

COR-IS1702: COMPUTATIONAL THINKING

WEEK 1A: INTRODUCTION

Vivien Soon

viviensoon@smu.edu.sg

<http://t.me/vhssoon>

Lesson Plan for Today

- 1st half:




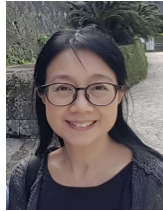












- Admin stuff
- Course Outline
- Counting

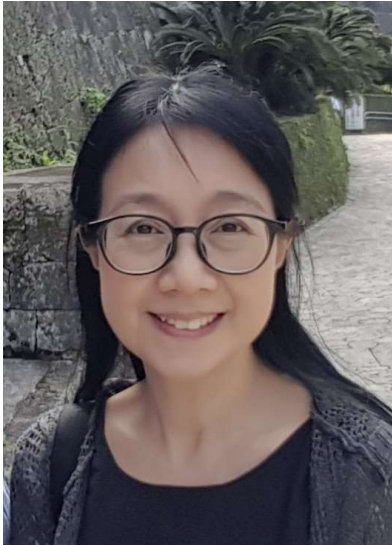
- 2nd half:

- Programming I

The Teaching Team for CT

	G1	G2	G4	G3	G8	G5	G6	G7
Faculty	Akshat Kumar 		Arunesh 	Mok Heng Ngee 			Vivien Soon 	
Instructor			Mok Heng Ngee 				Fiona Lee 	
TA	Po Qi Lin 	Goh Wan Xuan 	Tsang Bao Xian 	Bui Phuong Thao 	Syafiqah 	Lim Jia Peng 	Autumn Teo 	Pang Qi Xuan 

Your Teaching Team (G5 – Thu, 12pm)



Faculty: Vivien Soon

- viviensoon@smu.edu.sg
- Telegram: <http://t.me/vhssoon>



Instructor: Mok Heng Ngee

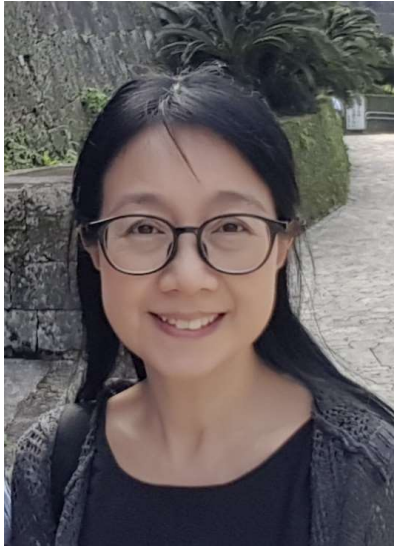
- hnmok@smu.edu.sg
- Telegram: <http://t.me/Mokkie>
- Tel: 68280541, 96630214



TA: Lim Jia Peng

- jiapeng.lim.2017@sis.smu.edu.sg
- Telegram: <http://t.me/jararap>

Your Teaching Team (G6 – Wed, 8:15am)



Faculty: Vivien Soon

- viviensoon@smu.edu.sg
- Telegram: <http://t.me/vhssoon>



Instructor: Fiona Lee

- fionalee@smu.edu.sg
- Telegram: <http://t.me/fionaleeyy>
- Tel: 68281982



TA: Autumn Teo Wei Jian

- autumn.teo.2018@sis.smu.edu.sg
- Telegram: <http://t.me/AmlAutumn>

Your Teaching Team (G7 – Fri, 8:15am)



Faculty: Vivien Soon

- viviensoon@smu.edu.sg
- Telegram: <http://t.me/vhssoon>



Instructor: Fiona Lee

- fionalee@smu.edu.sg
- Telegram: <http://t.me/fionaleeyy>
- Tel: 68281982



TA: Pang Qi Xuan

- qxpang.2018@accountancy.smu.edu.sg
Telegram: <http://t.me/Qixuannn>

Learning Outcomes

- Computational thinking skills
 - algorithmic, iterative and recursive thinking
 - abstraction, data representation
 - heuristics reasoning, pattern matching
- Problem solving skills
 - model problems as computational problems formally
 - identify and design appropriate representation and solution
 - implement in Python code
- Learning to learn skills
 - self-learn and use tools (PythonLabs) in exercises and assignments
 - independently investigate other techniques not taught in class
- Communication skills

