香港中文大學（深圳）
The Chinese University of Hong Kong

CSC3100 Data Structures
Course information

Yixiang Fang
School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

About instructor (Yixiang Fang)

- Background
  - PhD (2013-2017): HKU
  - Postdoc (2017-2018): HKU
  - Postdoc (2018-2020): UNSW
  - Associate Professor (2021-present): SDS, CUHK-SZ
- Contact
  - Email: fangyixiang@cuhk.edu.cn
  - Office: Room 417d, Dao Yuan Building
- Research
  - General area: querying, mining, and analytics of big data
  - Topics: graph queries, graph mining, social network analysis, etc.

Teaching location and time

- Teaching location
  - Room 101, Cheng Dao Building

- Teaching time
  - 6 hours/week
  - 10:30-12:20 on Mon/Wed/Fri
    - 10:30-11:20,
    - 11:30-12:20

A break of 10 mins

		30	31	1	2	3	4	5
	1	6	7	8	9	10	11	12
2021年	2	13	14	15	16	17	18	19
6月	3	20	21	22	23	24	25	26
	4	27	28	29	30	1	2	3
	5	4	5	6	7	8	9	10
2021年	6	11	12	13	14	15	16	17
7月	7	18	19	20	21	22	23	24
		25	26	27	28	29	30	31

Office hours

- Instructor
  - Time: 5:00~6:00pm on every Tuesday (except June 22)
  - Location: Room 417d, Dao Yuan Building

- TAs: help you with coursework and technical issues
  - Class TAs: answer questions in classes (raise hands!)

  - Office TAs: answer questions in offices

Please use BB as much as possible for daily discussions!

Office TAs

Name	Office	Time	Email
Panwen Hu (Leading TA)	Room 502, Zhi Ren Building zoom id: 461 695 1576	Tue, 9:00-10:00 am	panwenhu@link.cuhk.edu.cn
Yujian Zheng	Room 225, Dao Yuan Building zoom id: 598 072 8539, zoom pwd: CSC3100	Mon, 4:00-5:00 pm	yujianzheng@link.cuhk.edu.cn
Zizheng Yan	Room 203B, Zhi Xin Building zoom id: 454 103 1002	Thu, 4:00-5:00 pm	zizhengyan@link.cuhk.edu.cn
Xingchao Wang	Room 501, Zhi Ren Building	Mon, 2:00-3:00 pm	xingchaowang@link.cuhk.edu.cn
Chaoda Zheng	Room 204B, Zhi Xin Building	Wed, 4:00-5:00 pm	220019043@link.cuhk.edu.cn

Tutorials

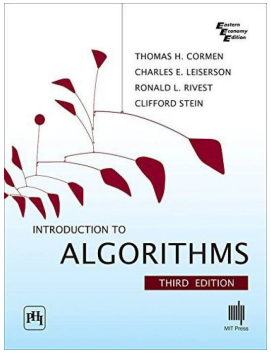
Time	Monday Jun 7	Tuesday Jun 8	Wednesday Jun 9	Thursday Jun 10	Friday Jun 11	Saturday Jun 12	Sunday Jun 13
08:30							
09:30							
10:30	CSC 3100 - L01 Lecture 10:30 - 12:20 Cheng Dao Building 101		CSC 3100 - L01 Lecture 10:30 - 12:20 Cheng Dao Building 101		CSC 3100 - L01 Lecture 10:30 - 12:20 Cheng Dao Building 101		
11:30							
12:30							
13:30							
14:30							
15:30							
16:30							
17:30	CSC 3100 - T01 Tutorial 18:00 - 18:50 Teaching D Buidling 111		CSC 3100 - T01 Tutorial 18:00 - 18:50 Teaching D Buidling 111				
18:30	CSC 3100 - T01 Tutorial 18:00 - 18:50 Teaching D Buidling 111 CSC 3100 - T03 Tutorial 19:00 - 19:50 Teaching D Buidling 111	CSC 3100 - T05 Tutorial 19:00 - 19:50 Teaching D Buidling 111	CSC 3100 - T01 Tutorial 18:00 - 18:50 Teaching D Buidling 111 CSC 3100 - T03 Tutorial 19:00 - 19:50 Teaching D Buidling 111	CSC 3100 - T05 Tutorial 19:00 - 19:50 Teaching D Buidling 111			
19:30	CSC 3100 - T03 Tutorial 19:00 - 19:50 Teaching D Buidling 111 CSC 3100 - T02 Tutorial 20:00 - 20:50 Teaching D Buidling 111	CSC 3100 - T05 Tutorial 19:00 - 19:50 Teaching D Buidling 111 CSC 3100 - T04 Tutorial 20:00 - 20:50 Teaching D Buidling 111	CSC 3100 - T03 Tutorial 19:00 - 19:50 Teaching D Buidling 111 CSC 3100 - T02 Tutorial 20:00 - 20:50 Teaching D Buidling 111	CSC 3100 - T05 Tutorial 19:00 - 19:50 Teaching D Buidling 111 CSC 3100 - T04 Tutorial 20:00 - 20:50 Teaching D Buidling 111			
20:30	CSC 3100 - T02 Tutorial 20:00 - 20:50 Teaching D Buidling 111	CSC 3100 - T04 Tutorial 20:00 - 20:50 Teaching D Buidling 111	CSC 3100 - T02 Tutorial 20:00 - 20:50 Teaching D Buidling 111	CSC 3100 - T04 Tutorial 20:00 - 20:50 Teaching D Buidling 111			
21:30							
22:00							

