Introduction

Atli FF

Árangursrík forritun og lausn verkefna

School of Computer Science
Reykjavík University

Course Overview

Welcome

- T-414-AFLV, Árangursrík forritun og lausn verkefna
- Arnar Bjarni Arnarson, arnarar@ru.is
- Atli Fannar Franklín, aff6@hi.is, atlif@ru.is

Learning goals

- At its heart this course is about problem solving.
- In this course you will learn to take a problem and:
    - analyse the constraints of the problem,
    - take those and find applicable algorithms and data structures,
    - convert those ideas into a functional program,
    - do this quickly and under pressure,
    - producing a program without bugs or other errors.

Getting there

- We will get to this point by going over a number of things:
  - look at common problem types,
  - cover different kinds of problem solving paradigms,
  - show common algorithms and data structures you should know already in action,
  - introduce new less common algorithms and data structures,
  - go over useful theories from a few branches of mathematics,
  - practice solving problems,
  - practice more,
  - practice more,
  - and practice more!

Teaching material

- This course will loosely follow Competitive Programming by Steven Halim
- First edition can be downloaded from the book homepage **https://cpbook.net/**
- We will loosely follow the third edition which can be ordered online
- The book is not mandatory reading, but can provide extra examples and code
- A different set of slides for additional reading can also be found at **https://github.com/Kakalinn/tol607g-glaerur**
- Just as a heads up, those slides are in Icelandic
- Other good material can be found on **cp-algorithms.com**, **codeforces.com**, **wiki.algo.is** and more

Piazza

- Piazza can be used to ask questions
- Can be accessed via Canvas, otherwise it can be found at `https://piazza.com/ru.is/fall2023/t414aflv`
- Before posting questions, read the pinned announcement on what questions can be made publically

Course Schedule

Week no.	Date	Topic
0	14.08	Warmup
1	21.08	Time complexity, languages and built-ins, ad-hoc
2	28.08	Complete search and greedy algorithms
3	04.09	Divide and conquer, dynamic programming part 1
4	11.09	Dynamic programming part 2
5	18.09	Data structures
6	25.09	Unweighted and undirected graphs
7	02.10	Weighted and/or directed graphs
8	09.10	Number Theory
9	16.10	Combinatorics
10	23.10	Geometry
11	30.10	String processing
12	05.11	Network flow & misc.

Problem Sets

- Each weekly topic will come with a corresponding problem set
- Groups of up to three people can discuss the problems, but each individual must write and hand in their own code
- We will check for similar submissions, and take action if we think that people are cheating
- Kattis also has a built in anti-cheat feature, which in my personal experience has been plenty good enough to catch a lot of cheaters
- The deadline for problem sets is always the next Sunday
- Late handins will not be accepted
- Kattis' verdict is law

Bonus Problems

- Some problem sets contain bonus problems
- These problems are often either harder or on bonus topics not covered in lectures
- Bonus problems can lift your grade
- Bonus problems can be turned in until the end of the course

Problem structure and Kattis

Problem Structure

- A typical programming contest problem usually consists of a
  - Problem description
  - Input description
  - Output description
  - Example input/output
  - A time limit in seconds
  - A memory limit in bytes
- You are asked to write a program that solves the problem for all valid inputs
- The program reads from stdin and writes to stdout, stderr is ignored
- The program must not exceed time or memory limits

Stdin/stdout

Language	Stdin	Stdout
C++	`cin` `scanf`	`cout` `printf`
Python	`input` `sys.stdin.readline`	`print` `sys.stdout.write`
Java	`Scanner(System.in)`	`System.out.println`

- For Java we recommend using Kattio

Types

Type	Size
`char`	$[-128, 127]$
`int`	$[-2^{31}, 2^{31} - 1]$
`unsigned int`	$[0, 2^{32} - 1]$
`long long`	$[-2^{63}, 2^{63} - 1]$
`unsigned long long`	$[0, 2^{64} - 1]$
`double`	Complicated, limited accuracy

- Java is similar.
- In python the floats are the same, but the integer types will automatically change to a dynamic type that can be as big as needed instead of overflowing.

Example Problem

Problem description
Write a program that multiplies pairs of integers.

Input description
Input starts with one line containing an integer T, where $1 \leq T \leq 100$, denoting the number of test cases. Then T lines follow, each containing a test case. Each test case consists of two integers A, B, where $-2^{20} \leq A, B \leq 2^{20}$, separated by a single space.

Output description
For each test case, output one line containing the value of $A \times B$.

Example Problem

Sample input	Sample output
4	
3 4	12
13 0	0
1 8	8
100 100	10000

Possible Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        int A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

Possible Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        int A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct?

Possible Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        int A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct?
- What if $A = B = 2^{20}$?

Possible Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        int A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct?
- What if $A = B = 2^{20}$? The output is 0

Possible Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        int A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct? No!
- What if $A = B = 2^{20}$? The output is 0

Fixed Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        long long A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

Fixed Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        long long A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct?

Fixed Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        long long A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct?
- The values are at most 2^{20} in absolute value, so their product is at most 2^{40} in absolute value, which fits.

