Olympiad in Informatics & Association for Computing Machinery–International Collegiate Programming Contest
Olympic Tin Học Sinh Viên OLP & ACM-ICPC

Nguyễn Quản Bá Hồng[1]

Ngày 14 tháng 6 năm 2025

[1]A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam. E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

Mục lục

Preface

Abstract

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Olympiad in Informatics & Association for Computing Machinery–International Collegiate Programming Contest – Olympic Tin Học Sinh Viên OLP & ICPC*.

  PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/NQBH_OLP_ICPC.pdf.

  TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/NQBH_OLP_ICPC.tex.

- Codes:
  - C: https://github.com/NQBH/advanced_STEM_beyond/tree/main/OLP_ICPC/C.
  - C++: https://github.com/NQBH/advanced_STEM_beyond/tree/main/OLP_ICPC/C++.
  - Python: https://github.com/NQBH/advanced_STEM_beyond/tree/main/OLP_ICPC/Python.

Preliminaries – Kiến thức chuẩn bị

Resources – Tài nguyên.

1. [WW16]. Yonghui Wu, Jiande Wang. *Data Structure Practice for Collegiate Programming Contests & Education.*
2. [WW18]. Yonghui Wu, Jiande Wang. *Algorithm Design Practice for Collegiate Programming Contests & Education.*
3. Codeforces https://codeforces.com/.
4. CSES Problem Sets. https://cses.fi/problemset/.

Some critical-thinking questions:

Question 1 (Generalization; main ideas of a solution/proof)**.** *What are main ideas of a solution or a proof of a problem that can be used to generalize the original problem?*

Question 2 (Link[1])**.** *Can we draw some link(s) between different problems? Even they are in different categories: algebra, analysis, & combinatorics.*

Remark 1 (Repeat & mathematical induction – Lặp & quy nạp toán học)**.** *Nếu bài toán có chứa $n \in \mathbb{N}^\star$ tổng quát hoặc chứa số tự nhiên của năm ra đề, e.g., 2025, thì đưa* 2025 *về $n \in \mathbb{N}^\star$, rồi sử dụng các kỹ thuật toán học để đưa về phép lặp, hoặc sử dụng phương pháp quy nạp toán học (method mathematical induction).*

Notation – Ký hiệu

- $\overline{m,n} := \{m, m+1, \dots, n-1, n\}$, $\forall m, n \in \mathbb{Z}$, $m \le n$. Hence the notation "for $i \in \overline{m,n}$" means "for $i = m, m+1, \dots, n$", i.e., chỉ số/biến chạy i chạy từ $m \in \mathbb{Z}$ đến $n \in \mathbb{Z}$. Trong trường hợp $a, b \in \mathbb{R}$, ký hiệu $\overline{a,b} := \overline{\lceil a \rceil, \lfloor b \rfloor}$ có nghĩa như định nghĩa trước đó với $m := \lceil a \rceil, n := \lfloor b \rfloor \in \mathbb{Z}$; khi đó ký hiệu "for $i \in \overline{a,b}$" với $a, b \in \mathbb{R}$, $a \le b$ có nghĩa là "for $i = \lceil a \rceil, \lceil a \rceil + 1, \dots, \lfloor b \rfloor - 1, \lfloor b \rfloor$, i.e., chỉ số/biến chạy i chạy từ $\lceil a \rceil$ đến $\lfloor b \rfloor \in \mathbb{Z}$.
- $\lfloor x \rfloor, \{x\}$ lần lượt được gọi là *phần nguyên & phần lẻ* (integer- & fractional parts) của $x \in \mathbb{R}$, see, e.g., Wikipedia/floor & ceiling functions, Wikipedia/fractional part.
- $x_+ := \max\{x, 0\}$, $x_- := \max\{-x, 0\} = -\min\{x, 0\}$ lần lượt được gọi là *phần dương & phần âm* (positive- & negative parts) của $x \in \mathbb{R}$.
- s.t.: abbreviation of 'such that'.
- w.l.o.g.: abbreviation of 'without loss of generality'.

Question 3 (MO $\mapsto$ OI: Mathematical Olympiads $\mapsto$ Olympiads of Informatics)**.** *Những bài Olympic Toán học, e.g., VMO, USAMO, China MO, đặc biệt là IMO, IMO shortlists, etc. nào có thể code được? Đưa các tiêu chí cho việc code-able.*

Answer. Trước hết các bài toán Olympic có thể code được gồm các bài về mảng số (arrays of real numbers, or finite sequence of real numbers), dãy số (infinite sequences of of real numbers), hình học tính toán (computational geometry), hình học tổ hợp (combinatorial geometry), số học/lý thuyết số (number theory). Đối với từng bài, mới biết có thể ra bài Olympic Tin tương ứng hay không, & còn tùy thuộc vào trình độ của người chế đề.

Các bài hình học thiên về thuần chứng minh thường khó ra đề Olympic Tin tương ứng 1 cách trực tiếp được, trừ khi là `AlphaGeometry`. Prove me wrong. □

[1] Watch, e.g., IMDb/Shi Guang Dai Li Ren ⋆ Link Click (2021–).

Chương 1

Basic Competitive Programming – Lập Trình Thi Đấu Cơ Bản

Contents

Resources – Tài nguyên.

1. [Laa20]. Antti Laaksonen. *Guide to Competitive Programming: Learning & Improving Algorithms Through Contests*.
2. [Thư+21a]. Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết. *Nhập Môn Lập Trình*.
3. [Thư+21b]. Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết, Đặng Bình Phương. *Kỹ Thuật Lập Trình*.
4. [TTK21]. Trần Đan Thư, Đinh Bá Tiến, Nguyễn Tấn Trần Minh Khang. *Phương Pháp Lập Trình Hướng Đối Tượng*.

1.1 The art of handling inputs & formatting outputs – Nghệ thuật xử lý các dạng đầu vào & định dạng các dạng đầu ra

To handle various types of inputs & format various types of outputs, see, e.g.:

- Peking University Judge Online for ACM/ICPC (POJ)/FAQ. See, e.g., [Laa20; Laa24, Chap. 2, Subsect. 2.1.1, pp. 10–11].

To compile a C++ program in Linux, run in Terminal:

```
$ g++ -O2 -Wall program_name.cpp -o program_name
$ ./program_name
```

or if you want to transfer input file into it & print output into Terminal screen:

```
$ ./program_name < program_name.inp
```

or if you want to transfer input file into it & print output into a file:

```
$ ./program_name < program_name.inp > program_name.out
```

- Geeks4Geeks/`std::endl` vs. `\n` in C++: https://www.geeksforgeeks.org/endl-vs-n-in-cpp/.
- `i++` vs. `++i`: StackOverflow/Is there a performance difference between i++ & ++i in C?

1.2 Repeat/Loop – Lặp

1.3 String data – Kiểu dữ liệu chuỗi

1.4 Array data – Kiểu dữ liệu mảng

Về mặt toán học, kiểu dữ liệu mảng là dãy số hữu hạn $(a_i)_{i=1}^n = (a_1, a_2, \ldots, a_n)$. Về mặt Tin học, kiểu dữ liệu mảng được ký hiệu bởi `a[1..n]`.

Bài toán 1 ([Đức22], 141., pp. 140–141: Count digit – Đếm chữ số)**.** *Cho dãy số n số nguyên dương $A[1..n]$ & 1 chữ số k. Đếm số lần xuất hiện chữ số k trong dãy A đã cho. E.g., với dãy $A[] = (11, 12, 13, 14, 15)$, thì chữ số $k = 1$ xuất hiện 6 lần trong dãy A.*

Input. *Dòng 1 của đầu vào chứa số nguyên $T \in \mathbb{N}^\star$ cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm: (i) Dòng đầu chứa lần lượt $n, k \in \mathbb{N}$ là số phần tử trong dãy $A[]$ & chữ số k. (ii) Dòng 2 chứa n số nguyên cách nhau 1 dấu cách, mô tả các phần tử của dãy A.*

Output. *Ứng với mỗi bộ dữ liệu, in ra 1 dòng chứa kết quả của bài toán tương ứng với bộ dữ liệu đầu vào đó.*

Constraint. $1 \le T \le 100, 1 \le n \le 100, 0 \le k \le 9, 1 \le A[i] \le 1000, \forall i = 1, \ldots, n$.

- Input: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/input/count_digit.inp.
- Output: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/output/count_digit.out.
- Python: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/Python/count_digit.py.
- C++: ?

Bài toán 2 ([Đức22], 141., pp. 140–141: Count digit – Đếm chữ số)**.** *Cho dãy số nguyên $a[1], a[2], \ldots, a[n]$. Thực hiện nhiệm vụ: Chia dãy thành 2 phần trái & phải, trong đó phần trái gồm $\frac{n}{2}$ phần tử đầu tiên & phần phải gồm các phần tử còn lại. Tính tổng các phần tử của mỗi phần, cuối cùng tính & in ra tích 2 tổng tìm được.*

Input. *Dòng 1 của đầu vào chứa $t \in \mathbb{N}^\star$ cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm: (i) Dòng đầu chứa $n \in \mathbb{N}^\star$ cho biết số phần tử của dãy. (ii) Dòng 2 chứa n số nguyên cách nhau bởi dấu cách, là các phần tử của dãy.*

Output. *Ứng với mỗi bộ dữ liệu, in ra 1 dong chứa kết quả của bài toán tương ứng với bộ dữ liệu đầu vào đó.*

Constraint. $1 \le t \le 100, 1 \le n \le 100, 1 \le A[i] \le 100, \forall i = 1, \ldots, n$.

- Input: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/input/prod_left_right_sums.inp.
- Output: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/output/prod_left_right_sums.out.
- Python: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/Python/prod_left_right_sums.py.

```
t = int(input())
for _ in range(t):
    n = int(input())
    a = list(map(int, input().split()))
    lsum = rsum = 0
    for i in range(n//2):
        lsum += a[i]
    for i in range(n//2, n):
        rsum += a[i]
    print(lsum * rsum)
```

- C++: ?

1.4.1 Kỹ thuật mảng chỉ số cho kiểu dữ liệu mảng

A general idea. Giả sử có dãy số $\{a_n\}_{n=1}^{n}$ được lưu với mảng `a = a[0], a[1],...,a[n - 1]` với a_i = `a[i - 1]`, $\forall i \in [n]$. Giả sử có $m \in \mathbb{N}^\star$ mảng chỉ số $\{f_i\}_{i=1}^{m}$ mà mỗi mảng có số phần tử là 1 hàm của n, $f_i : [n] \to \mathbb{R}$, mà m mảng chỉ số này lại liên quan hay ràng buộc với nhau theo những cách nào đấy, biểu diễn được bằng công thức toán, e.g., $f_i(n) = F_i(f_1, f_2, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n)$. Tìm hiểu cấu trúc toán học, cấu trúc giải thuật, & tạo ra các ví dụ để minh họa ý tưởng tổng quát này.

Kỹ thuật *sliding window* cũng là 1 trường hợp riêng của ý tưởng này với $m = 2$, $f_1(i)$ = left index (chỉ số trái), $f_2(i)$ = right index (chỉ số phải) & ta thường lấy tổng $\sum_{\texttt{left\_index}}^{\texttt{right\_index}} a[i]$, tích $\prod_{\texttt{left\_index}}^{\texttt{right\_index}} a[i]$, hoặc 1 hàm nào đấy của các phần tử bị giới hạn bởi 2 chỉ số trái & phải này, e.g., $\sum_{\texttt{left\_index}}^{\texttt{right\_index}} f(a[i])$ or $F(a[\texttt{left\_index}], \ldots, a[\texttt{right\_index}])$.

Problem 1 (Techniques of additional arrays – Kỹ thuật mảng bổ sung, R+4)**.** *Establish the general & rigorous frameworks for the idea of using additional arrays to micro manage or to get insights of a given array in some higher levels.*

– Thiết lập khuôn khổ chung & chặt chẽ cho ý tưởng sử dụng các mảng bổ sung để quản lý vi mô hoặc để có được thông tin chi tiết về một mảng nhất định ở một số cấp độ cao hơn.

Chương 2

Introductory Problems – Các Bài Toán Mở Đầu

Problem 2 (CSES Problem Set/weird algorithm). *Consider an algorithm that takes as input a positive integer n. If n is even, the algorithm divides it by 2, & if n is odd, the algorithm multiplies it by 3 & adds 1. The algorithm repeats this, until $n = 1$. E.g., the sequence for $n = 3$ is as follows: $3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$. Simulate the execution of the algorithm for a given value of n.*

Input. *The only input line contains an integer $n \in \mathbb{N}^\star$.*

Output. *rint a line that contains all values of n during the algorithm.*

Constraints. $n \in [10^6]$.

Sample.

weird_algorithm.inp	weird_algorithm.out
3	3 10 5 16 8 4 2 1

Problem 3 (CSES Problem Set/missing number). *You are given all numbers in $[n]$ except one. Find the missing number.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$. The 2nd line contains $n-1$ numbers. Each number is distinct & between 1 & n (inclusive) .*

Output. *Print the missing number.*

Constraints. $n\overline{2, 2 \cdot 10^5}$.

Sample.

missing_number.inp	missing_number.out
5 2 3 1 5	4

Problem 4 (CSES Problem Set/repetitions). *You are given a DNA sequence: a string consisting of characters* `A`, `C`, `G`, `T`. *Find the longest repetition in the sequence. This is a maximum-length substring containing only 1 type of character.*

Input. *The only input line contains a string of $n \in \mathbb{N}^\star$ characters.*

Output. *Print 1 integer: the length of the longest repetition.*

Constraints. $n \in [10^6]$.

Sample.

repetition.inp	repetition.out
ATTCGGGA	3

Problem 5 (CSES Problem Set/increasing array). *You are given an array of $n \in \mathbb{N}^\star$ integers. You want to modify the array so that it is increasing, i.e., every element is at least as large as the previous element. On each move, you may increase the value of any element by 1. What is the minimum number of moves required?*

Input. *The 1st input line contains an integer $n \in \mathbb{N}^\star$: the size of the array. Then, the 2nd line contains n integers $x_1, x_2, \ldots, x_n \in \mathbb{N}^\star$: the contents of the array.*

Output. *Print the minimum number of moves.*

Constraints. $n \in [2 \cdot 10^5], x_i \in [10^9], \forall i \in [n]$.

Sample.

increasing_array.inp	increasing_array.out
5 3 2 5 1 7	5

Problem 6 (CSES Problem Set/permutations)**.** *A permutation of* $[n]$ *is called* beautiful *if there are no adjacent elements whose difference is* 1*. Given* n*, construct a beautiful permutation if such a permutation exists.*

Input. *The 1st input line contains an integers* $n \in \mathbb{N}^\star$*.*

Output. *Print a beautiful permutation of integers* $1, 2, \ldots, n$*. If there are several solutions, you may print any of them. If there are no solutions, print* `NO SOLUTION`*.*

Constraints. $n \in [10^6]$.

Sample.

permutation.inp	permutation.out
5	4 2 5 3 1
3	NO SOLUTION

Problem 7 (CSES Problem Set/number spiral)**.** *A* number spiral *is an infinite grid whose upper-left square has number* 1*. Here are the 1st* 5 *layers of the spiral*

1	2	9	10	25
4	3	8	11	24
5	6	7	12	23
16	15	14	13	22
17	18	19	20	21

Find out the number is row y *&* *column* x*.*

Input. *The 1st input line contains an integer* $t \in \mathbb{N}^\star$*: the number of tests. After this, there are* t *lines, each containing integers* $y, z \in \mathbb{N}^\star$*.*

Output. *For each test, print the number in row* y *&* *column* x*.*

Constraints. $t \in [10^5], x, y \in [10^9]$.

Sample.

number_spiral.inp	number_spiral.out
3 2 3 1 1 4 2	8 1 15

Problem 8 (CSES Problem Set/two knights)**.** *Count for* $k \in [n]$ *the number of ways* 2 *knights can be placed on a* $k \times k$ *chessboard so that they do not attack each other.*

Input. *The only input line contains an integer* $n \in \mathbb{N}^\star$*.*

Output. *Print* n *integers: the results.*

Constraints. $n \in [10^4]$.

Sample.

two_knight.inp	two_knight.out
8	0 6 28 96 252 550 1056 1848

Problem 9 (CSES Problem Set/two sets). *Divide the numbers* $[n]$ *into 2 sets of equal sum.*

Input. *The only input line contains an integer* $n \in \mathbb{N}^\star$.

Output. *Print* `YES`*, if the division is possible, &* `NO` *otherwise. After this, if the division is possible, print an example of how to create the sets. 1st, print the number of elements in the 1st set followed by the elements themselves in a separate line, & then, print the 2nd set in a similar way.*

Constraints. $n \in [10^6]$.

Sample.

two_set.inp	two_set.out
7	YES 4 1 2 4 7 3 3 5 6
6	NO

Problem 10 (CSES Problem Set/bit strings). *Calculate the number of bit strings of length* $n \in \mathbb{N}^\star$*, e.g., if* $n = 3$*, the correct answer is* 8*, because the possible bit strings are* $000, 001, 010, 011, 100, 101, 110, 111$.

Input. *The only input line has an integer* $n \in \mathbb{N}^\star$.

Output. *Print the result modulo* $10^9 + 7$.

Constraints. $n \in [10^6]$.

Sample.

bit_string.inp	bit_string.out
3	8

Solution. The number of bit strings of length $n \in \mathbb{N}^\star$ is 2^n. □

Problem 11 (CSES Problem Set/trailing zeros). *Calculate the number of trailing zeros in the factorial* $n!$*, e.g.,* $20! = 2432902008176640$*& it has* 4 *trailing zeros.*

Input. *The only input line has an integer* $n \in \mathbb{N}^\star$.

Output. *Print the number of trailing zeros is* $n!$.

Constraints. $n \in [10^6]$.

Sample.

trailing_zero.inp	trailing_zero.out
20	4

Problem 12 (CSES Problem Set/coin piles). *You have* 2 *coin piles containing* $a, b \in \mathbb{N}$ *coins. On each move, you can either remove* 1 *coin from the left pile &* 2 *coins from the right pile, or* 2 *coins from the left pile &* 1 *coin from the right pile. Efficiently find out if you can empty both the piles.*

Input. *The 1st input line has an integer* t *integers* $n, m \in \mathbb{N}^\star$*: the number of tests. After this, there are* t *lines, each of which has* 2 *integers* $a, b \in \mathbb{N}$*: the numbers of coins in the piles.*

Output. *For each test, print* `YES` *if you can empty the piles &* `NO` *otherwise.*

Constraints. $t \in [10^5], a, b \in \overline{0, 10^9}$.

Sample.

coin_pile.inp	coin_pile.out
3 2 1 2 2 3 3	YES NO YES

Problem 13 (??CSES Problem Set/palindrome reorder)**.** *Given a string, recorder its letters in such a way that it becomes a palindrome (i.e., it reads the same forwards & backwards).*

Input. *The only input line has a string of length $n \in \mathbb{N}^\star$ consisting of characters* `A-Z`.

Output. *Print a palindrome consisting of the characters of the original string. You may print any valid solution. If there are no solutions, print* `NO SOLUTION`.

Constraints. $n \in [10^6]$.

Sample.

palindrome_reorder.inp	palindrome_reorder.out
AAAACACBA	AACABACAA

Problem 14 (CSES Problem Set/gray code)**.** *A* Gray code *(not Gay code) is a list of all 2^n bit strings of length $n \in \mathbb{N}^\star$, where any 2 successive strings differ in exactly 1 bit (i.e., their Hamming distance is 1). Create a Gray code for a given length n.*

Input. *The only input line has an integer $n \in \mathbb{N}^\star$.*

Output. *Print 2^n lines that describe the Gray code. You can print any valid solution.*

Constraints. $n \in [16]$.

Sample.

gray_code.inp	gray_code.out
2	00 01 11 10

Problem 15 (CSES Problem Set/tower of Hanoi)**.** *The Tower of Hanoi game consists of 3 stacks (left, middle, & right) & $n \in \mathbb{N}^\star$ round disks of different sizes. Initially, the left stack has all the disks, in increasing order of size from top to bottom. The goal is to move all the disks to the right stack using the middle stack. On each move you can move the uppermost disk from a stack to another stack. In addition, it is not allowed to place a larger disk on a smaller disk. Find a solution that minimizes the number of moves.*

Input. *The only input line has an integer $n \in \mathbb{N}^\star$: the number of of disks.*

Output. *1st print an integer $k \in \mathbb{N}^\star$: the minimum number of moves. After this, print k lines that describe the moves. Each line has 2 integers $a, b \in [3]$: you move a disk from stack a to stack b.*

Constraints. $n \in [16]$.

Sample.

tower_Hanoi.inp	tower_Hanoi.out
2	3 1 2 1 3 2 3

Problem 16 (CSES Problem Set/creating strings)**.** *Given a string, generate all different strings that can be created using its characters.*

Input. *The only input line has a string of length $n \in \mathbb{N}^\star$. Each character is between* `a-z`.

Output. *1st print an integer k: the number of strings. Then print k lines: the strings in alphabetical order.*

Constraints. $n \in [8]$.

Sample.

creating_string.inp	creating_string.out
aabac	20 aaabc aaacb aabac aabca aacab aacba abaac abaca abcaa acaab acaba acbaa baaac baaca bacaa bcaaa caaab caaba cabaa cbaaa

Problem 17 (CSES Problem Set/apple division)**.** *There are $n \in \mathbb{N}^\star$ apples with known weights. Divide the apples into 2 groups so that the difference between the weights of the groups is minimal.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of apples. The next line has n integers $p_1, p_2, \ldots, p_n$: the weight of each apple.*

Output. *Print 1 integer: the minimum difference between the weights of the groups.*

Constraints. $n \in [20], p_i \in [10^9]$.

Sample.

apple_division.inp	apple_division.out
5 3 2 7 4 1	1

Explanation. *Group 1 has weights $2, 3, 4$ (total weight 9), & group 2 has weights $1, 7$ (total weight 8).*

Problem 18 (CSES Problem Set/chessboard & queens)**.** *Place 8 queens on a chessboard so that no 2 queens are attacking each other. As an additional challenge, each square is either free or reserved, & you can only place queens on the free squares. However, the reserved squares do not prevent queens from attacking each other. Count the number of possible ways are there to place the queens.*

Input. *The input line has 8 lines, & each of them has 8 characters. Each square is either free . or reversed $*$.*

Output. *Print 1 integer: the number of ways you can place the queens.*

Constraints. $n \in [10^5], m \in [100], x_i \in \{0, 1, \ldots, m\}$.

Sample.

chessboard_queen.inp	chessboard_queen.out
....*..... ....**. ...*....	65

Problem 19 (CSES Problem Set/Raab game I). *Consider a* 2 *play game where each player has* $n \in \mathbb{N}^\star$ *cards numbered* $1, 2, \ldots, n$. *On each turn both players place* 1 *of their cards on the table. The player who placed the higher card gets* 1 *point. If the cards are equal, neither player gets a point. The game continues until all cards have been played. You are given the number of cards* n *& the players's scores at the end of the game,* $a, b \in \mathbb{N}$. *Give an example of how the game could have played out.*

Input. *The 1st input line contains* 1 *integer* $t \in \mathbb{N}^\star$*: the number of tests. Then there are* t *lines, each with* 3 *integers* $n, a, b \in \mathbb{N}^\star$.

Output. *For each test case print* `YES` *if there is a game with the given outcome &* `NO` *otherwise. If the answer is* `YES`*, print an example of* 1 *possible game. Print* 2 *line representing the order in which the players place their cards. You can give any valid example.*

Constraints. $t \in [10^3], n \in [100], a, b, \in \overline{0, n}$.

Sample.

Raab_game I.inp	Raab_game I.out
5 4 1 2 2 0 1 3 0 0 2 1 1 4 4 1	YES 1 4 3 2 2 1 3 4 NO YES 1 2 3 1 2 3 YES 1 2 2 1 NO

Problem 20 (CSES Problem Set/mex grid construction). *Construct an* $n \times n$ *grid where each square has the smallest nonnegative integer that does not appear to the left on the same row or above on the same column.*

Input. *The only input line has an integer* $n \in \mathbb{N}^\star$.

Output. *Print the grid according to the example.*

Constraints. $n \in [100]$.

Sample.

mex_grid_construction.inp	mex_grid_construction.out
5	 0 1 2 3 4 1 0 3 2 5 2 3 0 1 6 3 2 1 0 7 4 5 6 7 0

Problem 21 (CSES Problem Set/knight moves grid). *There is a knight on a* $n \times n$ *chessboard. For each square, print the minimum number of moves the knight needs to do to reach the top-left corner.*

Input. *The only input line has an integer* $n \in \mathbb{N}^\star$.

Output. *Print the minimum number of moves for each square.*

Constraints. $n \in \overline{4, 10^3}$.