Assessment

- Exams (55%)
 - Class Test (5%) week 6 (45 mins)
 - Common Midterm Exam (15%) week 9
 - Common Final Exam (35%)
- Common Assignments (30%)
 - Paper Assignment (10%) weeks 9-11
 - Project Assignment (20%) week 11-13
- Class Participation (15%)
 - section-specific

Important:

Reserve this slot
on your calendar
now for your
midterm:

16 Oct 2020 (Fri)
4 - 5:30pm

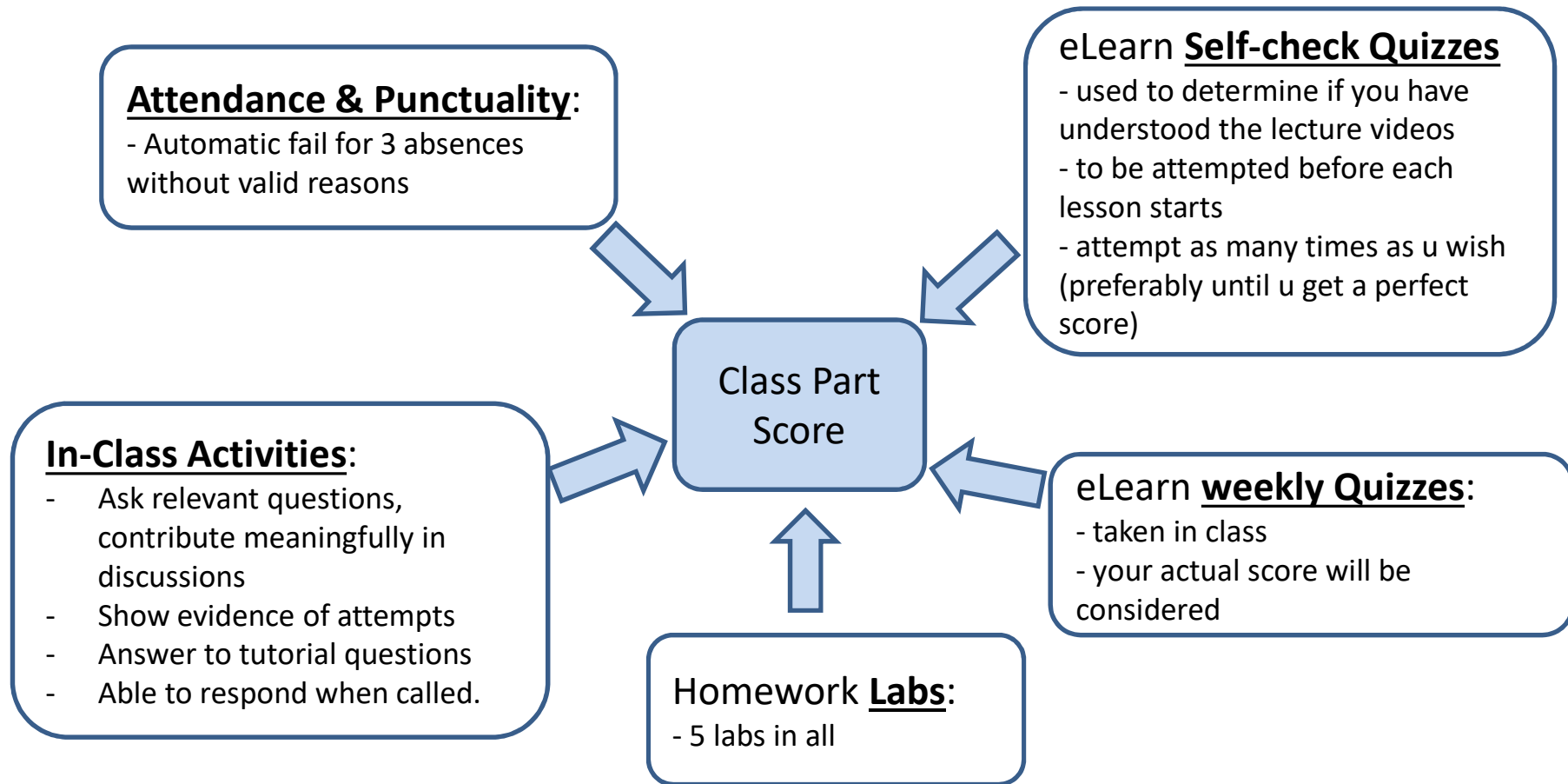
27 Nov 2020 (Fri)
8:30am

Lesson Plan

Week	Topic	Remarks
1	Intro / Counting / Python Programming	
2	Python Programming	
3	Complexity	
4	Iteration and Decomposition (Search, Sort)	
5	Recursion	
6	Linear Data Structures (Stack, Queue, Priority Queue)	Class Test (5%)
7	Binary Tree and Binary Search Tree	
8 (Recess Week)		
9	Graphs	Midterm Exam (15%) (16 Oct '20)
10	Heuristics (Greedy, Local Search)	Paper Assignment (10%)
11	Limits of Computation	
12	Project Week (no lesson)	Project Assignment (20%)
13	Review / Mock Exam	
14 (Study Week)		
15 (Exam Week)		Final Exam (35%) (27 Nov '20)

Assessment: Class Participation (15%)

For sections G5, G6 and G7:

