acm International Collegiate Programming Contest
IBM event sponsor
ICPC 2014 Ekaterinburg UrFU

NYU

CSCI-UA.0480-004 Algorithmic Problem Solving

Lecture 1
Introduction

Bowen Yu
Spring 2018

Course Administration

- **Class Website**
  - http://cs.nyu.edu/courses/spring18/CSCI-UA.0480-004/
- **Contact**
  - Instructor: Bowen Yu, bowen.yu@nyu.edu
  - Recitation Leader: Kai Chen, kc3443@nyu.edu
- **Recitations (required)**
  - With Kai
  - Fridays from 5:10pm-7:10pm @ WWH 101
  - **Bring your laptop**

NYU

Office Hours

- **Bowen Yu**
  - Thursday after class, 4:45pm-5:45pm @ WWH 308
  - Or by email appointments of other times if you have class conflicts
- **Kai Chen**
  - Friday before recitations, 4:00pm-5:00pm @ WWH 412

What is Algorithmic Problem Solving

- Systematic introduction/review on basic and advanced algorithms
    - What you may have learned in Basic Algorithms
    - Fancy algorithms you may not have seen before
- Go from ideas to working solutions in Java/C++
    - Heavy on programming exercise
    - Excellent coding and debugging skills
- Thorough hands-on practice with automated online judge system
    - Correctness is verified through rigorous black-box testing
- Other benefits:
    - Technical interview preparation
    - Creative thinking and fun of puzzle solving
    - Go competitive and join programming contests (e.g. ACM-ICPC)
    - ...

Prerequisite

- Familiarity with Java or C++ Programming
  - Coding environment setup: IDE, debugging tools, etc.
  - Know how to do basic stuffs:
    - String tokenization, 2D array, class/struct, sorting, etc.
  - You will become more proficient as we progress.

- Basic algorithms
  - Array, linked list, stack, queue, binary search tree…
  - We will review basic topics during the course.

NYU

How does problem solving work?

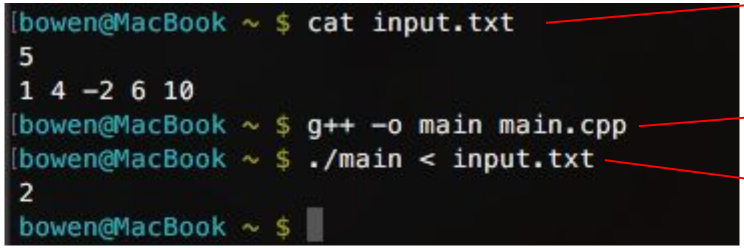
- 1D Closest Pair: Given a list of N integers, find the closest pair and output their difference.
    - 2 <= N <= 10^6
    - Numbers are no larger than 10^6 in their absolute values.
    - Sample
        - [1, 4, -2, 6, 10] => closest pair is (4, 6) => answer is 2

A Straightforward Approach

```
#include <iostream>
using namespace std;

#define MAXN 1000000
#define INFINITY (int)1e9
int a[MAXN];
int main() {
  int N;
  scanf("%d", &N);
  for (int i = 0; i < N; i++) scanf("%d", a + i);
  int ans = INFINITY;
  for (int i = 0; i < N; i++) {
    for (int j = i + 1; j < N; j++) {
      ans = min(ans, abs(a[i] - a[j]));
    }
  }
  printf("%d\n", ans);
  return 0;
}
```

Let’s verify our solution



```
[bowen@MacBook ~ $ cat input.txt
5
1 4 -2 6 10
[bowen@MacBook ~ $ g++ -o main main.cpp
[bowen@MacBook ~ $ ./main < input.txt
2
bowen@MacBook ~ $
```