Sample.

knight_move_grid.inp	knight_move_grid.out
8	0 3 2 3 2 3 4 5 3 4 1 2 3 4 3 4 2 1 4 3 2 3 4 5 3 2 3 2 3 4 3 4 2 3 2 3 4 3 4 5 3 4 3 4 3 4 5 4 4 3 4 3 4 5 4 5 5 4 5 4 5 4 5 6

Problem 22 (CSES Problem Set/grid coloring I). *You are given an $n \times m$ grid where each cell contains 1 character* `A, B, C` *or* `D`*. For each cell, you must change the character to* `A, B, C` *or* `D`*. The new character must be different from the old one. Change the characters in every cell s.t. no 2 adjacent cells have the same character.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of rows & columns. The next n lines each have m characters: the description of the grid.*

Output. *Print n lines each with m characters: the description of the final grid. You may print any valid solution. If no solution exists, just print* `IMPOSSIBLE`*.*

Constraints. $m, n \in [500]$.

Sample.

grid_coloring_I.inp	grid_coloring_I.out
3 4 AAAA BBBB CCDD	CDCD DCDC ABAB

Problem 23 (CSES Problem Set/digit queries). *Consider an infinite string that consists of all positive integers in increasing order:* 12345678910111213141516171819202122232425 . . . . *Process $q \in \mathbb{N}^\star$ queries of the form: what is the digit at position $k \in \mathbb{N}^\star$ in the string?*

Input. *The 1st input line has an integer $q \in \mathbb{N}^\star$: the number of queries. After this, there are q lines that describe the queries. Each line has an integer k: a 1-indexed position in the string.*

Output. *For each query, print the corresponding digit.*

Constraints. $q \in [10^3], k \in [10^{18}]$.

Sample.

digit_query.inp	digit_query.out
3 7 19 12	7 4 1

Problem 24 (CSES Problem Set/string reorder). *Reorder the characters of a string so that no 2 adjacent characters are the same. What is the lexicographically minimal such string?*

Input. *The only input line has a string of length $n \in \mathbb{N}^\star$ consisting of characters* `A-Z`*.*

Output. *Print the lexicographically minimal reordered string where no 2 adjacent characters are the same. If it is not possible to create such a string, print* `-1`*.*

Constraints. $n \in [10^6]$.

Sample.

string_reorder.inp	string_reorder.out
HATTIVATTI	AHATITITVT

Problem 25 (CSES Problem Set/grid path description)**.** *There are* 88418 *paths in a* 7×7 *grid from the upper-left square to the lower-left square. Each path corresponds to a* 48*-character description consisting of characters* `D, U, L, R` *(down, up, left, right, resp.). You are given a description of a path which may also contains characters* `?` *(any direction). Calculate the number of paths that match the description.*

Input. *The only input line has a* 48*-character string of characters* `?, D, U, L, R`*.*

Output. *Print* 1 *integer: the total number of paths.*

Sample.

grid_path_description.inp	grid_path_description.out
??????R??????U??????????????????????????LD????D?	201

Chương 3

Array & Sequence: Mảng & Dãy

Problem 26 (IMO2007P1)**.** *Real numbers $a_1, a_2, \ldots, a_n$ are given. For each $i \in [n]$ define*

$$d_i \coloneqq \max_{j\in[i]} - \min_{j\in\overline{i,n}} a_j = \max_{1\le j\le i} a_j - \min_{i\le j\le n} a_j,\ d = \max_{i\in[n]} d_i.$$

(a) Prove that, for any real numbers $x_1 \le x_2 \le \cdots \le x_n$,

$$\max_{i\in[n]} |x_i - a_i| \ge \frac{d}{2}.$$

(b) Show that there are real numbers $x_1 \le x_2 \le \cdots \le x_n$ s.t. equality holds.

Bài toán 3 ([VL24], 1., p. 10, IMO2007P1)**.** *Cho trước n số thực $a_1, a_2, \ldots, a_n$. Với mỗi $i \in [n]$, đặt*

$$d_i \coloneqq \max_{j\in[i]} - \min_{j\in\overline{i,n}} a_j = \max_{1\le j\le i} a_j - \min_{i\le j\le n} a_j,\ d = \max_{i\in[n]} d_i.$$

(a) Chứng minh: với n số thực $x_1 \le x_2 \le \cdots \le x_n$ tùy ý,

$$\max_{i\in[n]} |x_i - a_i| \ge \frac{d}{2}.$$

(b) Chỉ ra tồn tại n số thực $x_1 \le x_2 \le \cdots \le x_n$ sao cho bất đẳng thức cuối trở thành đẳng thức.

Problem 27. *Let $n \in \mathbb{N}^\star$ & let $a_1, \ldots, a_k \in [n]$, $k \ge 2$ s.t. $n | a_i(a_{i+1} - 1)$, $\forall i \in [k-1]$. Prove that $n \nmid a_k(a_1 - 1)$.*

Bài toán 4 ([VL24], 1., p. 14, IMO2009P1)**.** *Cho $n \in \mathbb{N}^\star$, xét $a_1, \ldots, a_k \in [n]$ là $k \in \mathbb{N}, k \ge 2$ số nguyên khác nhau sao cho $a_i(a_{i+1} - 1) \vdots n$, $\forall i \in [k-1]$. Chứng minh $a_k(a_1 - 1) \not\vdots n$.*

Bài toán 5 (CP version of IMO2009P1)**.** *Cho $n \in \mathbb{N}^\star$ được nhập vào. Đếm số tất cả các trường hợp có thể của dãy số $a_1, \ldots, a_k \in [n]$ là $k \in \mathbb{N}, k \ge 2$ số nguyên khác nhau sao cho $a_i(a_{i+1} - 1) \vdots n$, $\forall i \in [k-1]$, & xét tính chia hết của $a_k(a_1 - 1)$ cho n. (a) Sử dụng duyệt vét cạn thông thường. (b) Sử dụng phân tích thừa số nguyên tố.*

Problem 28 (IMO2009P3)**.** *Suppose that $\{s_n\}_{n=1}^\infty$ is a strictly increasing sequence of positive integers s.t. the subsequences $\{s_{s_n}\}_{n=1}^\infty, \{s_{s_n+1}\}_{n=1}^\infty$ are both arithmetic progressions. Prove that the sequence $\{s_n\}_{n=1}^\infty$ is itself an arithmetic progression.*

Bài toán 6 ([VL24], 1., p. 14, IMO2009P3)**.** *Cho biết $\{s_n\}_{n=1}^\infty$ là dãy tăng thực sự gồm các số nguyên dương sao cho 2 đãy con $\{s_{s_n}\}_{n=1}^\infty, \{s_{s_n+1}\}_{n=1}^\infty$ đều tạo thành 2 cấp số cộng. Chứng minh dãy $\{s_n\}_{n=1}^\infty$ cũng là cấp số cộng.*

Bài toán 7 (CP version of IMO2009P3)**.** *(a) Minh họa cho bài toán IMO2009P3, e.g., khởi tạo $\{s_n\}_{n=1}^\infty = \{n\}_{n=1}^\infty$, rồi điều chỉnh bằng cách tăng giảm các số hạng s_n để 2 đãy con $\{s_{s_n}\}_{n=1}^\infty, \{s_{s_n+1}\}_{n=1}^\infty$ đều tạo thành 2 cấp số cộng, sau đó kiểm tra dãy ban đầu có phải là cấp số cộng không. (b) Bài toán có mở rộng ra cho cấp số nhân được không?*

Problem 29 (IMO2010P6)**.** *Let $\{a_n\}_{n=1}^\infty \subset (0, \infty)$ be a sequence of positive real numbers. Suppose that for some positive integer s, we have $a_n = \max\{a_i + a_{n-i}; 1 \le i \le n-1\}$, $\forall n > s$. Prove that there exist $l \in [s], N \in \mathbb{N}^\star$ s.t. $a_n = a_l + a_{n-l}$, $\forall n \ge N$.*

Bài toán 8 ([VL24], 6., p. 16, IMO2010P6)**.** *Giả sử $\{a_n\}_{n=1}^\infty \subset (0, \infty)$ là 1 dãy số thực dương. Giả sử với số nguyên dương s cố định nào đó, ta có $a_n = \max\{a_i + a_{n-i}; 1 \le i \le n-1\}$, $\forall n > s$. Chứng minh tồn tại $l \in [s], N \in \mathbb{N}^\star$ sao cho $a_n = a_l + a_{n-l}$, $\forall n \ge N$.*

Bài toán 9 (CP version of IMO2010P6)**.** *Viết thuật toán & chương trình* C/C++, Pascal, Python *để minh hoạ IMO2010P6.*

Chương 4

Sorting & Searching – Sắp Xếp & Tìm Kiếm

Problem 30 (CSES Problem Set/distinct numbers)**.** *You are given a list of $n \in \mathbb{N}^\star$ integers. Calculate the number of* distinct *values in the list.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of values. The 2nd line has n integers $x_1, x_2, \ldots, x_n \in \mathbb{Z}$.*

Output. *Print 1 integer: the*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x_i \in [10^9]$.

Sample.

distinct_number.inp	distinct_number.out
5 2 3 2 2 3	2

Problem 31 (CSES Problem Set/apartments)**.** *There are $n \in \mathbb{N}^\star$ applicants & $m \in \mathbb{N}^\star$ free apartments. Distribute the apartments so that as many applicants as possible will get an apartment. Each applicant has a desired apartment size, & they will accept any apartment whose size is close enough to the desired size.*

Input. *The 1st input line has 3 integers $n, m, k \in \mathbb{N}^\star$: the number of applicants, the number of apartments, & the maximum allowed difference. The next line contains n integers $a_1, a_2, \ldots, a_n$: the desired apartment size of each applicant. If the desired size of an applicant is x, they will accept any apartment whose size is between $x - k$ & $x + k$. The last line contains m integers $b_1, b_2, \ldots, b_m$: the size of each apartment.*

Output. *Print 1 integer: the number of applicants who will get an apartment.*

Constraints. $m, n \in [2 \cdot 10^5], k \in \overline{0, 10^9}, a_i, b_j \in [10^9]$, $\forall i \in [n]$, $\forall j \in [m]$.

Sample.

apartment.inp	apartment.out
4 3 5 60 45 80 60 30 60 75	2

Problem 32 (CSES Problem Set/Ferris wheel)**.** *There are $n \in \mathbb{N}^\star$ children who want to go to a Ferris wheel. Find a gondola for each child. Each gondola may have 1 or 2 children on it, & in addition, the total weight in a gondola may not exceed x. You know the weight of every child. What is the minimum number of gondolas needed for the children?*

Input. *The 1st input line contains 2 integers $n, x \in \mathbb{N}^\star$: the number of children & the maximum allowed weight. The next line contains n integers $p_1, p_2, \ldots, p_n$: the weight of each child.*

Output. *Print 1 integer: the minimum number of gondolas.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x \in [10^9], p_i \in [x]$, $\forall i \in [n]$.

Sample.

Ferris_wheel.inp	Ferris_wheel.out
4 10 7 2 3 9	3

Problem 33 (CSES Problem Set/concert tickets). *There are $n \in \mathbb{N}^\star$ concert tickets available, each with a certain price. Then, $m \in \mathbb{N}^\star$ customers arrive, one after another. Each customer announces the maximum price they are willing to pay for a ticket, & after this, they will get a ticket with the nearest possible price s.t. it does not exceed the maximum price.*

Input. *The 1st input line contains 2 integers $n, m \in \mathbb{N}^\star$: the number of tickets & the number of customers. The next line contains n integers $h_1, h_2, \ldots, h_n$: the price of each ticket. The last line contains m integers $t_1, t_2, \ldots, t_m$: the maximum price for each customer in the order they arrive.*

Output. *Print, for each customer, the price that they will pay for their ticket. After this, the ticket cannot be purchased again. If a customer cannot get any ticket, print* `-1`.

Constraints. $m, n \in [2 \cdot 10^5], h_i, t_j \in [10^9]$, $\forall i \in [n]$, $\forall j \in [m]$.

Sample.

concert_ticket.inp	concert_ticket.out
5 3 5 3 7 8 5 4 8 3	3 8 −1

Problem 34 (CSES Problem Set/restaurant customers). *You are given the arrival & leaving times of $n \in \mathbb{N}^\star$ customers in a restaurant. What was the maximum number of customers in the restaurant at any time?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of customers. After this, there are n lines that describe the customers. Each line has 2 integers $a, b \in \mathbb{N}^\star$: the arrival & leaving times of a customer. You may assume that all arrival & leaving times are distinct.*

Output. *Print 1 integer: the maximum number of customers.*

Constraints. $n \in [2 \cdot 10^5], 1 \le a < b \le 10^9$.

Sample.

restaurant_customer.inp	restaurant_customer.out
3 5 8 2 4 3 9	2

Problem 35 (CSES Problem Set/movie festival). *In a movie festival $n \in \mathbb{N}^\star$ movies will be shown. You know the starting & ending time of each movie. What is the maximum number of movies you can watch entirely?*

Input. *The 1st input line has an integer $n, m \in \mathbb{N}^\star$: the number of movies. After this, there are n lines that describe the movies. Each line has 2 integers $a, b \in \mathbb{N}$: the starting & ending times of a movie.*

Output. *Print 1 integer: the maximum number of movies.*

Constraints. $n \in [2 \cdot 10^5], 1 \le a < b \le 10^9$.

Sample.

movie_festival.inp	movie_festival.out
3 3 5 4 9 5 8	2

Problem 36 (CSES Problem Set/sum of 2 values). *You are given an array of $n \in \mathbb{N}^\star$ integers. Find 2 values (at distinct positions) whose sum is x.*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the array size & the target sum. The 2nd line has n integers $a_1, a_2, \ldots, a_n$: the array values.*

Output. *Print 2 integers: the positions of the values. If there are several solutions, you may print any of them. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in [2 \cdot 10^5], x, a_i \in [10^9]$, $\forall i \in [n]$.

Sample.

sum_2_value.inp	sum_2_value.out
4 8 2 7 5 1	2 4

Problem 37 (CSES Problem Set/maximum subarray sum). *Given an array of $n \in \mathbb{N}^\star$ integers. Find the maximum sum of values in a contiguous, nomepty subarray.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the size of the array. The 2nd line has n integers $x_1, x_2, \ldots, x_n \in \mathbb{Z}$: the array values.*

Output. *Print 1 integer: the maximum subarray sum.*

Constraints. $n \in [2 \cdot 10^5], x_i \in \overline{-10^9, 10^9}$.

Sample.

max_subarray_sum.inp	max_subarray_sum.out
8 -1 3 -2 5 3 -5 2 2	9

Problem 38 (CSES Problem Set/stick lengths). *There are $n \in \mathbb{N}^\star$ sticks with some lengths. Modify the sticks so that each stick has the same length. You can either lengthen & shorten each stick. Both operations cost x where x is the difference between the new & original length. What is the minimum total cost?*

Input. *The 1st input line contains an integer $n \in \mathbb{N}^\star$: the number of sticks. Then there are n integers: $p_1, p_2, \ldots, p_n$: the lengths of the sticks.*

Output. *Print 1 integer: the minimum total cost.*

Constraints. $n \in [2 \cdot 10^5], p_i \in [10^9]$, $\forall i \in [n]$.

Sample.

stick_length.inp	stick_length.out
5 2 3 1 5 2	6

Problem 39 (CSES Problem Set/missing coin sum). *You have $n \in \mathbb{N}^\star$ coins with positive integer values. What is the smallest sum you cannot create using a subset of the coins?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of coins. the 2nd line has n integers $x_1, x_2, \ldots, x_n$: the value of each coin.*

Output. *Print 1 integer: the smallest coin sum.*

Constraints. $n \in [2 \cdot 10^5], , x_i \in [10^9]$, $\forall i \in [n]$.

Sample.

missing_coin_sum.inp	missing_coin_sum.out
5 2 9 1 2 7	6

Problem 40 (CSES Problem Set/collecting numbers). *You are given an array that contains each number $\in [n]$ exactly once. Collect the numbers from 1 to n in increasing order. On each round, you go through the array from left to right & collect as many numbers as possible. What will be the total number of rounds?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the array size. The next line has n integers $x_1, x_2, \ldots, x_n$: the numbers in the array.*

Output. *Print 1 integer: the number of rounds.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x_i \in \{0, 1, \ldots, m\}$.

Sample.

collecting_number.inp	collecting_number.out
5 4 2 1 5 3	3

Problem 41 (CSES Problem Set/collecting numbers II). *You are given an array that contains each number $\in [n]$ exactly once. Collect the numbers from 1 to n in increasing order. On each round, you go through the array from left to right & collect as many numbers as possible. Giving $m \in \mathbb{N}^\star$ operations that swap 2 numbers in the array, report the number of rounds after each operation.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the array size & the number of operations. The next line has n integers $x_1, x_2, \ldots, x_n$: the numbers in the array. Finally, there are m lines that describe the operations. Each line has 2 integers $a, b \in \mathbb{N}^\star$: the numbers at positions a, b are swapped.*

Output. *Print m integers: the number of rounds after each swap.*

Constraints. $m, n \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

collecting_number_II.inp	collecting_number_II.out
5 3 4 2 1 5 3 2 3 1 5 2 3	2 3 4

Problem 42 (CSES Problem Set/playlist). *You are given a playlist of a radio station since its establishment. The playlist has a total of $n \in \mathbb{N}^\star$ songs. What is the longest sequence of successive songs where each song is unique?*

Input. *The 1st input line contains an integer $n \in \mathbb{N}^\star$: the number of songs. The next line has n integers $k_1, k_2, \ldots, k_n$: the id number of each song.*

Output. *Print the length of the longest sequence of unique songs.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], k_i \in [10^9], \forall i \in [n]$.

Sample.

playlist.inp	playlist.out
8 1 2 1 3 2 7 4 2	5

Problem 43 (CSES Problem Set/towers). *You are given $n \in \mathbb{N}^\star$ cubes in a certain order. Build towers using them. Whenever 2 cubes are 1 on top of the other, the upper cube must be smaller than the lower cube. You must process the cubes in the given order. You can always either place the cube on top of an existing tower, or begin a new tower. What is the minimum possible number of towers?*

Input. *The 1st input line contains an integer $n \in \mathbb{N}^\star$: the number of cubes. The next line contains n integers $k_1, k_2, \ldots, k_n$: the sizes of the cubes.*

Output. *Print 1 integer: the minimum number of towers.*

Constraints. $n \in [2 \cdot 10^5], k_i \in [10^9]$.

Sample.

tower.inp	tower.out
5 3 8 2 1 5	2

Problem 44 (CSES Problem Set/traffic lights). *There is a street of length x whose positions are numbered $0, 1, \ldots, x$. Initially there are no traffic lights, but $n \in \mathbb{N}^\star$ sets of traffic lights are added to the street one after another. Calculate the length of the longest passage without traffic lights after each addition.*

Input. *The 1st input line has 2 integers $x, n \in \mathbb{N}^\star$: the length of the street & the number of sets of traffic lights. Then, the next line contains n integers $p_1, p_2, \ldots, p_n$: the position of each set of traffic lights. Each position is distinct.*

Output. *Print the length of the longest passage without traffic lights after each addition.*

Constraints. $x \in [10^9], n \in [2 \cdot 10^5], p_i \in [x-1]$, $\forall i \in [n]$.

Sample.

traffic_light.inp	traffic_light.out
8 3 3 6 2	5 3 3

Problem 45 (CSES Problem Set/distinct values subarrays)**.** *Given an array of $n \in \mathbb{N}^\star$ integers, count the number of subarrays where each element is dictinct.*

Input. *The 1st input line has an integer $m \in \mathbb{N}^\star$: the array size. The 2nd line has n integers $x_1, x_2, \ldots, x_n$: the array contents.*

Output. *Print the number of subarrays with distinct elements.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x_i \in [10^9]$, $\forall i \in [n]$.

Sample.

distinct_values_subarray.inp	distinct_values_subarray.out
4 1 2 1 3	8

Explanation. *The subarrays are* $[1]$ *(2 times),* $[2], [3], [1,2], [1,3], [2,1], [2,1,3]$.

Problem 46 (CSES Problem Set/distinct values subsequences)**.** *Given an array of $n \in \mathbb{N}^\star$ integers, count the number of subsequences where each element is distinct.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the array size. The 2nd line has n integers $x_1, x_2, \ldots, x_n$: the array contents.*

Output. *Print the number of subsequences with distinct elements. The answer can be large, so print it modulo $10^9 + 7$.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x_i \in [10^9]$, $\forall i \in [n]$.

Sample.

distinct_value_subsequence.inp	distinct_value_subsequence.out
4 1 2 1 3	11

Explanation. *The subsequences are* $[1]$ *(2 times),* $[2], [3], [1,2], [1,3]$ *(2 times),* $[2,1], [2,3], [1,2,3], [2,1,3]$.

Problem 47 (CSES Problem Set/Josephus problem I)**.** *Consider a game where there are $n \in \mathbb{N}^\star$ children (numbered $1, 2, \ldots, n$) in a circle. During the game, every other child is removed from the circle until there are no children left. In which order will the children be removed?*

Input. *The only input line has an integer $n \in \mathbb{N}^\star$.*

Output. *Print n integers: the removal order.*

Constraints. $n \in [2 \cdot 10^5]$.

Sample.

Josephus_problem_I.inp	Josephus_problem_I.out
7	2 4 6 1 5 3 7

Problem 48 (CSES Problem Set/Josephus problem II)**.** *Consider a game where there are $n \in \mathbb{N}^\star$ children (numbered $1, 2, \ldots, n$) in a circle. During the game, repeatedly $k \in \mathbb{N}^\star$ children are skipped & 1 child is removed from the circle. In which order will the children be removed?*

Input. *The only input line has 2 integers $n, k \in \mathbb{N}^\star$.*

Output. *Print n integers: the removal order.*

Constraints. $n \in [2 \cdot 10^5] k \in \overline{0, 10^9}$.

Sample.

Josephus_problem_II.inp	Josephus_problem_II.out
7 2	3 6 2 7 5 1 4

Problem 49 (CSES Problem Set/nested ranges check). *Given $n \in \mathbb{N}^\star$ ranges, determine for each range if it contains some order range & if some other range contains it. Range $[a, b]$ contains range $[c, d]$ if $a \leq c$ & $d \leq b$.*

Input. *The 1st input line has an integer $n, m \in \mathbb{N}^\star$: the number of ranges. After this, there are n lines that describe the ranges. Each line has 2 integers $x, y \in \mathbb{Z}$: the range is $[x, y]$. You may assume that no range appears more than once in the input.*

Output. *1st print a line that describes for each range (in the input order) if it contains some other range* `1` *or not* `0`*. Then print a line that describes for each range (in the input order) if some other range contains it* `1` *or not* `0`*.*

Constraints. $n \in [2 \cdot 10^5], 1 \leq x < y \leq 10^9$.

Sample.

nested_range_check.inp	nested_range_check.out
4 1 6 2 4 4 8 3 6	 1 0 0 0 0 1 0 1

Problem 50 (CSES Problem Set/nested ranges count). *Given $n \in \mathbb{N}^\star$ ranges, count for each range how many other ranges it contains & how many other ranges contain it. Range $[a, b]$ contains range $[c, d]$ if $a \leq c$ & $d \leq b$.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of ranges. After this, there are n lines that describe the ranges. Each line has 2 integers $x, y \in \mathbb{Z}$: the range is $[x, y]$. You may assume that no range appears more than once in the input.*

Output. *1st print a line that describes for each range (in the input order) how many other ranges it contains. Then print a line that describes for each range (in the input order) how many other ranges contain it.*

Constraints. $n \in [2 \cdot 10^5], 1 \leq x < y \leq 10^9$.

Sample.

nested_ranges_count.inp	nested_ranges_count.out
4 1 6 2 4 4 8 3 6	2 0 0 0 0 1 0 1

Problem 51 (CSES Problem Set/room allocation). *There is a large hotel, & $n \in \mathbb{N}^\star$ customers will arrive soon. Each customer wants to have a single room. You know each customer's arrival & departure day. 2 customers can stay in the same room if the departure day of the 1st customer is earlier than the arrival day of the 2nd customer. What is the minimum number of rooms that are needed to accommodate all customers? & how can the rooms be allocated?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of of customers. Then there are n lines, each of which describes 1 customer. Each line has 2 integers $a, b \in \mathbb{N}^\star$: the arrival & departure day.*

Output. *1st print an integer k: the minimum number of rooms required. After that, print a line that contains the room number of each customer in the same order in the input. The rooms are numbered $1, 2, \ldots, k$. You can print any valid solution.*

Constraints. $n \in [2 \cdot 10^5], 1 \leq a \leq b \leq 10^9$.