Mixed mode teaching

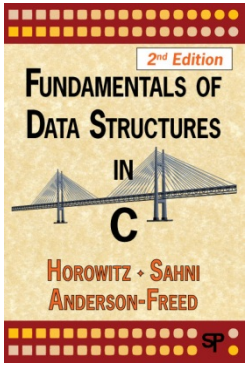
- Lectures on Zoom
  - ID: 949 5625 5767
  - Password: 123456
- Tutorials on Zoom
  - ID: 924 042 4060
  - No password
- The videos of lectures and tutorials will be uploaded to zoom servers automatically

Textbook and references

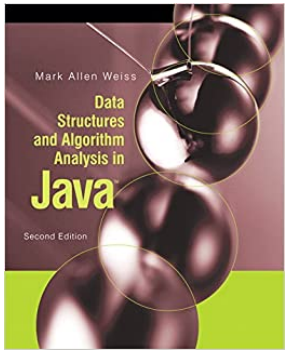
- Textbook:
  - T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, The MIT Press, 2009.



- References:
  - Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed, Fundamentals of Data Structures in C.



  - M.A. Weiss, Data Structures and Algorithm Analysis in Java, 2nd Edition, Addison-Wesley, 2007.



Teaching speed

- 7 weeks instead of 14 weeks means
  - Same (or more) amount of content, but less time to digest the content
  - Finish 2x homework within the same amount of time
  - Longer class time per day and you can get tired or lost in the class easily if you are not focus, especially second half

Teaching methods

- Lectures (6 hours/week)
  - Important materials from the textbook and references will be covered in class
  - Feel free to interrupt to ask questions
  - Ensure you stay up with the material

- Tutorials (2 hours/week)
  - Illustrate more on the parts that are difficult to understand
  - Show students more examples
  - Discuss assignment/exercise
  - Answer students' questions

Students with diverse backgrounds

Students distribution w.r.t. years

Year	1st year	2nd year	3rd year	4th year	Total
Number	12.5%	70.7%	15.8%	1%	100%

- Students distribution w.r.t. programs
  - Bachelor of Business Admin: 7.5%
    - Economics: 4; Finance: 5; Global business studies: 1; Professional Accountancy: 1
  - Bachelor of Engineering: 45.8%
    - Common plan of BEng: 5; Common Plan of BEng SDS: 11; Computer Sci and Engineering: 39; Electronic Info Engineering: 14;
  - Bachelor of Science: 30%
    - Bioinformatics: 3; Mathematics and Applied Mathem: 22; Statistics: 21;
  - FE Bachelor of Business Admin: 16.7%
    - Financial Engineering: 28

Please try to consider your classmates before complaining

How to do well in this course?

- Common suggestions
  - Slides will be uploaded to BB before lectures; learn them in advance
  - Use examples to facilitate learning
  - Within limited time, think more, do more
  - Write more codes!!!