Prepare an input file with the test case

Compile the source code into executable

Run the executable on the test case

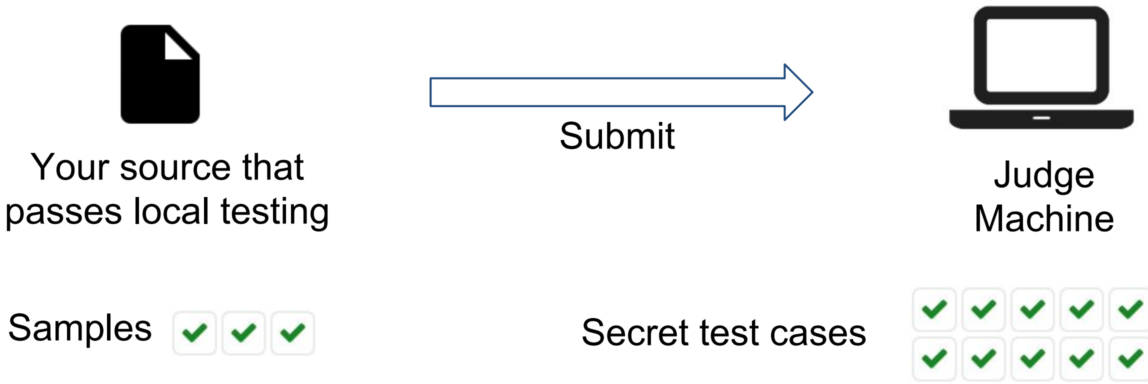
* You may have IDE to do these things for you

1D Closest Pair

- We have written a solution program that:
  - Read from stdin to get the problem’s input
  - Compute the answer for the given task
  - Write to stdout to produce the answer
  - We tested it locally to verify that it works on sample input.
- When we finish the solution, we submit it to an online judge system that will verify its correctness.

- **Note:** The particular details of the implementation are not important:
  - We are language-neutral in our discussion. In practice, it is up to you to implement the idea in your language of choice.
  - There are many ways to implement a same idea. Feel free to use your own coding style.

Online Judge (OJ) Black-box Testing


Your source that passes local testing
Submit
Judge Machine
Samples ✓ ✓ ✓
Secret test cases ✓

You should always do local tests:

- Submit only if your solutions passes samples
- Recheck samples on code changes

The judge checks that:

- Your code produces expected output
- Your code finishes in the time limit
- All cases must pass

NYU

Why Online Judge

- Fast and responsive (instant feedback)
  - A great pleasure of this course is to get your code **Accepted**!
- Truly objective
  - No more argument of correct or wrong
- Thorough and zero error tolerance (strong test cases)
  - It greatly helps hone our coding skills because bugs are never allowed.
  - This is different from algorithm solutions on paper, which may miss tiny but critical detail.
- Fully automated

(No more graders, yay!)

1D Closest Pair

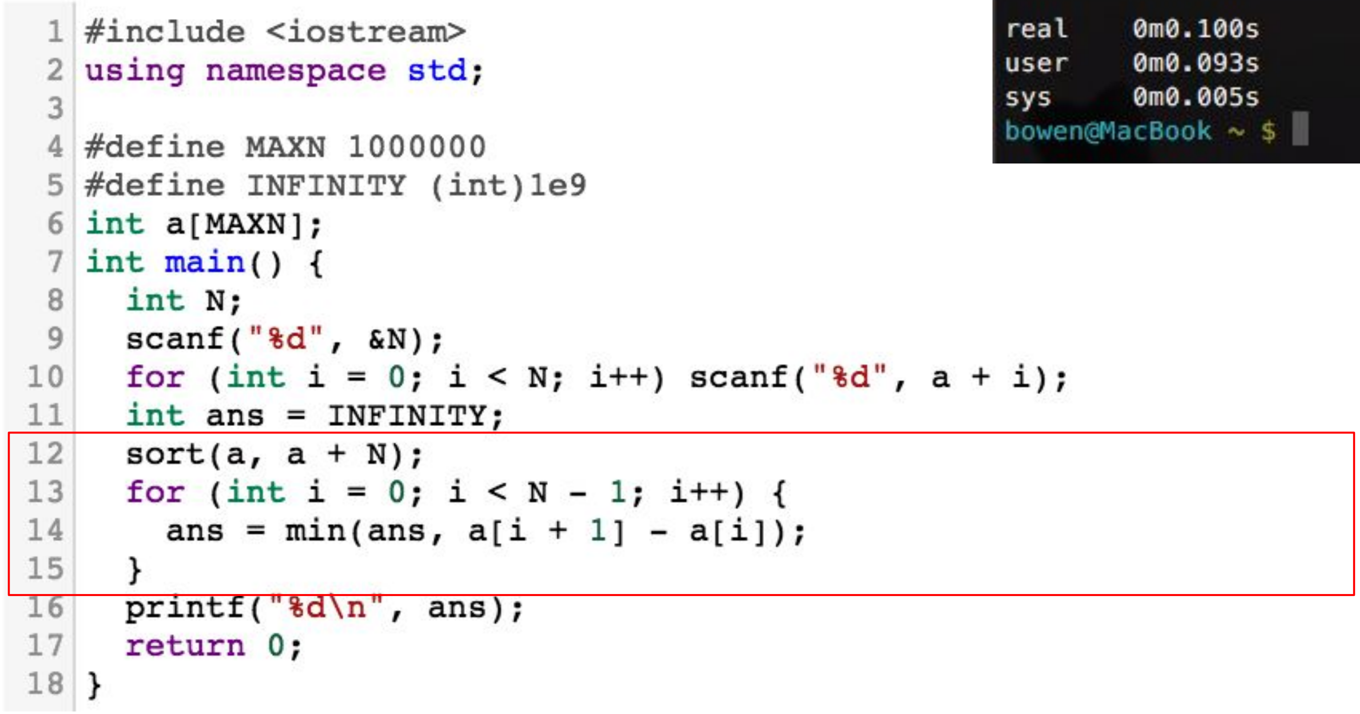
- Now that we submit our closest pair solution...
- We got **Time Limit Exceeded (TLE)** verdict from the online judge.
  - On some case(s) the solution did not finish within the problem's time limit.
- The straightforward approach produces correct output, but it is generally considered **incorrect**:
  - Recall that we have the list length 2 <= N <= 10^6.
  - If we give it a list that is 10^6 long (we can randomly generate one into input.txt), it may not finish in 10 minutes.
  - In this course we only accept solutions that will finish in several seconds.

