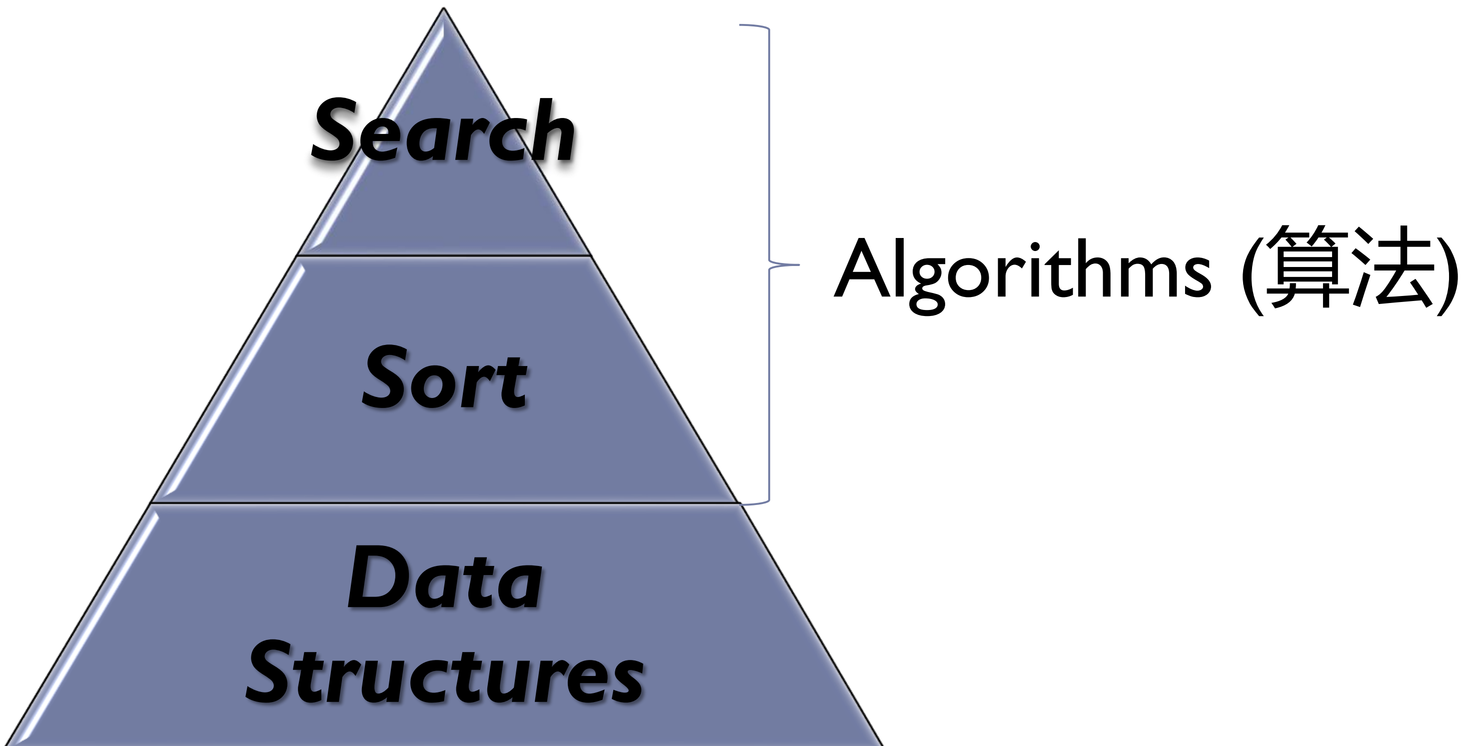






Data Structures (Generalized Definition)

Data Structures and Algorithms (DSA)

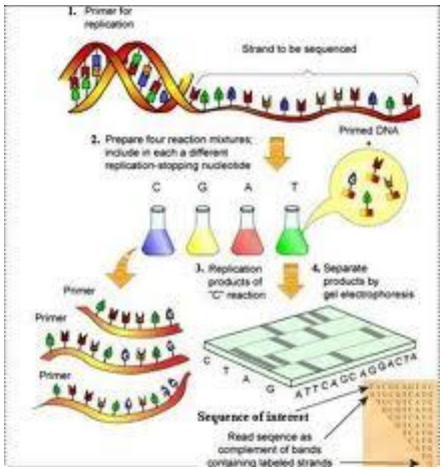


From Al-khwarizmi to Algorithm

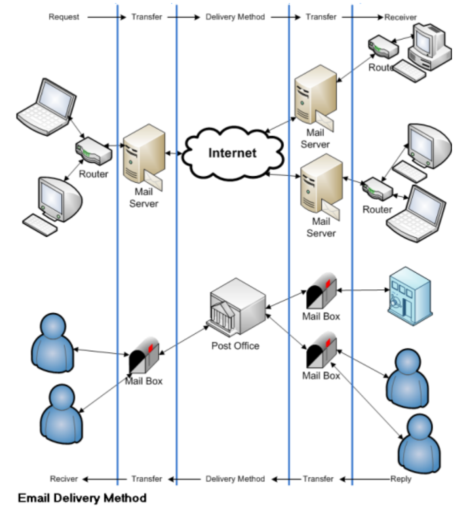
- Abū Abdallāh Muḥammad ibn Mūsā al-Khwārizmī (c. 780, Khwārizm – c. 850) was a Persian mathematician, astronomer and geographer, a scholar in the House of Wisdom in Baghdad.
- His ***Kitab al-Jabr wa-l-Muqabala*** presented the first systematic solution of linear and quadratic equations. He is considered the founder of algebra, a credit he shares with Diophantus (丢番图). In the twelfth century, Latin translations of his work on the Indian numerals, introduced the decimal positional number system to the Western world. He revised Ptolemy's Geography and wrote on astronomy and astrology.
- His contributions had a great impact on language. "Algebra" is derived from al-jabr, one of the two operations he used to solve quadratic equations. Algorism and algorithm stem from Algoritmi, the Latin form of his name. His name is the origin of (Spanish) guarismo and of (Portuguese) algarismo, both meaning digit.