Fixed Solution

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

int main() {
    int T;
    cin >> T;
    for(int t = 0; t < T; t++) {
        long long A, B;
        cin >> A >> B;
        cout << A * B << endl;
    }
    return 0;
}
```

- Is this correct? Yes!
- The values are at most 2^{20} in absolute value, so their product is at most 2^{40} in absolute value, which fits.

Automatic Judging

- The problems will be on **https://ru.kattis.com/**
- You will submit your solutions to Kattis, and get immediate feedback about the solution
- You may submit in whatever language you prefer (which Kattis supports), but keep in mind that if you choose something other than C/C++, Java and Python we might not be as able to help

Verdicts

- Feedback is (intentionally) limited
- You will (almost always) receive one of the following verdicts:
  - Accepted (AC)
  - Wrong Answer (WA)
  - Compile Error (CE)
  - Run Time Error (RTE)
  - Time Limit Exceeded (TLE)
  - Memory Limit Exceeded (MLE)
- Neither we nor Kattis will give away info on the test data used to test solutions

Time complexities:

What is a time complexity?

- Saying a program runs in $\mathcal{O}(f(n))$ means that for some C, n_0 the program will take at most $Cf(n)$ steps to finish for $n \geq n_0$
- Ignoring constants is necessary, otherwise you could change the time complexity just by making the CPU faster or adding more cores
- Time complexities are very useful for napkin math on whether a solution will pass time constraints

Calculate time complexities

- A good rule of thumb is that we have 10^8 operations per second

Calculate time complexities

- A good rule of thumb is that we have 10^8 operations per second
- Say we want to sort $n \leq 10^6$ integers in 3 seconds.
- Can we use a $\mathcal{O}(n^2)$ bubble sort or do we need to implement the more complex $\mathcal{O}(n \log(n))$ merge sort?

Calculate time complexities

- A good rule of thumb is that we have 10^8 operations per second
- Say we want to sort $n \leq 10^6$ integers in 3 seconds.
- Can we use a $\mathcal{O}(n^2)$ bubble sort or do we need to implement the more complex $\mathcal{O}(n \log(n))$ merge sort?
- Bubble sort would take $\sim 10^{12}$ operations or about 10^4 seconds, which is far too slow.
- The merge sort would be around 0.2 seconds, which suffices.

Time complexities cntd.

- Always use the simplest solution that suffices. If n had been 10^3 bubble sort would suffice.
- It can be good to be able to estimate these things quick in your head.
- Rules of thumb can be useful, things like $2^{10} \approx 10^3$.
- Logarithms are usually base 2, so like earlier if $n = 10^6$ for $n \log(n)$ we can estimate it as $10^6 \log_2(2^{20})$ or $2 \cdot 10^7$.

Complexity overview

n	Slowest Accepted Algorithm	Example
≤ 10	$O(n!)$	Enumerating a permutation
≤ 15	$O(2^n \times n^2)$	Traveling salesperson DP
≤ 20	$O(2^n), O(n^5)$	Bitmask DP
≤ 50	$O(n^4)$	Blossom algorithm
$\leq 10^2$	$O(n^3)$	Floyd Warshall algorithm
$\leq 10^3$	$O(n^2)$	Bubble/Selection/Insertion sort
$\leq 10^5$	$O(n \log_2 n)$	Merge sort, building a Segment tree
$\leq 10^6$	$O(n)$	Linear scans like prefix sums
$> 10^8$	$O(\log_2 n), O(1)$	Direct formulas or digit operations

Language features

- Kattis allows the use of standard libraries, so get acquainted with what your language of choice has to offer.
- C++ sorts with `sort(a.begin(), a.end())`, python has `a.sort()` and Java has `Arrays.sort(a)`.
- All three languages support common mathematical operations like square roots and complex numbers.
- C++ can do binary search with `lower_bound` and `upper_bound`, python can `import bisect`.
- There's plenty more! Regex, pseudo-randomness and plenty of data structures.

Built-in data structures overview

- The simplest data structures are arrays, which allow $O(1)$ access to a linear sequence of elements.
- Vectors / ArrayLists allow $O(1)$ appending or popping from the back additionally.
- Set / TreeSet allow $O(\log(n))$ insertion, deletion and lookup and keep the elements ordered, so they can be iterated over. Python has no equivalent.
- Unordered set / HashSet / Python Set allow $O(1)$ insertion, deletion and lookup on average.

Ad hoc problems

Ad hoc

- Ad hoc problems are usually the simplest kind of problem (though certainly not always the easiest)
- Just do what the description says
- Time limits should not be an issue
- Sometimes have long and misleading descriptions or tricky edge cases
- Harder such problems are often hard to implement

Example - Baby bites

Arild just turned 1 year old, and is currently learning how to count. His favorite thing to count is how many mouthfuls he has in a meal: every time he gets a bite, he will count it by saying the number out loud.

Unfortunately, talking while having a mouthful sometimes causes Arild to mumble incomprehensibly, making it hard to know how far he has counted. Sometimes you even suspect he loses his count! You decide to write a program to determine whether Arild's counting makes sense or not.

Input

The first line of input contains an integer n ($1 \leq n \leq 1\,000$), the number of bites Arild receives. Then second line contains n space-separated words spoken by Arild, the i'th of which is either a non-negative integer a_i ($0 \leq a_i \leq 10\,000$) or the string "`mumble`".

Output

If Arild's counting might make sense, print the string "`makes sense`". Otherwise, print the string "`something is fishy`".

Solution

```
n = int(input())
words = input().split()
valid = True

for i in range(n):
    if words[i] != "mumble" and l[i] != str(i + 1):
        valid = False
        break

if valid:
    print("makes sense")
else:
    print("something is fishy")
```