1D Closest Pair

- The straightforward solution looks at N^2 pairs.

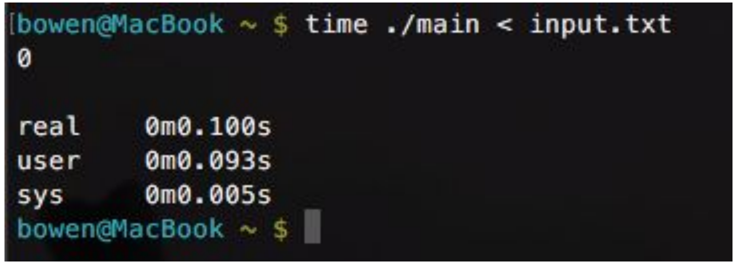
- We optimize it by:
  - First sort the list
    - We know that sorting a list is much faster than checking N^2 pairs. More precisely, sorting does roughly NlogN comparisons. (here comes our *algorithmic* way)
  - Then look at only adjacent pairs. There are only N-1 pairs to check.
  - Example:
    - [1, 4, -2, 6, 10] => sort and get [-2, 1, 4, 6, 10]
    - Only look at adjacent pairs (-2, 1), (1, 4), (4, 6) and (6, 10)

An Improved Approach



```
#include <iostream>
using namespace std;

#define MAXN 1000000
#define INFINITY (int)1e9
int a[MAXN];
int main() {
  int N;
  scanf("%d", &N);
  for (int i = 0; i < N; i++) scanf("%d", a + i);
  int ans = INFINITY;
  sort(a, a + N);
  for (int i = 0; i < N - 1; i++) {
    ans = min(ans, a[i + 1] - a[i]);
  }
  printf("%d\n", ans);
  return 0;
}
```

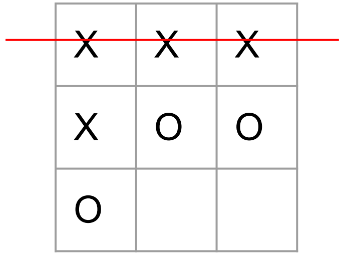


```
bowen@MacBook ~ $ time ./main < input.txt
0

real    0m0.100s
user    0m0.093s
sys     0m0.005s
bowen@MacBook ~ $
```

Another Example: Tic-Tac-Toe

- Tic-Tac-Toe: A classic, simple 2-player game
  - Players take turns to place markers on a 3x3 grid.
  - One player uses cross X, the other uses circle O. X goes first.
  - Whoever reaches three markers in a row/column/diagonal wins.