Source: http://en.wikipedia.org/

More Than Sorting Numbers ……

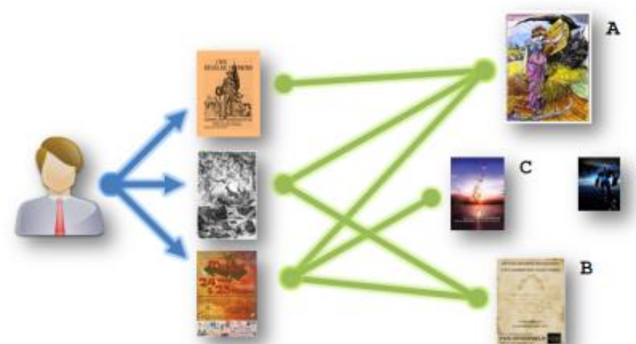


DNA Sequencing



Email routing optimization

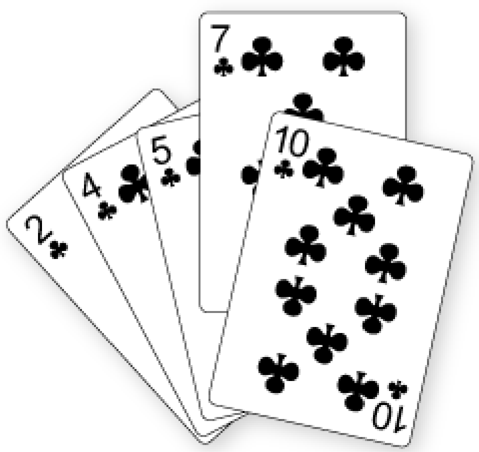
Search Engine



Online Recommender System

Algorithm is not as simple as we think

- How to insert a card properly?
  - Locate the appropriate position;
  - Make a space;
  - Insert the new card.



Getting into the Course

What is This Course All About

Methodology

Learn to practice, practice to learn and have fun!

Algorithms

Step-by-step procedure to perform some task in a finite amount of time

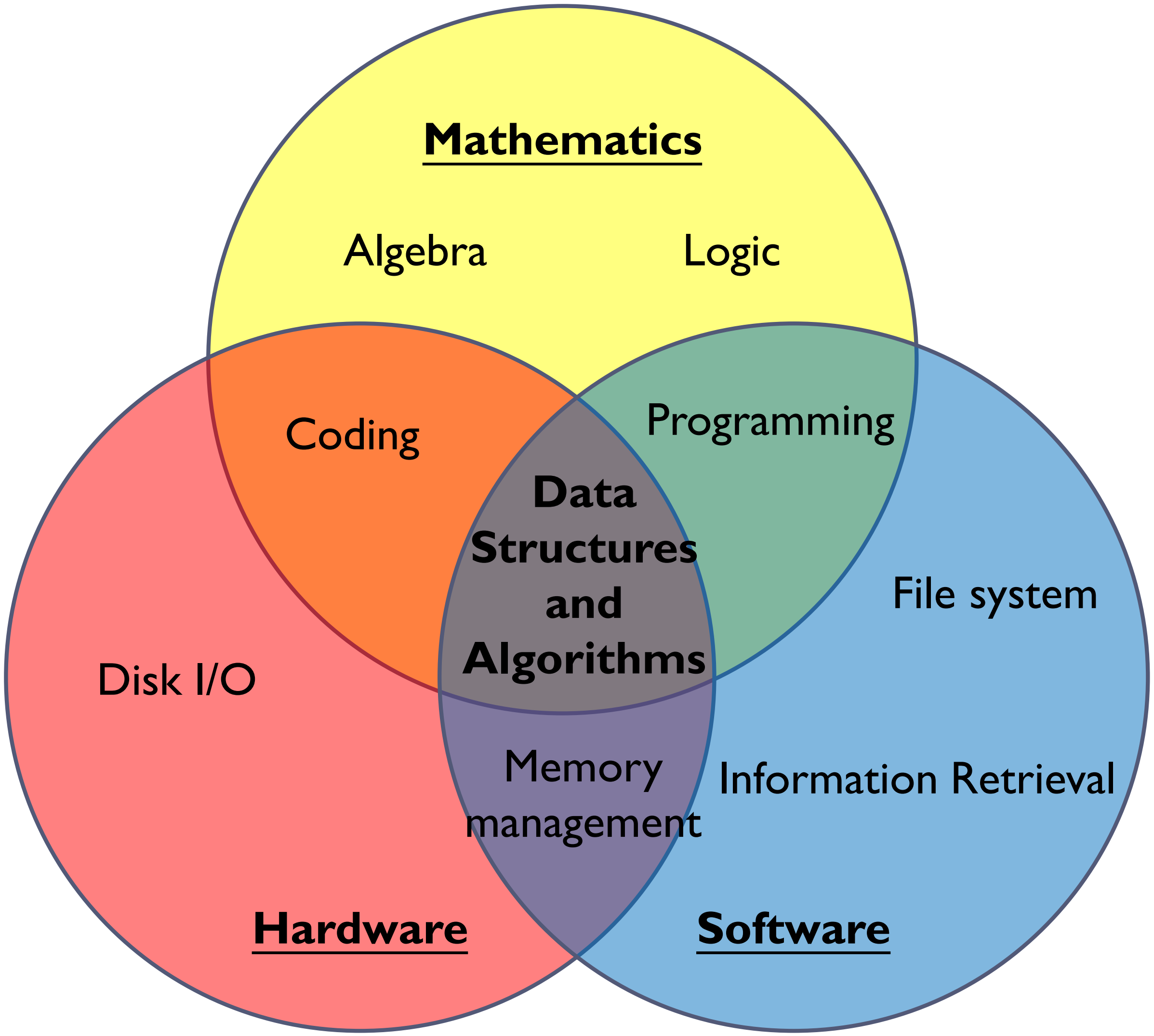
Data Structures

Systematic way of organizing and accessing a large amount of data

What We Want to Achieve

- Understand fundamental data structures and their relationships to algorithms
- Appreciate the interplay between data structures and algorithms
- Become a skilled modelers
- Become a skilled C-based analyst

The Position of DSA



Data Structures and Algorithms (DSA)

- Algorithm: Idea behind program, sequence of steps for solving problems
  - Finite
  - Well-defined
  - Each step tractable