- Special suggestions
  - If you feel difficult,
    - Try to focus on the content of slides and keep asking questions (TAs and instructor are ready to answer your questions!)
  - If you feel easy,
    - Read more details (e.g., theoretical analysis) in the textbook
    - Use the learned techniques to solve ACM-ICPC questions

Learn more and practice more

- ACM-ICPC
  - ACM: Association for Computing Machinery
  - ICPC: International Collegiate Programming Contest

- Online Judge (OJ) systems
  - CUHKSZ online judges: https://oj.computercomity.com/ or oj.polarisstudio.cn
  - SJTU online judge: https://acm.sjtu.edu.cn/OnlineJudge/
  - ACM online judge: https://onlinejudge.org/

Course assessment

Homework 40% + Midterm exam 20% + Final exam 40%

	Release date	Due date	Weighting
Programming assignment 1	June 13	June 27	12%
Written assignment 1	June 20	July 4	8%
Written assignment 2	June 27	July 11	12%
Midterm exam	July 5	--	20%
Programming assignment 2	July 11	July 25	8%
Final exam	July 26-28	--	40%

*The above table is subject to change as the instructor sees fit.

Course policy

- Your work MUST be your own
  - Cheating is against "fair-play" and will not be tolerated under any circumstances
- Assignment
  - Penalty will be imposed on copying assignments; minimum penalty is zero mark for the assignments
  - There will be penalty for late submission:
    - 0~24 hours after deadline: final score = your score x **0.8**
    - 24~72 hours after deadline: final score = your score x **0.5**
    - 72+ hours after deadline: final score = your score x **0**
- Check important things from Blackboard!
  - Lecture notes, announcement, assignments, etc.

Extra bonuses

- Extra bonuses for students who perform very well
  - Recommend industry internship in BATH companies
  - Offer research assistant (RA) positions and you may have the chances to publish academic papers!
  - High admission chances for your PhD application in my research team
  - Write recommendation letters for your PhD applications

Students' feedback

- Feedback is important and also welcome!
  - Talk to me or send me emails
  - Talk to TAs or send them emails
  - Please complain to me before complaining to others
  - Questionnaire

Acknowledgements

Lecture materials:

Prof. Xiang Wan (CUHK-SZ)
Prof. Kaiming Shen (CUHK-SZ)
Prof. Minming Li (City University of Hong Kong)
Prof. Zengfeng Huang (Fudan University)
Prof. Jane You (The Hong Kong Polytechnic University)
Prof. Sibo Wang (The Chinese University of Hong Kong)

Tutorial materials:

Mr. Xingchao Wang (CUHK-SZ)

香港中文大學（深圳）
The Chinese University of Hong Kong

CSC3100 Data Structures
Lecture 1: Introduction

Yixiang Fang
School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

Outline

- Introduction
  - Why take this course?
  - Basic concepts, e.g., abstract data type
  - Relationship of algorithms, data structures, and programming
  - Mathematics review

Why take this course?

- Required course
  - Also very important and very useful
  - A fundamental course in computer science

- How useful?
  - e.g., validate one Chinese ID in 1.4 billion people
  - find the best driving route

- Learn to save the data (data structure) and manipulate the data (algorithms) effectively and efficiently

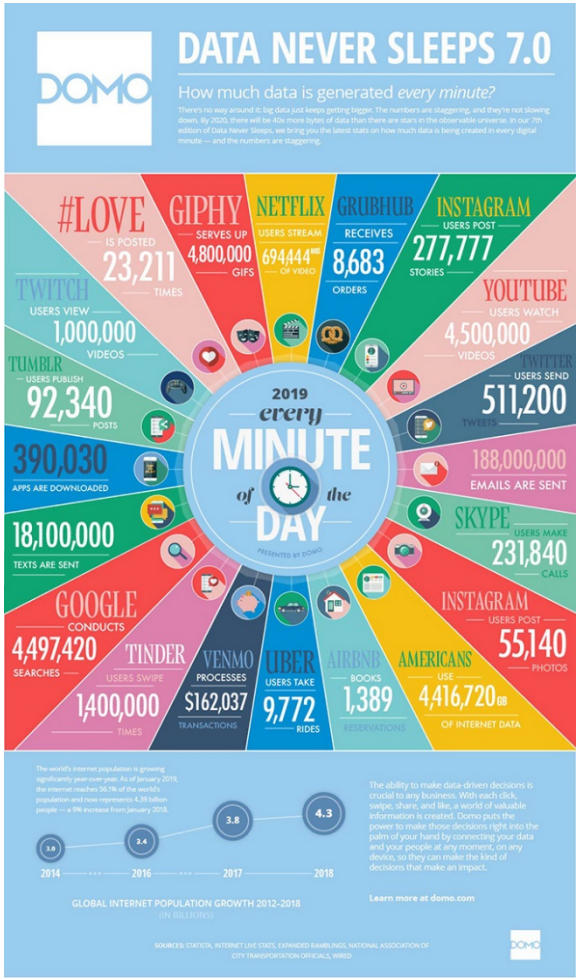
Real examples

- Route planning
  - Find the shortest path between two specific locations
  - Input:
    - A road network
    - A starting node (location)
    - A end node (location)
  - Output
    - A path, or a sequence of edges, with the shortest total distance

Why need data structures?