X wins

- **Problem:** Given a board configuration, determine:
  - If it is a valid board (Some board is impossible)
  - If it is valid, check if one of the players has won
  - If nobody wins, find who shall play next, or it is a draw.
  - Samples:

	X	X
X	O	X
O		

invalid

X	X	X
X	O	O
O		

X wins

X	X	
O		
O	X	

O next

X	X	O
O	X	X
X	O	O

draw

Tic-Tac-Toe: Some Thoughts

- When will a board be invalid?
    - The number of X's and O's shall differ by at most one.
    - O cannot be more than X.
    - Not both players win.
- When will a player win?
    - If there exists a row/column/diagonal with three same markers.
- Who is the next player?
    - If there are more X's than O's, then "O next";
    - Otherwise it is "X next".
- When is it a draw?
    - If the board is full and valid, then it is a draw.

- However, after submission we got **Wrong Answer (WA)** verdict from the online judge.
  - On some test case(s) our solution does not produce correct output.
  - In most cases, we do not know the failed case's input as the judge's test data is *secret*.
    - Why secret?
      - Learning how to debug your code by proofreading (instead of others finding bugs for you) is a critical programming skill.
      - In practice, it is the programmer's responsibility to ensure the code's correctness. It is often up to yourself to come up with the test cases.
- What can go wrong?
  - There is a bug in the code. We didn't correctly implement the logic from the previous slide.
    - Very likely (*"code without bugs is not code"*, hehe)
  - The judge is wrong
    - Very, very, very unlikely. Usually judge's answer is carefully matched by multiple writer/tester's solutions.
  - Our idea is wrong.
    - The algorithm misses some condition(s).

A Breaker

- When will a board be invalid?
  - The number of X’s and O’s shall differ by at most one. ✓
  - O cannot be more than X. ✓
  - Not both players win. ✓
- When will a player win?
  - If there exists a row/column/diagonal with three same markers.
- Who is the next player?
  - If there are more X’s than O’s, then “O next”;
  - Otherwise it is “X next”.
- When is it a draw?
  - If the board is full and valid, then it is a draw.

X		O
X		O
X	O	

Our wrong output:
X wins

Correct output
invalid

NYU

Tic-Tac-Toe: A Fix

- When will a board be invalid?
  - The number of X’s and O’s shall differ by at most one.
  - O cannot be more than X.
  - Not both players win.
  - When number of X’s equals number of O’s, then X must not win.
- When will a player win?
  - If there exists a row/column/diagonal with three same markers.
- Who is the next player?
  - If there are more X’s than O’s, then “O next”;
  - Otherwise it is “X next”.
- When is it a draw?
  - If the board is full and valid, then it is a draw.

NYU

Tic-Tac-Toe: After the Fix

- When will a board be invalid?
  - The number of X’s and O’s shall differ by at most one.
  - O cannot be more than X.
  - Not both players win.
  - When number of X’s equals number of O’s, then X must not win.
- When will a player win?
  - If there exists a row/column/diagonal with three same markers.
- Who is the next player?
  - If there are more X’s than O’s, then “O next”;
  - Otherwise it is “X next”.
- When is it a draw?
  - If the board is full and valid, then it is a draw.

But we still got **WA**. Why?

We will leave it as an exercise.

NYU

Tic-Tac-Toe: One Step Further

- For the same settings (tic-tac-toe game), we can also ask a different (more interesting) question: “Given a valid board, which player will win, assuming both players play optimally?”
- We will learn how to address such questions later into the course.

Let’s summarize. Here is how we do problem solving:

- We are given a (fun) task
- We think about possible solutions and pick one (using our algorithm knowledge)
- We analyze the solutions’ correctness and efficiency (using our algorithm knowledge)
- We write the solution as working code
- We test the solution
- We submit the solution
- Aha! (or we go back and fix things)

Our problem solution requires:

- Good understanding of the task (what and where is its challenge?)
- A *correct* and *efficient* algorithm
    - Efficiency is as important as correctness!
- Careful implementation so that our code is bug-free
    - Proficient coding skill
    - Code design (e.g. avoid code repetition, avoid huge if-else branches, etc.)