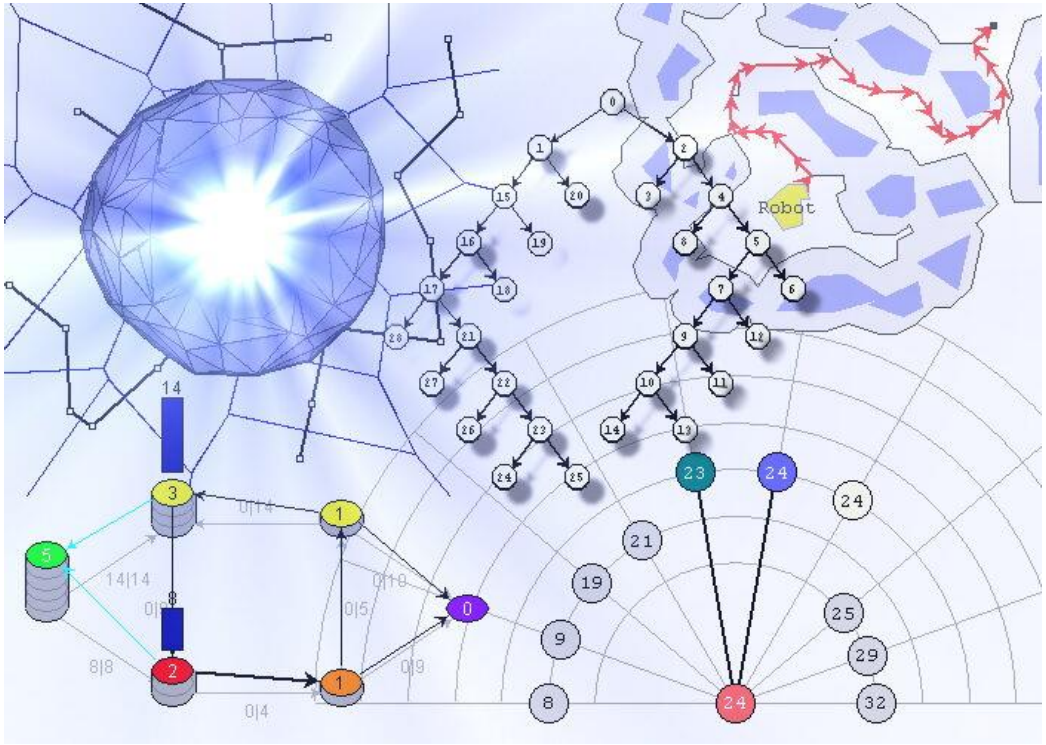
    **** (problems CANNOT be "compute π exactly" or "find meaning of life")***

- Algorithm is a mapping (映射):
  - A: {inputs} → {outputs}
  - Outputs could be bits/decisions: prime or not?

    or larger structure: position of number; sorted sequence.

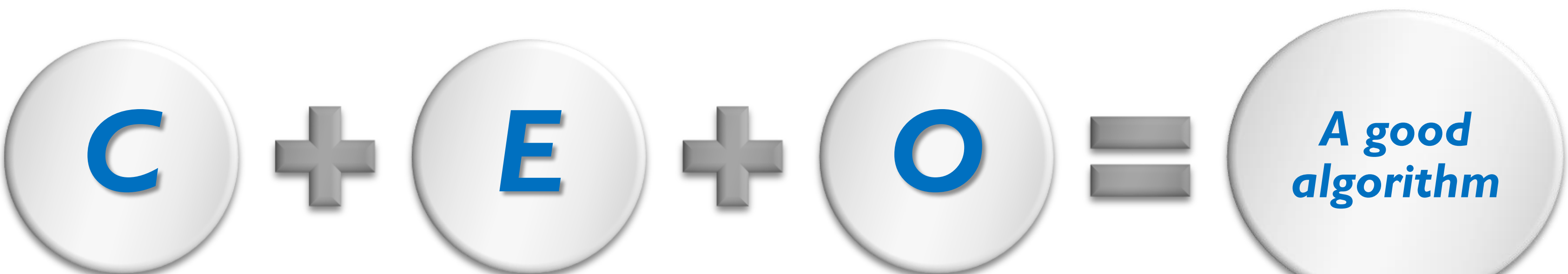
DSA (continued…)

- More than 90% problems could be modeled with data structures, if computer systems are needed, e.g.,
  - Kepler's Equations
  - Inventory management system
  - Alipay
  - Sort 2 numbers? 100 numbers? … N numbers?
  - ……



Algorithm's Goals

- Ideally, a good algorithm should be **CEO**:



Correct/ Effective

Efficient

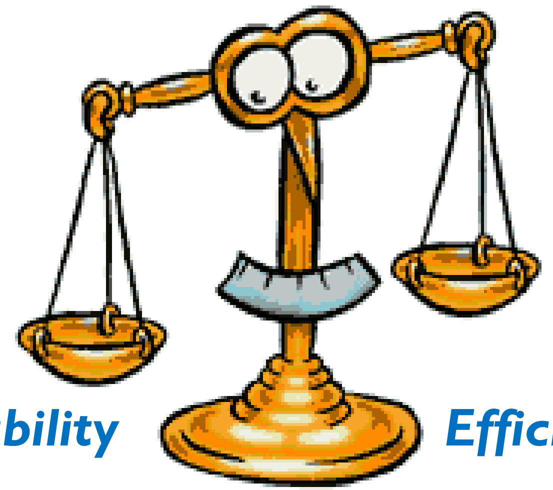
or at least Optimal

Each of the above three criteria could be difficult to achieve.

Algorithm’s Goals (continued…)

- Sometimes, a challenge between ***understandability*** and ***efficiency***. Generally,
  - Easy-to-understand algorithm may always be slow (inefficient).
  - Fast algorithm may be difficult to understand (or to find).

- Not really surprising: “obvious” solution is often slow.



Example 1

- *K*th largest selection problem:
  - Given N numbers, try to detect the *k*th largest.
  - E.g., given 7, 6, 9, 3, 5, 2, 4, 8, 1, find the 3rd largest,
  - which is actually 7.

- Naïve (straightforward) Strategy 1:
  - Read the N numbers in an array (if possible);
  - Sort them (in a descent order);
  - Return the element in position *k*.

Example 1 (continued…)

- Strategy 2 (Better):
  - Read the first *k* elements in an array;
  - Sort the *k* elements (in decreasing order);
  - Read each remaining element in a temporary variable ***a*** one by one;
    - Compare the new element ***a*** with the *k*th element ***b*** in the array:
      - □ If ***a*** <= ***b***, do nothing;
      - □ If ***a*** > ***b***, then place ***a*** in the correct spot and update in the array and bump the old ***b*** out of the array.

- ***Better because of less sorting operations.***

Example 1 (continued…)

- For example, given the following array with 10 elements:

  {18, 32, 97, 65, 24, 75, 54, 9, 47, 85}

- Try to find the 3rd largest number.

- Strategy 1:
  - Sort 10 elements;
  - Return the 3rd element.

- Strategy 2:
  - Sort the first 3 elements;
  - Compare the rest 7 elements one by one and place into the correct spot if possible.
  - Return the 3rd element.

Example 2

- Pick all the prime integers (素数) among 1 to N.