Sample.

room_allocation.inp	room_allocation.out
3 1 2 2 4 4 4	2 1 2 1

Problem 52 (CSES Problem Set/factory machines). *A factory has $n \in \mathbb{N}^\star$ machines which can be used to make products. Make a total of $t \in \mathbb{N}^\star$ products. For each machine, you know the number of seconds it needs to make a single product. The machines can work simultaneously, & you can freely decide their schedule. What is the shortest time needed to make t products?*

Input. *The 1st input line has 2 integers $n, t \in \mathbb{N}^\star$: the number of machines & products. The next line has n integers $k_1, k_2, \ldots, k_n$: the time needed to make a product using each machine.*

Output. *Print 1 integer: the minimum time needed to make t products.*

Constraints. $n \in [2 \cdot 10^5], t, k_i \in [10^9]$, $\forall i \in [n]$.

Sample.

factory_machine.inp	factory_machine.out
3 7 3 2 5	8

Explanation. *Machine 1 makes 2 products, machine 2 makes 4 products & machine 3 makes 1 product.*

Problem 53 (CSES Problem Set/tasks & deadlines). *You have to process $n \in \mathbb{N}^\star$ tasks. Each task has a duration & a deadline, & you will process the tasks in some order one after another. Your reward for a task is $d - f$ where d is its deadline & f is your finishing time. (The starting time is 0, & you have to process all tasks even if a task would yield negative reward.) What is your maximum reward if you act optimally?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of tasks. After this, there are n lines that describe the tasks. Each line has 2 integers $a, d \in \mathbb{N}^\star$: the duration & deadline of the task.*

Output. *Print 1 integer: the maximum reward.*

Constraints. $n \in [2 \cdot 10^5], a, d \in [10^6]$.

Sample.

task_deadline.inp	task_deadline.out
3 6 10 8 15 5 12	2

Problem 54 (CSES Problem Set/reading books). *There are $n \in \mathbb{N}^\star$ books, &* Kotivalo & Justiina *are going to read them all. For each book, you know the time it takes to read it. They both read each book from beginning to end, & they cannot read a book at the same time. What is the minimum total time required?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of books. The 2nd line has n integers $t_1, t_2, \ldots, t_n$: the time required to read each book.*

Output. *Print 1 integer: the minimum total time.*

Constraints. $n \in [2 \cdot 10^5], t_i \in [10^9]$, $\forall i \in [n]$.

Sample.

reading_book.inp	reading_book.out
3 2 8 3	16

Problem 55 (CSES Problem Set/sum of 3 values). *You are given an array of $n \in \mathbb{N}^\star$ integers. Find 3 values (at distinct positions) whose sum is x.*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the array size & the target sum. The 2nd line has n integers $a_1, a_2, \ldots, a_n$: the array values.*

Output. *Print 3 integers: the positions of the values. If there are several solutions, you may print any of them. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in [5000], m \in [100], x, a_i \in [10^9]$, $\forall i \in [n]$.

Sample.

sum_3_value.inp	sum_3_value.out
4 8 2 7 5 1	1 3 4

Problem 56 (CSES Problem Set/sum of 4 values). *You are given an array of $n \in \mathbb{N}^\star$ integers. Find 4 values (at distinct positions) whose sum is x.*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the array size & the target sum. The 2nd line has n integers $a_1, a_2, \ldots, a_n$: the array values.*

Output. *Print 4 integer: the positions of the values. If there are several solutions, you may print any of them. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in [1000], x, a_i \in [10^9]$, $\forall i \in [n]$.

Sample.

sum_4_value.inp	sum_4_value.out
8 15 3 2 5 8 1 3 2 3	2 4 6 7

Problem 57 (CSES Problem Set/nearest smaller values). *Given an array of $n \in \mathbb{N}^\star$ integers. Find for each array position the nearest position to its left having a smaller value.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the size of the array. The 2nd line has n integers $x_1, x_2, \ldots, x_n$: the array values.*

Output. *Print n integer: for each array position the nearest position with a smaller value. If there is no such position, print 0.*

Constraints. $n \in [2 \cdot 10^5], x_i \in [10^9]$, $\forall i \in [n]$.

Sample.

nearest_smaller_values.inp	nearest_smaller_values.out
8 2 5 1 4 8 3 2 5	0 1 0 3 4 3 3 7

Problem 58 (CSES Problem Set/subarray sums I). *Given an array of $n \in \mathbb{N}^\star$ positive integers. Count the number of subarrays having sum x.*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the size of the array & the target sum x. The 2nd line has n integers $a_1, a_2, \ldots, a_n \in \mathbb{N}^\star$: the contents of the array.*

Output. *Print 1 integer: the required number of subarrays.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x, a_i \in [10^9]$, $\forall i \in [n]$.

Sample.

subarray_sums_I.inp	subarray_sums_I.out
5 7 2 4 1 2 7	3

Problem 59 (CSES Problem Set/subarray sums II). *Given an array of $n \in \mathbb{N}^\star$ integers. Count the number of subarrays having sum x.*

Input. *The 1st input line has 2 integers $n \in \mathbb{N}^\star, x \in \mathbb{Z}$: the size of the array & the target sum x. The 2nd line has n integers $a_1, a_2, \ldots, a_n \in \mathbb{Z}$: the contents of the array.*

Output. *Print 1 integer: the required number of subarrays.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x, a_i \in \overline{-10^9, 10^9}$, $\forall i \in [n]$.

Sample.

subarray_sums_II.inp	subarray_sums_II.out
5 7 2 -1 3 5 -2	2

Problem 60 (CSES Problem Set/subarray divisibility). *Given an array of $n \in \mathbb{N}^\star$ integers. Count the number of subarrays where the sum of values is divisible by n.*

Input. *The 1st input line has an integers $n \in \mathbb{N}^\star$: the size of the array. The 2nd line has n integers $a_1, a_2, \ldots, a_n \in \mathbb{Z}$: the contents of the array.*

Output. *Print 1 integer: the required number of subarrays.*

Constraints. $n \in [2 \cdot 10^5], a_i \in \overline{-10^9, 10^9}, \forall i \in [n]$.

Sample.

subarray_divisibility.inp	subarray_divisibility.out
5 3 1 2 7 4	1

Problem 61 (CSES Problem Set/distinct values subarrays II). *Given an array of $n \in \mathbb{N}^\star$ integers. Calculate the number of subarrays that have at most k distinct values.*

Input. *The 1st input line has 2 integers $n, k \in \mathbb{N}^\star$. The next line has n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*

Output. *Print 1 integer: the number of subarrays.*

Constraints. $1 \le k \le n \le 2 \cdot 10^5, x_i \in [10^9], \forall i \in [n]$.

Sample.

distinct_values_subarray_II.inp	distinct_values_subarray_II.out
5 2 1 2 3 1 1	10

Problem 62 (CSES Problem Set/array division). *You are given an array containing $n \in \mathbb{N}^\star$ positive integers. Divide the array into $k \in \mathbb{N}^\star$ subarrays so that the maximum sum in a subarray is as small as possible.*

Input. *The 1st input line has 2 integers $n, k \in \mathbb{N}^\star$: the size of the array & the number of subarrays in the division. The next line contains n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*

Output. *Print 1 integer: the maximum sum in a subarray in the optimal division.*

Constraints. $n \in [2 \cdot 10^5], k \in [n], x_i \in [10^9]$.

Sample.

array_division.inp	array_division.out
5 3 2 4 7 3 5	8

Explanation. *An optimal division is $[2, 4], [7], [3, 5]$ where the sums of the subarrays are $6, 7, 8$. The largest sum is the last sum 8.*

Problem 63 (CSES Problem Set/movie festival II). *In a movie festival, $n \in \mathbb{N}^\star$ movies will be shown. Syrjälä's movie club consists of $k \in \mathbb{N}^\star$ members, who will be all attending the festival. You know the starting & ending time of each movie. What is the maximum total number of movies the club members can watch entirely if they act optimally?*

Input. *The 1st input line has 2 integers $n, k \in \mathbb{N}^\star$: the number of movies & club members. After this, there are n lines that describe the movies. Each line has 2 integers $a, b \in \mathbb{B}$: the starting & ending time of a movie.*

Output. *Print 1 integer: the maximum total number of movies.*

Constraints. $1 \le k \le n \le 2 \cdot 10^5, 1 \le a < b \le 10^9$.

Sample.

movie_festival_II.inp	movie_festival_II.out
5 2 1 5 8 10 3 6 2 5 6 9	4

Problem 64 (CSES Problem Set/maximum subarray II)**.** *Given an array of $n \in \mathbb{N}^\star$ integers. Find the maximum sum of values in a contiguous subarray with length between a, b.*

Input. *The 1st input line has 3 integers $n, a, b \in \mathbb{N}^\star$: the size of the array & the minimum & maximum subarray length. The 2nd line has n integers $x_1, x_2, \ldots, x_n \in \mathbb{Z}$: the array values.*

Output. *Print 1 integer: the maximum subarray sum.*

Constraints. $n \in [2 \cdot 10^5], 1 \le a \le b \le n, x_i \in \overline{-10^9, 10^9}, \forall i \in [n]$.

Sample.

maximum_subarray_II.inp	maximum_subarray_II.out
8 1 2 -1 3 -2 5 3 -5 2 2	8

Chương 5

Practice for Simple Computing – Thực Hành Tính Toán Đơn Giản

Resources – Tài nguyên.

1. [WW16]. Yonghui Wu, Jiande Wang. *Data Structure Practice for Collegiate Programming Contests & Education.*

2. [WW18]. Yonghui Wu, Jiande Wang. *Algorithm Design Practice for Collegiate Programming Contests & Education.*

Problem 65 ([WW16], p. 4: financial management)**.** Larry *graduated this year & finally has a job. He's making a lot of money, but somehow never seems to have enough.* Larry *has decided that he needs to get a hold of his financial portfolio & solve his financial problems. The 1st step is to figure out what's been going on with his money.* Larry *has his bank account statements & wants to see how much money he has. Help* Larry *by writing a program to take his closing balance from each of the past 12 months & calculate his average account balance.*

Input. *The input will be 12 lines. Each line will contain the closing balance of his bank account for a particular month. Each number will be positive & displayed to the penny. No dollar sign will be included.*

Output. *The output will be a single number, the average (mean) of the closing balances for the 12 months. It will be rounded to the nearest penny, preceded immediately by a dollar sign, & followed by the end of the line. There will be no other spaces or characters in the output.*

Source. *ACM Mid-Atlantic United States 2001.*

IDs for online judges. *POJ 1004, ZOJ 1048, UVA 2362.*

Math Analysis. Let $\{a_i\}_{i=1}^{12} \subset [0, \infty)$ be monthly incomes of 12 months. Compute their average by the formula $\overline{a} = \frac{1}{12} \sum_{i=1}^{12} a_i$. This can be generalized to $n \in \mathbb{N}^\star$ months with a sequence of monthly incomes $\{a_i\}_{i=1}^{n} \subset [0, \infty)$ with its average value given by the formula $\overline{a} := \frac{1}{n} \sum_{i=1}^{n} a_i$.

CS Analysis. The income of 12 months `a[0..11]` is input by a `for` statement & the total income `sum` $:= \sum_{i=0}^{11} a[i]$ is calculated. Then the average monthly income `avg = sum/12` is calculated. Finally, `avg` is output in accordance with the problem's output format by utilizing `printf`'s format functionalities via `printf("$%.2f", avg)`.

- Input: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/input/financial_management.inp.
- Output: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/output/financial_management.out.
- C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/financial_management.cpp.

```
#include <iostream>
using namespace std;
int main() {
    double avg, sum = 0.0, a[12] = {0};
    for (int i = 0; i < 12; ++i) { // input income of 12 months a[0..11] & summation
        cin >> a[i];
        sum += a[i];
    }
    avg = sum/12; // compute average monthly
    printf("$%.2f", avg); // output average monthly
    return 0;
}
```

Remark 2 (array of 0s). *The technique* `a[12] = {0}` *initializes an array a with all zero elements.*

- Python: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/Python/financial_management.py.

```
sum = 0
for __ in range(12):
a = float(input())
sum += a
print("${:.2f}".format(sum / 12))
```

Bài toán 10 (Basic statistical data sample – mẫu dữ liệu thống kê cơ bản). *Cho 1 mẫu dữ liệu* $(a_i)_{i=1}^n$. *Tính trung bình, độ lệch chuẩn, phương sai của mẫu.*

Problem 66 ([WW16], pp. 5–6: doubles). *As part of an arithmetic competency program, your students will be given randomly generated lists of 2–15 unique positive integers & asked to determine how many items in each list are twice some other item in same list. You will need a program to help you with the grading. This program should be able to scan the lists & output the correct answer for each one. E.g., given the list* `1 4 3 2 9 7 18 22` *your program should answer* 3, *as* 2 *is twice* 1, 4 *is twice* 2, *&* 18 *is twice* 9.

Input. *The input file will consist of 1 or more lists of numbers. There will be 1 list of numbers per line. Each list will contain from 2–15 unique positive integers. No integer will be* > 99. *Each line will be terminated with the integer* 0, *which is not considered part of the list. A line with the single number* -1 *will mark the end of the file. Some lists may not contain any doubles.*

Output. *The output will consist of 1 line per input list, containing a count of the items that are double some other item.*

Source. *ACM Mid-Central United States 2003.*

IDs for online judges. *POJ 1552, ZOJ 1760, UVA 2787.*

Remark 3 (Multiple test cases – đa bộ test). *For any problem with multiple test cases, a loop is used to deal with multiple test cases. The loop enumerates every test case.*

– Đối với bất kỳ vấn đề nào có nhiều trường hợp thử nghiệm, một vòng lặp được sử dụng để xử lý nhiều trường hợp thử nghiệm. Vòng lặp liệt kê mọi trường hợp thử nghiệm.

- Input: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/input/double.inp.
- Output: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/output/double.out.
- C++:
  - https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/double.cpp.

```
#include <iostream>
using namespace std;
int main() {
 int i, j, n, count, a[20];
 cin >> a[0]; // input 1st element
 while (a[0] != -1) { // if it is not end of input, input a new test case
  n = 1;
  for ( ; ; ++n) {
   cin >> a[n];
   if (a[n] == 0) break;
  }
  count = 0; // determine how many items in each list are twice some other item
  for (i = 0; i < n - 1; ++i) { // enumerate all pairs
   for (j = i + 1; j < n; ++j) {
    if (a[i]*2 == a[j] || a[j]*2 == a[i]) // accumulation
    ++count;
   }
  }
  cout << count << endl; // output result
  cin >> a[0]; // input 1st element of next test case
 }
```

```
    return 0;
}
```

- https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/double_DPAK.cpp: use map & vector.

Bài toán có thể mở rộng từ double thành triple, multiple, or just equal.

Problem 67 ([WW16], pp. 7–8: sum of consecutive prime numbers)**.** *Some positive integers can be represented by a sum of 1 or more consecutive prime numbers. How many such representations does a give positive integer have? E.g., the integer* 53 *has 2 representations* $5+7+11+13+17$ *&* 53*. The integer* 41 *has 3 representations:* $2+3+5+7+11+13, 11+13+17$*, &* 41*. The integer* 3 *has only 1 representation, which is* 3*. The integer* 20 *has no such representations. Note: summands must be consecutive prime numbers, so neither* $7+13$ *nor* $3+5+5+7$ *is a valid representation for the integer* 20*. Your mission is to write a program that reports the number of representations for the given positive integer.*

Input. *The input is a sequence of positive integers, each in a separate line. The integers are between* 2 *&* 10000*, inclusive. The end of the input is indicated by a zero.*

Output. *The output should be composed of lines each corresponding to an input line, except the last zero. An output line includes the number of representations for the input integer as the sum of 1 or more consecutive prime numbers. No other characters should be inserted in the output.*

Source. *ACM Japan 2005.*

IDs for online judges. *POJ 2739, UVA 3399.*

Problem 68 ([WW16], pp. 9–10: I think I need a houseboat)**.** Fred Mapper *is considering purchasing some land in Louisiana to build his house on. In the process of investigating the land, he learned that the state of Louisiana is actually shrinking by* 50 *square miles each year, due to erosion caused by the Mississippi River. Since* Fred *is hoping to live in this house for the rest of his life, he needs to know if his land is going to be lost to erosion.*

After doing more research, Fred *has learned that the land that is being lost forms a semicircle. This semicircle is part of a circle centered at* $(0,0)$*, with the line that bisects the circle being the* x *axis. Locations below the* x *axis are in the water. The semicircle has an area of* 0 *at the beginning of year* 1*.*

Input. *The 1st line of input will be a positive integer indicating how many data sets will be included* N*. Each of the next* N *lines will contain the* x, y *Cartesian coordinates of the land* Fred *is considering. These will be floating-point numbers measured in miles. The* y *coordinate will be nonnegative.* $(0,0)$ *will not be given.*

Output. *For each data set, a single line of output should appear. This line should take the form of*

Property N*: This property will begin eroding in year* Z*.*

where N *is the data set (counting from* 1*) &* Z *is the 1st year (start from* 1*) this property will be within the semicircle AT THE END OF YEAR* Z*.* Z *must be an integer. After the last data set, this should print out "END OF OUTPUT."*

Source. *ACM Mid-Atlantic United States 2001.*

Note. *No property will appear exactly on the semicircle boundary: it will be either inside or outside. This problem will be judged automatically. Your answer must match exactly, including the capitalization, punctuation, & white space. This includes the periods at the ends of the lines. All locations are given in miles.*

IDs for online judges. *POJ 1005, ZOJ 1049, UVA 2363.*

Mathematics analysis. Gọi S_n, r_n lần lượt là tổng diện tích đất sạt lở & bán kính của hình bán nguyệt của mảnh đất sạt lở đến hết năm n, $S_1=0, S_n=S_{n-1}+50=50(n-1)=\dfrac{\pi r_n^2}{2}\Rightarrow r_n=\sqrt{\dfrac{100(n-1)}{\pi}}$, $\forall n\in\mathbb{N}^\star$. Tìm n thỏa mãn $r_{n-1}<\sqrt{x^2+y^2}<r_n$, tuwong □

Problem 69 ([WW16], p. 12: Hangover)**.** *How far can you make a stack of cards overhang a table? If you have 1 card, you can create a maximum overhang of half a card length. (We're assuming that the cards must be perpendicular to the table.) With 2 cards, you can make the top card overhang the bottom one by half a card length, & the bottom one overhang the table by a third of a card length, for a total maximum overhang of* $\frac{1}{2}+\frac{1}{3}=\frac{5}{6}$ *card lengths. In general, you can make* n *cards overhang by* $\sum_{i=2}^{n+1}\frac{1}{i}=\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n+1}$ *card lengths, where the top card overhangs the 2nd by* $\frac{1}{2}$*, the 2nd overhangs the 3rd by* $\frac{1}{3}$*, the 3rd overhangs the 4th by* $\frac{1}{4}$*, & so on, & the bottom card overhangs the table by* $\frac{1}{n+1}$*.*

Input. *The input consists of 1 or more test cases, followed by a line containing the number* 0.00 *that signals the end of the input. Each test case is a single line containing a positive floating-point number* c *whose value is at least* 0.01 *& at most* 5.20*;* c *will contain exactly 3 digits.*

Output. *For each test case, output the minimum number of cards necessary to achieve an overhang of at least c card lengths. Use the exact output format shown in the examples.*

Source. *ACM Mid-Central United States 2001. item* IDs for online judges. *POJ 1003, UVA 2294.*

Problem 70 ([WW16], p. 17: Sum)**.** *Your task is to find the sum of all integer numbers lying between 1 & N inclusive.*

Input. *The input consists of a single integer N that is not greater than 10000 by its absolute value.*

Output. *Write a single integer number that is the sum of all integer numbers lying between 1 & N inclusive.*

Source. *Source: ACM 2000, Northeastern European Regional Programming Contest (test tour).*

ID for online judge. *Ural 1068.*

Bài toán 11 ([Tru23b], HSG12 Tp. Hà Nội 2020–2021, Prob. 1, p. 80: Find mid – Tìm giữa)**.** *(a) Cho $l, r \in \mathbb{N}^\star$. Tìm $m \in [l, r) \cap \mathbb{N}^\star$ để chênh lệch giữa tổng các số nguyên liên tiếp từ l đến m & tổng các số nguyên liên tiếp từ $m+1$ đến r là nhỏ nhất. (b) Mở rộng cho $l, r \in \mathbb{Z}$. (c⋆) Thay tổng bởi tổng bình phương, tổng lập phương, tổng lũy thừa bậc $a \in \mathbb{R}$.*

Input. *2 số $l, r \in \mathbb{N}^\star$, $l < r \leq 10^9$.*

Output. *Gồm 1 số nguyên duy nhất là m thỏa mãn.*

Limits. *Subtask 1:* 60% *các test có $l < r \leq 10^3$. Subtask 2:* 40% *các test còn lại có $l < r \leq 10^9$.*

- Input: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/input/find_mid.inp.
- Output: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/output/find_mid.out.
- C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/find_mid.cpp.
- Python: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/Python/find_mid.py.

Chương 6

Ad Hoc Problems

Contents

- **ad hoc** [a] (from Latin) arranged or happening when necessary & not planned in advance; [adv] (from Latin) in a way that is arranged or happens when necessary & is not planned in advance.

  – (từ tiếng Latin) được sắp xếp hoặc xảy ra khi cần thiết & không được lên kế hoạch trước; [adv] (từ tiếng Latin) theo cách được sắp xếp hoặc xảy ra khi cần thiết & không được lên kế hoạch trước.

For the definitions of "ad hoc", see also, e.g., viWikipedia/ad hoc, enWikipedia/ad hoc.

6.1 Solving Ad Hoc Problems by Mechanism Analysis

Problem 71 ([WW18], 1.1.1, p. 1, Factstone Benchmark)**.** AMTEL *has announced that it will release a 128-bit computer chip by 2010, a 256-bit computer by 2020, & so on, continuing its strategy of doubling the word size every 10 years. (*AMTEL *released a 64-bit computer in 2000, a 32-bit computer in 1990, a 16-bit computer in 1980, an 8-bit computer in 1970, & a 4-bit computer, its 1st, in 1960.)* AMTEL *will use a new benchmark – the* Factstone *– to advertise the vastly improved capacity of its new chips. The* Factstone *rating is defined to be the largest integer* n *such that* $n!$ *can be represented as an unsigned integer in a computer word. Given a year* $1960 \le y \le 2160$*, what will be the* Factstone *rating of* AMTEL*'s most recently released chip?*

Input. *There are several test cases. For each test case, there is 1 line of input containing* y*. A line containing* 0 *follows that last test case.*

Output. *For each test case, output a line giving the Factstone rating.*

Source. *Waterloo local 2005.09.24*

IDs for Online Judges. *POJ 2661, UVA 10916*

Bài toán 12 ([WW18], 1.1.1, p. 1, Điểm chuẩn Factstone)**.** AMTEL *đã thông báo rằng họ sẽ phát hành một con chip máy tính 128-bit vào năm 2010, một máy tính 256-bit vào năm 2020, & cứ thế, tiếp tục chiến lược tăng gấp đôi kích thước từ sau mỗi 10 năm. (*AMTEL *đã phát hành một máy tính 64-bit vào năm 2000, một máy tính 32-bit vào năm 1990, một máy tính 16-bit vào năm 1980, một máy tính 8-bit vào năm 1970, & một máy tính 4-bit, máy tính đầu tiên của họ, vào năm 1960.)* AMTEL *sẽ sử dụng một chuẩn mực mới –* Factstone *– để quảng cáo cho khả năng được cải thiện đáng kể của các con chip mới của mình. Xếp hạng* Factstone *được định nghĩa là số nguyên* n *lớn nhất sao cho* $n!$ *có thể được biểu diễn dưới dạng một số nguyên không dấu trong một từ máy tính. Với năm* $1960 \le y \le 2160$*, xếp hạng* Factstone *của chip mới nhất được phát hành của* AMTEL *sẽ là bao nhiêu?*

Đầu vào. *Có một số trường hợp thử nghiệm. Đối với mỗi trường hợp thử nghiệm, có 1 dòng đầu vào chứa* y*. Một dòng chứa* 0 *theo sau trường hợp thử nghiệm cuối cùng đó.*

Đầu ra. *Đối với mỗi trường hợp thử nghiệm, đầu ra là một dòng cho biết xếp hạng Factstone.*

Analysis. For a given year $y \in [1960, 2160] \cap \mathbb{N}$, 1st the number of bits for the computer in this year is calculated, & then the largest integer n, i.e., the *Factstone rating*, that $n!$ can be represented as an unsigned integer in a computer word is calculated. Since the computer was a 4-bit computer in 1960 & AMTEL doubles the word size every 10 years, the number of bits for the computer in year y is $b = 2^{2+\lfloor \frac{y-1960}{10} \rfloor} \in \mathbb{N}^\star$. The largest unsigned integer for b-bit is $2^b - 1 \in \mathbb{N}^\star$. If $n!$ is the largest unsigned integer $\le 2^b - 1$, then n is the Factstone rating in year y. There are 2 calculation methods:

1. Calculate $n!$ directly, which is slow & easily leads to overflow.
2. Logarithms are used to calculate $n!$, based on the following inequality

$$n! \le 2^b - 1 \Rightarrow \log_2 n! = \sum_{i=1}^{n} \log_2 i = \log_2 1 + \log_2 2 + \cdots + \log_2 n \le \log_2(2^b - 1) < \log_2 2^b = b,$$

n can be calculated by a loop: Initially $i \coloneqq 1$, repeat `++i`, & $\log_2 i$ is accumulated until the sum is $> b$. Then $i - 1$ is the Factstone rating.