NYU

Syllabus (Tentative)

- Basic coding skills
  - input parsing, output formatting, I/O techniques
- Code performance analysis
  - asymptotic complexity: tell if an algorithm will be fast enough
- Data structures
  - fundamental: Array, linked list, stack, queue, binary search tree, hash table, heap
  - advanced: bitmask, segment/fenwick tree, square-root structures
- Problem solving techniques
  - complete search (brute-force), greedy algorithms, divide-and-conquer
- Dynamic programming
  - classic models: subset sum, LCS, LIS, Knapsack, edit distance
  - advanced: higher-dimensional DP, games, adhoc DP formulation, DP optimization

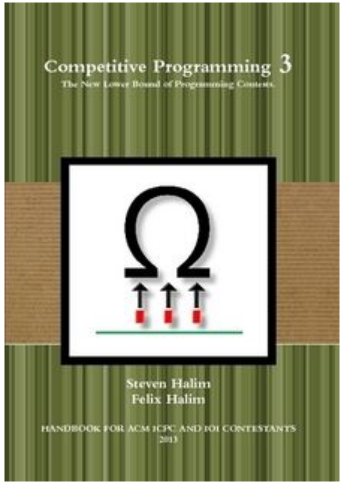
NYU

Syllabus (cont'd)

- Graph algorithms
    - traversal, shortest path, minimum spanning tree, topological sort, strongly connected components, trees, graph matching, flow algorithms
- String matching
    - KMP, Rabin-Karp, suffix structures
- Mathematics
    - basic number theory, combinatorics, probability
- Computational geometry
    - lines, segments, circles
    - convex hull

Textbook/Reference

- Steven and Felix Halim's **Competitive Programming**, 3rd Ed.
  - https://cpbook.net/
  - Paperback version is $26.99
  - eBook (PDF) is $19.99


Competitive Programming 3
The New Lower Bound of Programming Contests
Steven Halim
Felix Halim
HANDBOOK FOR ACM ICPC AND IOI CONTESTANTS
2013

NYU

Grading

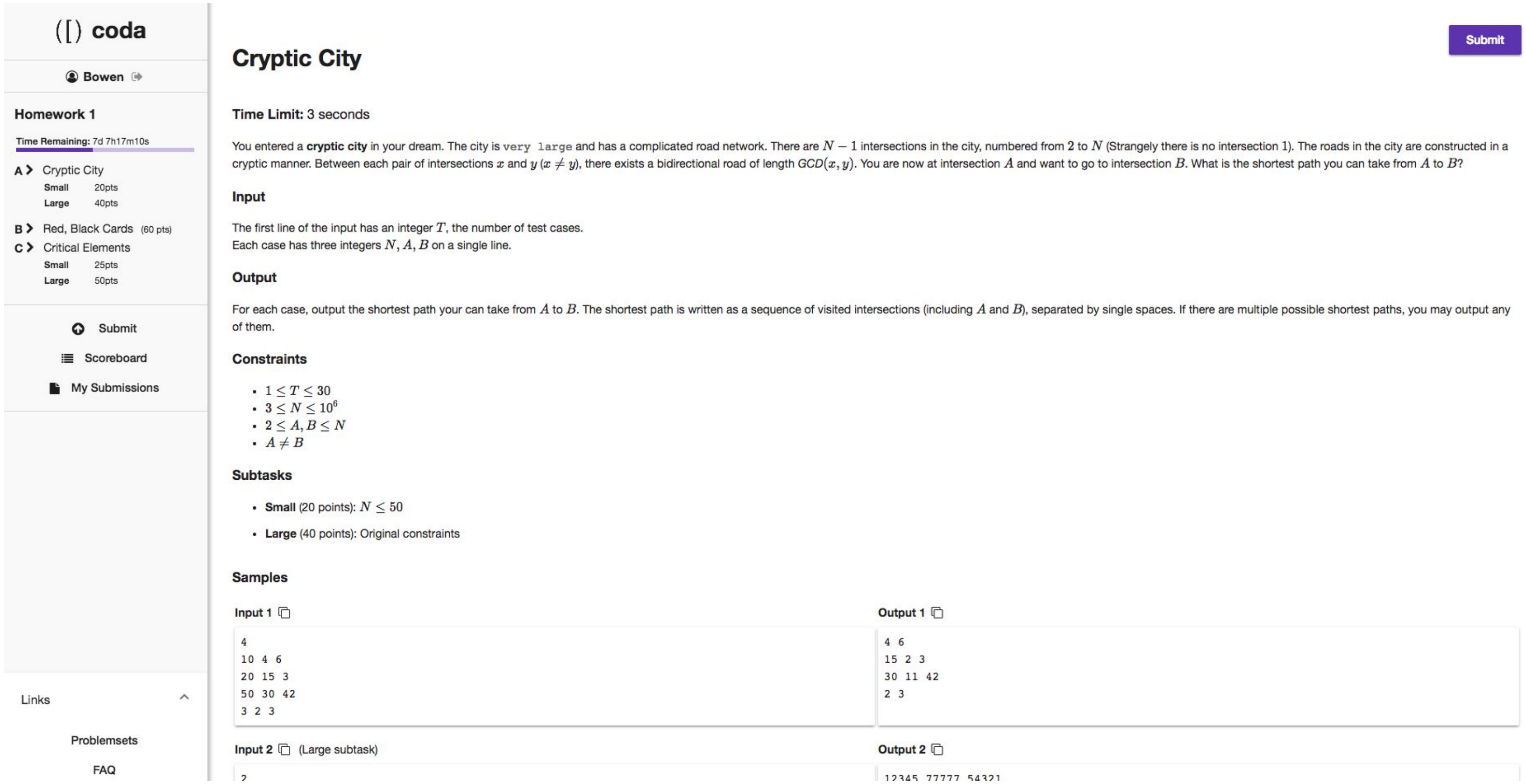
- Homework: 15%
- Recitation: 10%
- Free Study: 10%
- Quizzes: 20%
- Midterm: 20%
- Final: 25%
- Extra credit