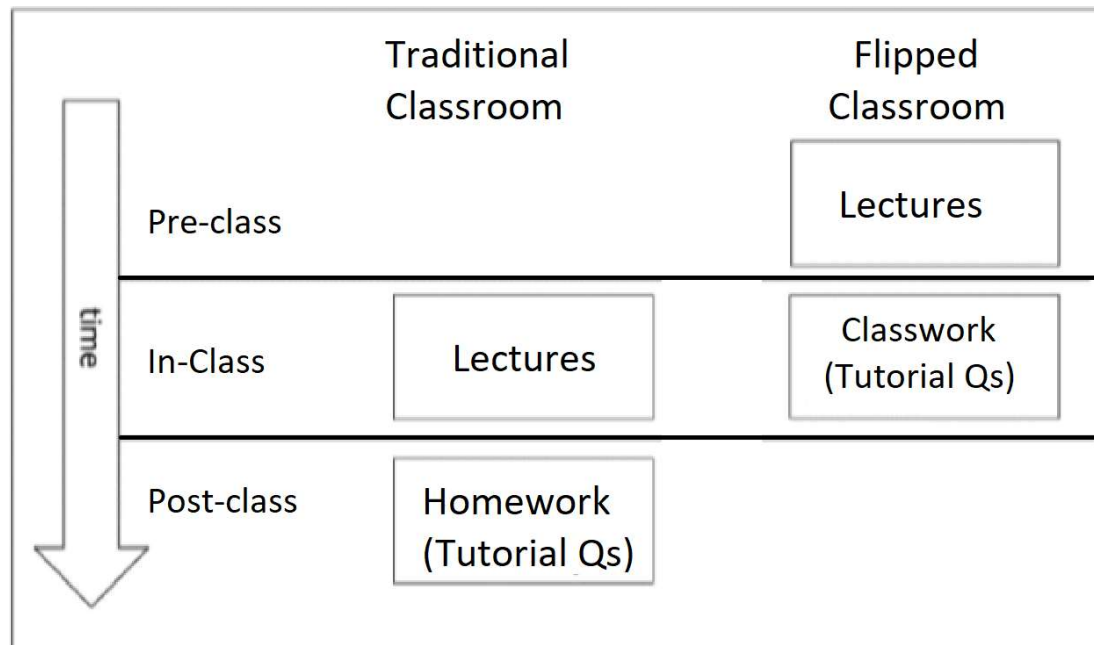


Admin Issues

- Announcements/reminders:
 - Will be sent via eLearn/Telegram
 - See “Announcements” at red.smu.edu.sg for project-related announcements
- Voice out problems ASAP and seek help
- Consultation hours:
 - For Vivien: every Monday 9 – 10 a.m. (Join scheduled Zoom meetings)
 - For TAs: ad-hoc. Please contact Jia Peng(G5), Autumn(G6), Qi Xuan(G7)
- Attending another section’s class:
 - Not encouraged. Occasionally allowed with good reason
 - Let me know ASAP if necessary

Pedagogy for this Section

- From week 3 onwards, this section will be adopting the “Flipped Classroom”
 - What is this all about? → https://ink.library.smu.edu.sg/sis_research/2363/



Comparing the traditional classroom with the flipped classroom. Diagram adapted from Mok's paper

...Pedagogy

- Some rules for this to work:
 - You must prepare for each class by **watching all the assigned video lectures + attempt all the self-check quizzes (on eLearn)**
 - Attend a consultation session if you need help understanding the lectures
 - The assumption is that everyone comes to class with a common knowledge baseline

Academic Integrity

- SMU's code of academic integrity:
<https://oasis.smu.edu.sg/Pages/DOS-WKLSWC/UCSC.aspx> (Click on “Code of Academic Integrity”)
- Behave ethically and honorably
- No undesirable activities
 - Cheating, plagiarism, fabrication, misrepresentation of academic records, facilitating academic dishonesty, unfair advantage
- For marked assignments/projects:
 - Always acknowledge discussions and/or contributions of others
 - Do not post solutions or partial solutions (including any discussion forum or public code repositories such as GitHub); however, you are permitted to discuss general ideas and problem-solving approaches, and not specific answers.
- When in doubt, consult the instructors

What is an Algorithm

- Definition:

- “*An algorithm is a finite sequence of precise instructions for performing a computation or for solving a problem*”

