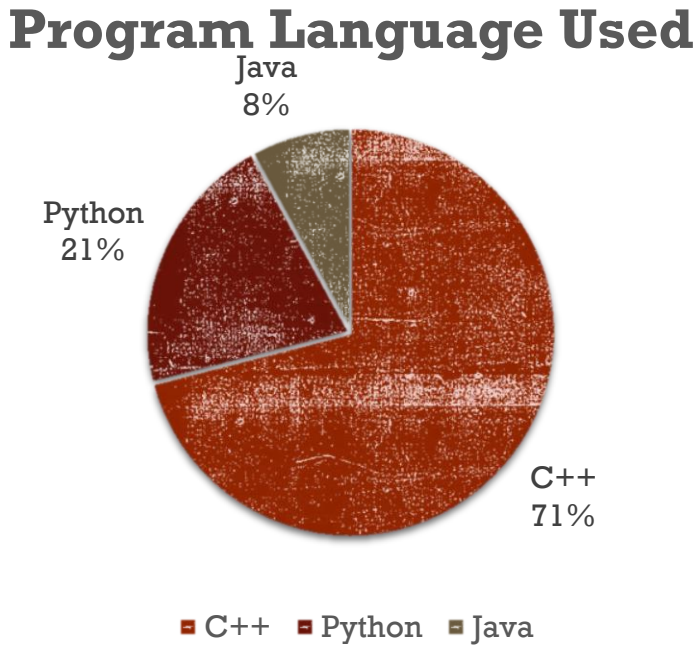
CIS 4930 INTRODUCTION TO COMPETITIVE PROGRAMMING

Fall 2020

Instructor: Ronnie Zhang

WHAT IS COMPETITIVE PROGRAMMING

- Aim: write code to solve given problems
  - Design of algorithms
    - Inventing efficient algorithms to solve well-defined problems
  - Implementation of algorithms
    - Short programs
      - Should be written quickly
    - Language?
      - 71% of top 4500 participants in Google Code Jam 2019 using C++
- "Competitive Programming" is this: "Given well-known Computer Science (CS) problems, solve them as quickly as possible!"



WHY SHOULD I TAKE THIS CLASS?

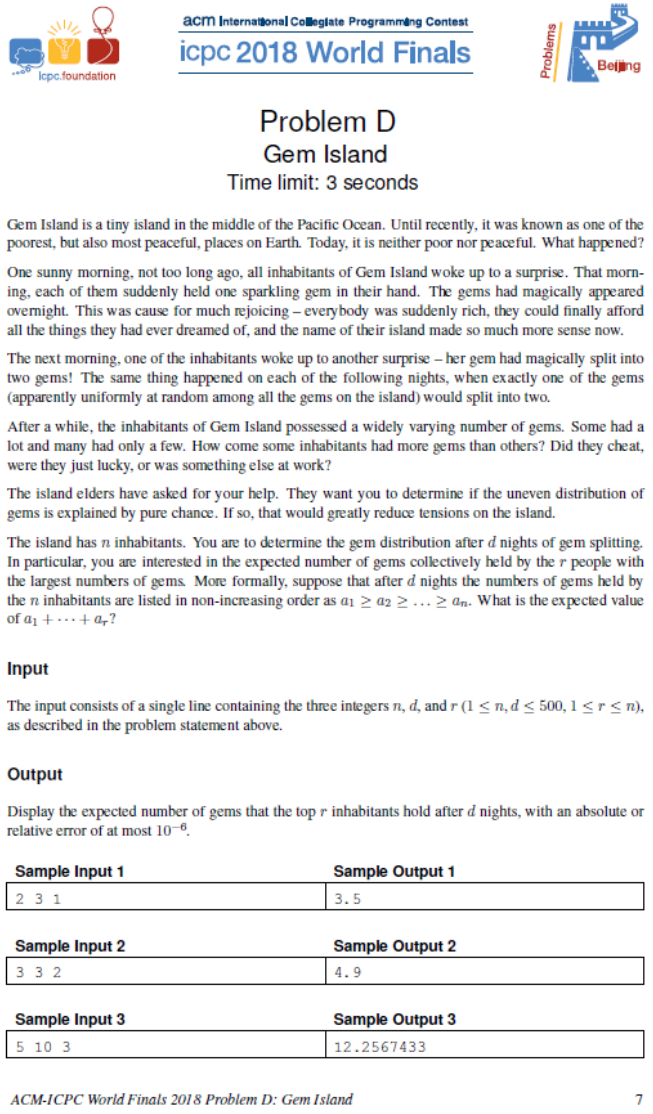
- Success in programming contests, tech interviews, and future career
- Fill the gap between algorithms and problem solving
- Goal:
  - Sharpen programming and problem-solving skills
  - Able to select appropriate algorithms for a given problem
  - Able to integrate multiple algorithms to solve a complex problem
  - Able to implement advanced algorithms in a timely manner
  - Solve problems in teams
- Not immerse you with thousands of competition questions
  - But you still need to practice A LOT