- In the era of big data
  - The size of data is huge
  - Data is growing faster than ever

- We need to handle such data carefully so as to support efficient operations
  - Different data structures work for different applications
    - Arrays
      - Are friendly to search (if sorted)
      - Not friendly to updates
    - Lists
      - are more friendly to updates
      - Not friendly to search



One sentence about this course

This course is about how to use computing resources and tools to solve practical problems **correctly** and **efficiently**

Logical thinking and math

Good at data structures, algorithms, and programming

What's the course about?

- Why study data structures and algorithms?
  - Answer: To solve the problem efficiently!
- To solve the problem efficiently:
  - Efficient Management of Data ⇔ Data Structures
  - Efficient Actions ⇔ Algorithms
- Examples: How does Google find the documents matching your query so fast?
  - Uses sophisticated algorithms to create index structures, which are just data structures;
  - Algorithms and data structures are everywhere.

30 trillion (30×10^{12}) webpages

Ready for the course?

- No pain, no gain
- Heavy workload
- Require analytic skills and programming skills
  - Java + Eclipse will be used for programming assignments
- Fully committed

Sanity check whether I am suitable for this summer course

- You may consider dropping this course in summer if any of the following happens to you:
  - The only programming language I know is Python/PHP/ JavaScript and I also don't want to learn Java
  - I have another math course this semester and I am not sure about the workload

Data abstraction

- A clear separation between
  - the abstract properties of a data type
  - and concrete details of its implementation

- Example: smartphone
  - Users do not know
    - how calls are made; how the phone accesses the Internet; how data is stored in the phone
  - Users do know
    - To make a call: select a contact's phone number
    - To access the Internet: open the browser

Abstract Data Types (ADT)

- An ADT is for encapsulation (information hiding)
  - The implementation of an ADT and its operations can be localized to one section of the program
  - Procedures that make use of the ADT can safely ignore its implementation details

- Benefit of ADT
  - User-friendly: users do not need to know the mechanisms of how to connect to the Internet
  - Designer-friendly: designer can change mechanisms without affecting users
  - Protection: others cannot know your secrets!

How to separate?

- An ADT only provides the definition of operations
  - it consists of names of every operation (function), the type of its arguments, and the type of its result

```
1  ADT IntegerSet
2   IntegerSet createEmptySet();
3   IntegerSet addElementToSet(integer, SetA);
4   Boolean search(integer, SetA);
5   IntegerSet intersection(setA, setB);
6   IntegerSet union(setA, setB);
7   IntegerSet difference(setA, setB);
```

  - The definition does not reveal the internal representation or implementation details
    - How the set is represented? Array, List, Tree, etc.?
    - How the operations are implemented? Many different ways

Relationships: ADT, data Structure, algorithms

- A data structure: implementation of an ADT
  - List
    - Implementation: ArrayList or LinkedList
  - Set
    - Implementation: Hash Table or Red-Black Tree

- An algorithm: implementation of operations in ADT

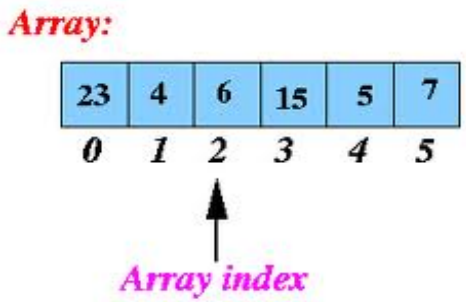
Input
A sequence of numbers: $a_1, a_2, \ldots a_n$

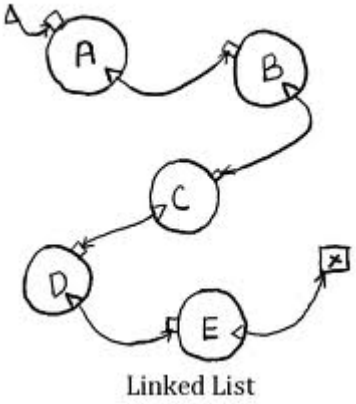
→ Sorting Algorithm →

Output
A permutation (reordering) $\langle a_1', a_2', \ldots a_n' \rangle$ of the input sequence, $a_1' \leq a_2' \leq \ldots \leq a_n'$