- Naïve Strategy:
  - For each integer i ($1 < i \le N$), compute whether *mod*(i, p) = 0, where p = 1, 2, …, $\lfloor sqrt(i) \rfloor$.

* $\lfloor x \rfloor$ = the largest integer which is smaller than x, e.g., $\lfloor 8.2 \rfloor = 8$, $\lfloor 78.9 \rfloor = 78$. For simplicity, since only integer numbers will be discussed in this course, we will use |x| to represent $\lfloor x \rfloor$ in the rest of the course.

Example 2 (continued…)

- Elimination Strategy (better):
  - (1) i = 2;
  - (2) If i < N, eliminate all the i*k <= N, k = 2, 3, …., otherwise terminate and output the un-eliminated numbers which are prime.
  - (3) Let i = next un-eliminated number and go to (2).

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, ...
2, 3, , 5, , 7, , 9, , 11, , 13, , 15, , 17, , 19, , 21, , 23, , 25, , ...
2, 3, , 5, , 7, , , , 11, , 13, , , , 17, , 19, , , , 23, , 25, , ...
2, 3, , 5, , 7, , , , 11, , 13, , , , 17, , 19, , , , 23, , , , ...

……

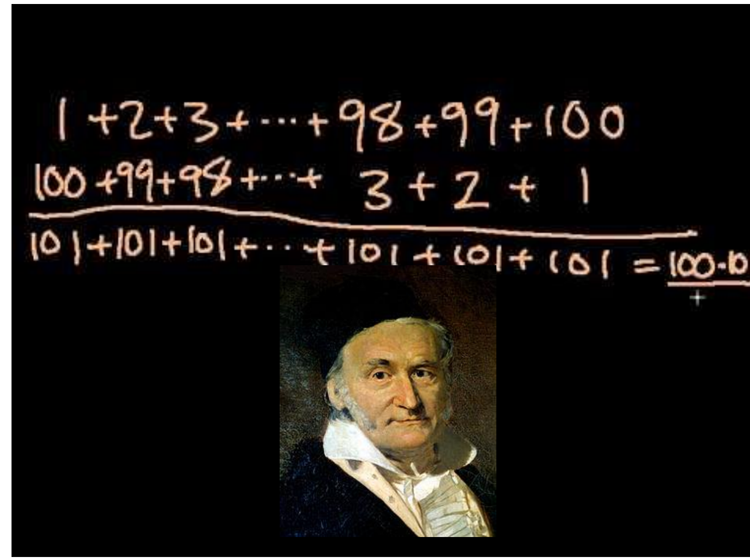
Example 2 (continued…)

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Example 3

- Sum of integers: add 1 to N (N > 1), what is the sum?



- 2 algorithmic strategies:
  - Strategy 1: 1+2+3+4+5+…
  - Strategy 2: Gauss' method
- A Question raised: Which strategy is better?

It seems that Strategy 2 is better.

But if you think as a machine …

Example 3 (continued…)

- Strategy 1 Processing details
  - Read 1 integer then add 1 integer
- Time Cost/Complexity
  - N times of read operations, i.e., $N*t_r$
  - (N – 1) times of add operations, i.e., $(N-1)*t_a$
  - Roughly, $N(t_r + t_a)$.

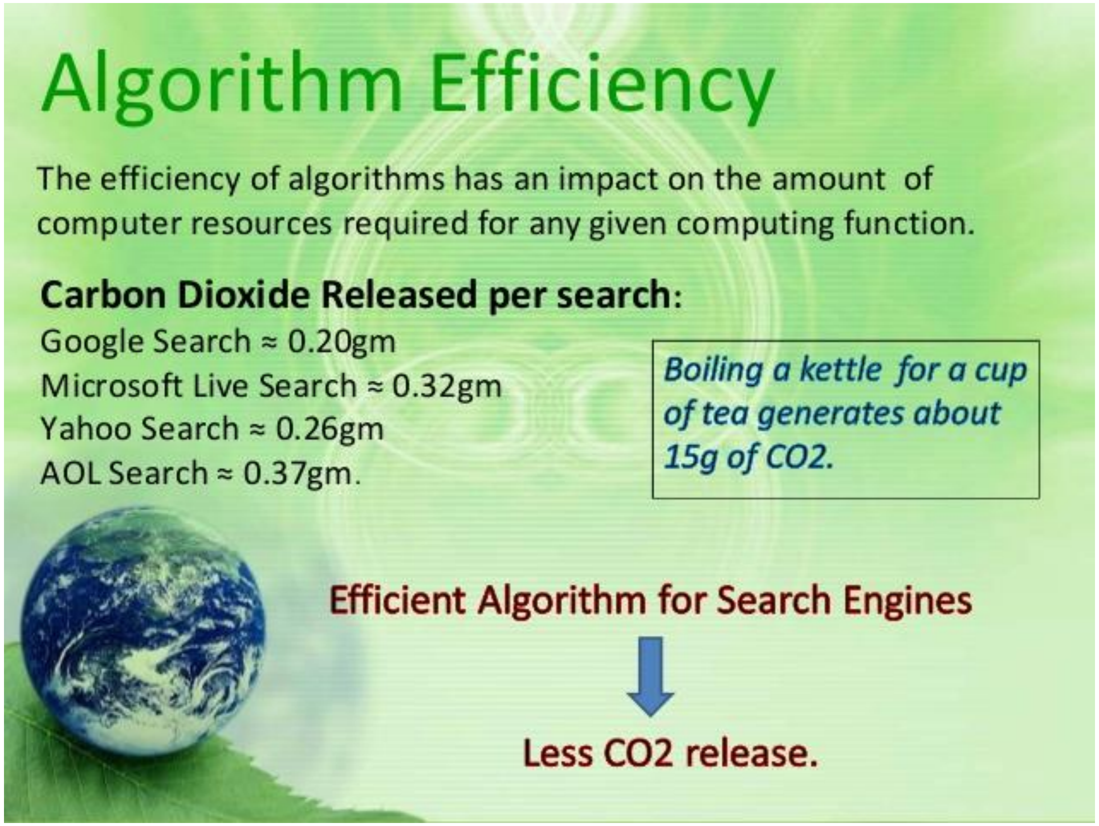
- Strategy 2 Processing details
  - Read N integers to count total number
  - Compute (1+N)/2
  - Compute (1+N)/2*N
- Time Cost/Complexity
  - N times of read operations, e.g., $N*t_r$
  - Other operations, e.g., $2*t_o$
  - Totally, $Nt_r + 2t_o$.