COMPETITION

- ICPC – the International Collegiate Programming Contest
- Online competitions
  - TopCoder
  - Google's Coding Competitions
    - Google Code jam
    - Google Hash Code
    - Google Kick Start
  - Microsoft Imagine Cup
  - Facebook Hacker Cup
  - CodeChef
  - Codeforces
  - Etc.

QUESTION FORMAT

- Anatomy of a programming contest problem
  - Problem Statements
    - Background story/problem description
  - Input and Output description
  - Constraints
    - Input size: memory limit
    - Running time: Time limit
    - Accuracy
  - Sample Input and Sample Output
    - Help you to understand the problem and debug



icpc.foundation

acm International Collegiate Programming Contest
icpc 2018 World Finals

Problems Beijing

Problem D
Gem Island
Time limit: 3 seconds

Gem Island is a tiny island in the middle of the Pacific Ocean. Until recently, it was known as one of the poorest, but also most peaceful, places on Earth. Today, it is neither poor nor peaceful. What happened?

One sunny morning, not too long ago, all inhabitants of Gem Island woke up to a surprise. That morning, each of them suddenly held one sparkling gem in their hand. The gems had magically appeared overnight. This was cause for much rejoicing – everybody was suddenly rich, they could finally afford all the things they had ever dreamed of, and the name of their island made so much more sense now.

The next morning, one of the inhabitants woke up to another surprise – her gem had magically split into two gems! The same thing happened on each of the following nights, when exactly one of the gems (apparently uniformly at random among all the gems on the island) would split into two.

After a while, the inhabitants of Gem Island possessed a widely varying number of gems. Some had a lot and many had only a few. How come some inhabitants had more gems than others? Did they cheat, were they just lucky, or was something else at work?

The island elders have asked for your help. They want you to determine if the uneven distribution of gems is explained by pure chance. If so, that would greatly reduce tensions on the island.

The island has n inhabitants. You are to determine the gem distribution after d nights of gem splitting. In particular, you are interested in the expected number of gems collectively held by the r people with the largest numbers of gems. More formally, suppose that after d nights the numbers of gems held by the n inhabitants are listed in non-increasing order as $a_1 \geq a_2 \geq \ldots \geq a_n$. What is the expected value of $a_1 + \cdots + a_r$?

Input

The input consists of a single line containing the three integers n, d, and r ($1 \leq n, d \leq 500$, $1 \leq r \leq n$), as described in the problem statement above.

Output

Display the expected number of gems that the top r inhabitants hold after d nights, with an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
2 3 1	3.5

Sample Input 2	Sample Output 2
3 3 2	4.9

Sample Input 3	Sample Output 3
5 10 3	12.2567433

ACM-ICPC World Finals 2018 Problem D: Gem Island 7

SIMPLE EXAMPLE

- Leetcode 1557: Minimum Number of Vertices to Reach All Nodes

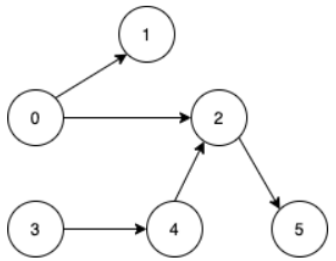
Given a **directed acyclic graph**, with `n` vertices numbered from `0` to `n-1`, and an array `edges` where `edges[i] = [from_i, to_i]` represents a directed edge from node `from_i` to node `to_i`.
Find *the smallest set of vertices from which all nodes in the graph are reachable*. It's guaranteed that a unique solution exists.
Notice that you can return the vertices in any order.