Homework

- When
  - Released within a week after a topic is covered in lecture.
  - Due one week after release.
- What
  - Each homework is a problemset that consists of 2-3 problems we designed to help you learn the course material.
  - 1-2 problemsets will be released per week based on lecture progress.
- How
  - Use the coda system for submitting and grading
  - Each homework problemset is weighted equally
  - **3 homework freebies:** 3 lowest scores are dropped

NYU

coda



Cryptic City

Time Limit: 3 seconds

You entered a **cryptic city** in your dream. The city is `very large` and has a complicated road network. There are $N-1$ intersections in the city, numbered from 2 to N (Strangely there is no intersection 1). The roads in the city are constructed in a cryptic manner. Between each pair of intersections x and y ($x \neq y$), there exists a bidirectional road of length $GCD(x, y)$. You are now at intersection A and want to go to intersection B. What is the shortest path you can take from A to B?

Input

The first line of the input has an integer T, the number of test cases.
Each case has three integers N, A, B on a single line.

Output

For each case, output the shortest path your can take from A to B. The shortest path is written as a sequence of visited intersections (including A and B), separated by single spaces. If there are multiple possible shortest paths, you may output any of them.

Constraints

- $1 \leq T \leq 30$
- $3 \leq N \leq 10^6$
- $2 \leq A, B \leq N$
- $A \neq B$

Subtasks

- **Small** (20 points): $N \leq 50$
- **Large** (40 points): Original constraints