Codes:

- C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/Factstone_benchmark.cpp.

```
#include <stdio.h>
#include <math.h>

int main() {
    int y;
    while (1 == scanf("%d", &y) && y) { // input test cases
        double w = log(4);
        for (int Y = 1960; Y <= y; Y += 10)
            w *= 2;
        int i = 1; // accumulation log2 i until > w
        double f = 0; // f: sum of accumulation for log2 i
        while (f < w)
            f += log((double)++i);
        printf("%d\n", i - 1); // output Factstone rating
    }
    if (y) printf("fishy ending %d\n", y);
}
```

- Python:

Bài toán 13 (Mở rộng [WW18], 1.1.1, p. 1, Điểm chuẩn Factstone)**.** Amtel *đã thông báo rằng kể từ năm $y_0 \in \mathbb{N}^\star$ cho trước, họ sẽ phát hành 1 con chip b^{b_0}-"bit" computer ("bit" ở đây được hiểu theo nghĩa rộng là theo cơ số $b \in \mathbb{N}^\star$, $b \ge 2$, chứ không phải hệ nhị phân 2-bit), & cứ sau δ_y năm & δ_m tháng ($\delta_m \in \overline{1,11}$), số "bit" sẽ gấp $b \in \mathbb{N}$, $b \ge 2$ lên so với trước đó.* Amtel *sẽ sử dụng một chuẩn mực mới –* Factstone *– để quảng cáo cho khả năng được cải thiện đáng kể của các con chip mới của mình. Xếp hạng* Factstone *được định nghĩa là số nguyên n lớn nhất sao cho $n!$ có thể được biểu diễn dưới dạng một số nguyên không dấu trong một từ máy tính. Với năm $y_0 \le y$, xếp hạng* Factstone *của chip mới nhất được phát hành của* Amtel *sẽ là bao nhiêu?*

Đầu vào. *Có một số trường hợp thử nghiệm. Đối với mỗi trường hợp thử nghiệm, có 1 dòng đầu vào chứa y. Một dòng chứa 0 theo sau trường hợp thử nghiệm cuối cùng đó.*

Đầu ra. *Đối với mỗi trường hợp thử nghiệm, đầu ra là một dòng cho biết xếp hạng Factstone.*

Remark 4 (log: product $\mapsto$ sum)**.** *When you encounter a product function of n, i.e., $f(n)$, e.g. $n!$ above, use logarithm to transform products into sums.*

Question 4 (Sum $\leftrightarrows$ Product)**.** *Làm sao để chuyển 1 tổng thành 1 tích? Làm sao chuyển 1 tích thành 1 tổng?*

Answer. Chuyển tổng thành tích: $e^{a+b} = e^a e^b$, $\forall a, b \in \mathbb{R}$. Tổng quát:

$$e^{\sum_{i=1}^n a_i} = \prod_{i=1}^{n} e^{a_i},\ \forall a_i \in \mathbb{R},\ \forall n \in \mathbb{N}^\star,\ \forall i = 1, \ldots, n.$$

Chuyển tích thành tổng: $\ln(ab) = \ln a + \ln b$, $\forall a, b \in (0, \infty)$. Tổng quát:

$$\ln \prod_{i=1}^{n} a_i = \sum_{i=1}^{n} \ln a_i,\ \forall a_i \in (0, \infty),\ \forall n \in \mathbb{N}^\star,\ \forall i = 1, \ldots, n.$$

Note: Có thể thay $\ln x$ bởi $\log x, \log_a x$ với $a \in (0, \infty)$ bất kỳ. □

Problem 72 ([WW18], 1.1.2, p. 3, Bridge)**.** *Consider that n people wish to cross a bridge at night. A group of at most 2 people may cross at any time, & each group must have a flashlight. Only 1 flashlight is available among the n people, so some sort of shuttle arrangement must be arranged in order to return the flashlight so that more people may cross.*

Each person has a different crossing speed; the speed of a group is determined by the speed of the slower member. Your job is to determine a strategy that gets all n people across the bridge in the minimum time.

Input. *The 1st line of input contains n, followed by n lines giving the crossing times for each of the people. There are not more than* 1000 *people, & nobody takes more than* 100 *seconds to cross the bridge.*

Output. *The 1st line of output must contain the total number of seconds required for all n people to cross the bridge. The following lines give a strategy for achieving this time. Each line contains either 1 or 2 integers, indicating which person or people form the next group to cross. (Each person is indicated by the crossing time specified in the input. Although many people may have the same crossing time, the ambiguity is of no consequence.) Note that the crossings alternate directions, as it is necessary to return the flashlight so that more may cross. If more than 1 strategy yields the minimal time, any one will do.*

Source. *POJ 2573, ZOJ 1877, UVA 10037*

IDs for Online Judge. *Waterloo local 2000.09.30*

Bài toán 14 ([WW18], 1.1.2, p. 3, "Đi cầu")**.** *Hãy xem xét n người muốn đi qua cầu vào ban đêm. Một nhóm tối đa 2 người có thể đi qua bất kỳ lúc nào, & mỗi nhóm phải có một chiếc đèn pin. Chỉ có 1 chiếc đèn pin trong số n người, vì vậy phải sắp xếp một số loại hình sắp xếp đưa đón để trả lại đèn pin để nhiều người hơn có thể đi qua.*

Mỗi người có tốc độ đi qua khác nhau; tốc độ của một nhóm được xác định bởi tốc độ của thành viên chậm hơn. Nhiệm vụ của bạn là xác định một chiến lược giúp tất cả n người đi qua cầu trong thời gian ngắn nhất.

Đầu vào. *Dòng 1 của đầu vào chứa n, theo sau là n dòng cho biết thời gian đi qua của mỗi người. Không có quá* 1000 *người, & không ai mất hơn* 100 *giây để đi qua cầu.*

Đầu ra. *Dòng 1 của đầu ra phải chứa tổng số giây cần thiết để tất cả n người đi qua cầu. Các dòng sau đưa ra một chiến lược để đạt được thời gian này. Mỗi dòng chứa 1 hoặc 2 số nguyên, cho biết người hoặc những người nào tạo thành nhóm tiếp theo để vượt sông. (Mỗi người được chỉ định theo thời gian vượt sông được chỉ định trong đầu vào. Mặc dù nhiều người có thể có cùng thời gian vượt sông, nhưng sự mơ hồ không quan trọng.) Lưu ý rằng các lần vượt sông thay đổi hướng, vì cần phải trả lại đèn pin để nhiều người có thể vượt sông. Nếu có nhiều hơn 1 chiến lược tạo ra thời gian tối thiểu, bất kỳ chiến lược nào cũng được.*

Computer Science Analysis. The strategy that gets all n people, numbered $P_1, \ldots, P_n$, across the bridge in the minimum time is: fast people should return the flashlight to help slow people. Because a group of ≤ 2 people may cross the bridge each time, we solve the problem by analyzing members of groups. 1st, n people's crossing times, denoted by $t_1, \ldots, t_n$, are sorted in descending order: $t_{i_1} \geq t_{i_2} \geq \cdots t_{i_n}$ where $(i_1, \ldots, i_n)$ is some rearrangement of $(1, \ldots, n)$, i.e., $\{i_1, \ldots, i_n\} = \{1, \ldots, n\}$. Suppose that in the current sequence (i.e., after some people have crossed the bridge & hence being not counted in the current sequence), A, B are the current fastest person P_A & the current 2nd fastest person P_B's crossing times, respectively, a, b are the current slowest person P_a & the current 2nd slowest person P_b's crossing time, respectively. Obviously, $A < B < b < a$. There are 2 methods for making the current slowest person & the current 2nd slowest person to cross the bridge:

- *Method 1*: The fastest person P_A helps the slowest person P_a & the 2nd slowest person P_b to cross the bridge. The steps:
  1. The fastest person P_A & the slowest person P_a cross the bridge, which takes time $\max\{A, a\} = a$.
  2. The fastest person P_A is back, which takes time A.
  3. The fastest person P_A & the 2nd slowest person P_b cross the bridge, which takes time $\max\{A, b\} = b$.
  4. The fastest person is back, which takes time A.

  It takes times $a + A + b + A = 2A + a + b$.
- *Method 2*: The fastest person P_A & the 2nd fastest person P_B help the current slowest person P_a & the current 2nd slowest person P_b to cross the bridge. The steps:
  1. The fastest person P_A & the 2nd fastest person P_B cross the bridge, which takes time $\max\{A, B\} = B$.
  2. The fastest person P_A is back & returns the flashlight to the slowest person P_a & the 2nd slowest person P_b, which takes time A.
  3. The slowest person P_a & the 2nd slowest person P_b cross the bridge & give the flashlight to the 2nd fastest person P_B, which takes time $\max\{a, b\} = a$.
  4. The 2nd faster person P_B is back, which takes time B.

  It takes time $B + A + a + B = 2B + A + a$. Note: In Method 2, the roles of the fastest person P_A & the 2nd fastest person P_B are the same & hence they will take the same time, indeed: $B + B + a + A = 2B + a + A$.

Each time, we need to compare Method 1 & Method 2. If $2A + a + b < 2B + A + a \Leftrightarrow A + b < 2B$, then we use Method 1, else we use Method 2 (in the case $2A + a + b = 2B + A + a \Leftrightarrow A + b = 2B$, either of them can be used). & each time the current slowest person & the current 2nd slowest person cross the bridge. Finally, there are 2 cases depending on n being even or odd (since only 2 persons can cross the bridge in each turn):

- Case 1: If there are only 2 persons who need to cross the bridge, then the 2 persons cross the bridge. It takes time B.
- Case 2: There are 3 persons who need to cross the bridge. 1st, the fastest person & the slowest person cross the bridge. Then, the fastest person is back. Finally, the last 2 persons cross the bridge. It takes time $\max\{A, a\} + A + \max\{A, b\} = a + A + b$.

Mathematical Analysis.

Codes:

- C++:

6.2 Solving Ad Hoc Problems by Statistical Analysis

Unlike mechanism analysis, statistical analysis begins with a partial solution to the problem, & the overall global solution is found based on analyzing the partial solution. Solving problems by statistical analysis is a bottom-up method.

– Không giống như phân tích cơ chế, phân tích thống kê bắt đầu bằng một giải pháp cục bộ cho vấn đề, & giải pháp toàn cục tổng thể được tìm thấy dựa trên việc phân tích giải pháp cục bộ. Giải quyết vấn đề bằng phân tích thống kê là phương pháp từ dưới lên.

Problem 73 ([WW18], 1.2.1., p. 6, Ants)**.** *An army of ants walk on a horizontal pole of length l* cm*, each with a constant speed of* 1 cm/s*. When a walking ant reaches an end of the pole, it immediately falls off it. When 2 ants meet, they turn back & start walking in opposite directions. We know the original positions of ants on the pole; unfortunately, we do not know the directions in which the ants are walking. Your task is to compute the earliest & the latest possible times needed for all ants to fall off the pole.*

Input. *The 1st line of input contains 1 integer giving the number of cases that follow. The data for each case start with 2 integer numbers: the length of pole (in* cm*) & n, the number of ants residing on the pole. These 2 numbers are followed by n integers giving the position of each ant on the pole as the distance measured from the left end of the pole, in no particular order. All input integers are $\leq 10^6$, & they are separated by whitespace.*

Output. *For each case of input, output 2 numbers separated by a single space. The 1st number is the earliest possible time when all ants fall off the pole (if the directions of their walks are chosen appropriately), & the 2nd number is the latest possible such time.*

Source. *Waterloo local 2004.09.19*

IDs for Online judges. *POJ 1852, ZOJ 2376, UVA 10714*

Bài toán 15 ([WW18], 1.2.1., p. 6, Kiến)**.** *Một đội quân kiến đi trên một cột nằm ngang dài l* cm*, mỗi con có tốc độ không đổi là* 1 cm/s*. Khi một con kiến đi đến một đầu của cột, nó ngay lập tức rơi khỏi cột. Khi 2 con kiến gặp nhau, chúng quay lại & bắt đầu đi theo hướng ngược nhau. Chúng ta biết vị trí ban đầu của các con kiến trên cột; thật không may, chúng ta không biết hướng mà các con kiến đang đi. Nhiệm vụ của bạn là tính toán thời gian sớm nhất & thời gian muộn nhất có thể cần thiết để tất cả các con kiến rơi khỏi cột.*

Đầu vào. *Dòng 1 của đầu vào chứa 1 số nguyên cho biết số trường hợp theo sau. Dữ liệu cho mỗi trường hợp bắt đầu bằng 2 số nguyên: chiều dài của cột (tính bằng* cm*) & n, số kiến trú ngụ trên cột. 2 số này được theo sau bởi n số nguyên cho biết vị trí của mỗi con kiến trên cột là khoảng cách được đo từ đầu bên trái của cột, không theo thứ tự cụ thể. Tất cả các số nguyên đầu vào là $\leq 10^6$, & chúng được phân cách bằng khoảng trắng.*

Đầu ra. *Đối với mỗi trường hợp đầu vào, đầu ra 2 số được phân cách bằng một khoảng trắng. Số thứ nhất là thời gian sớm nhất có thể khi tất cả các con kiến rơi khỏi cột (nếu hướng đi của chúng được chọn một cách thích hợp), & số thứ 2 là thời gian muộn nhất có thể như vậy.*

Analysis.

Problem 74 ([WW18], 1.3.1., pp. 12–13, Perfection)**.** *From the article Number Theory in the 1994 Microsoft Encarta: "If $a, b, c \in \mathbb{Z}$ s.t. $a = bc$, a is called a multiple of b or of c, & b or c is called a* divisor *or* factor *of a. If $c \neq \pm 1$, b is called a* proper divisor *of a.*

Chương 7

VNOI

Bài toán 16 (gcd in Pascal triangle – ƯCLN trong tam giác Pascal, https://oj.vnoi.info/problem/gpt)**.** *Tam giác Pascal là 1 cách sắp xếp hình học của các hệ số nhị thức vào 1 tam giác. Hàng thứ $n \in \mathbb{N}$ của tam giác bao gồm các hệ số trong khai triển của đa thức $f(x,y) = (x+y)^n$. I.e., phần tử tại cột thứ k, hàng thứ n của tam giác Pascal là $C_n^k = \binom{n}{k}$, i.e., tổ hợp chập k của n phần tử $0 \le k \le n$. Cho $n \in \mathbb{N}$. Tính* $\mathrm{GPT}(n)$ *là ƯCLN của các số nằm giữa 2 số 1 trên hàng thứ n của tam giác Pascal.*

Input. *Dòng đầu ghi T là số lượng test. T dòng tiếp theo, mỗi dòng ghi 1 số nguyên n.*

Output. *Gồm T dòng, mỗi dòng ghi* $\mathrm{GPT}(n)$ *tương ứng.*

Constraint. $1 \le T \le 20$, $2 \le n \le 10^9$.

Phân tích. Công thức khai triển nhị thức Newton: $(a+b)^n = \sum_{i=0}^{n} C_n^i a^{n-i} b^i$, $\forall n \in \mathbb{N}$, see, e.g., Wikipedia/binomial theorem. Cần tính $\gcd(\{C_n^i; 1 \le i \le n-1\}) = \gcd(C_n^1, C_n^2, \ldots, C_n^{n-1})$. Chú ý mỗi hàng của tam giác Pascal có tính chất đối xứng nên chỉ cần xét "1 nửa" là đủ. Cụ thể hơn: $C_n^k = C_n^{n-k}$, $\forall k \in \mathbb{N}$, $k \le n$, nên

$$\{C_n^1, \ldots, C_n^{n-1}\} = \{C_n^1, \ldots, C_n^{\lfloor \frac{n}{2} \rfloor}\} = \begin{cases} \{C_n^1, \ldots, C_n^{\frac{n-1}{2}}\} & \text{if } n \not\vdots 2, \\ \{C_n^1, \ldots, C_n^{\frac{n}{2}}\} & \text{if } n \vdots 2, \end{cases}$$

nên thay vì xét $i = 1, \ldots, n-1$, chỉ cần xét $i = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ là đủ.

Theorem 1.

$$\gcd\{C_n^i\}_{i=1}^{n-1} = \begin{cases} p & \text{if } n = p^k \text{ for some prime } p \text{ \& some } n \in \mathbb{N}^\star, \\ 1 & \text{if } n \ne p^k \text{ for all prime } p \text{ \& any } n \in \mathbb{N}^\star. \end{cases}$$

See also, e.g.:

- Mathematics StackExchange/gcd of binomial coefficients.

Chương 8

Recurrence Relation – Quan Hệ Hồi Quy

Contents

Resources – Tài nguyên.

1. [AB20]. Dorin Andrica, Ovidiu Bagdasar. *Recurrent Sequences: Key Results, Applications, & Problems.*

Let X be an arbitrary set. A function $f : \mathbb{N} \to X$ defines a *sequence* $(x_n)_{n=0}^{\infty}$ of elements of X, where $x_n = f(n)$, $\forall n \in \mathbb{N}$. The set of all sequences with elements in X is denoted by $X^{\mathbb{N}}$, while X^n denotes Cartesian product of n copies of X, where X will be chosen as $\mathbb{C}$, the Euclidean space $\mathbb{R}^m$, the algebra $M_r(A)$ of the $r \times r$ matrices with entries in a ring A, etc. The set $X^{\mathbb{N}}$ has numerous important subsets. E.g., when $X = \mathbb{R}$, the set of real numbers $\mathbb{R}^{\mathbb{N}}$ includes sequences which are bounded, monotonous, convergent, positive, nonzero, periodic, etc.

– Cho X là 1 tập hợp tùy ý. Một hàm $f : \mathbb{N} \to X$ định nghĩa một *dãy* $(x_n)_{n=0}^{\infty}$ các phần tử của X, trong đó $x_n = f(n)$, $\forall n \in \mathbb{N}$. Tập hợp tất cả các dãy có các phần tử trong X được ký hiệu là $X^{\mathbb{N}}$, trong khi X^n biểu thị tích Descartes của n bản sao của X, trong đó X sẽ được chọn là $\mathbb{C}$, không gian Euclidean $\mathbb{R}^m$, đại số $M_r(A)$ của các ma trận $r \times r$ có các phần tử trong vành A, v.v. Tập hợp $X^{\mathbb{N}}$ có nhiều tập con quan trọng. Ví dụ, khi $X = \mathbb{R}$, tập hợp các số thực $\mathbb{R}^{\mathbb{N}}$ bao gồm các dãy số bị chặn, đơn điệu, hội tụ, dương, khác không, tuần hoàn, v.v.

When $a \in X$ is fixed, in *implicit form*, a recurrence relation is defined by

$$F_n(x_n, x_{n-1}, \ldots, x_0) = a, \ \forall n \in \mathbb{N}^\star, \tag{8.1}$$

where $F_n : X^{n+1} \to X$ is a function of $n+1$ variables, $n \in \mathbb{N}^\star$. In general, the implicit form of a recurrence relation does not define uniquely the sequence $(x_n)_{n=0}^{\infty}$.

The *explicit form* of a recurrence relation is

$$x_n = f_n(x_{n-1}, \ldots, x_0), \ \forall n \in \mathbb{N}^\star, \tag{8.2}$$

where $f_n : X^n \to X$ is a function, $\forall n \in \mathbb{N}^\star$. The relations (8.2) give the rule to construct the term x_n of the sequence $(x_n)_{n\geq 0}$ from the 1st term x_0: $x_1 = f_1(x_0), x_2 = f_2(x_1, x_0), \ldots$, i.e., (8.2) is a functional type relation.

In mathematics, a *recurrence relation* is an equation according to which the nth term of a sequence of numbers is equal to some combination of the previous terms, i.e.:

$$\begin{cases} u_0 \in \mathbb{F}, \\ u_n = f_n(n, u_0, u_1, \ldots, u_{n-1}), \ \forall n \in \mathbb{N}^\star, \end{cases} \tag{8.3}$$

if u_0 is the initial element, which is an element of the given field $\mathbb{F}$, of the sequence $\{u_n\}_{n=0}^{\infty}$, where $f_n : \mathbb{N}^\star \times \mathbb{F}^n \to \mathbb{F}$ is a scalar-valued function of $(n+1)$-dimensional-vector-valued argument, $\forall n \in \mathbb{N}^\star$; &

$$\begin{cases} u_1 \in \mathbb{F}, \\ u_n = f_n(n, u_1, \ldots, u_{n-1}), \ \forall n \in \mathbb{N}, \ n \geq 2, \end{cases} \tag{8.4}$$

if u_1 is the initial element, which is an element of the given field $\mathbb{F}$, of the sequence $\{u_n\}_{n=1}^{\infty}$, where $f_n : \mathbb{N}_{\geq 2} \times \mathbb{F}^{n-1} \to \mathbb{F}$ is a scalar-valued function of n-dimensional-vector-valued argument, $\forall n \in \mathbb{N}$, $n \geq 2$.

Question 5. *What define uniquely* (8.3)*?*

Answer. The solutions $\{u_n\}_{n=0}^{\infty}$ defined by (8.3), are uniquely determined in terms of $u_0 \in \mathbb{F}, \{f_n\}_{n=1}^{\infty}$. Analogously, the solutions $\{u_n\}_{n=0}^{\infty}$ defined by (8.4), are uniquely determined in terms of $u_1 \in \mathbb{F}, \{f_n\}_{n=2}^{\infty}$. □

Remark 5 (Starting index of a sequence). *The starting index of a sequence $\{u_n\}_{n\in\{0,1\}}^{\infty}$ can be* 0*, which is commonly used in Computer Science & various programming languages, or* 1*, which is commonly used in Mathematics.*

Often, only k previous terms of the sequence appear in the equation, for a parameter k that is independent of n; this number k is called the *order* of the relation. If the values of the 1st k numbers in the sequence have been given, the rest of the sequence can be calculated by repeatedly applying the equation.

In *linear recurrences*, the nth term is equated to a linear function of the k previous terms. A famous example is the recurrence for the Fibonacci numbers

$$\begin{cases} F_0 = F_1 = 1, \\ F_n = F_{n-1} + F_{n-2}, \ \forall n \in \mathbb{N}, \ n \geq 2, \end{cases} \tag{Fib}$$

where the order $k = 2$ & the linear function merely adds the 2 previous terms. This example is a linear recurrence with constant coefficients, because the coefficients of the linear function (1 & 1) are constants that do not depend on n. For these recurrences, one can express the general term of the sequence as a closed-form expression of n. Linear recurrences with polynomial coefficients depending on n are also important, because many common [elementary functions] & special functions have a Taylor series whose coefficients satisfy such a recurrence relation (see Wikipedia/holonomic function).

Def: Solving a recurrence relation means obtaining a closed-form solution: a non-recursive function of n.

The concept of a recurrence relation can be extended to multidimensional arrays, i.e., indexed families that are indexed by tuples of naturals.

Definition 1 (Recurrence relation). *A* recurrence relation *is an equation that expresses each element of a sequence as a function of preceding ones. More precisely, in the case where only the immediately preceding element is involved, a* 1st order recurrence relation *has the form*

$$\boxed{\begin{cases} u_0 \in X, \\ u_n = \varphi(n, u_{n-1}), \ \forall n \in \mathbb{N}^\star, \end{cases}} \tag{8.5}$$

where $\varphi : \mathbb{N} \times X \to X$ is a function, where X is a set to which the elements of a sequence must belong. For any $u_0 \in X$, this defines a unique sequence with u_0 as its 1st element, called the initial value*, which is easy to modify the definition for getting sequences starting from the term of index* 1 *or higher.*

A recurrence relation of order $k \in \mathbb{N}^\star$ *has the form*

$$\boxed{\begin{cases} u_0, u_1, \ldots, u_{k-1} \in X, \\ \qquad\qquad u_n = \varphi(n, k, u_{n-1}, u_{n-2}, \ldots, u_{n-k}), \ \forall n \in \mathbb{N}, \ n \geq k, \end{cases}} \tag{8.6}$$

where $\varphi : \mathbb{N}^2 \times X^k \to X$ is a function that involves k consecutive elements of the sequence. In this case, k initial values are needed for defining a sequence.