Constraints:

- `2 <= n <= 10^5`
- `1 <= edges.length <= min(10^5, n * (n - 1) / 2)`
- `edges[i].length == 2`
- `0 <= from_i, to_i < n`
- **All pairs** `(from_i, to_i)` are distinct

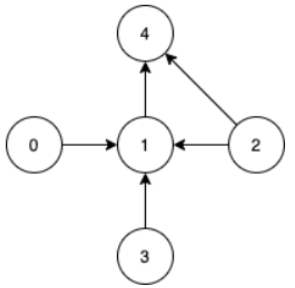
What if the given numbers are HUGE?
Not all the input constraints are explicit
Always think about the worst case scenario, edge cases, etc

Example 1:



Input: n = 6, edges = [[0,1],[0,2],[2,5],[3,4],[4,2]]
Output: [0,3]

Example 2:



Input: n = 5, edges = [[0,1],[2,1],[3,1],[1,4],[2,4]]
Output: [0,2,3]

COMMON VERDICT INFORMATION

- In Queue (QU)
- Accepted (AC): Congratulations!
- Presentation Error (PE)
- Wrong Answer (WA)
- Compile Error (CE)
- Runtime Error (RE)
- Time Limit Exceeded (TL)
- Memory Limit Exceeded (ML)
- Output Limit Exceeded (OL)

PRE-REQUISITE

- Basic knowledge in programming methodology
  - Familiar with at least one of the following programming languages
    - C++, Java, Python
  - Confident in coding debugging, and testing
- Data structure
  - Array, stack, queues, deques
  - Graph, tree, string
- Algorithm
  - Recursion
  - Sorting/Searching algorithm
  - Dynamic programming
- Math
  - Binaries, fractions and complex numbers.
  - Matrix multiplication
  - Trigonometry

Minus one programmer, next.

https://me.me/i/thuaha-hadi-santa-pinta-dear-santa-i-want-to-be-1f2714f290fe48ca86c36cd24a6079b1

TENTATIVE COURSE SCHEDULE

- Week 1: Introduction and Efficiency
- Week 2: Data Structure
- Week 3: Sorting and Searching
- Week 4: Greedy Algorithm
- Week 5: Dynamic Programming
- Week 6: Dynamic Programming – cont.
- Week 7: Graph Algorithms
- Week 8: Algorithm Design
- Week 9: Range Queries
- Week 10: Tree Algorithms
- Week 11: String Algorithms
- Week 12: Geometry Algorithms
- Week 13: Bit Manipulations

*schedule is subject to change

RECOMMENDED MATERIALS

- Guide to Competitive Programming
  - Authors: Antti Laaksonen
  - Publisher: Springer
  - ISBN: 3319725467
- Algorithms Unlocked
  - Authors: Cormen, Thomas H
  - Publisher: MIT Press
  - ISBN: 0262518805
- Others

CLASS FORMAT AND POLICIES

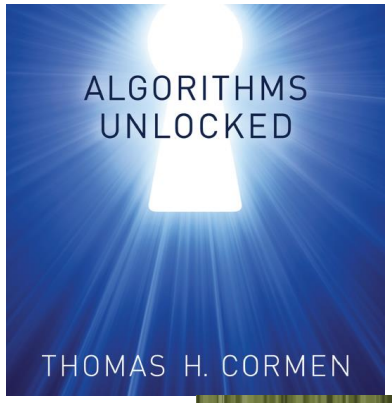
- 50 minutes lecture, 50 minutes examples, 50 minutes problem solving
  - 2-3 questions if team
  - 2 questions if individual
- Homework every week
  - Late submission
- Midterm and Final
  - Midterm: 3 hours 8 questions team
  - Final: 3 hours 6 questions individual
- Evaluation of Grade