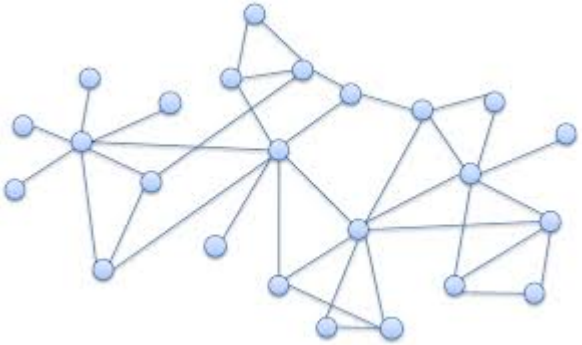
What are the common ADTs?

- *Array, Lists, stacks, queues, trees, graphs*, etc.

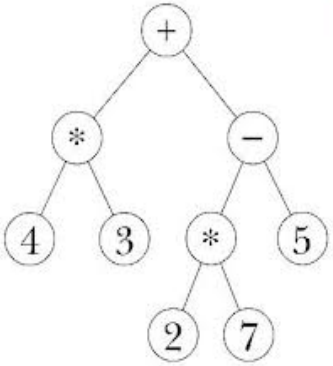




Linked List




Tree



Topics covered

- How to define "efficient"?
  - Complexity analysis
- No single data structure fits all scenarios
  - Array: Efficient for sorting and search, not for updates
  - List: Efficient for updates, but not in searching/sorting
  - Stack and queues: Efficient for certain types of updates, not applicable for searching/sorting
  - Trees: Heap & Binary search trees
  - Hash tables: Hash functions
  - Graphs: Graph algorithms
- Efficient operations
  - Sorting
  - Searching/Updates on different data structures (Discussed when learning the specific data structure)

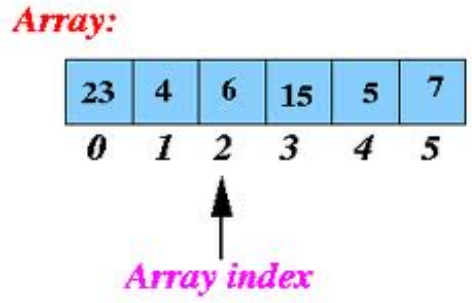
Tentative teaching plan

Week	Monday	Wednesday	Friday
1	Introduction (★)	Java (★)	Insertion sort, merge sort (★★)
2	Dragon boat festival	Complexity analysis (★★)	Complexity of recursion, divide-and-conquer (★★)
3	List (★★)	Stack, queue (★★)	Trees, binary trees (★)
4	Binary search tree (★★)	Priority queue, heap, heapsort (★★★)	Black-Red trees (★★★)
5	Midterm exam	Midterm exam questions, sorting algorithms (★★)	Sorting algorithms (★★)
6	Hashing (★★)	Graphs, BFS, DFS (★★)	Graph minimum spanning tree (★★★)
7	Graph shortest path (★★★)	Graph shortest path (★★★)	DAG checking, topological sort (★★), review

*This tentative teaching plan is subject to change as the instructor sees fit.

Array

- An array is used to store the **same** objects together, and access the objects by their indices.
  - Pro: If you know the index (where), you can find the object (content) with one basic operation (efficient). Efficient search if array is sorted
  - Con: (i)If you want to insert an object in a place between two objects in an array, you have to move the objects first. (ii) The capacity is fixed.



Array ADT

```
1  ADT Array
2   Array createArray(n)
3   Item retrieve(arr, i)
4   Item store(arr, i, itemToStore)
```

- createArray(n)
  - Initialized an array of size n to store Item
- retrieve(arr,i)
  - arr[i],
  - Return the item stored in the i-th position of the array
- store(arr, i, itemToStore)
  - Store itemToStore to the i-th position of the array
  - arr[i] = itemToStore

Advantage of ADT

- We can design algorithms without knowing its underlying implementation.
  - Example: Design an algorithm to do linear search with an array using the ADT

Algorithm 1: ***Linear Search***
Input: An array *a* of integers with length n, an integer searchnum
Output: the index i such that the value stored in the i-th position equals searchnum , or -1 if no such i exists.

```
1  int i;
2  for (i=0; i<n; i++){
3      if( retrieve(a,i) == searchnum )
4          return i;
5  }
6  return -1;
```