Remark 6 (Explicit- vs. implicit recurrence relations). *The explicit recurrence relations are the recurrence relations that can be given as (8.3) or (8.4); meanwhile the implicit recurrence relations are the recurrence relations that can be given as*

$$\begin{cases} u_0 \in \mathbb{F}, \\ f_n(n, u_0, u_1, \ldots, u_{n-1}, u_n) = 0, \ \forall n \in \mathbb{N}^\star, \end{cases} \tag{8.7}$$

if u_0 is the initial element, which is an element of the given field $\mathbb{F}$, of the sequence $\{u_n\}_{n=0}^{\infty}$, where $f_n : \mathbb{N}^\star \times \mathbb{F}^{n+1} \to \mathbb{F}$ is a scalar-valued function of $(n+1)$-dimensional-vector-valued argument, $\forall n \in \mathbb{N}^\star$; &

$$\begin{cases} u_1 \in \mathbb{F}, \\ f_n(n, u_1, \ldots, u_{n-1}, u_n) = 0, \ \forall n \in \mathbb{N}, \ n \geq 2, \end{cases} \tag{8.8}$$

if u_1 is the initial element, which is an element of the given field $\mathbb{F}$, of the sequence $\{u_n\}_{n=1}^{\infty}$, where $f_n : \mathbb{N}_{\geq 2} \times \mathbb{F}^n \to \mathbb{F}$ is a scalar-valued function of n-dimensional-vector-valued argument, $\forall n \in \mathbb{N}$, $n \geq 2$. The wellposednesses of (8.7) & (8.8) require that the corresponding recurrent equation has a unique solution to be able to define u_n uniquely.

Example 1 (Factorial). *The factorial is defined by the recurrence relation $n! = n \cdot (n-1)!$, which is (8.5) with $X = \mathbb{N}^\star$, $u_0 = 0! = 1$, $\varphi(x, y) = xy$, $\forall x, y \in X = \mathbb{N}^\star$ so that $u_n = \varphi(n, u_{n-1}) = nu_{n-1} = n(n-1)! = n!$, $\forall n \in \mathbb{N}^\star$. This is an example of a* linear recurrence with polynomial coefficients *of order* 1*, with the simple polynomial (in n) n as its only coefficient.*

Example 2 (Logistic map). *An example of a recurrence relation is the logistic map defined by*

$$\begin{cases} x_0 \in \mathbb{R}, \\ x_{n+1} = rx_n(1 - x_n), \end{cases} \tag{lgt}$$

for a given constant r. The behavior of the sequence depends dramatically on r, but is stable when the initial condition x_0 varies (proofs?)

8.1 Linear recurrence with constant coefficients – Hồi quy tuyến tính với hệ số hằng

See, e.g., Wikipedia/linear recurrence with constant coefficients. In mathematics (including combinatorics, linear algebra, & dynamical system), a *linear recurrence with constant coefficients* (also known as a *linear recurrence relation* or *linear difference equation*) sets equal to 0 a polynomial that is linear in the various iterates of a variable – i.e., in the values of the elements of a sequence. The polynomial's linearity means that each of its terms has degree 0 or 1. A linear recurrence denotes the evolution of some variable over time, with the current time period or discrete moment in time denoted as t, 1 period earlier denoted as $t-1$, 1 period later as $t+1$, etc.

The solution of such an equation is a function of t, & not of any iterate values, giving the value of the iterate at any time. To find the solution, it is necessary to know the specific values (known as *initial conditions*) of n of the iterates, & normally these are the n iterates that are oldest. The equation or its variable is said to be *stable* if from any set of initial conditions the variable's limit as time goes to ∞ exists; this limit is called the *steady state*.

Difference equations are used in a variety of contexts, e.g. in economics to model the evolution through time of variables e.g. gross domestic product, the inflation rate, the exchange rate, etc. They are used in modeling such time series because values of these variables are only measured at discrete intervals. In econometric applications, linear difference equations are modeled with stochastic terms in the form of autoregressive (AR) models & in models e.g. vector autoregression (VAR) & autoregressive moving average (ARMA) models that combine AR with other features.

Definition 2 (Linear recurrence with constant coefficients). *A* linear recurrence with constant coefficients *is an equation of the following form, written in terms of parameters $a_1, \ldots, a_n, b$:*

$$y_n = \sum_{i=1}^{k} a_i y_{n-i} + b, \tag{8.9}$$

or equivalently as

$$y_{n+k} = \sum_{i=1}^{n} a_i y_{n+k-i} + b, \tag{8.10}$$

Chương 9

Dynamic Programming – Quy Hoạch Động

Resource – Tài nguyên.

1. Wikipedia/dynamic programming.
2. [Thư+21b]. Trần Dan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết, Đặng Bình Phương. *Kỹ Thuật Lập Trình*. Chap. 9: Kỹ Thuật Quy Hoạch Động.
3. [Ber05; Ber17]. Dimitri P. Bertsekas. *Dynamic Programming & Optimal Control. Vol. I*. 3e. 4e (can't download yet).
4. [Ber07; Ber12] Dimitri P. Bertsekas. *Dynamic Programming & Optimal Control. Vol. II*. 3e. 4e (can't download yet).

Bài toán 17 ([Thư+21b], p. 441)**.** *(a) Tìm $(x,y) \in \mathbb{R}^2$ thỏa $x^2+y^2 \le 1$ để $x+y$ đạt* GTNN, GTLN*. (b) Tìm $(x,y) \in \mathbb{R}^2$ thỏa $x^2+y^2 \le r^2$ để $x+y$ đạt* GTNN, GTLN *với $r \in (0,\infty)$. (c) Phát biểu ý nghĩa hình học của bài toán.*

Phát biểu bài toán tối ưu/bài toán quy hoạch:

$$\max_{x^2+y^2\le r^2} x+y = \sqrt{2},\ \arg\max_{x^2+y^2\le r^2} x+y = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right),$$
$$\min_{x^2+y^2\le r^2} x+y = -\sqrt{2},\ \arg\min_{x^2+y^2\le r^2} x+y = \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right),$$

Definition 3 (Fibonacci sequences)**.** Fibonacci sequences *are defined by*

$$\begin{cases} F_0 = 0,\ F_1 = 1, \\ F_n = F_{n-1} + F_{n-2},\ \forall n \in \mathbb{N},\ n \ge 2. \end{cases}$$

Definition 4 (Lucas sequences)**.** *The sequence of* Lucas numbers *are defined by*

$$\begin{cases} L_0 = 2,\ L_1 = 1, \\ L_n = L_{n-1} + L_{n-2},\ \forall n \in \mathbb{N},\ n \ge 2. \end{cases}$$

Bài toán 18 (Fibonacci numbers – Số Fibonacci)**.** *(a) Tính dãy số Fibonacci & dãy Lucas bằng: (i) Truy hồi $O(a^n)$ với $a \approx 1.61803$. (ii) Quy hoạch động $O(n)$. (iii) Quy hoạch động cải tiến. (b) Trong mỗi thuật toán, tính cụ thể số lần gọi hàm tính F_i, L_i, với $i = 0,1,\dots,n$, số phép cộng đã thực hiện. Tính time- & space complexities. (c) Mở rộng bài toán cho dãy số truy hồi cấp 2 với hệ số hằng $\{u_n\}_{n=1}^{\infty}$ được xác định bởi:*

$$\begin{cases} u_0 = \alpha,\ u_1 = \beta, \\ u_{n+2} = au_{n+1} + bu_n,\ \forall n \in \mathbb{N}, \end{cases}$$

với $a,b,\alpha,\beta \in \mathbb{C}$ cho trước. (d) Mở rộng bài toán cho dãy số truy hồi cấp 2 với hệ số thay đổi $\{u_n\}_{n=1}^{\infty}$ được xác định bởi:

$$\begin{cases} u_0 = \alpha,\ u_1 = \beta, \\ u_n = a(n)u_{n-1} + b(n)u_{n-2},\ \forall n \in \mathbb{N},\ n \ge 2. \end{cases}$$

với $\alpha,\beta \in \mathbb{C}$, $a,b : \mathbb{N}_{\ge 2} \to \mathbb{C}$ là 2 hàm giá trị phức cho trước.

Chứng minh. Cho $n \in \mathbb{N}$. Đặt $f(n,i)$ là số lần xuất hiện (frequency) của F_i khi tính F_n bằng công thức truy hồi. Dễ chứng minh bằng phương pháp quy nạp Toán học: $f(n, n-i) = F_{i+1}, \forall i = 0, 1, \ldots, n$.

- Số lần call hàm truy hồi $= \sum_{i=0}^{n} f_{n-i} = \sum_{i=0}^{n} F_{i+1} = \sum_{i=1}^{n+1} F_i = F_{n+3} - 1$.
- Số lần thực hiện phép cộng $= F_{n+1} - 1$.
- Tốn $n+1$ ô nhớ để chứa $F_0, F_1, \ldots, F_n$.

□

C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/Fibonacci.cpp.

```
#include <iostream>
using namespace std;
const long nMAX = 10000;

long fib(long i) {
    if (i == 1 || i == 2)
        return 1;
    else
        return fib(i - 1) + fib(i - 2);
}

// \cite{Thu_Phuong_Tien_Triet_Phuong_KTLT}, p. 443
long fib_recurrence(long n) {
    long ans, Fn_1, Fn_2;
    if (n <= 2)
        ans = 1;
    else {
        Fn_1 = fib_recurrence(n - 1);
        Fn_2 = fib_recurrence(n - 2);
        ans = Fn_1 + Fn_2;
    }
    return ans;
}

// \cite{Thu_Phuong_Tien_Triet_Phuong_KTLT}, p. 443
long fib_dynamic(long n) {
    long F[nMAX + 1];
    F[0] = 0;
    F[1] = F[2] = 1;
    for (int i = 2; i <=n; ++i)
        F[i] = F[i - 1] + F[i - 2];
    return F[n];
}

// \cite{Thu_Phuong_Tien_Triet_Phuong_KTLT}, p. 443
long fib_dynamic_improved(long n) {
    long lastF = 1, F = 1;
    int i = 1;
    while (i < n) {
        F += lastF;
        lastF = F - lastF;
        ++i;
    }
    return F;
}

int main() {
    long n, i;
```

```
    cin >> n;
    cout << "Fibonacci sequence of length " << n << ":\n";

    for (i = 0; i <= n; ++i)
        cout << fib(i) << " ";
    cout << "\n";

    for (i = 0; i <= n; ++i)
        cout << fib_recurrence(i) << " ";
    cout << "\n";

    for (i = 0; i <= n; ++i)
        cout << fib_dynamic(i) << " ";
    cout << "\n";

    for (i = 0; i <= n; ++i)
        cout << fib_dynamic_improved(i) << " ";
    cout << "\n";
}
```

Bài toán 19 ([Tru23a], Đăk Nông THCS 2022–2023, 4: virus, p. 32)**.** *Flashback là 1 loại virus máy tính sinh sản rất nhanh khi gặp môi trường thuận lợi & là 1 loại virus nguy hiểm, có tốc độ lây lan nhanh trong môi trường mạng. Flashback lần đầu được phát hiện vào năm 2011 bởi công ty diệt virus Intego dưới dạng 1 bản cài đặt flash giả & chúng sinh sản theo quy luật:*

- *Ngày đầu tiên (ngày thứ* 0*) có* n *cá thể ở mức* 1*.*
- *Ở mỗi ngày tiếp theo, mỗi cá thể mức* i *sinh ra* i *cá thể mức* 1*, các cá thể mới sẽ sinh sôi, phát triển từ ngày hôm sau.*
- *Bản thân cá thể thứ* i *sẽ phát triển thành mức* $i + 1$ *& chu kỳ phát triển trong ngày chấm dứt.*

Requirement. *Xác định sau* k *ngày trong môi trường có bao nhiêu cá thể.*

Input. *Vào từ file* `flashback.inp` *gồm: Dòng 1 chứa số bộ test* $t \in \mathbb{N}^\star$*.* t *dòng tiếp theo, mỗi dòng chứa* $n, k \in \mathbb{N}^\star$*, ràng buộc* $n \in [1000], k \leq [10^5]$*.*

Output. *Ghi ra file* `flashback.out` *1 số nguyên duy nhất là số dư của kết quả tìm được chia cho* $10^9 + 7$*.*

Limit.

- *Subtask 1: có 40% số test ứng với* $n \leq 100, k \leq 1000$*.*
- *Subtask 2: có 60% số test ứng với* $n \leq 1000, k \leq 10^5$*.*

Sample.

flashback.inp	flashback.out
5	65
5 3	130
10 3	170
5 4	2563
11 6	232767
999 6	

(a) Mô tả thành mô hình toán học. (b) Viết chương trình C/C++, Pascal, Python *để giải.*

Chứng minh. Gọi $a(d, l)$ là số cá cá thể ở mức l vào ngày d (d: day, l: level). Thiết lập công thức truy hồi:
Kết quả cuối cùng: $n(F_1 + \sum_{i=1}^{k} F_{2i}) = n \sum_{i=1}^{k} F_{2i} = nF_{2k+1}$.
C++ codes:

- DAK's C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C%2B%2B/DAK_virus.cpp.
- NHT's C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C%2B%2B/NHT_virus.cpp.

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;

const int MOD = 1e9 + 7;
int fib[2000009];

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    fib[0] = 0;
    fib[1] = 1;
    for(int i = 2; i <= 200005; i++) {
        fib[i] = (fib[i - 1] + fib[i - 2]) % MOD;
    }