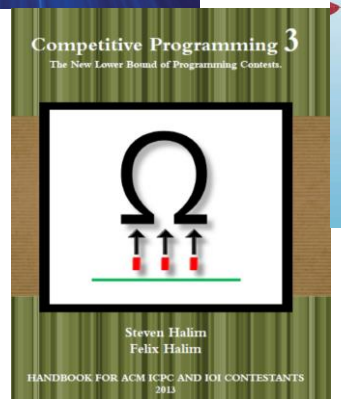
Assignment	Total Points	Percentage of Final Grade
Homework Sets (11)	100 each	50%
Midterm Exam	100	15%
Final Exam	100	20%
Class activities	1 each	15%
		100%

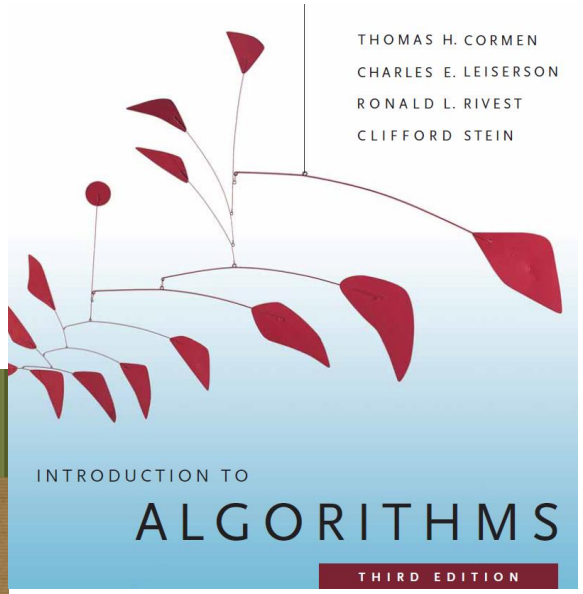
Percent	Grade	Grade Points
94.0 - 100	A	4.00
90.0 – 93.9	A-	3.67
87.0 - 89.9	B+	3.33
84.0 - 86.9	B	3.00
80.0 - 83.9	B-	2.67
77.0 - 79.9	C+	2.33
74.0 - 76.9	C	2.00
70.0 - 73.9	C-	1.67
67.0 - 69.9	D+	1.33
64.0 - 66.9	D	1.00
61.0 - 63.9	D-	0.67
0 – 60.9	E	0.00

TIPS FOR PRACTICING

- Number of solved problems is not as important as the quality of the problem
- Start with questions you feel comfortable
  - Get used to pace and adjust speed
  - Check other people's code to improve
    - Shorter, faster
  - Try some online contest
- Now try some hard problems
  - Try something a little above your level
  - Refine your code, test and debug
  - Know your defects
- Enhancing Your Theoretical Background
  - Arithmetic, combinatorics, number theory, game theory







UPCOMING COMPETITIONS

- 2020 ACM Southeast USA regional programming contest
- 2021 The North American Invitational Programming Contest
- 2021 ICPC North America Championship

	Category	**Frequency**
1	Ad Hoc	1-2
2	Complete Search (Iterative/Recursive)	1-2
3	Divide and Conquer	0-1
4	Greedy (usually the original ones)	0-1
5	Dynamic Programming (usually the original ones)	1-3
6	Graph	1-2
7	Mathematics	1-2
8	String Processing	1
9	Computational Geometry	1
10	Some Harder/Rare Problems	1-2

OTHER INFORMATION

Instructor: Rong Zhang

- Office location: CSE E526
- E-mail address: rzhang1@ufl.edu
- Office hours: M period 5 and 6 (11:45am – 1:40pm) or by appointment

Peer Mentors:

- Shawn Hatchwell <shawn.hatchwell@ufl.edu>, office hours: Thursday 3-4pm and Friday 2-3pm
- Neill Johnston <neilljohnston@ufl.edu>, office hours: Friday 4-5pm