~ From Rosen chapter 3 (p.191)

- Algorithms must have the following properties:

- Input and output
- *Definiteness*: each step must be defined precisely
- *Correctness*: must produce the correct output values for each set of input values
- *Finiteness*: will not go on forever
- *Generality*: applicable for all problems of the desired form, not just for a particular set of input values

~ From Rosen chapter 3 (p.193)

This Course

- Introduction to:
 - Algorithms
 - Data Structures
 - Coding in Python
- We will:
 - Study a few well-known algorithms for certain problems
 - Given multiple algorithms, compare them in terms of time complexity to determine which is “better”
 - Given a problem, come up with an algorithm to solve it
 - Code & execute your algorithm in Python

Course Emphasis

- This course is NOT:
 - A pure CS course on data structures, algorithms, or discrete mathematics
 - A programming course
- What we emphasize:
 - Design
 - Complexity
 - Efficiency and scalability
- What we will learn:
 - Problem formulation
 - Problem solving methods
 - **Real-world applications**

-
- Show full example of:
 - a real problem
 - the algorithm used to solve it

Example of a Problem

- You are the emcee of a “couple matching” game show
 - There are 8 participants: 4 males and 4 females
 - After a series of self introductions, each M ranks each F from most preferred (1) to least preferred (4). Each woman does the same.
- Objective:
 - All matches are stable if, for any one particular couple (M_x - F_x),
 - there is no F whom M_x prefers (over F_x), and who prefers M_x (over her current partner)
 - There is no M whom F_x prefers (over M_x), and who prefers F_x (over his current partner)
 - In other words, no two people of opposite sex who would both rather have each other than their current partners
 - You need to match them to form 4 “stable” couples

Ranking by Males:

	F1	F2	F3	F4
M1	1	3	2	4
M2	1	2	4	3
M3	1	2	3	4
M4	2	3	1	4

e.g. M1 prefers F1 the most, followed by F3, F2 and F4.