    int T;
    cin >> T;
    while(T--) {
        int n, k;
        cin >> n >> k;
        cout << n*fib[2*k+1] << '\n';
    }
    return 0;
}
```

□

Bài toán 20 ([Tru23a], Hà Nội HSG9 2021–2022, 5: cổ phiếu VNI, p. 37)**.** *Bình mua bán cổ phiếu VNI trên thị trường chứng khoán. Giả sử giá của 1 cổ phiếu VNI trong $n \in \mathbb{N}^\star$ ngày lần lượt là $a_1, a_2, \ldots, a_n$. Biết mỗi ngày Bình chỉ thực hiện 1 trong 3 hoạt động:*

1. *Mua 1 cổ phiếu VNI*
2. *Bán số lượng cổ phiếu bất kỳ mà Bình đang sở hữu*
3. *Không thực hiện bất kỳ giao dịch nào.*

Requirement. *Bình thực hiện mua bán cổ phiếu VNI như thế nào để thu được lợi nhuận lớn nhất nếu Bình tham gia mua bán bắt đầu từ ngày thứ $t \in \mathbb{N}^\star$ cho trước?*

Input. *Đọc từ file* `vni.inp` *gồm:*

- *Dòng 1: $n \in [10^5]$ là số ngày biết giá cổ phiếu.*
- *Dòng 2: gồm n số nguyên dương $a_1, a_2, \ldots, a_n$ tương ứng là giá cổ phiếu VNI trong từng ngày $a_i \in [10^9]$, $1 \le i \le n$.*
- *Dòng 3: $q \in [10^5]$ là số lượng truy vấn.*
- *q dòng tiếp theo, mỗi dòng gồm $t \in [n]$ thể hiện cho ngày đầu tiên mà Bình tham gia mua bán cổ phiếu VNI.*

Output. *Ghi ra file* `vni.out` *gồm q dòng, mỗi dòng 1 số nguyên duy nhất là lợi nhuận lớn nhất mà Bình thu được ở mỗi truy vấn tương ứng.*

Limit.

- *Subtask 1: có 50% số test tương ứng $n \in [10^3], q = 1$.*
- *Subtask 2: có 30% số test tương ứng $n \in [10^5], q = 1$.*
- *Subtask 3: có 20% số test còn lại không có ràng buộc gì thêm.*

Sample.

flashback.inp	flashback.out
4 1 2 5 4 2 1 3	7 0

Bài toán 21 ([Tru23a], BRVT HSG9 2022–2023, 2: đố vui tin học, p. 57)**.** *Để tổng kết phát thưởng cho cuộc thi Đố vui Tin học. Ban tổ chức có $n \in \mathbb{N}^\star$ phần quà được đánh số thứ tự từ 1 đến n, phần quà thứ i có giá trị là a_i. Ban tổ chức yêu cầu học sinh chọn các phần quà theo quy tắc:*

- *Phần quà chọn sau phải có số thứ tự lớn hơn phần quà chọn trước đó.*
- *Phần quà chọn sau phải có giá trị lớn hơn phần quà chọn trước đó ít nhất k giá trị.*

Requirement. *Giúp các học sinh lựa chọn theo quy tắc ban tổ chức đặt ra sao cho số lượng phần quà được chọn là nhiều nhất.*

Input. *Vào từ file* `gift.inp`*:*

- *Dòng 1 chứa số bộ test $t \in \mathbb{N}^\star$.*
- *Với mỗi bộ test:*
  - *Dòng 1 chứa $n \in [10^4], k \in [10^3]$.*
  - *Dòng 2 chứa n số nguyên dương $a_i \in [10^6]$ là giá trị của phần quà thứ i, $\forall i \in [n]$.*

Output. *Ghi ra file* `gift.out` *1 dòng duy nhất chứa số lượng phần quà nhiều nhất thỏa mãn yêu cầu của Ban tổ chức.*

Sample.

gift.inp	gift.out
4 5 2 4 5 6 4 8 10 2 4 3 6 5 7 6 9 10 8 12 10 3 4 3 6 5 7 6 9 10 8 12 10 4 4 3 6 5 7 6 9 10 8 12	3 5 4 3

Bài toán 22 ([Tru23a], BRVT HSG9 2022–2023, 3: trò chơi, p. 58)**.** *Nhân dịp kỷ niệm ngày thành lập Đoàn, cô Tổng phụ trách tổ chức 1 trò chơi có thưởng cho các bạn lớp 9: Có $n \in \mathbb{N}^\star$ ô vuông được vẽ thẳng hàng trên sân trường, các ô vuông được đánh số thứ tự từ 1 đến n. Mỗi ô vuông i có giá trị năng lượng là h_i. 1 học sinh đang ở ô thứ i, bạn ấy có thể nhảy tới ô vuông tiếp theo theo các cách:*

- *Nếu đang ở ô thứ i, có thể nhảy đến ô vuông thứ tự $i+1, i+2, \ldots, i+k$.*
- *Chi phí năng lượng tiêu hao cho 1 lần nhảy là $|h_i - h_j|$ với h_j là ô đích mà bạn nhảy tới.*

Học sinh nào di chuyển từ ô 1 đến ô n với chi phí năng lượng thấp nhất sẽ được cô thưởng 1 phần quà.

Yêu cầu. *Tìm chi phí thấp nhất để giúp các học sinh nhảy từ ô vuông thứ 1 đến ô vuông thứ n.*

Input. *Vào từ file* `game.inp`*:*

- *Dòng 1 chứa số bộ test $t \in \mathbb{N}^\star$.*
- *Với mỗi bộ test:*
  - *Dòng 1 chứa $2 \le n \le 10^5, k \in [100]$, lần lượt là số ô vuông & số ô vuông tối đa mà học sinh có thể nhảy qua.*
  - *Dòng 2 chứa n giá trị $h_i \in [10^4]$, mỗi số cách nhau 1 ký tự trắng là chi phí năng lượng của ô thứ i, $\forall i \in [n]$.*

Output. *Ghi ra file* `game.out` *1 số nguyên là chi phí năng lượng ít nhất.*

Sample.

game.inp	game.out
1 5 3 10 25 35 40 20	20

Bài toán 23 ([Tru23a], Lâm Đồng HSG9 2022–2023, 4: biểu diễn văn nghệ, p. 82)**.** *Trong 1 chương trình nghệ thuật diễn ra liên tục trong $n \in \mathbb{N}^\star$ giờ, công ty X có danh sách $m \in \mathbb{N}^\star$ nghệ sĩ khác nhau có thể thuê để biểu diễn. Thời điểm bắt đầu biểu diễn được tính bằng 0. Để đơn giản trong quản lý & sắp xếp, các nghệ sĩ được đánh số thứ tự từ 1 đến m, nghệ sĩ thứ $i \in [n]$ biểu diễn trong thời điểm s_i đến thời điểm t_i, $0 \le s_i < t_i \le n$, với tiền công là c_i, $0 \le c_i \le 10^6$.*

Requirement. *Viết chương trình thuê các nghệ sĩ để bất cứ thời điểm nào cũng luôn có ít nhất 1 nghệ sĩ biểu diễn đồng thời tổng chi phí thuê là nhỏ nhất.*

Input. *Vào từ file* `art_performance.inp` *gồm:*

- *Dòng 1 chứa số bộ test $t \in \mathbb{N}^\star$.*
- *Với mỗi bộ test:*
  - *Dòng 1 chứa $n, m \in [400]$.*
  - *m dòng tiếp theo, mỗi dòng chứa 3 số nguyên không âm s_i, t_i, c_i.*

Output. *Ghi ra file* `art_performance.out` *1 số nguyên là chi phí thuê nhỏ nhất.*

Sample.

art_performance.inp	art_performance.out
1 9 5 0 5 25 1 3 18 3 7 21 4 6 38 7 9 20	20

Bài toán 24 ([Tru23a], TS10 chuyên Tin Bình Dương 2023–2024, 2: ghép tranh, p. 93)**.** *Trò chơi thứ 2 của lớp 9A là trò chơi ghép tranh. Cô chủ nhiệm cho 1 bức tranh có $n \in \mathbb{N}^\star$ mảnh ghép & $k \in \mathbb{N}^\star$, $1 \le k \le n \le 50$. Lần lượt từng tổ sẽ thay phiên nhau lên ghép tranh với số mảnh ghép sử dụng không vượt quá k. An nhận thấy phải tìm được tất cả các cách ghép tranh mới có thể chiến thắng trò chơi. Có thể có nhiều cách ghép tranh, 2 cách ghép khác nhau nếu tồn tại 1 cách ghép giúp hoàn thành được bức tranh & bị bỏ qua ở cách kia.*

Requirement. *Giúp An xác định số cách ghép tranh khác nhau để tổ của An có thể ghép hoàn thành bức tranh.*

Input. *Vào từ file* `picture.inp` *gồm:*

- *Dòng 1 chứa số bộ test $t \in \mathbb{N}^\star$.*
- *Mỗi bộ test gồm 1 dòng chứa $n, k \in \mathbb{N}^\star$.*

Output. *Với mỗi bộ test, ghi ra file* `picture.out` *1 số nguyên duy nhất là số cách ghép tranh khác nhau.*

picture.inp	picture.out
1 4 3	7

Bài toán 25 ([Tru23a], Vĩnh Phúc HSG9 2022–2023, 2: lật ký tự, p. 210)**.** *Cho xâu S gồm $n \in \mathbb{N}^\star$ ký tự* `.` *và* `#` *được đánh số từ 1 đến n. Thao tác lật ký tự của xâu được định nghĩa như sau:*

- *Chọn $i \in [n]$.*
- *Nếu ký tự thứ i của xâu S là* `.` *thì nó sẽ được thay thế bằng* `#`*. Ngược lại, nếu là* `#` *thì sẽ được thay thế bằng* `.`*.*

Requirement. *Lập trình tính xem cần thực hiện ít nhất bao nhiêu thao tác để trong xâu không có ký tự* `.` *nào ở ngay bên phải ký tự* `#`.

Input. *Vào từ file* `pchar.inp`*: Dòng 1 chứa số bộ test* $t \in \mathbb{N}^\star$. *Với mỗi bộ test:*

- *Dòng 1 ghi* $n \in [200000]$ *là số lượng ký tự xâu trong* S.
- *Dòng 2 ghi xâu* S *gồm* n *ký tự* `.` `#`.

Subtask.

- *Subtask 1: Ràng buộc* $1 \leq n \leq 2000$
- *Subtask 2: Không có ràng buộc bổ sung.*

Output. *Ghi ra file* `pchar.out` *1 số nguyên duy nhất là số thao tác ít nhất cần thực hiện để xâu* S *không có bất kỳ thứ tự* `.` *nào ở ngay bên phải ký tự* `#`.

Sample.

pchar.inp	pchar.out
3 3 #.# 5 #.##. 9	1 2 0

Định nghĩa 1 (Dãy phần tử cực trị của 1 dãy số thực cho trước)**.** *Cho dãy số thực* $\{a_i\}_{i=1}^n$. Dãy phần tử lớn nhất *của dãy số thực* $\{a_i\}_{i=1}^n$ *được định nghĩa bởi:*

$$a_i^{\max} := \max_{i \leq j \leq n} a_j.$$

Tương tự, dãy phần tử nhỏ nhất *của dãy số thực* $\{a_i\}_{i=1}^n$ *được định nghĩa bởi:*

$$a_i^{\min} := \min_{i \leq j \leq n} a_j.$$

Bài toán 26 ([Tru23b], HSG12 Nam Định 2020–2021, 4: work, pp. 21–2)**.** *Trong 1 dây chuyền làm việc của công ty có* n *công nhân làm* n *việc. Người ta đánh số cho công nhân từ* 1 *đến* n *theo thứ tự đứng trong dây chuyền. Thời gian hoàn thành 1 công việc của người thứ* i *là* t_i *phút. Mỗi người cần làm xong công việc của mình nhưng được quyền làm tối đa* 2 *việc. Vì thế họ có thể phối hợp với người đứng ngay trước mình cùng làm, nếu người thứ* i *& người thứ* $i+1$ *phối hợp thì thời gian làm xong việc cho* 2 *người là* p_i.

Bài toán 27 ([Thư+21b], p. 446, tính C_n^k)**.** *(a) Viết chương trình* C/C++, Python *để tính* C_n^k *– số tổ hợp chập* k *của* n *phần tử bằng công thức truy hồi nhờ đẳng thức Pascal:*

$$C_n^k = \begin{cases} 1 & \text{if } k = 0 \vee k = n, \\ C_{n-1}^k + C_{n-1}^{k-1} & \text{if } 0 < k < n, \end{cases}$$

với $n, k \in \mathbb{N}$, $0 \leq k \leq n$, *được nhập vào. (b) Có thể sử dụng mảng 1 chiều để lưu lại dòng trước đó mà không cần phải lưu lại tất cả.*

Bài toán 28 ([Thư+21b], II.1, p. 447, tìm dãy con đơn điệu tăng dài nhất)**.** *Cho dãy số nguyên* $a = \{a_i\}_{i=1}^n = a_1, \ldots, a_n$ *gồm* $n \in \mathbb{N}^\star$ *phần tử. 1 dãy con của* a *là 1 cách chọn trong* a *1 số phần tử giữ nguyên thứ tự (có tất cả* 2^n *dãy con của 1 dãy có* n *phần tử). Tìm dãy con đơn điệu tăng (resp., giảm, không tăng, không giảm) của* a *có độ dài lớn nhất.*

CS solution. Giả sử dãy ban đầu gồm $a[1], a[2], \ldots, a[n]$. Bổ sung vào a 2 phần tử $a[0] = -\infty$ & $a[n+1] = \infty$ (khi viết chương trình $\pm\infty$ sẽ được cài đặt các giá trị thích hợp). Khi đó dãy con đơn điệu tăng dài nhất sẽ bắt đầu từ $a[0]$ & kết thúc ở $a[n+1]$. Đặt $l(i)$ là độ dài dãy con đơn điệu tăng dài nhất bắt đầu từ $a[i]$, $\forall i = 0, 1, \ldots, n+1$. Cần tính tất cả $l(i)$ này. Đáp số bài toán sẽ là dãy ứng với $l(i_0)$ có GTLN.

Cơ sở quy hoạch động (bài toán nhỏ nhất). Trường hợp đặc biệt, $l(n+1)$ là độ dài dãy con đơn điệu tăng dài nhất bắt đầu tại $a[n+1] = \infty$. Do dãy con này chỉ gồm 1 phần tử ∞ nên $l(n+1) = 1$.

Công thức truy hồi. Cần tính $l(i)$ với $i = n, \ldots, 1, 0$. Giá trị $l(i)$ sẽ được tính trong điều kiện $l(i+1), \ldots, l(n+1)$ đã biết. Dãy con đơn điệu tăng dài nhất bắt đầu từ $a[i]$ sẽ được thành lập bằng cách lấy $a[i]$ ghép vào đầu 1 trong số các dãy con đơn điệu tăng dài nhất bắt đầu từ vị trí $a[j] > a[i]$ (để đảm bảo tính tăng) nào đó đứng sau $a[i]$ & chọn dãy dài nhất trong số đó để ghép $a[i]$ vào đầu để đảm bảo tính dài nhất. Nên $l(i)$ được tính bằng cách xét tất cả các chỉ số $j = i+1, \ldots, n+1$ mà $a[j] > a[i]$, chọn ra chỉ số $j_{\max}$ có $l(j_{\max})$ lớn nhất:

$$l(i) = l(j_{\max}) + 1 = \max\{l(j); i < j \le n+1,\ a[i] < a[j]\} + 1.$$

□

Bài toán 29 ([Thư+21b], II.1.4.1., p. 451, bố trí phòng họp). *Có $n \in \mathbb{N}^\star$ cuộc họp, cuộc họp thứ i bắt đầu vào thời điểm s_i (start) & kết thúc vào thời điểm f_i (final). Do chỉ có 1 phòng hội thảo nên 2 cuộc họp bất kỳ sẽ được cùng bố trí phục vụ nếu khoảng thời gian làm việc của chúng chỉ giao nhau tại 1 đầu mút. Bố trí phòng họp để phục vụ được nhiều cuộc họp nhất.*

Bài toán 30 ([Thư+21b], II.1.4.2., p. 451, cho thuê máy). *Trung tâm tính toán hiệu năng cao nhận được đơn đặt hàng của $n \in \mathbb{N}^\star$ khách hàng. Khách hàng i muốn sử dụng máy trong khoảng thời gian từ s_i (start) đến f_i (final) & thả tiền thuê là c_i. Bố trí lịch thuê máy để tổng số tiền thu được là lớn nhất mà thời gian sử dụng máy của 2 khách bất kỳ được phục vụ đều không giao nhau.*

Bài toán 31 ([Thư+21b], II.1.4.3., p. 451, dãy tam giác bao nhau). *Cho $n \in \mathbb{N}^\star$ tam giác trên mặt phẳng. Tam giác i bao tam giác j nếu 3 đỉnh của tam giác j đều nằm trong tam giác i (có thể nằm trên cạnh). Tìm dãy tam giác bao nhau có nhiều tam giác nhất.*

Problem 75 (CSES Problem Set/dice combinations). *Count the number of ways to construct sum $n \in \mathbb{N}^\star$ by throwing a dice 1 or more times. Each throw produces an outcome between 1 & 6. E.g., if $n = 3$, there are 4 ways: $3 = 1+1+1 = 1+2 = 2+1 = 3$.*

Input. *The only input line has an integer $n \in \mathbb{N}^\star$.*

Output. *Print the number of ways module $10^9 + 7$.*

Constraints. $n \in [10^6]$.

Problem 76 (CSES Problem Set/minimizing coins). *Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money x using the available coins in such a way that the number of coins is minimal. E.g., if the coins are $\{1, 5, 7\}$ & the desired sum is 11, an optimal solution is $5 + 5 + 1$ which requires 3 coins.*

Input. *The 1st input line has 2 integer $n, x \in \mathbb{N}^\star$: the number of coins & the desired sum of money. The 2nd line has n distinct integers $\{c_i\}_{i=1}^n = c_1, c_2, \ldots, c_n$: the value of each coin.*

Output. *Print 1 integer: the minimum number of coins. If it is not possible to produce the desired sum, print* `-1`.

Constraints. $n \in [100], x \in [10^6], c_i \in [10^6]$.

Sample.

minimizing_coin.inp	minimizing_coin.out
3 11 1 5 7	3

Problem 77 (CSES Problem Set/coin combinations I). *Consider a money system consisting of n coins. Each coin has a positive integer value. Calculate the number of distinct ways you can produce a money sum x using the available coins. E.g., if the coins are $\{2, 3, 5\}$ & the desired sum is 9, there are 8 ways: $8 = 2+2+5 = 2+5+2 = 5+2+2 = 3+3+3 = 2+2+2+3 = 2+2+3+2 = 2+3+2+2 = 3+2+2+2$.*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the number of coins & the desired sum of money. The 2nd line has n distinct integers $\{c_i\}_{i=1}^n = c_1, c_2, \ldots, c_n$: the value of each coin.*

Output. *Print 1 integer: the number of ways module $10^9 + 7$.*

Constraints. $n \in [100], x \in [10^6], c_i \in [10^6]$.

Sample.

coin_combination_I.inp	coin_combination_I.out
3 9 2 3 5	8

Problem 78 (CSES Problem Set/coin combinations II). *Consider a money system consisting of n coins. Each coin has a positive integer value. Calculate the number of distinct ordered ways you can produce a money sum x using the available coins. E.g., if the coins are $\{2,3,5\}$ & the desired sum is 9, there are 3 ways: $8=2+2+5=3+3+3=2+2+2+3$.*

Input. *The 1st input line has 2 integers $n,x\in\mathbb{N}^\star$: the number of coins & the desired sum of money. The 2nd line has n distinct integers $\{c_i\}_{i=1}^n=c_1,c_2,\ldots,c_n$: the value of each coin.*

Output. *Print 1 integer: the number of ways module 10^9+7.*

Constraints. $n\in[100],x\in[10^6],c_i\in[10^6]$.

Sample.

coin_combination_II.inp	coin_combination_II.out
3 9 2 3 5	3

Problem 79 (CSES Problem Set/removing digits). *You are given an integer $n\in\mathbb{N}^\star$. On each step, you may subtract 1 of the digits from the number. How many steps are required to make the number equal to 0?*

Input. *The only input line has an integer $n\in\mathbb{N}^\star$.*

Output. *Print 1 integer: the minimum number of steps.*

Constraints. $n\in[10^6]$.

Sample.

removing_digit.inp	removing_digit.out
27	5

Explanation: An optimal solution is $27\to20\to18\to10\to9\to0$.

Problem 80 (CSES Problem Set/grid paths I). *Consider an $n\times n$ grid whose squares may have traps. It is not allowed to move to a square with a trap. Calculate the number of paths from the upper-left square to the lower-right square. You can only move right or down.*

Input. *The 1st input line has an integer $n\in\mathbb{N}^\star$: the size of the grid. After this, there are n lines that describe the grid. Each line has n character: . denotes an empty cell, & * denotes a trap.*

Output. *Print the number of paths modulo 10^9+7.*

Constraints. $n\in[10^3]$.

Sample.

grid_path_I.inp	grid_path_I.out
4 .... .*.. ...* *...	3

Problem 81 (CSES Problem Set/book shop). *You are in a book shop which sells $n\in\mathbb{N}^\star$ different books. You know the price & number of pages of each book. You have decided that the total price of your purchases will be at most x. What is the maximum number of pages you can buy? You can buy each book at most once.*

Input. *The 1st input line contains 2 integers $n,x\in\mathbb{N}^\star$: the number of books & the maximum total price. The next line contains n integers $h_1,h_2,\ldots,h_n$: the price of each book. The last line contains n integers $s_1,s_2,\ldots,s_n$: the number of pages of each book.*

Output. *Print 1 integer: the maximum number of pages.*

Constraints. $n\in[10^3],x\in[10^5],h_i,s_i\in[10^3],\forall i\in[n]$.

Sample.

book_shop.inp	book_shop.out
4 10 4 8 5 3 5 12 8 1	13

Problem 82 (CSES Problem Set/array description). *You know that an array has $n \in \mathbb{N}^\star$ integers in $[m]$, & the absolute difference between 2 adjacent values is at most 1. Given a description of the array where some values may be unknown, count the number of arrays that match the description.*

Input. *The 1st input line has integers $n, m \in \mathbb{N}^\star$: the array size & the upper bound for each value. The next line has n integers $x_1, x_2, \ldots, x_n \in \mathbb{N}^\star$: the contents of the array. Value 0 denotes an unknown value.*

Output. *Print 1 integer: the number of arrays module $10^9 + 7$.*

Constraints. $n \in [10^5], m \in [100], x_i \in \{0, 1, \ldots, m\}, \forall i \in [n]$.

Sample.

array_description.inp	array_description.out
3 5 2 0 2	3

Explanation: The arrays $[2, 1, 2], [2, 2, 2], [2, 3, 2]$ match the description.

Problem 83 (CSES Problem Set/counting towers). *Build a tower whose width is 2 & height is n. You have an unlimited supply of blocks whose width & height are integers. Given n, how many different towers can you build? Mirrored & rotated towers are counted separately if they look different.*

Input. *The 1st input line has integers $t \in \mathbb{N}^\star$: the number of tests. After this, there are t lines, & each line contains an integer $n \in \mathbb{N}^\star$: the height of the tower.*

Output. *For each test, print the number of towers module $10^9 + 7$.*

Constraints. $t \in [100], n \in [10^6]$.

Sample.

counting_tower.inp	counting_tower.out
3 2 6 1337	8 2864 640403945

Problem 84 (CSES Problem Set/edit distance). *The* edit distance *between 2 strings is the minimum number of operations required to transform 1 string into the other. The allowed operations are:*

- *Add 1 character to the string.*
- *Remove 1 character from the string.*
- *Replace 1 character in the string.*

E.g., the edit distance between `LOVE` *&* `MOVIE` *is 2, because you can 1st replace* `L` *with* `M`*, & then add* `I`*. Calculate the edit distance between 2 strings.*

Input. *The 1st input line has a string that contains $n \in \mathbb{N}^\star$ characters between* `A-Z`*. The 2nd input line has a string that contains $m \in \mathbb{N}^\star$ characters between* `A-Z`*.*

Output. *Print 1 integer: the edit distance between the strings.*

Constraints. $m, n \in [5000]$.

Sample.

edit_distance.inp	edit_distance.out
LOVE MOVIE	2

Problem 85 (CSES Problem Set/longest common subsequence). *Given 2 arrays of integers, find their longest common subsequence. A subsequence is a sequence of array elements from left to right that can contain gaps. A common subsequence is a subsequence that appears in both arrays.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the size of the arrays. The 2nd line has n integers $a_1, a_2, \ldots, a_n$: the contents of the 1st array. The 3rd line has m integers $b_1, b_2, \ldots, b_m$: the contents of the 2nd array.*

Output. *1st print the length of the longest common subsequence. After that, print an example of such a sequence. If there are several solutions, you can print any of them.*

Constraints. $m, n \in [10^3], a_i, b_i \in [10^9], \forall i \in [n]$.

Sample.

longest_common_subsequence.inp	longest_common_subsequence.out
8 6 3 1 3 2 7 4 8 2 6 5 1 2 3 4	3 1 2 4

Problem 86 (CSES Problem Set/rectangle cutting). *Given an $a \times b$ rectangle, cut it into squares. On each move, you can select a rectangle & cut it into 2 rectangles in such a way that all side lengths remain integers. What is the minimum possible number of moves?*

Input. *The only input line has 2 integers $a, b \in \mathbb{N}^\star$:*

Output. *Print 1 integer: the minimum number of moves.*

Constraints. $a, b \in [500]$.

Sample.

rectangle_cutting.inp	rectangle_cutting.out
3 5	3

Problem 87 (CSES Problem Set/minimal grid path). *You are given an $n \times n$ grid whose each square contains a letter. You should move from the upper-left square to the lower-right square. You can only move right or down. What is the lexicographically minimal string you can construct?*

Input. *The 1st input line has integers $n, m \in \mathbb{N}^\star$: the size of the grid. After this, there are n lines that describe the grid. Each line has n letters betweeen* `A-Z`.

Output. *Print the lexicographically minimal string.*

Constraints. $n \in [3000]$.

Sample.

minimal_grid_path.inp	minimal_grid_path.out
4 AACA BABC ABDA AACA	AAABACA

Problem 88 (CSES Problem Set/money sums). *You have n coins with certain values. Find all money sums you can create using these coins.*

Input. *The 1st input line has integers $n, m \in \mathbb{N}^\star$: the number of coins. The next line has n integers $x_1, x_2, \ldots, x_n$: the values of the coins.*

Output. *1st print an integer $k \in \mathbb{N}^\star$: the number of distinct money sums. After this, print all possible sums in increasing order.*

Constraints. $n \in [100], x_i \in [10^3], \forall i \in [n]$.

Sample.

money_sum.inp	money_sum.out
4 4 2 5 2	9 2 4 5 6 7 8 9 11 13

Problem 89 (CSES Problem Set/removal game). *There is a list of $n \in \mathbb{N}^\star$ numbers & 2 players who move alternately. On each move, a player removes either the 1st or last number from the list, & their score increases by that number. Both players try to maximize their scores. What is the maximum possible score for the 1st player when both players play optimally?*

Input. *The 1st input line has integers $n, m \in \mathbb{N}^\star$: the size of the list. The next line has n integers $x_1, x_2, \ldots, x_n$: the contents of the list.*

Output. *Print the maximum possible score for the 1st player.*

Constraints. $n \in [5000], m \in [100], x_i \in \overline{-10^9, 10^9}$, $\forall i \in [n]$.

Sample.

removal_game.inp	removal_game.out
4 4 5 1 3	8

Problem 90 (CSES Problem Set/2 sets II). *Count the number of ways numbers $[n]$ can be divided into 2 sets of equal sum. E.g., if $n = 7$, there are 4 solutions: $\{1, 3, 4, 6\}$ & $\{2, 5, 7\}$, $\{1, 2, 5, 6\}$ & $\{3, 4, 7\}$, $\{1, 2, 4, 7\}$ & $\{3, 5, 6\}$, $\{1, 6, 7\}$ & $\{2, 3, 4, 4\}$.*

Input. *The only input line contains an integer $n \in \mathbb{N}^\star$:*

Output. *Print the answer modulo $10^9 + 7$.*

Constraints. $n \in [500]$.

Sample.

two_sets_II.inp	two_sets_II.out
7	4

Problem 91 (CSES Problem Set/mountain range). *There are $n \in \mathbb{N}^\star$ mountains in a row, each with a specific height. You begin your hang gliding route from some mountain. You can glide from mountain a to mountain b if mountain a is taller than mountain b & all mountains between a & b. What is the maximum number of mountains you can visit on your route?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of mountains. The next line has n integers $h_1, h_2, \ldots, h_n$: the heights of the mountains.*

Output. *Print 1 integer: the maximum number of mountains.*

Constraints. $n \in [2 \cdot 10^5], h_i \in [10^9]$, $\forall i \in [n]$.

Sample.

.inp	.out
10 20 15 17 35 25 40 12 19 13 12	5

Problem 92 (CSES Problem Set/increasing subsequence). *You are given an array containing $n \in \mathbb{N}^\star$ integers. Determine the longest increasing subsequence in the array, i.e., the longest subsequence where every element is larger than the previous one. A subsequence is a sequence that can be derived from the array by deleting some elements without changing the order of the remaining elements.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the size of the array. After this there are n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*

Output. *Print the length of the longest increasing subsequence.*

Constraints. $n \in [2 \cdot 10^5], x_i \in [10^9]$, $\forall i \in [n]$.

Sample.

increasing_subsequence.inp	increasing_subsequence.out
8 7 3 5 3 6 2 9 8	4

Problem 93 (CSES Problem Set/projects). *There are $n \in \mathbb{N}^\star$ you can attend. For each project, you know its starting & ending days & the amount of money you would get as reward. You can only attend 1 project during a day. What is the maximum amount of money you can earn?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of projects. After this, there are n lines. Each such line has 3 integers $a_i, b_i, p_i \in \mathbb{N}^\star$: the starting day, the ending day, & the reward.*

Output. *Print 1 integer: the maximum amount of money you can earn.*

Constraints. $n \in [2 \cdot 10^5], a_i, b_i, p_i[10^9]$, $a_i \leq b_i$, $\forall i \in [n]$.

Sample.

project.inp	project.out
4 2 4 4 3 6 6 6 8 2 5 7 3	7

Problem 94 (CSES Problem Set/elevator rides). *There are $n \in \mathbb{N}^\star$ people who want to get to the top of a building which has only 1 elevator. You know the weight of each person & the maximum allowed weight in the elevator. What is the minimum number of elevator rides?*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the number of people & the maximum allowed weight in the elevator. The 2nd line has n integers $w_1, w_2, \ldots, w_n$: the weight of each person.*

Output. *Print 1 integer: the minimum number of rides.*

Constraints. $n \in [20], x \in [10^9], w_i \in [x]$.

Sample.

elevator_ride.inp	elevator_ride.out
4 10 4 8 6 1	2

Problem 95 (CSES Problem Set/counting tilings). *Count the number of ways you can fill an $n \times m$ grid using 1×2 & 2×1 tiles.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$:*

Output. *Print 1 integer: the number of ways module $10^9 + 7$.*

Constraints. $n \in [10], m \in [1000]$.

Sample.

counting_tiling.inp	counting_tiling.out
4 7	781

Problem 96 (CSES Problem Set/counting numbers). *Count the number of integers between $a, b \in \mathbb{N}\star$ where no 2 adjacent digits are the same.*

Input. *The 1st input line has 2 integers $a, b \in \mathbb{N}^\star$:*

Output. *Print 1 integer: the answer to the problem.*

Constraints. $0 \leq a \leq b \leq 10^{18}$.

Sample.

counting_number.inp	counting_number.out
123	321

Problem 97 (CSES Problem Set/increasing subsequence II). *Given an array of $n \in \mathbb{N}^\star$ integers, calculate the number of increasing subsequences it contains. If 2 subsequences have the same values but in different positions in the array, they are counted separately.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the size of the array. The 2nd line has n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*

Output. *Print 1 integer: the number of increasing subsequences module $10^9 + 7$.*

Constraints. $n \in [2 \cdot 10^5], x_i \in [10^9]$.

Sample.

increasing_subsequence_II.inp	increasing_subsequence_II.out
3 2 1 3	5

Explanation: The increasing subsequences are $[2], [1], [3], [2, 3], [1, 3]$.

Bài toán 32 (VNOI/cắt hình chữ nhật)**.** *Người ta dùng máy cắt để cắt 1 hình chữ nhật kích thước* $m \times n$*, với* $m, n \in [5000]$*, thành 1 số ít nhất các hình vuông có kích thước nguyên dương & có các cạnh song song với cạnh hình chữ nhật ban đầu. Máy cắt khi cắt luôn cắt theo phương song song với 1 trong 2 cạnh của hình chữ nhật & chia hình chữ nhạt thành 2 phần.*

Input. *Gồm 2 số là kích thước* m, n *cách nhau bởi dấu cách.*

Output. *Ghi số* k *là số hình vuông nhỏ nhất được tạo ra.*

Sample.

cut_rectangle.inp	cut_rectangle.out
5 6	5

Chương 10

Graph Algorithms – Thuật Toán Đồ Thị

Problem 98 (CSES Problem Set/counting rooms)**.** *You are given a map of a building, & your task is to count the number of its rooms. The size of the map is $n \times m$ squares, & each square is either floor or wall. You can walk left, right, up, & down through the floor squares.*

Input. *The 1st input lines has 2 integers n, m: the height & width of the map. Then there are n lines of m characters describing the map. Each character is either* . *(floor) or* # *(wall).*

Output. *Print 1 integer: the number of rooms.*

Constraints. $1 \leq n, m \leq 10^3$.

Sample.

counting_room.inp	counting_room.out
5 8 ######## #..#...# ####.#.# #..#...# ########	3

Bài toán 33. *Bạn được cung cấp 1 bản đồ của một tòa nhà, & nhiệm vụ của bạn là đếm số phòng trong đó. Kích thước của bản đồ là $n \times m$ ô vuông, & mỗi ô vuông là sàn hoặc tường. Bạn có thể đi sang trái, phải, lên, & xuống qua các ô vuông trên sàn.*

Đầu vào. *Dòng đầu vào thứ nhất có 2 số nguyên n, m: chiều cao & chiều rộng của bản đồ. Sau đó, có n dòng gồm m ký tự mô tả bản đồ. Mỗi ký tự là* . *(sàn) hoặc* # *(tường).*

Đầu ra. *In 1 số nguyên: số phòng.*

Ràng buộc. $1 \leq n, m \leq 10^3$.

Sample.

counting_room.inp	counting_room.out
5 8 ######## #..#...# ####.#.# #..#...# ########	3

Solution. C++:

1. PGL's C++: counting room: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C%2B%2B/PGL_counting_room.cpp

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

```
4
5  void dfs(vector<vector<char>> &a, int r, int c, int n, int m) {
6      if ( r < 0 || c < 0 || r >= n || c >= m || a[r][c] != '.' ) {
7          return;
8      }
9
10     a[r][c] = '#';
11
12     dfs(a, r + 1, c, n, m);
13     dfs(a, r - 1, c, n, m);
14     dfs(a, r, c + 1, n, m);
15     dfs(a, r, c - 1, n, m);
16 }
17
18 void Countroom(vector<vector<char>> &a, int n, int m) {
19     int count = 0;
20     for (int i = 0; i < n; i++) {
21         for (int j = 0; j < m; j++) {
22             if (a[i][j] == '.') {
23                 dfs(a, i, j, n, m);
24                 count++;
25             }
26         }
27     }
28     cout << count << endl;
29 }
30
31 int main() {
32     int n, m; cin >> n >> m;
33     vector<vector<char>> a(n, vector<char>(m)) ;
34     for (int i = 0; i < n; i++) {
35         for (int j = 0; j < m; j++) {
36             cin >> a[i][j];
37         }
38     }
39     Countroom(a, n, m);
40 }
```

□

Problem 99 (CSES Problem Set/labyrinth). *You are given a map of a labyrinth, task: find a path from start to end. You can walk left, right, up, & down.*

Input. *The 1st input line has 2 integers n, m: the height & width of the map. Then there are n lines m characters describing the labyrinth. Each character is* `.` *(floor),* `#` *(wall),* `A` *(start), or* `B` *(end). There is exactly 1* `A` *& 1* `B` *in the input.*

Output. *1st print* `YES`*, if there is a path, &* `No` *otherwise. If there is a path, print the length of the shortest such path & its description as a string consisting of characters* `L` *(left),* `R` *(right),* `U` *(up), &* `D` *(down). You can print any valid solution.*

Constraints. $1 \leq n, m \leq 1000$.

Sample. *Input:*