Example 3 (continued…)

- Normally, t_r, t_a, t_o and 2 are regarded as constants.
- What if N is very large
  - Compared with N, the values of t_r, t_a, t_o and 2 can be regarded as small volumes.
  - So both of the complexity of Strategy 1 and Strategy 2 are on O(N) level.
  - Strategy 2 is not so good as we expect…
- For example, if N is large enough,
  - $O(N^3 + 2N^2 + 8N) = O(N^3)$.

Large-Scaled Data Environment

- Algorithm's efficiency is one of the key factors for seizing competitive advantage.
  - How to build appropriate data structures.
  - How to correspondingly construct efficient algorithms.



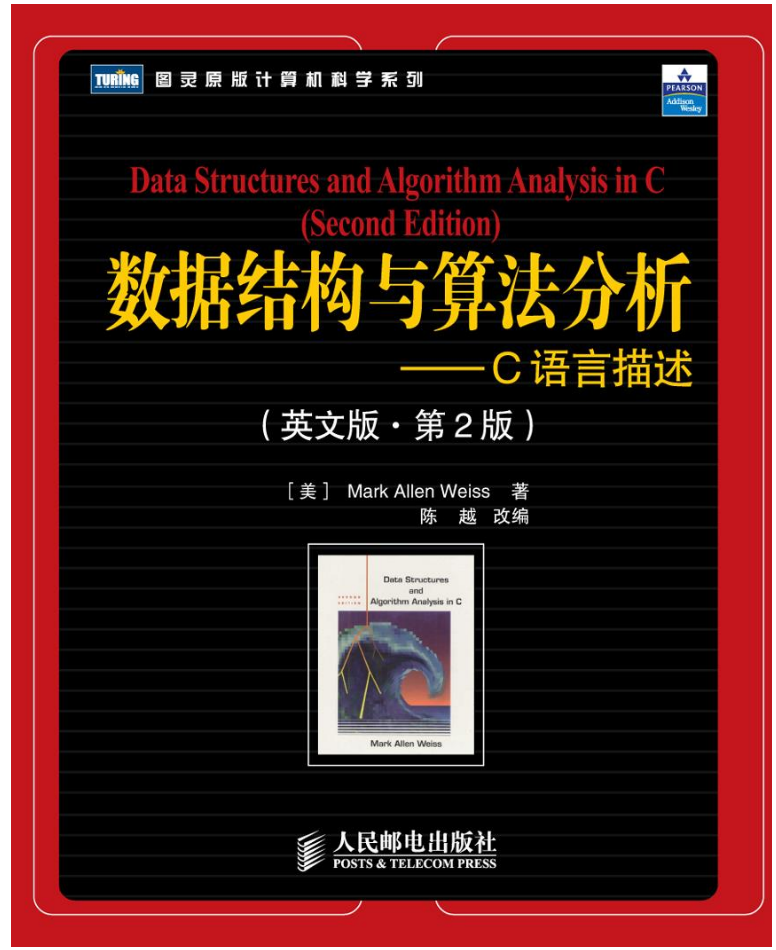
About the Course

Tsinghua-SEM

Course Information

- Lecturer: WEI Qiang (卫强)
  - Email address: **weiq@sem.tsinghua.edu.cn**
  - Office: Room 443, Weilun Building, 62789824
  - Info system!

- Textbook:
  - 数据结构与算法分析, Data Structures and Algorithms Analysis in C, Mark Allen Weiss, (英文版)。



Tsinghua-SEM

Agenda

- 01 – Opening Remarks
- 02 – C Basis: Pointer and Structure
- 03 – Mathematical Basis and Algorithm Analysis
- 04 – Lists, Stacks and Queues (1)
- 05 – Lists, Stacks and Queues (2)
- 06 – Google PageRank
- 07 – Trees (1)
- 08 – Trees (2)
- 09 – Sorting (1)
- 10 – Sorting (2)
- 11 – Hashing & Encryption
- 12 – Graph Algorithms (1)
- 13 – Graph Algorithms (2) + Social Network Analysis
- 14 – Patten Matching & AI
- 15 – Summary