Samples

Input 1

```
4
10 4 6
20 15 3
50 30 42
3 2 3
```

Output 1

```
4 6
15 2 3
30 11 42
2 3
```

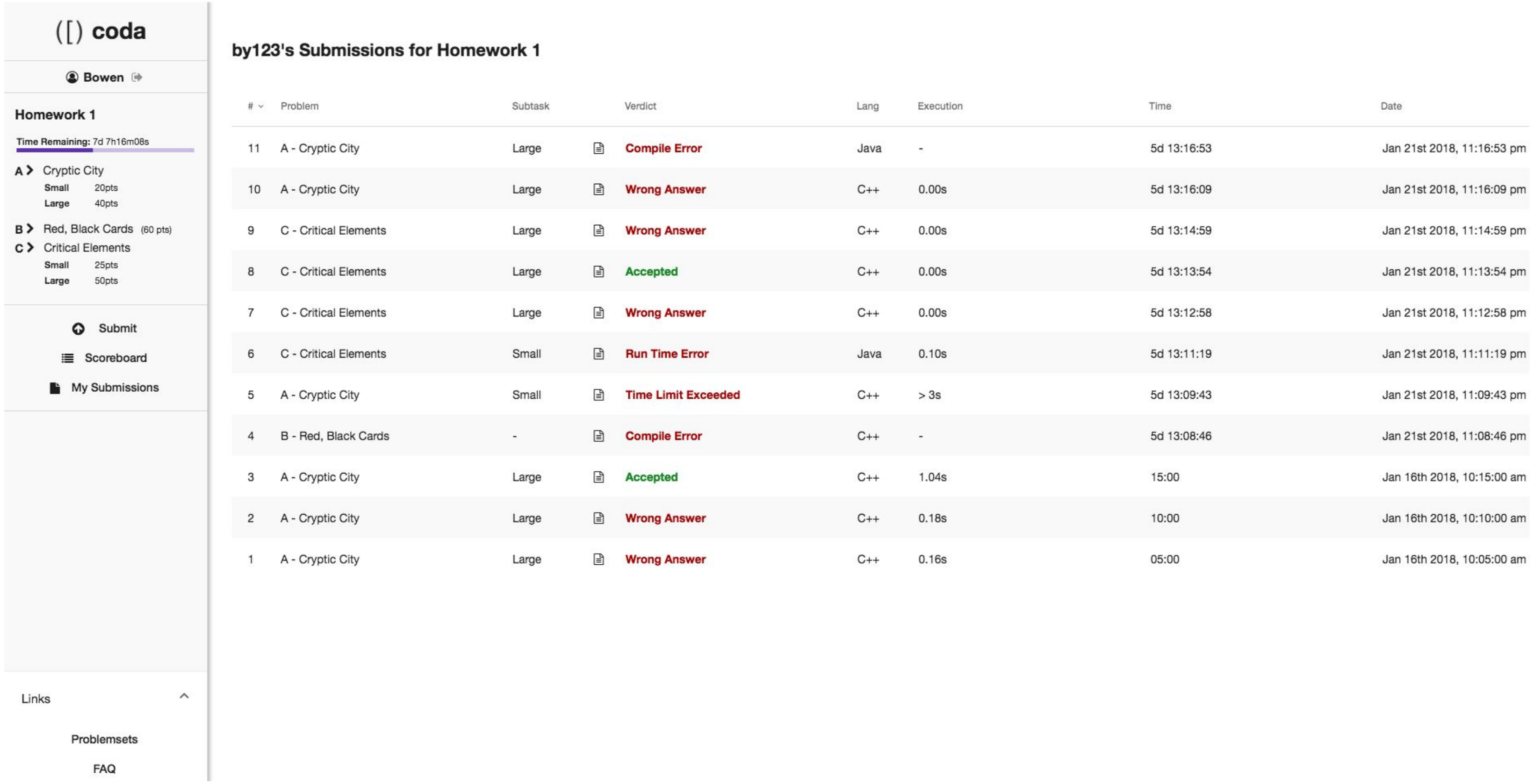
Input 2 (Large subtask)

```
2
```

Output 2

```
12345 77777 54321
```

coda



([) coda

Bowen

Homework 1

Time Remaining: 7d 7h16m08s

A > Cryptic City
Small 20pts
Large 40pts

B > Red, Black Cards (60 pts)