```
5 8
########
#.A#...#
#.##.#B#
#......#
########
```

Output:

```
YES
9
LDDRRRRRU
```

Problem 100 (CSES Problem Set/building roads). *Byteland has n cities, & m roads between them. Goal: construct new roads so that there is a route between any 2 cities. Task: find out the minimum number of roads required, & also determine which roads should be built.*

Input. *The 1st input line has 2 integers n, m: the number of cities & roads. The cities are numbered $1, 2, \ldots, n$. After that, there are m lines describing the roads. Each lines has 2 integers a, b: there is a road between those cities. A road always connects 2 different cities, & there is at most 1 road between any 2 cities.*

Output. *1st print an integer k: the number of required roads. Then, print k lines that describe the new roads. You can print any valid solution.*

Constraints. $1 \le n \le 10^5, 1 \le m \le 2 \cdot 10^5, 1 \le a, b \le n$.

Sample.

build_road.inp	build_road.out
4 2 1 2 3 4	1 2 3

Problem 101 (https://cses.fi/problemset/task/1667CSES Problem Set/message route). *Syrjälä's network has n computers & m connections. Task: find out if Uolevi can send a message to Maija, & if it is possible, what is the minimum number of computers on such a route.*

Input. *The 1st input line has 2 integers n, m: the number of computers & connections. The computers are numbered $1, 2, \ldots, n$. Uolevi's computer is 1 & Maija's computer is n. Then, there are m lines describing the connections. Each line has 2 integers a, b: there is a connection between those computers. Every connection is between 2 different computers, & there is at most 1 connection between any 2 computers.*

Output. *If it is possible to send a message, 1st print k: the minimum number of computers on a valid route. After this, print an example of such a route. You can print any valid solution. If there are no routes, print* `IMPOSSIBLE`.

Constraints. $2 \le n \le 10^5, 1 \le m \le 2 \cdot 10^5, 1 \le a, b \le n$.

Sample.

message_route.inp	message_route.out
5 5 1 2 1 3 1 4 2 3 5 4	3 1 4 5

Problem 102 (CSES Problem Set/building team). *There are n pupils in Uolevi's class, & m friendships between them. Task: divide pupils into 2 teams in such a way that no 2 pupils in a team are friends. You can freely choose the sizes of the teams.*

Input. *The 1st input line has 2 integers n, m: the number of pupils & friendships. The pupils are numbered $1, 2, \ldots, n$. Then, there are m lines describing the friendships. Each line has 2 integers a, b: pupils a, b are friends. Every friendship is between 2 different pupils. You can assume that there is at most 1 friendship between any 2 pupils.*

Output. *Print an example of how to build the teams. For each pupil, print* `1` *or* `2` *depending on to which team the pupil will be assigned. You can print any valid team. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $1 \le n \le 10^5, 1 \le m \le 2 \cdot 10^5, 1 \le a, b \le n$.

Sample.

build_team.inp	build_team.out
5 3 1 2 1 3 4 5	1 2 2 1 2

Problem 103 (CSES Problem Set/round trip). *Byteland has n cities & m roads between them. Task: design a round trip that begins in a city, goes through 2 or more other cities, & finally returns to starting city. Every intermediate city on the route has to be distinct.*

Input. *The 1st input line has 2 integers n, m: the number of cities & roads. The cities are numbered $1, 2, \ldots, n$. Then, there are m lines describing the roads. Each line has 2 integers a, b: there is a road between those cities. Every road is between 2 different cities, & there is at most 1 road between any 2 cities.*

Output. *1st print an integer k: the number of cities on the route. Then print k cities in order they will be visited. You can print any valid solution. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $1 \leq n \leq 10^5, 1 \leq m \leq 2 \cdot 10^5, 1 \leq a, b \leq n$.

Sample.

build_team.inp	build_team.out
5 6 1 3 1 2 5 3 1 5 2 4 4 5	4 3 5 1 3

Problem 104 (CSES Problem Set/monsters). *You & some monsters are in a labyrinth. When taking a step to some direction in the labyrinth, each monster may simultaneously take 1 as well. Goal: reach 1 of the boundary squares without ever sharing a square with a monster. Task: find out if your goal is possible, & if it is, print a path that you can follow. Your plan has to work in any situation; even if the monsters know your path beforehand.*

Input. *The 1st input line has 2 integers n, m: the height & width of the map. After this there are n lines of m characters describing the map. Each character is* `.` *(floor),* `#` *(wall),* `A` *(start), or* `M` *(monster). There is exactly 1* `A` *in the input.*

Output. *1st print* `YES` *if your goal is possible, &* `NO` *otherwise. If your goal is possible, also print an example of a valid path (the length of the path & its description using characters* `D, U, L, R`*). You can print any path, as long as its length is at most mn steps.*

Constraints. $1 \leq m, n \leq 10^3$.

Sample. *Input:*

```
5 8
########
#M..A..#
#.#.M#.#
#M#..#..
#.######
```

Output:

```
YES
5
RRDDR
```

Problem 105 (CSES Problem Set/shortest routes I). *There are n cities & m flight connections between them. Task: determine the length of the shortest route from Syrjälä to every city.*

Input. *The 1st input line has 2 integers n, m: the number of cities & flight connections. The cities are numbered $1, 2, \ldots, n$, & city 1 is Syrjälä. After that, there are m lines describing the flight connections. Each line has 3 integers a, b, c: a flight begins at city a, ends at city b, & its length is c. Each flight is a 1-way flight. You can assume that it is possible to travel from Syrjälä to all other cities.*

Output. *Print n integers: the shortest route lengths from Syrjälä to cities $1, 2, \ldots, n$.*

Constraints. $1 \leq n \leq 10^5, 1 \leq m \leq 2 \cdot 10^5, 1 \leq a, b \leq n, 1 \leq c \leq 10^9$.

Sample.

shortest_route_I.inp	shortest_route_I.out
3 4 1 2 6 1 3 2 3 2 3 1 3 4	0 5 2

Problem 106 (CSES Problem Set/shortest routes II). *There are n cities & m roads between them. Task: process q queries where you have to determine the length of the shortest route between 2 given cities.*

Input. *The 1st input line has 3 integers n, m, q: the number of cities, roads, & queries. Then, there are m lines describing the roads. Each line has 3 integers a, b, c: there is a road between cities a & b whose length is c. All roads are 2-way roads. Finally, there are q lines describing the queries. Each line has 2 integers a, b: determine the length of the shortest route between cities a & b.*

Output. *Print the length of the shortest route for each query. If there is no route, print* `-1` *instead.*

Constraints. $1 \le n \le 500, 1 \le m \le n^2, 1 \le q \le 10^5, 1 \le a, b \le n, 1 \le c \le 10^9$.

Sample.

shortest_route_II.inp	shortest_route_II.out
4 3 5 1 2 5 1 3 9 2 3 3 1 2 2 1 1 3 1 4 3 2	5 5 8 -1 3

Problem 107 (CSES Problem Set/high score). *You play a game consisting of n rooms & m tunnels. Your initial score is 0, & each tunnel increases your score by x where x may be both positive or negative. You may go through a tunnel several times. Task: walk from room 1 to room n. What is the maximum score you can get?*

Input. *The 1st input line has 2 integers n, m: the number of rooms & tunnels. The rooms are numbered $1, 2, \ldots, n$. Then, there are m lines describing the tunnels. Each line has 3 integers a, b, x: the tunnel starts at room a, ends at room b, & it increases your score by x. All tunnels are 1-way tunnels. You can assume that it is possible to get from room 1 to room n.*

Output. *Print 1 integer: the maximum score you can get. However, if you can get an arbitrarily large score, print* `-1`.

Constraints. $1 \le n \le 2500, 1 \le m \le 5000, 1 \le a, b \le n, -10^9 \le x \le 10^9$.

Sample.

high_score.inp	high_score.out
4 5 1 2 3 2 4 -1 1 3 -2 3 4 7 1 4 4	5

Problem 108 (CSES Problem Set/flight discount). *Task: find a minimum-price flight route from Syrjälä to Metsälä. You have 1 discount coupon, using which you can halve the price of any single flight during the route. However, you can only use the coupon once. When you use the discount coupon for a flight whose price is x, its price becomes $\lfloor \frac{x}{2} \rfloor$.*

Input. *The 1st input line has 2 integers n, m: the number of cities & flight connections. The cities are numbered $1, 2, \ldots, n$. City 1 is Syrjälä, & city n is Metsälä. After this there are m lines describing the flights. Each line has 3 integers a, b, c: a flight begins at city a, ends at city b, & its price is c. Each flight is unidirectional. You can assume that it is always possible to get from Syrjälä to Metsälä.*

Output. *Print 1 integer: the price of the cheapest route from Syrjälä to Metsälä.*

Constraints. $1 \le n \le 10^5, 1 \le m \le 2 \cdot 10^5, 1 \le a, b \le n, 1 \le c \le 10^9$.

Sample.

flight_discount.inp	flight_discount.out
3 4 1 2 3 2 3 1 1 3 7 2 1 5	2

Problem 109 (CSES Problem Set/cycle finding). *You are given a directed graph, & task: find out if it contains a negative cycle, & also give an example of such a cycle.*

Input. *The 1st input line has 2 integers n, m: the number of nodes & edges. The nodes are numbered $1, 2, \ldots, n$. After this, the input has m lines describing the edges. Each line has 3 integers a, b, c: there is an edge from node a to node b whose length is c.*

Output. *If the graph contains a negative cycle, print 1st* `YES`*, & then the nodes in the cycle in their correct order. If there are several negative cycles, you can print any of them. If there are no negative cycles, print* `NO`.

Constraints. $1 \le n \le 2500, 1 \le m \le 5000, 1 \le a, b \le n, -10^9 \le c \le 10^9$.

Sample.

cycle_finding.inp	cycle_finding.out
4 5 1 2 1 2 4 1 3 1 1 4 1 -3 4 3 -2	YES 1 2 4 1

Problem 110 (CSES Problem Set/flight routes). *Find the $k \in \mathbb{N}^\star$ shortest flight routes from Syrjälä to Metsälä. A route can visit the same city several times. Note that there can be several routes with the same price & each of them should be considered*

Input. *The 1st input line has 3 integers $n, m, k \in \mathbb{N}^\star$: the number of cities, the number of flights, & the parameter k. The cities are numbered $1, 2, \ldots, n$. City 1 is Syrjälä, & city n is Metsälä. After this, the input has m lines describing the flights. Each line has 3 integers a, b, c: a flight begins at city a, ends at city b, & its price is c. All flights are 1-way flights. You may assume that there are at least k distinct routes from Syrjälä to Metsälä.*

Output. *Print k integers: the prices of the k cheapest routes sorted according to their prices.*

Constraints. $n \in \overline{2, 10^5}, m \in [2 \cdot 10^5], a, b \in [n], c \in [10^9], k \in [10]$.

Sample.

flight_route.inp	flight_route.out
4 6 3 1 2 1 1 3 3 2 3 2 2 4 6 3 2 8 3 4 1	4 4 7

Explanation. *The cheapest routes are $1 \to 3 \to 4$ (price 4), $1 \to 2 \to 3 \to 4$ (price 4), & $1 \to 2 \to 4$ (price 7).*

Problem 111 (CSES Problem Set/round trip II). *Byteland has $n \in \mathbb{N}^\star$ & $m \in \mathbb{N}^\star$ flight connections. Design a round trip that begins in a city, goes through 1 or more other cities, & finally returns to the starting city. Every intermediate city on the route has to be distinct.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & flights. The cities are numbered $1, 2, \ldots, n$. Then, there are m lines describing the flights. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a flight connection from city a to city b. All connections are 1-way flights from a city to another city.*

Output. *1st print an integer $k \in \mathbb{N}^\star$: the number of cities on the route. Then print k cities in the order they will be visited. You can print any valid solution. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

round_trip_II.inp	round_trip_II.out
4 5 1 3 2 1 2 4 3 2 3 4	4 2 1 3 2

Problem 112 (CSES Problem Set/course schedule)**.** *You have to complete $n \in \mathbb{N}^\star$ courses. There are $m \in \mathbb{N}^\star$ requirements of the form "course a has to be completed before course b". Find an order in which you can complete the courses.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of courses & requirements. The courses are numbered $1, 2, \ldots, n$. After this, there are m lines describing the requirements. Each line has 2 integers $a, b \in \mathbb{N}^\star$: course a has o be completed before course b.*

Output. *Print an order in which you can complete the courses. You can print any valid order that includes all the courses. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

course_schedule.inp	course_schedule.out
5 3 1 2 3 1 4 5	3 4 1 5 2

Problem 113 (CSES Problem Set/longest flight route)**.** *Uolevi has won a contest, & the prize is a free flight trip that can consist of 1 or more flights through cities. Of course, Uolevi wants to choose a trip that has as many cities as possible. Uolevi wants to fly from Syrjälä to Lehmälä so that he visits the maximum number of cities. You are given the list of possible flights, & you know that there are no directed cycles in the flight network.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & flights. The cities are numbered $1, 2, \ldots, n$. City 1 is Syrjälä, & city n is Lehmälä. After this, there are m lines describing the flights. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a flight from city a to city b. Each flight is a 1-way flight.*

Output. *1st print the maximum number of cities on the route. After this, print the cities in the order they will be visited. You can print any valid solution. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in \overline{2, 10^5}], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

longest_flight_route.inp	longest_flight_route.out
5 5 1 2 2 5 1 3 3 4 4 5	4 1 3 4 5

Problem 114 (CSES Problem Set/game routes)**.** *A game has $n \in \mathbb{N}^\star$ levels, connected by $m \in \mathbb{N}^\star$ teleporters. Get from level 1 to level n. The game has been designed so that there are no directed cycles in the underlying graph. In how many ways can you complete the game?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of levels & teleporters. The levels are numbered $1, 2, \ldots, n$. After this, there are m lines describing the teleporters. Each line has 2 integers a, b: there is a teleporter from level a to level b.*

Output. *Print* 1 *integer: the number of ways you can complete the game. Since the result may be large, print it modulo* $10^9 + 7$.

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

game_route.inp	game_route.out
4 5 1 2 2 4 1 3 3 4 1 4	3

Problem 115 (CSES Problem Set/investigation). *You are going to travel from Syrjälä to Lehmälä by plane. You would like to find answers to the following questions:*

- *What is the minimum price of such a route?*
- *How many minimum-price routes are there? (modulo* $10^9 + 7$*)?*
- *What is the minimum number of flights in a minimum-price route?*
- *What is the maximum number of flights in a minimum-price route?*

Input. *The 1st input line has* 2 *integers* $n, m \in \mathbb{N}^\star$*: the number of cities & the number of flights. The cities are numbered* $1, 2, \ldots, n$*. City* 1 *is Syrjälä, & city* n *is Lehmälä. After this, there are* m *lines describing the flights. Each line has* 3 *integers* $a, b, c \in \mathbb{N}^\star$*: there is a flight from city* a *to city* b *with price* c*. All flights are* 1*-way flights. You may assume that there is a route from Syrjälä to Lehmälä.*

Output. *Print* 4 *integers according to the problem statement.*

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n], c \in [10^9]$.

Sample.

investigation.inp	investigation.out
4 5 1 4 5 1 2 4 2 4 5 1 3 2 3 4 3	5 2 1 2

Problem 116 (CSES Problem Set/planets queries I). *You are playing a game consisting of* $n \in \mathbb{N}^\star$ *planets. Each planet has a teleporter to another planet (or the planet itself). process* $q \in \mathbb{N}^\star$ *queries of the form: when you begin on planet* x *& travel through* $k \in \mathbb{N}^\star$ *teleporters, which planet will you reach?*

Input. *The 1st input line has* 2 *integers* $n, q \in \mathbb{N}^\star$*: the number of planets & queries. The planets are numbered* $1, 2, \ldots, n$*. The 2nd line has* n *integers* $t_1, t_2, \ldots, t_n$*: for each planet, the destination of the teleporter. It is possible that* $t_i = i$*. Finally, there are* q *lines describing the queries. Each line has* 2 *integers* $x, k \in \mathbb{N}^\star$*: you start on planet* x *& travel through* k *teleporters.*

Output. *Print the answer to each query.*

Constraints. $n, q \in [2 \cdot 10^5], x \in [n], k \in \overline{0, 10^9}, t_i \in [n], \forall i \in [n]$.

Sample.

planet_query_I.inp	planet_query_I.out
4 3 2 1 1 4 1 2 3 4 4 1	1 2 4

Problem 117 (CSES Problem Set/planet queries II). *You are playing a game consisting of $n \in \mathbb{N}^\star$ planets. Each planet has a teleporter to another planet (or the planet itself). You have to process $q \in \mathbb{N}^\star$ queries of the form: You are now on planet a & want to reach planet b. What is the minimum number of teleportations?*

Input. *The 1st input line has 2 integers $n, q \in \mathbb{N}^\star$: the number of the number of planets & queries. The planets are numbered $1, 2, \ldots, n$. The 2nd line contains n integers $t_1, t_2, \ldots, t_n$: for each planet, the destination of the teleporter. Finally, there are q lines describing the queries. Each line has 2 integers $a, b \in \mathbb{N}^\star$: you are now on planet a & want to reach planet b.*

Output. *For each query, print the minimum number of teleportations. If it is not possible to reach the destination, print -1.*

Constraints. $n, q \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

planet_query_II.inp	planet_query_II.out
5 3 2 3 2 3 2 1 2 1 3 1 4	1 2 -1

Problem 118 (CSES Problem Set/planets cycles). *You are playing a game consisting of $n \in \mathbb{N}^\star$ planets. Each planet has a teleporter to another planet (or the planet itself). You start on a planet & then travel through teleporters until you reach a planet that you have already visited before. Calculate for each planet the number of teleportations there would be if you started on that planet.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of planets. The planets are numbered $1, 2, \ldots, n$. The 2nd line has n integers $t_1, t_2, \ldots, t_n$: for each planet, the destination of the teleporter. It is possible that $t_i = i$.*

Output. *Print n integers according to the problem statement.*

Constraints. $n \in [2 \cdot 10^5], t_i \in [n], \forall i \in [n]$.

Sample.

planet_cycle.inp	planet_cycle.out
5 2 4 3 1 4	3 3 1 3 4

Problem 119 (CSES Problem Set/road reparation). *There are $n \in \mathbb{N}^\star$ cities & $m \in \mathbb{N}^\star$ roads between them. Unfortunately, the condition of the roads is so poor that they cannot be used. Repair some of the roads so that there will be a decent route between any 2 cities. For each road, you know its reparation cost, & you should find a solution where the total cost is as small as possible.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & roads. The cities are numbered $1, 2, \ldots, n$. Then, there are m lines describing the roads. Each line has 3 integers $a, b, c \in \mathbb{N}^\star$: there is a road between cities a & b, & its reparation cost is c. All roads are 2-way roads. Every road is between 2 different cities, & there is at most 1 road between 2 cities.*

Output. *Print 1 integer: the minimum total reparation cost. However, if there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n], c \in [10^9]$.

Sample.

road_reparation.inp	road_reparation.out
5 6 1 2 3 2 3 5 2 4 2 3 4 8 5 1 7 5 4 4	14

Problem 120 (CSES Problem Set/road construction). *There are $n \in \mathbb{N}^\star$ cities & initially no roads between them. However, every day a new road will be constructed, & there will be a total of $m \in \mathbb{N}^\star$ roads. A component is a group of cities where there is a route between any 2 cities using the roads. After each day, find the number of components & the size of the largest component.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & roads. The cities are numbered $1, 2, \ldots, n$. Then, there are m lines describing the roads. Each line has 2 integers $a, b \in \mathbb{N}^\star$: a new road is constructed between cities a, b. You may assume that every road will be constructed between 2 different cities.*

Output. *Print m lines: the required information after each day.*

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

road_construction.inp	road_construction.out
5 3 1 2 1 3 4 5	4 2 3 3 2 3

Problem 121 (CSES Problem Set/flight routes check)**.** *There are $n \in \mathbb{N}^\star$ cities & $m \in \mathbb{N}^\star$ flight connections. Check if you can travel from any city to any other city using the available flights.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & flights. The cities are numbered $1, 2, \ldots, n$. After this, there are m lines describing the flights. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a flight from city a to city b. All flights are 1-way flights.*

Output. *Print* `YES` *if all routes are possible, &* `NO` *otherwise. In the latter case also print 2 cities $a, b \in \mathbb{N}^\star$ s.t. you cannot travel from city a to city b. If there are several possible solutions, you can print any of them.*

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

flight_routes_check.inp	flight_routes_check.out
4 5 1 2 2 3 3 1 1 4 3 4	NO 4 2

Problem 122 (CSES Problem Set/planets & kingdoms)**.** *A game has $n \in \mathbb{N}^\star$ planets, connected by $m \in \mathbb{N}^\star$ teleporters. 2 planets a, b belong to the same kingdom exactly when there is a route both from a to b & from b to a. Determine for each planet its kingdom.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of planets & teleporters. The planets are numbered $1, 2, \ldots, n$. After this, there are m lines describing the teleporters. Each line has 2 integers $a, b \in \mathbb{N}^\star$: you can travel from planet a to planet b through a teleporter.*

Output. *1st print an integer $k \in \mathbb{N}^\star$: the number of kingdoms. After this, print for each planet a kingdom label between 1 & k. You can print any valid solution.*

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

planet_kingdom.inp	planet_kingdom.out
5 6 1 2 2 3 3 1 3 4 4 5 5 4	2 1 1 1 2 2

Problem 123 (CSES Problem Set/giant pizza)**.** *Uolevi's family is going to order a large pizza & eat it together. A total of $n \in \mathbb{N}^\star$ family members will join the order, & there are $m \in \mathbb{N}^\star$ possible toppings. The pizza may have any number of toppings. Each family member gives 2 wishes concerning the toppings of the pizza. The wishes are of the form "topping x is good/bad". Choose the toppings so that at least 1 wish from everybody becomes true (a good topping is included in the pizza or a bad topping is not included).*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of family members & toppings. The toppings are numbered $1, 2, \ldots, m$. After this, there are n lines describing the wishes. Each line has 2 wishes of the form "$+x$" (topping x is good) or "$-x$" (topping x is bad).*

Output. *Print a line with m symbols: for each topping + if it is included & - if it is not included. You can print any valid solution.*

Constraints. $m, n \in [10^5], x \in [m]$.

Sample.

giant_pizza.inp	giant_pizza.out
3 5 + 1 + 2 - 1 + 3 + 4 - 2	- + + + -

Problem 124 (CSES Problem Set/coin collector)**.** *A game has $n \in \mathbb{N}^\star$ rooms & $m \in \mathbb{N}^\star$ tunnels between them. Each room has a certain number of coins. What is the maximum number of coins you can collect while moving through the tunnels when you can freely choose your starting & ending room?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of rooms & tunnels. The rooms are numbered $1, 2, \ldots, n$. Then, there are n integers $k_1, k_2, \ldots, k_n$: the number of coins in each room. Finally, there are m lines describing the tunnels. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a tunnel from room a to room b. Each tunnel is a 1-way tunnel.*

Output. *Print 1 integer: the maximum number of coins you can collect.*

Constraints. $n \in [10^5], m \in [2 \cdot 10^5], a, b \in [n], k_i \in [10^9], \forall i \in [n]$.

Sample.

coin_collector.inp	coin_collector.out
4 4 4 5 2 7 1 2 2 1 1 3 2 4	16

Problem 125 (CSES Problem Set/mail delivery)**.** *Deliver mail to the inhabitants of a city. For this reason, you want to find a route whose starting & ending point are the post office, & that goes through every street exactly once.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of crossings & streets. The crossings are numbered $1, 2, \ldots, n$, & he post office is located at crossing 1. After that, there are m lines describing the streets. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a street between crossings a, b. All streets are 2-way streets. Every street is between 2 different crossings, & there is at most 1 street between 2 crossings.*

Output. *Print all the crossings on the route in the order you will visit them. You can print any valid solution. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in \overline{2, 10^5}, m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

mail_delivery.inp	mail_delivery.out
6 8 1 2 1 3 2 3 2 4 2 6 3 5 3 6 4 5	1 2 6 3 2 4 5 3 1

Problem 126 (CSES Problem Set/de Bruijn sequence)**.** *Construct a minimum-length bit string that contains all possible substrings of length $n \in \mathbb{N}^\star$, e.g., when $n = 2$, the string 00110 is a valid solution, because its substrings of lengt 2 are $00, 01, 10, 11$.*

Input. *The only input line has an integer* $n \in \mathbb{N}^\star$.

Output. *Print a minimum-length bit string that contains all substrings of length* n. *You can print any valid solution.*

Constraints. $n \in [15]$.

Sample.

de_Bruijn_sequence.inp	de_Bruijn_sequence.out
2	00110

Problem 127 (CSES Problem Set/teleporters path). *A game has* $n \in \mathbb{N}^\star$ *levels &* $m \in \mathbb{N}^\star$ *teleportes between them. You win the game if you move from level* 1 *to level* n *using every teleporter exactly once. Can you win the game, & what is a possible way to do it?*

Input. *The 1st input line has* 2 *integers* $n, m \in \mathbb{N}^\star$*: the number of levels & teleporters. The levels are numbered* $1, 2, \ldots, n$. *Then, there are* m *lines describing the teleporters. Each line has* 2 *integers* a, b*: there is a teleporter from level* a *to level* b. *You can assume that each pair* (a, b) *in the input is distinct.*

Output. *Print* $m + 1$ *integers: the sequence in which you visit the levels during the game. You can print any valid solution. If there are no solutions, print* `IMPOSSIBLE`.

Constraints. $n \in \overline{2, 10^5}, m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

teleporter_path.inp	teleporter_path.out
5 6 1 2 1 3 2 4 2 5 3 1 4 2	1 3 1 2 4 2 5

Problem 128 (CSES Problem Set/Hamiltonian flights). *There are* $n \in \mathbb{N}^\star$ *cities &* $m \in \mathbb{N}^\star$ *flight connections between them. You want to travel from Syrjälä to Lehmälä so that you visit each city exactly once. How many possible routes are there?*

Input. *The 1st input line has* 2 *integers* $n, m \in \mathbb{N}^\star$*: the number of cities & flights. The cities are numbered* $1, 2, \ldots, n$. *City* 1 *is Syrjälä, & city* n *is Lehmälä. Then, there are* m *lines describing the flights. Each line has* 2 *integers* $a, b \in \mathbb{N}^\star$*: there is a flight from city* a *to city* b. *All flights are* 1*-way flights.*

Output. *Print* 1 *integer: the number of routes modulo* $10^9 + 7$.

Constraints. $n \in \overline{2, 20}, m \in [n^2], a, b \in [n]$.

Sample.

Hamiltonian_flight.inp	Hamiltonian_flight.out
4 6 1 2 1 3 2 3 3 2 2 4 3 4	2

Problem 129 (CSES Problem Set/knight's tour). *Given a starting position of a knight on an* 8×8, *find a sequence of moves s.t. it visits every square exactly once. On each move, the knight may either move* 2 *steps horizontally &* 1 *step vertically, or* 1 *step horizontally &* 2 *steps vertically.*

Input. *The only input line has* 2 *integers* $x, y \in \mathbb{N}^\star$*: the knight's starting position.*

Output. *Print a grid that shows how the knight moves (according to the example). You can print any valid solution.*

Constraints. $x, y \in [8]$.

Sample.

knight_tour.inp	knight_tour.out
2 1	8 1 10 13 6 3 20 17 11 14 7 2 19 16 23 4 26 9 12 15 24 5 18 21 49 58 25 28 51 22 33 30 40 27 50 59 32 29 52 35 57 48 41 44 37 34 31 62 42 39 46 55 60 63 36 53 47 56 43 38 45 54 61 64

Problem 130 (CSES Problem Set/download speed). *Consider a network consisting of $n \in \mathbb{N}^\star$ computers & $m \in \mathbb{N}^\star$ connections. Each connection specifies how fast a computer can send data to another computer.* Kotivalo *wants to download some data from a server. What is the maximum speed he can do this, using the connections in the network?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of computers & connections. The computers are numbered $1, 2, \ldots, n$. Computer 1 is the server & computer n is Kotivalo's computer. After this, there are m lines describing the connections. Each line has 3 integers $a, b, c \in \mathbb{N}^\star$: computer a can send data to computer b at speed c.*

Output. *Print 1 integer: the maximum speed Kotivalo can download data.*

Constraints. $n \in [500], m \in [1000], a, b \in [n], c \in [10^9]$.

Sample.

download_speed.inp	download_speed.out
4 5 1 2 3 2 4 2 1 3 4 3 4 5 4 1 3	6

Problem 131 (CSES Problem Set/police chase). *Kaaleppi has just robbed a bank & is now heading to the harbor. However, the police wants to stop him by closing some streets of the city. What is the minimum number of streets that should be closed so that there is no route between the bank & the harbor?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of crossings & streets. The crossings are numbered $1, 2, \ldots, n$. The bank is located at crossing 1, & the harbor is located at crossing n. After this, there are m lines that describing the streets. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a street between crossings a, b. All streets are 2-way streets, & there is at most 1 street between 2 crossings.*

Output. *1st print 1 integer k: the minimum number of streets that should be closed. After this, print k lines describing the streets. You can print any valid solution.*

Constraints. $n \in \overline{2, 500}, m \in [1000], a, b \in [n]$.

Sample.

police_chase.inp	police_chase.out
4 5 1 2 1 3 2 3 3 4 1 4	2 3 4 1 4

Problem 132 (CSES Problem Set/school dance). *There are $n \in \mathbb{N}^\star$ boys & $m \in \mathbb{N}^\star$ girls in a school. Next week a school dance will be organized. A dance pair consists of a boy & a girl, & there are $k \in \mathbb{N}^\star$ potential pairs. Find out the maximum number of dance pairs & show how this number can be achieved.*

Input. *The 1st input line has 3 integers $n, m, k \in \mathbb{N}^\star$: the number of boys, girls, & potential pairs. The boys are numbered $1, 2, \ldots, n$ & the girls are numbered $1, 2, \ldots, m$. After this, there are k lines describing the potential pairs. Each line has 2 integers $a, b \in \mathbb{N}^\star$: boy a & girl b are willing to dance together.*

Output. *1st print 1 integer r: the maximum number of dance pairs. After this, print r lines describing the pairs. You can print any valid solution.*

Constraints. $m, n \in [500], k \in [1000], a \in [n], b \in [m]$.

Sample.

school_dance.inp	school_dance.out
3 2 4 1 1 1 2 2 1 3 1	2 1 2 3 1

Problem 133 (CSES Problem Set/distinct routes)**.** *A game consists of $n \in \mathbb{N}^\star$ rooms & $m \in \mathbb{N}^\star$ teleporters. At the beginning of each day, you start in room 1 & you have to reach room n. You can use each teleporter at most once during the game. How many days can you play if you choose your routes optimally?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of rooms & teleporters. The rooms are numbered $1, 2, \ldots, n$. After this, there are m lines describing the teleporters. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a teleporter from room a to room b. There are no 2 teleporters whose starting & ending room are the same.*

Output. *1st print an integer k: the maximum number of days you can play the game. Then, print k route descriptions according to the example. You can print any valid solution.*

Constraints. $n \in \overline{2, 500}, m \in [1000], a, b \in [n]$.

Sample.

distinct_route.inp	distinct_route.out
6 7 1 2 1 3 2 6 3 4 3 5 4 6 5 6	2 3 1 2 6 4 1 3 4 6

Problem 134 (IMO2007P3)**.** *In a mathematical competition some competitors are friends. Friendship is always mutual. Call a group of competitors a* clique *if each 2 of them are friends. (In particular, any group of < 2 competitors is a clique.) The number of members of a clique is called its* size*.*

Give that, in this competition, the largest size of a clique is even, prove that the competitors can be arranged in 2 rooms s.t. the largest size of a clique contained in 1 room is the same as the largest size of a clique contained in the other room.

Bài toán 34 ([VL24], 3., p. 10, IMO2007P3)**.** *Trong 1 kỳ thi học sinh giỏi Toán, có vài thí sinh là bạn bè của nhau. Quan hệ bạn bè luôn là quan hệ 2 chiều. Gọi 1 nhóm các thí sinh là* nhóm bạn bè *nếu như 2 người bất kỳ trong nhóm này đều là bạn bè của nhau. (1 nhóm tùy ý có < 2 thí sinh cũng vẫn được coi là 1 nhóm bạn bè). Số lượng các thí sinh của 1 nhóm bạn bè được gọi là cỡ của nó. Cho biết trong kỳ thi này, cỡ của 1 nhóm bạn bè có nhiều người nhất là 1 số chẵn. Chứng minh: có thể xếp tất cả các thí sinh vào 2 phòng sao cho cỡ của nhóm bạn bè có nhiều người nhất trong phòng này cũng bằng cỡ của nhóm bạn bè có nhiều người nhất trong phòng kia.*

Bài toán 35 (CP version of IMO2007P3)**.** *Cho $n \in \mathbb{N}^\star$ người, được đánh số $1, 2, \ldots, n$, tạo thành 1 đồ thị đơn vô hướng hữu hạn (finite undirected simple graph) $G = ([n], E)$. $e_{ij} \in E \Leftrightarrow$ người i & người j quen lẫn nhau, $\forall i, j \in [n]$. 1* nhóm bạn bè *được định nghĩa là 1 đồ thị con đầy đủ (complete subgraph $G' = (V', E')$ với $V' \subset [n], |E'| = \frac{|V'|(|V'|-1)}{2}$), i.e., 2 đỉnh bất kỳ của G' được nối với nhau. Viết chương trình* C/C++, Pascal, Python *để tính bậc lớn nhất có thể có của 1 nhóm bạn bè. Sau đó tìm cách phân hoạch tập $[n]$ thành 2 tập con (tương ứng 2 phòng R_1, R_2) sao cho bậc lớn nhất của thể của 1 nhóm bạn bè trong mỗi phòng bằng nhau.*

Input. *Dòng 1 chứa số $n \in \mathbb{N}^\star$. Dòng thứ $i \in [n]$ trong n dòng tiếp theo chứa các đỉnh nối với đỉnh i, i.e., tập láng giềng (neighborhood) của đỉnh i: $N(i) := \{j \in [n]; e_{ij} \in E\}$.*

Output. *Xuất ra dòng đầu tiên gồm 1 số nguyên dương: cỡ lớn nhất của nhóm bạn. 2 dòng tiếp theo chứa các đỉnh trong phòng R_1, R_2 thỏa mãn. Dòng cuối chứa cỡ lớn nhất chung của 2 phòng R_1, R_2.*

Sample.

IMO2007_P3.inp	IMO2007_P3.out
5 2 3 4 1 3 4 1 2 4 5 1 2 3 5	4 1 3 2 4 5 2

Solution. C++:

1. DXH's C++: IMO2007 P3:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

// Class đại diện cho từng sinh viên
class sv {
    protected:
    int id;
    vector<int> friends;

    public:
    // Constructor
    sv(int id) : id(id) {}

    // Phương thức thêm bạn bè
    void addFriend(int friendId) {
        friends.push_back(friendId);
    }

    // Getter: để lấy id từ bên ngoài khi cần
    int getId() const { return id; }

    // Getter: để lấy danh sách bạn bè
    vector<int> getFriends() const { return friends; }
};

// Class quản lý toàn bộ hệ thống kết nối sinh viên
class sodoketnoisv {
    private:
    map<int, sv*> students;

    public:
    void addsv(int id) {
        students[id] = new sv(id);
    }

    void addbanbe(int id, int friendId) {
        students[id]->addFriend(friendId);
        students[friendId]->addFriend(id);
    }

    // Getter để lấy toàn bộ danh sách sinh viên ra
    map<int, sv*> getStudents() {
        return students;
    }
};