Grading

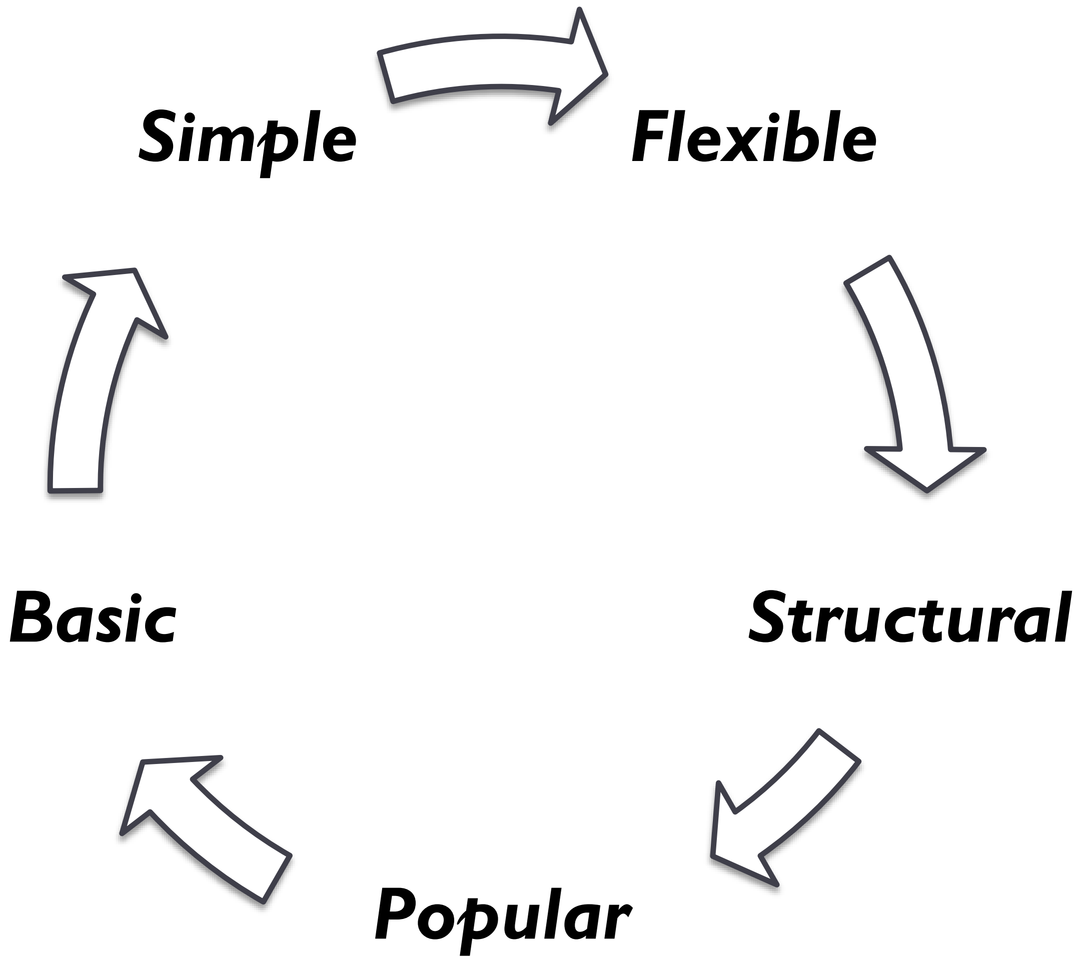
- Prerequisite: C/C++/Java programming

- Grading:
  - Homework: 40%
  - Participation/Attendance: 10%
  - Final (closed book): 50%

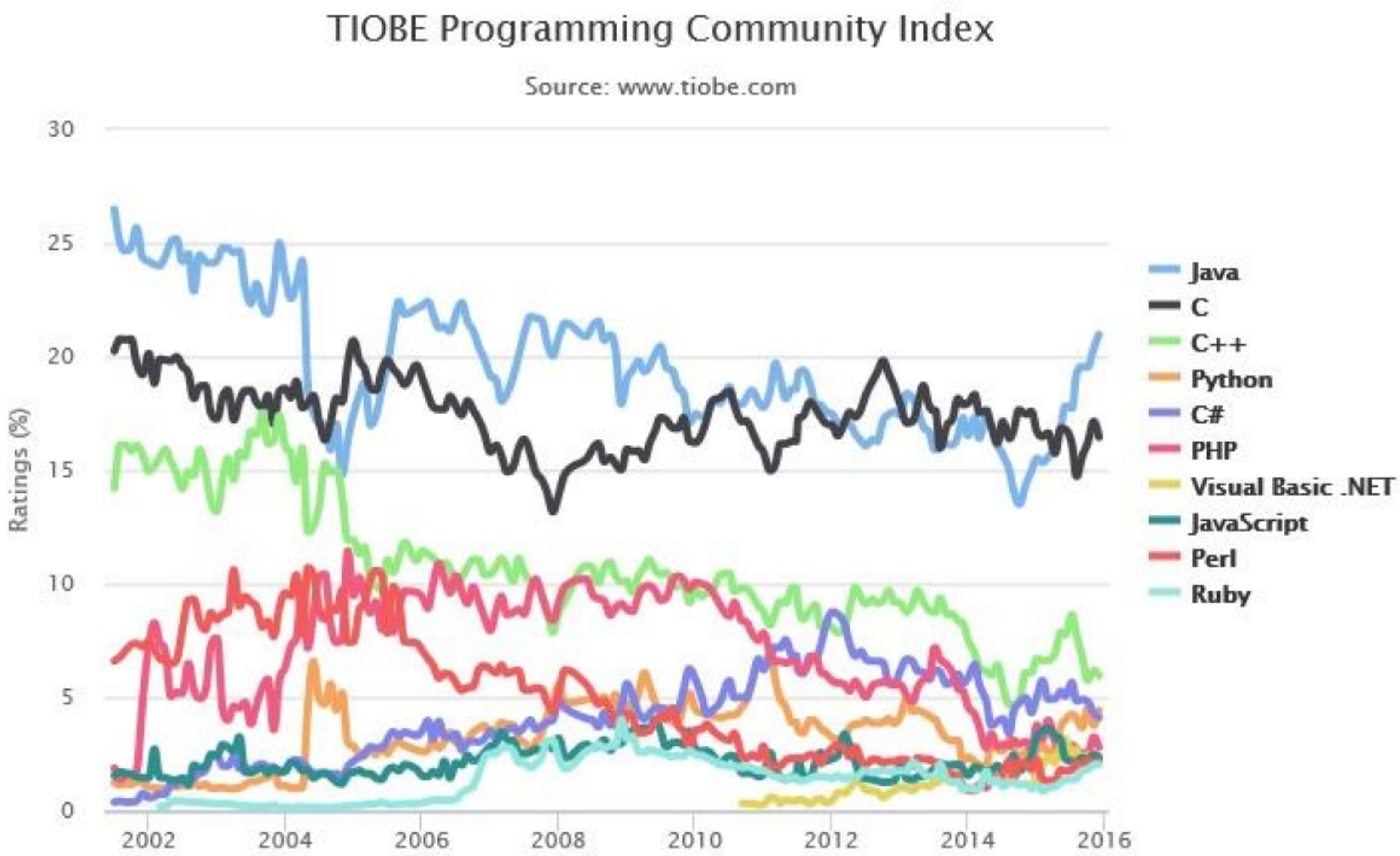
- Attendance required
- Homework done individually
- Non-cited use of others' solutions, codes, etc.

C Programming Revisited

Characteristics of C



Popularity of C Family Language



Development Platform

- Turbo C++
- Visual C++
- Borland C++
- Etc.

C Programs

- Output "Hello, world!" on screen.

Start point of any C programs

Output control code

```
void main()
{
    printf("Hello, world!\n");
    getchar();
}
```

For windows specifically

C Programs (continued…)

- C Program for calculating the sin() value of an inputted value.

```
// include the necessary header files, i.e., *.h
#include <math.h>
#include <stdio.h>

void main()
{  double x,s;                  // define 2 double float variables
   printf("input a angle:");    // display the prompt
   scanf("%1f",&x);             // get a double float number
                                // from keyboard and store in x
   s = sin(x*3.14159265/180);  // calculate the sin value of x
   printf("sine of %1f is %1f\n", x, s);       // output to screen
   printf("Strike any key to continue!\n");
   getchar();                   // stay in DOS command window
}
```

C Programs (continued…)

- C program with user-defined function/routine to output the larger one in 2 numbers.

```
#include <stdio.h>
void main()
{  int x,y,z;   int max(int, int);
   printf("Input two numbers:\n");
   scanf("%d%d", &x, &y);
   z = max(x, y);        // call the user-defined routine/function
   printf("maximum = %d\n", z);
   printf("Strike any key to continued!\n");
   getchar();
}

int max(int a, int b)  // return the big one in two numbers
{  if (a > b) return a;        else return b;
}
```

Clock() Function

- Return the number of clock ticks used by the program.