C > Critical Elements
Small 25pts
Large 50pts

Submit

Scoreboard

My Submissions

Links

Problemsets

FAQ

by123's Submissions for Homework 1

#	Problem	Subtask	Verdict	Lang	Execution	Time	Date
11	A - Cryptic City	Large	Compile Error	Java	-	5d 13:16:53	Jan 21st 2018, 11:16:53 pm
10	A - Cryptic City	Large	Wrong Answer	C++	0.00s	5d 13:16:09	Jan 21st 2018, 11:16:09 pm
9	C - Critical Elements	Large	Wrong Answer	C++	0.00s	5d 13:14:59	Jan 21st 2018, 11:14:59 pm
8	C - Critical Elements	Large	Accepted	C++	0.00s	5d 13:13:54	Jan 21st 2018, 11:13:54 pm
7	C - Critical Elements	Large	Wrong Answer	C++	0.00s	5d 13:12:58	Jan 21st 2018, 11:12:58 pm
6	C - Critical Elements	Small	Run Time Error	Java	0.10s	5d 13:11:19	Jan 21st 2018, 11:11:19 pm
5	A - Cryptic City	Small	Time Limit Exceeded	C++	> 3s	5d 13:09:43	Jan 21st 2018, 11:09:43 pm
4	B - Red, Black Cards	-	Compile Error	C++	-	5d 13:08:46	Jan 21st 2018, 11:08:46 pm
3	A - Cryptic City	Large	Accepted	C++	1.04s	15:00	Jan 16th 2018, 10:15:00 am
2	A - Cryptic City	Large	Wrong Answer	C++	0.18s	10:00	Jan 16th 2018, 10:10:00 am
1	A - Cryptic City	Large	Wrong Answer	C++	0.16s	05:00	Jan 16th 2018, 10:05:00 am

NYU

Homework Grading

- Each problem has a pre-assigned score.
    - Some problems have subtasks that allow partial scoring.
    - Judgment for each subtask is binary (full score if correct, otherwise zero).
- We highly discourage blindly submitting just for testing. Therefore we have score penalty for failed submissions:
    - Each incorrect submission results in 10% loss of a problem's score.
        - You get a minimum of 30%.
    - To allow for some leniency, a problemset may have X freebies: the first X wrong submissions for each problem are exempted from penalty.
    - Example:
        - Suppose a problem has X=3 freebies, and John solves a problem of 50 points with 5 attempts (4 wrong and 1 correct), how many points does John get?
        - Since X=3, only one wrong submission has a penalty of 50 * 10% = 5 points. So John receives 45 for this problem.
- See coda FAQ and scoreboard for more details.

Recitation

- What
  - Practice with additional problems from external sources (other OJs) that will help you go over the course materials.
- How
  - Organized as practice contests:
    - 3-5 problems
    - Time constrained
      - Why? This will help with quizzes and exams.
  - Recitation grades are given by attendance
    - Solving at least 1 problem will count as attendance
    - Recitations are weighted equally
  - Problem solutions are discussed after the contest.
  - Use the vjudge system: https://vjudge.net/
  - **2 recitation freebies:** You can skip two recitations.

NYU

Free Study

- When
  - In the second half of the course, after we cover enough topics you can start the free study.
  - Submit your report before the final.
- How
  - 2-3 students per group
  - Select a topic that is relevant to the course materials and dive deeper into it by:
    - Searching for problems that are relevant from other resources
    - Solve these problems and write their solutions
    - Summarize the techniques applied and what you’ve learned
  - We will provide a list of topics for you to choose from. But you are also free to choose your own topic as long as it is related to APS. We will give more details later into the course.
  - Graded by report quality: correctness, depth, etc.

NYU

Quizzes and Exams

- When
  - **At recitation time!**
    - Except for the final that is scheduled otherwise by the department
  - Please see Piazza for dates
- What
  - Essentially they are like homework problemsets weighted a bit larger.
- How to prepare
  - Go over course materials: lecture slides, textbook.
  - Do homework and recitations
    - Avoid using too much help from others
  - Extra problems from other OJs

Important Logistics

- Piazza
  - Course announcement and discussion.
  - Signup: https://piazza.com/nyu/spring2018/csciua0480004
- Coda (HW and quizzes/exams)
  - signup instructions will be posted on Piazza
- Vjudge (recitation)
  - Create an account with username: {netID}_CS480S18
    - For example, if your netID is by123, your username is: by123_CS480S18
  - Apply to the group https://vjudge.net/group/aps-spring18

- Grades will be released on NYUClasses
  - Note that you can calculate the grades yourself as HW/Recitation/Exam scores can be directly found on coda/vjudge.
  - It is also used for receiving free study reports.

Homework Today

- Complete the logistics signup’s as on previous slide.

- HW1 Warmup
    - Remember that you have to read from stdin and output to stdout
        - Refer to sample I/O code