Ranking by Females:

	M1	M2	M3	M4
F1	2	1	3	4
F2	4	3	2	1
F3	1	2	3	4
F4	3	4	2	1

e.g. F2 prefers M4 the most, followed by M3, M2 and M1.

Consider the following solutions – are they stable? If not, give an example of a couple which will elope:

- (i) (M1-F1), (M2-F3), (M3-F2), (M4-F4)
- (ii) (M1-F3), (M2-F1), (M3-F4), (M4-F2)

Gale-Shapley Deferred-Choice Algorithm

Given n males and n females.

Each male proposes to his highest ranked female.

If a female is not matched yet, she automatically accepts.

If she is already matched, she picks the more preferred male.

The rejected male moves on to his next desired female.

*When each man is engaged, the problem is solved.
(The order in which males propose is not important.)*

Gale-Shapley Deferred-Choice Algorithm

```
Each person can be in 1 of 2 states at any one time: matched or free
All males and females are initially free

While there exists an unmatched male m who has not proposed to all females:
    Let f be the highest ranked female to whom m has not proposed

    If f is free:
        (m, f) are matched
    Else it means that some other pair (m', f) already exists:
        If f prefers m to m'
            (m, f) are matched
            m' becomes free
        Else
            # do nothing!
            (m', f) remain matched
            m remains free
    End
End
End
```

- For N males and N females, at most N^2 number of proposals
 - E.g. if $N = 22$, there are up to 484 proposals (much more efficient than billions)
- Guarantees: Everyone gets married, and all marriages are stable

2012 Nobel Prize in Economics

Lloyd Shapley. Stable matching theory and Gale–Shapley algorithm.

COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only g . Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the g best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept.

original applications:
college admissions and
opposite-sex marriage

Alvin Roth. Applied Gale–Shapley to matching med-school students with hospitals, students with schools, and organ donors with patients.



Lloyd Shapley



Alvin Roth



Source:

<https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/01StableMatching.pdf>

- Gale-Shapley algorithm and kidney donations. Watch: https://www.youtube.com/watch?v=GVCBI_DepDE

Problem Solving

- Problem-solving methodology that we will learn in the course:
 - Formulating a real-world problem in computational terms
 - Characterizing the difficulty or complexity of the problem
 - Finding ways (algorithms) to solve the problem efficiently
- By the end of the course, you are expected to be able to carry out a similar problem-solving methodology for this and other problems

Things You Will Learn

How to design algorithms to solve problems?

How to organize data to solve problems more efficiently?

How to deal with complex problems?

Course Outline

Algorithm Design and Analysis

(Weeks 1 – 5)

Fundamental Data Structures

(Weeks 6 – 10)

Computational Intractability and Heuristic Reasoning

(Weeks 10 – 13)

Course Outline

Algorithm Design and Analysis

- Week 1: Introduction, Counting, Programming
- Week 2: Programming
- Week 3: Complexity
- Week 4: Iteration & Decomposition
- Week 5: Recursion

Course Outline

Fundamental Data Structures

- Week 6: Linear Data Structures (Stacks, Queues)
- Week 7: Hierarchical Data Structure (Binary Trees)
- **Week 8: Mid-term Break**
- Week 9: Networked Data Structure (Graphs)

Course Outline

Computational Intractability and Heuristic Reasoning

- Week 10: Heuristics
- Week 11: Limits of Computation
- Week 12: Project Week (no class)
- Week 13: Review

References

- Explorations in Computing by John S. Conery (JSC)
 - Publisher: Taylor & Francis CRC Press
 - Homepage: <http://ix.cs.uoregon.edu/~conery/eic/python/index.html>
 - PDF: <ftp://ftp.cs.uoregon.edu/pub/conery/freeman/EiC.Mar.27.pdf>
- Discrete Mathematics and its Applications
 - Authors: Kenneth H. Rosen (KHR)
 - Publisher: McGraw-Hill, 2012
 - Edition: 7th
 - Chapter 6 on Counting (provided on eLearn)
- Supplementary Handout on Fundamental Data Structures
 - Provided by Instructors on eLearn (Handout)