Array ADT: example

- We can design algorithms without knowing its underlying implementation.
  - Implement the dimension-product given two arrays a_1 and a_2 and output to a_3 such that $a_3[i] = a_1[i] \cdot a_2[i]$

Algorithm 2: ***Dimension Product***
Input: Vector a_1, a_2, a_3 of length n
Output: Dimension-product of a_1 and a_2 which stored in a_3.

```
1  int i,i_dimension_coordinate;
2  for (i=0; i<n; i++){
3        i_dimension_coordinate = retrieve(a1,i)*retrieve(a2,i);
4        store(a3, I, i_dimension_coordinate);
5  }
6  return a3;
```

Question

- Given a problem, is it enough to write a correct working program to solve it?

Example: selection problem

- Problem: given a group of N numbers, determine the k^{th} largest, where N > k.

- Solution 1:
  1) read N number into an array;
  2) sort the array in descending order;
  3) return the element in position k;

- Solution 2:
  1) read the first k elements into an array and sort them in descending order,
  2) each remaining element is read one by one,
     - 2.1) it is ignored if it is smaller than or equal to the k^{th} element in the array
     - 2.2) otherwise, it is placed in its correct spot in the array, bumping one element out of the array.
  3) the element in the k^{th} position is returned as the answer.

Example: Selection Problem

- Two natural questions:
  - Which solution is better ?
    - By simulation
    - By theoretical analysis
  - Is either algorithm good enough (particularly when N is very large)?
    - A simulation using 1 million elements and k = 500,000 will show that NEITHER algorithm finishes in a reasonable amount of time
    - Is there a better algorithm?
- What is the conclusion?

Conclusion from the previous example

- An important concept
  - In many problems, writing a working program is not good enough
  - If the program is to be run on a large data set, then the running time becomes an issue
  - The optimum solution: a correct algorithm with minimum resources and minimum running time

Throughout this course…

- We will see
  - How to estimate the running time of a program for large inputs
  - How to compare the running times of two programs
  - Techniques to improve the speed of a program, and to determine the program bottlenecks

Mathematics review

- Exponents

$$X^A X^B = X^{A+B}$$

$$\frac{X^A}{X^B} = X^{A-B}$$

$$(X^A)^B = X^{AB}$$

$$X^N + X^N = 2X^N \neq X^{2N}$$

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

Mathematics review

- Logarithm definition:

$$X^A = B \text{ if and only if } \log_x B = A$$

- All log are to be base 2 unless specified otherwise.

– *Useful equalities*

$$\log_A B = \frac{\log_c B}{\log_c A}; C > 0$$

$$\log AB = \log A + \log B$$

$$\log\left(A/B\right) = \log A - \log B$$

$$\log x < x, \quad \forall x > 0$$

$$\log 1 = 0, \log 2 = 1, \log 1024 = 10, \log 65536 = 16$$

Mathematics review

- Series: arithmetic series

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

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

Example: To find the sum

2+5+8+….+ (3k-1)

= 3(1+2+3+…+k)
- (1+1+1+….+1)

$$= \frac{3k(k+1)}{2} - k$$

Mathematics review

- Series: geometric series

$$\sum_{i=0}^{N} A^i = \frac{1 - A^{N+1}}{1 - A}$$

$$If\ 0 < A < 1, then$$

$$\sum_{i=0}^{N} A^i \leq \frac{1}{1 - A}$$

- Derivation

Let $S = 1 + A + A^2 + \ldots\ldots$ (1)

where, $0 < A < 1$

then $AS = A + A^2 + A^3 + \ldots$ (2)

Subtracting (1) and (2), we get $S - AS <= 1$, i.e.

$$S \leq \frac{1}{1 - A}$$

Mathematics review

- To prove a false statement:

  -proof by counter example

  e.g., Fibonacci number:

  $F_0 = 1, F_1 = 1, F_{k+1} = F_k + F_{k-1}$

  To show the statement $F_k \leq k^2$ is false, we can compute a concrete counter example, e.g.,

  $F_{11} = 144 > 11^2$.

- To prove a correct statement

  - proof by induction

  (1) proving a base case

  (2) inductive hypothesis

  - proof by contradiction

  (1) assume it is false

  (2) show that this assumption is false

Exercise

- Is it possible to find a better algorithm to select the k^{th} largest number?