- Synopsis:
  - `#include <time.h>        // don't forget to include <time.h>`
  - `clock_t clock(void);     // don't forget to declare as clock_t`

- Description:
  - The clock() function returns the number of clock ticks of processor time used by the program since it started executing. You can convert the number of ticks into seconds by dividing the value CLOCKS_PER_SEC.

Example 4: (1*1+1*2+…+1*10)+…+(10*1+…+10*10)

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>

void compute(void)
{ int i, j; double x = 0.0;
  for( i = 1; i <= 10; i++ )       // nested loop
  {    for( j = 1;  j <= 10;  j++ ) {  x = x + i * j ; }
  }
  printf ( "%16.7f\n", x );
}
void main(void)
{ clock_t start_time, end_time;
  start_time = clock();  compute();      end_time = clock();
  printf (“Time: %l seconds.\n", (end_time-
  start_time)/CLOCKS_PER_SEC);
  return (0);
}
```

Mathematics Review

Exponents (指数) Review

$$2^n + 2^n = 2^{n+1}$$

$$(x^a)^b = x^{(ab)}$$

$$x^a * x^b = x^{(a+b)}$$

$$x^a / x^b = x^{(a-b)}$$

$$x^n + x^n = 2x^n$$

Log (对数) Review

- In computer science and information systems, all log() are based to 2 unless specified otherwise.
- $\log_A B = \log B / \log A$
- $\log A + \log B = \log AB$
- $\log(A/B) = \log A - \log B$
- $\log(A^B) = B\log A$
- $\log X < X$ for all $X > 0$
- $\log 1 = 0$, $\log 2 = 1$, $\log 1024 = \log(2^{10}) = 10$, $\log(\text{million}) \approx 20$

Series (排列组合)

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

$$\sum_{i=0}^{N} i = \frac{N(N+1)}{2} \approx \frac{N^2}{2}$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1}$$

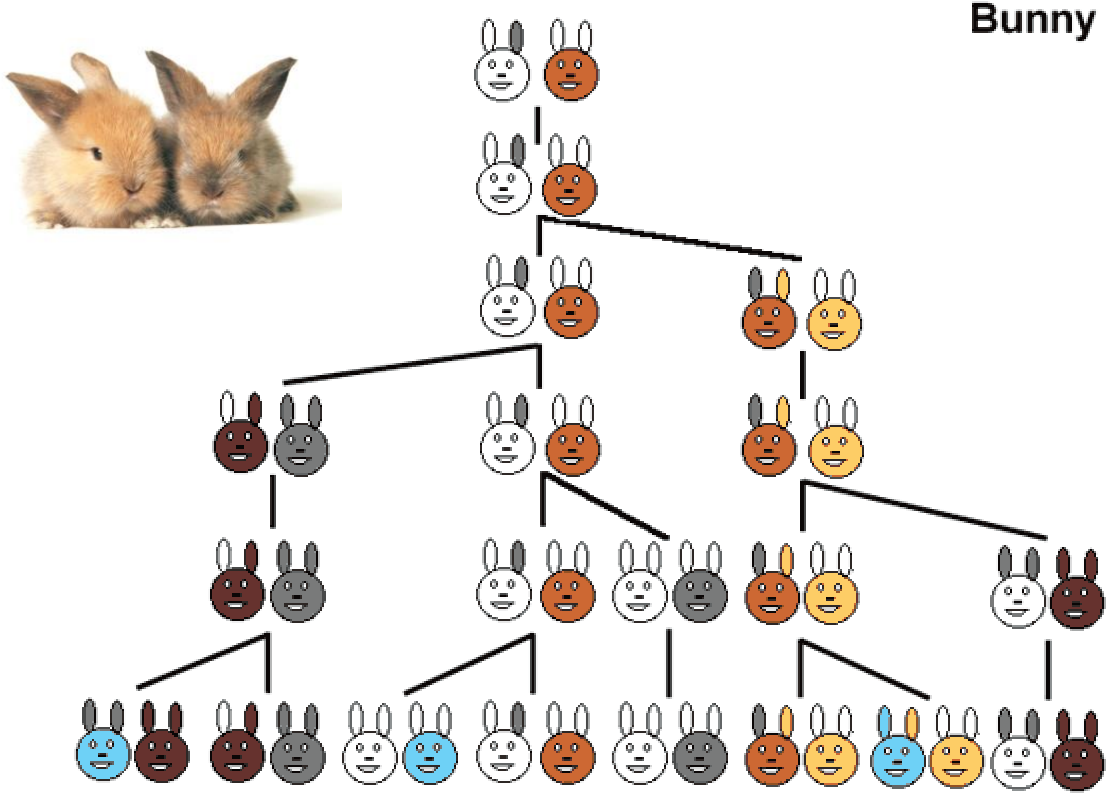
$$\sum_{i=0}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} \approx \frac{N^3}{3}$$

$$\sum_{i=0}^{\infty} a^i = \frac{1}{1-a}, if\ \ 0 < a < 1$$

$$\sum_{i=0}^{N} i^k \approx \frac{N^{k+1}}{|k+1|}, when\ k \neq -1$$

Sum of Harmonic numbers: $H_N \approx \sum_{i=1}^{N} \frac{1}{i} \approx \log_e N = \ln N$

Fibonacci Sequence (菲波纳契级数)



Bunny Pairs	Periods
1	0
1	1
2	2
3	3
5	4
8	5
……	……

Proof by Induction (归纳法)

- Given the Fibonacci (菲波纳契) numbers:
  - $F_0 = 1, F_1 = 1, F_2 = 2, F_3 = 3, F_4 = 5, \ldots$, where $F_i = F_{i-1} + F_{i-2}$

- Theorem: $F_i < (5/3)^i$.
- Proof:
  - $F_1 = 1 < 5/3, F_2 = 2 < (5/3)^2$. (inductive basis)
  - Assume the theorem is true for $i = 1, 2, \ldots, k$. (inductive assumption)
  - We have $F_{k+1} = F_k + F_{k-1}$, then
  - $F_{k+1} < (5/3)^k + (5/3)^{k-1}$
  - $F_{k+1} < (3/5)(5/3)^{k+1} + (3/5)^2(5/3)^{k+1}$
  - $F_{k+1} < (3/5 + 9/25)(5/3)^{k+1}$
  - $F_{k+1} < (24/25)(5/3)^{k+1}$
  - $F_{k+1} < (5/3)^{k+1}$. □

Tsinghua-SEM

A Brief Introduction to Recursion

A Regular Function/Routine

- Simple function/routine to convert degrees Fahrenheit to degrees Celsius, e.g.,
  - C = 5(F – 32)/9
- C code form:

```
float C, F;
F = 100.00;
C = 5*(F - 32)/9;
```

- C function form:

```
float convert(float F)
{	float C;
	C = 5*(F - 32)/9;
	return(C);
}
```

Recursion (递归)

- Many functions are not so simple, e.g.,
  - F(0) = 0, and
  - $F(X) = 2F(X-1) + X^2$.

- A function that is defined in terms of itself is called a ***recursive function***.