Summary

- Computational thinking is for everyone
- Elements of computational thinking include:
 - Abstract and algorithmic thinking.
 - Iteration, Decomposition, Complexity, Heuristic reasoning.
- Course emphasizes:
 - Conceptual thinking and reasoning, *not* programming
 - Formulating a problem in terms of how it can be computed
 - Characterizing the complexity of problems and solutions
 - Designing solutions with simplicity and efficiency in mind
- We are here to support your learning journey

Lesson Plan for Today

- 1st half:
 - Admin stuff
 - Course Outline
 - ➔ – Counting
- 2nd half:
 - Programming I

Need to know some "high school Maths"

- Basic algebra
- Logarithms
- Sum of series in AP/GP

Exponent Rules

Assume that a and b are nonzero real numbers, and m and n are any integers.

1) Zero Property of Exponent

$$b^0 = 1$$

2) Negative Property of Exponent

$$b^{-n} = \frac{1}{b^n} \quad \text{OR} \quad \frac{1}{b^{-n}} = b^n$$

3) Product Property of Exponent

$$(b^m)(b^n) = b^{m+n}$$

4) Quotient Property of Exponent

$$\frac{b^m}{b^n} = b^{m-n}$$

5) Power of a Power Property of Exponent

$$(b^m)^n = b^{mn}$$

6) Power of a Product Property of Exponent

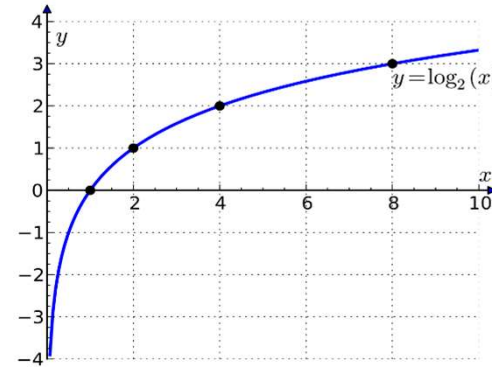
$$(ab)^m = a^m b^m$$

7) Power of a Quotient Property of Exponent

$$\left(\frac{a}{b}\right)^m = \frac{a^m}{b^m}$$

Logarithm Rules

$$a^y = x \quad \Leftrightarrow \quad y = \log_a x$$



$$\log_a xy = \log_a x + \log_a y$$

$$\log_a \frac{x}{y} = \log_a x - \log_a y$$

$$\log_a x^n = n \log_a x$$

$$\log_a b = \frac{\log_c b}{\log_c a}$$

$$\log_a b = \frac{1}{\log_b a}$$

The following can be derived from the rules.

$$\log_a 1 = 0$$

$$\log_a a = 1$$

$$\log_a a^r = r$$

$$\log_a \frac{1}{b} = -\log_a b$$

$$\log_{\frac{1}{a}} b = -\log_a b$$

Sum of Arithmetic Progression Series

The sum of the arithmetic series, S_n , by starting with the first term a and successively adding the common difference d :

$$S_n = \overset{\text{1st}}{a} + \overset{\text{2nd}}{(a + d)} + \overset{\text{3rd}}{(a + 2d)} + \dots + \overset{\text{n}^{\text{th}}}{[a + (n-1)d]}$$

$$= \frac{n}{2} [2a + (n - 1)d]$$

$$= \frac{n}{2} [a + a_n] \quad \text{where } a_n = a + (n-1)d$$

Sum of Geometric Progression Series

The sum of the geometric series, S_n , by starting with the first term a and successively multiplying the common ratio r ($r \neq 1$):

$$S_n = \overset{1\text{st}}{a} + \overset{2\text{nd}}{ar} + \overset{3\text{rd}}{ar^2} + \dots + \overset{n\text{th}}{ar^{n-1}}, \quad r \neq 1$$

$$rS_n = ar + ar^2 + ar^3 + \dots + ar^n$$

$$(r-1)S_n = a(r^n - 1)$$

$$S_n = \frac{a(r^n - 1)}{(r-1)}$$