```
int F(int N)
{   if(N == 0)
      return 0;
    else
      return(2*F(N-1)+N^2);
}
```

Base case

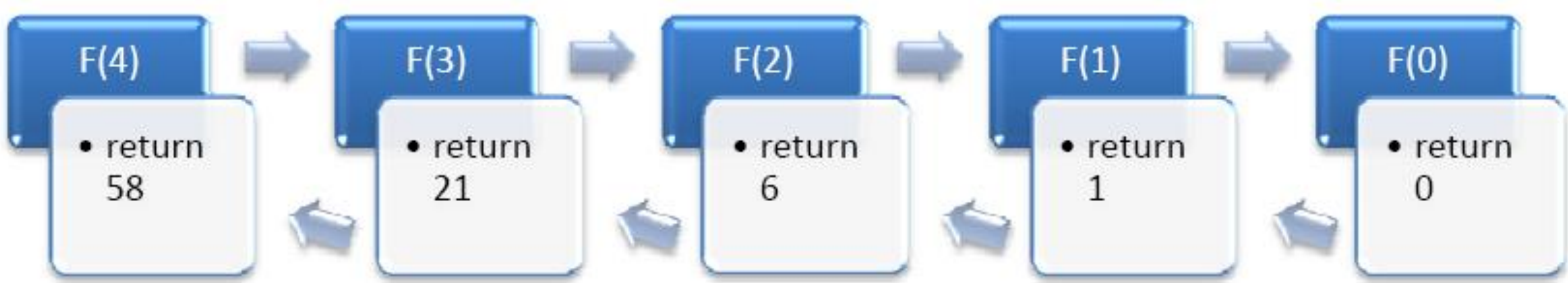
Make recursive call

Recursion (continued…)

- Recursion progress:

  e.g., call F(4)

```
int F(int N)
{   if(N == 0)
        return 0;
    else
        return(2*F(N-1)+N^2);
}
```



Non-terminating Pitfall!

- Check the terminating condition carefully!

```
int Bad(int N)
{     if(N == 0)
            return 0;
      else
            return Bad(N/3 + 1) + N - 1;
}
```

- If call Bad(1), then Bad(1) will be called again, again, …
  - Memory overflow error caused.
  - This frequently happens.

Example 5 – Fibonacci Numbers

- Fibonacci Numbers:
  - $F(1) = 1, F(2) = 1, F(N) = F(N-1) + F(N-2)$
  - The first two: Base cases

```
int FIB(int N)
{ if(N == 1 || N == 2) return (1);
  return(FIB(N - 1)+FIB(N - 2));
}
```

- Base: 1, 2 are correct
- Then $FIB(N) = FIB(N-1) + FIB(N-2)$

Problems with Recursion

- But is this routine good?
  - It is NOT GOOD!
  - What's the problem?
- Consider call FIB(-2) → infinite recursive loop
  - FIB(-2) calls FIB(-3) and FIB(-4); FIB(-3) calls FIB(-4) and FIB(-5), ……
- Also, consider time elapsed: ***Lots of duplications!***
  - To compute FIB(10), first do FIB(8) and FIB(9)
    - To compute FIB(9), first do FIB(8) (***again***) and FIB(7)
    - ……

Recursion Rules

- ***Base cases:*** You **MUST** always have some base cases, which can be solved without recursion.
- ***Making progress:*** For the cases that are to be solved recursively, the recursive call must always be to a case that **MAKES PROGRESS TOWARD** a base case.
- ***Design rule:*** **ASSUME** that all the recursive calls work.
- ***Compound interest rule:*** **NEVER DUPLICATE** work by solving the same instance of a problem in separate recursive calls.

Recursion Rules (continued…)

- Short version:
  - Must always reach one of the bases
  - Each step must make progress
  - In writing code, assume all previous steps come for free
  - Don't duplicate work
  - ***Don't use recursion unless you have to.***

- For the Fibonacci example
  - Recursion solution is correct (**effectiveness**) but duplicating too many (**low efficiency**).

Recursion vs. Iteration (迭代)

- **Q: When is recursion necessary?**
- **A: NEVER !!!!**

- "Proof": Any recursion will be "stripped" out by complier!

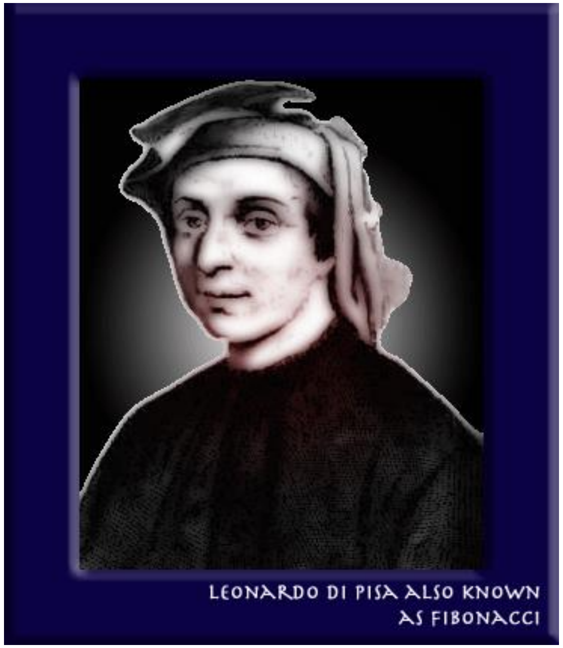
- Why still use?
  - ***(The codes are) often easier, more elegant!***

Iteration Fibonacci

- Better **NOT** use recursion for Fibonacci.

Iteration version is better than recursion version.

```
int FIB2(int N)
{       int prev1 = 1, prev2 = 1;
        int curr = 2, temp;
        while(curr < N)
        {       temp = prev1 + prev2;
                prev1 = prev2;
                prev2 = temp;
                curr ++;
        }
        return(prev1);
}
```



Summary

Summary

- DSA is one of the computing basis for IT era.
- Big data → Complexity & Efficiency
- C language is the tool for the course.
- Recursion is very useful technique in algorithm design, though not necessary.