```

```
48 // Hàm chia phòng
49 void chiaphong(sodoketnoisv &sinhvien) {
50     // Giả sử tìm được clique sẵn
51     vector<int> clique = {203, 204, 205, 206};
52 
53     vector<int> roomA, roomB;
54 
55     // Chia clique vào 2 phòng
56     for (int i = 0; i < static_cast<int>(clique.size()); i++) {
57         if (i < static_cast<int>(clique.size() / 2))
58         roomA.push_back(clique[i]);
59         else
60         roomB.push_back(clique[i]);
61     }
62 
63     // Tìm những sinh viên còn lại (vệ tinh)
64     map<int, sv*> allStudents = sinhvien.getStudents();
65     vector<int> others;
66 
67     for (auto pair : allStudents) {
68         int id = pair.first;
69         if (find(clique.begin(), clique.end(), id) == clique.end()) {
70             others.push_back(id);
71         }
72     }
73 
74     // Phân bổ vệ tinh vào 2 phòng
75     for (int id : others) {
76         int countA = 0, countB = 0;
77         vector<int> friends = sinhvien.getStudents()[id]->getFriends();
78 
79         for (int friendId : friends) {
80             if (find(roomA.begin(), roomA.end(), friendId) != roomA.end())
81             countA++;
82             if (find(roomB.begin(), roomB.end(), friendId) != roomB.end())
83             countB++;
84         }
85 
86         if (countA <= countB)
87         roomA.push_back(id);
88         else
89         roomB.push_back(id);
90     }
91 
92     // In kết quả chia phòng ra màn hình (kiểm tra)
93     cout << "Room A: ";
94     for (int id : roomA)
95     cout << id << " ";
96     cout << endl;
97 
98     cout << "Room B: ";
99     for (int id : roomB)
100    cout << id << " ";
101    cout << endl;
102 }
103 
104 int main() {
105     ios::sync_with_stdio(0);
106 
```

```
    sodoketnoisv sinhvien;

    // Thêm sinh viên
    sinhvien.addsv(203);
    sinhvien.addsv(204);
    sinhvien.addsv(205);
    sinhvien.addsv(206);
    sinhvien.addsv(207);
    sinhvien.addsv(208);

    // Thêm các cặp bạn bè (ví dụ)
    sinhvien.addbanbe(203, 204);
    sinhvien.addbanbe(204, 205);
    sinhvien.addbanbe(205, 206);
    sinhvien.addbanbe(203, 205);
    sinhvien.addbanbe(203, 206);
    sinhvien.addbanbe(204, 206);
    sinhvien.addbanbe(207, 203);
    sinhvien.addbanbe(208, 204);

    // Gọi hàm chia phòng
    chiaphong(sinhvien);
}
```

2. DAK's C++: IMO2007 P3:

```
#include <bits/stdc++.h>
using namespace std;
const int N = 65;
int n;
bool adj[N][N];
vector<int> best_clique_so_far;

void find_max_clique_recursive(const vector<int>& potential_nodes, vector<int> current_clique) {
    if (current_clique.size() > best_clique_so_far.size())
    best_clique_so_far = current_clique;

    for (size_t i = 0; i < potential_nodes.size(); ++i) {
        int v = potential_nodes[i];
        if (current_clique.size() + (potential_nodes.size() - i) <= best_clique_so_far.size())
        return;

        vector<int> new_potential_nodes;
        for (size_t j = i + 1; j < potential_nodes.size(); ++j)
        if (adj[v][potential_nodes[j]])
        new_potential_nodes.push_back(potential_nodes[j]);

        vector<int> next_clique = current_clique;
        next_clique.push_back(v);
        find_max_clique_recursive(new_potential_nodes, next_clique);
    }
}

vector<int> get_max_clique(const vector<int>& nodes) {
    if (nodes.empty())
    return {};
    best_clique_so_far.clear();
    find_max_clique_recursive(nodes, {});
    return best_clique_so_far;
}
```

```
35
36  void print_vector(const vector<int>& vec) {
37      for (size_t i = 0; i < vec.size(); ++i)
38      cout << vec[i] << (i == vec.size() - 1 ? "" : " ");
39      cout << '\n';
40  }
41
42  int main() {
43      ios_base::sync_with_stdio(false);
44      cin.tie(NULL);
45      cin >> n;
46      cin.ignore();
47
48      for (int i = 1; i <= n; ++i) {
49          string line;
50          getline(cin, line);
51          stringstream ss(line);
52          int neighbor;
53          while (ss >> neighbor)
54          adj[i][neighbor] = true;
55      }
56
57      vector<int> all_nodes(n);
58      iota(all_nodes.begin(), all_nodes.end(), 1);
59
60      vector<int> C = get_max_clique(all_nodes);
61      cout << C.size() << '\n';
62
63      vector<int> room1 = C;
64      vector<int> room2;
65
66      sort(C.begin(), C.end());
67      sort(all_nodes.begin(), all_nodes.end());
68
69      set_difference(all_nodes.begin(), all_nodes.end(),
70      C.begin(), C.end(),
71      back_inserter(room2));
72
73      for (size_t i = 0; i <= C.size(); ++i) {
74          sort(room1.begin(), room1.end());
75          sort(room2.begin(), room2.end());
76
77          int size1 = get_max_clique(room1).size();
78          int size2 = get_max_clique(room2).size();
79
80          if (size1 == size2) {
81              print_vector(room1);
82              print_vector(room2);
83              cout << size1 << '\n';
84              return 0;
85          }
86
87          if (i < C.size()) {
88              int node_to_move = C[i];
89
90              room1.erase(remove(room1.begin(), room1.end(), node_to_move), room1.end());
91              room2.push_back(node_to_move);
92          }
93      }
```

```
94
95      return 1;
96  }
```

□

Chương 11

Range Queries – Truy Vấn Phạm Vi

Chương 12

Tree Algorithms – Thuật Toán Trên Cây

Chương 13

Mathematics

Problem 135 (CSES/Josephus Queries, https://cses.fi/problemset/task/2164). *Consider a game where there are $n \in \mathbb{N}^\star$ children, numbered $1, 2, \ldots, n$, in a circle. During the game, every 2nd child is removed from circle, until there are no children left. Task: process q queries of the form: "when there are n children, who is the kth child that will be removed?"*

- Input. *The 1st input line has an integer q: the number of queries. After this, there are q lines that describe the queries. Each line has 2 integers n, k: the number of children & the position of the child.*
- Output. *Print q integers: the answer for each query.*

  It seems to me that Jack97 (nickname: `abortion_grandmaster`) proposed this problem.
  Codes:
- C++: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C++/gcd_Pascal_triangle.cpp.

Problem 136 (CSES/Dice Probability, https://cses.fi/problemset/task/1725). *Throw a dice $n \in \mathbb{N}^\star$ times, & every throw produces an outcome between 1 & 6. What is the probability that the sum of outcomes is between $a, b \in \mathbb{Z}$?*

- Input. *The only input line contains 3 integers $n, a, b \in \mathbb{N}^\star$.*
- Output. *Print probability rounded to 6 decimal places (rounding half to even).*
- Constraints. $1 \le n \le 100, 1 \le a \le b \le 6n$.
- Example. *Input:* `2 9 10`. *Output:* `0.194444`.

  Phân tích. Gọi n outcomes là $a_1, \ldots, a_n \in \{1, \ldots, 6\}$. Sum of outcomes: $S := \sum_{i=1}^n a_i \in \{n, \ldots, 6n\}$.

Chương 14

String Algorithms

Chương 15

Computational Geometry – Hình Học Tính Toán

Contents

15.1 Computational Elementary Geometry – Hình Học Sơ Cấp Tính Toán

Resources – Tài nguyên.

1. [Đàm+19b]. Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, Nguyễn Thanh Hùng. *Tài Liệu Chuyên Tin Học Quyển 3*.
2. [Đàm+19a]. Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, Nguyễn Thanh Hùng. *Tài Liệu Chuyên Tin Học Bài Tập Quyển 3*.

Hình học tính toán (computational geometry) là 1 nhánh của ngành khoa học máy tính (Computer Science), chuyên nghiên cứu về thuật toán giải quyết các bài toán liên quan tới các đối tượng hình học (geometrical object). Trong Toán học & công nghệ hiện đại, hình học tính toán có ứng dụng khá rộng rãi trong các lĩnh vực về đồ họa máy tính, thiết kế, mô phỏng, etc.

15.1.1 Cartesian coordinate system – Hệ tọa độ Descartes

Resources – Tài nguyên.

1. Wikipedia/Cartesian coordinate system.

In geometry, a *Cartesian coordinate system* in a plane is a coordinate system that specifies each point uniquely by a pair of real numbers called *coordinates*, which are the signed distances to the point from 2 fixed perpendicular oriented lines, called *coordinate lines*, *coordinate axes* or just *axes* (plural of axis) of the system. The point where the axes meet is called the *origin* & has $(0, 0)$ as coordinates. The axes directions represent an orthogonal basis. The combination of origin & basis forms a coordinate frame called the *Cartesian frame*.

– Trong hình học, một *hệ tọa độ Descartes* trong một mặt phẳng là một hệ tọa độ chỉ định mỗi điểm duy nhất bởi một cặp số thực gọi là *tọa độ*, là khoảng cách có dấu đến điểm từ 2 đường thẳng cố định vuông góc với nhau, được gọi là *đường tọa độ*, *trục tọa độ* hoặc chỉ là *trục* (số nhiều của trục) của hệ thống. Điểm mà các trục gặp nhau được gọi là *gốc* & có $(0, 0)$ làm tọa độ. Các hướng của trục biểu diễn một cơ sở trực giao. Sự kết hợp của gốc & cơ sở tạo thành một khung tọa độ được gọi là *khung Descartes*.

Similarly, the position of any point in 3D space can be specified by 3 *Cartesian coordinates*, which are the signed distances from the point to 3 mutually perpendicular planes. More generally, n Cartesian coordinates specify the point in an n-dimensional Euclidean space for any dimension n. These coordinates are the signed distances from the point to n mutually perpendicular fixed hyperplanes.

– Tương tự như vậy, vị trí của bất kỳ điểm nào trong không gian 3 chiều có thể được chỉ định bởi 3 *tọa độ Descartes*, là khoảng cách có dấu từ điểm đến 3 mặt phẳng vuông góc với nhau. Tổng quát hơn, n tọa độ Descartes chỉ định điểm trong không gian Euclidean n chiều cho bất kỳ chiều n nào. Các tọa độ này là khoảng cách có dấu từ điểm đến n siêu phẳng cố định vuông góc với nhau.

Cartesian coordinates are named for René Descartes, whose invention of them in the 17th century revolutionized mathematics by allowing the expression of problems of geometry in terms of algebra & calculus. Using the Cartesian coordinate system,

geometric shapes (e.g. curves) can be described by equations involving the coordinates of points of the shape. E.g., a circle of radius 2, centered at the origin of the plane, may be described as the set of all points whose coordinates x, y satisfy the equation $x^2 + y^2 = 4$; the area, the perimeter, & the tangent line at any point can be computed from this equation by using integrals & derivatives, in a way that can be applied to any curve.

– Tọa độ Descartes được đặt theo tên của René Descartes, người đã phát minh ra chúng vào thế kỷ 17 đã cách mạng hóa toán học bằng cách cho phép biểu diễn các bài toán hình học theo đại số & phép tính. Sử dụng hệ tọa độ Descartes, các hình dạng hình học (ví dụ: đường cong) có thể được mô tả bằng các phương trình liên quan đến tọa độ của các điểm của hình dạng đó. Ví dụ, một đường tròn bán kính 2, có tâm tại gốc của mặt phẳng, có thể được mô tả là tập hợp tất cả các điểm có tọa độ x, y thỏa mãn phương trình $x^2 + y^2 = 4$; diện tích, chu vi, & đường tiếp tuyến tại bất kỳ điểm nào có thể được tính toán từ phương trình này bằng cách sử dụng tích phân & đạo hàm, theo cách có thể áp dụng cho bất kỳ đường cong nào.

Cartesian coordinates are the foundation of analytic geometry, & provide enlightening geometric interpretations for many other branches of mathematics, e.g. linear algebra, complex analysis, differential geometry, multivariate calculus, group theory, & more. A familiar example is the concept of the graph of a function. Cartesian coordinates are also essential tools for most applied disciplines that deal with geometry, including astronomy, physics, engineering, & many more. They are the most common coordinate system used in computer graphics, computer-aided geometric design, & other geometry-related data processing.

– Tọa độ Descartes là nền tảng của hình học giải tích, & cung cấp các diễn giải hình học bổ ích cho nhiều nhánh toán học khác, ví dụ như đại số tuyến tính, phân tích phức, hình học vi phân, phép tính đa biến, lý thuyết nhóm, & nhiều hơn nữa. Một ví dụ quen thuộc là khái niệm đồ thị của một hàm. Tọa độ Descartes cũng là công cụ thiết yếu cho hầu hết các ngành ứng dụng liên quan đến hình học, bao gồm thiên văn học, vật lý, kỹ thuật, & nhiều hơn nữa. Chúng là hệ tọa độ phổ biến nhất được sử dụng trong đồ họa máy tính, thiết kế hình học hỗ trợ máy tính, & xử lý dữ liệu liên quan đến hình học khác.

15.1.1.1 History of Cartesian coordinate system

The adjective *Cartesian* refers to the French mathematician & philosopher René Descartes, who published this idea in 1637 while he was resident in the Netherlands. It was independently discovered by Pierre de Fermat, who also worked in 3D, although Fermat did not publish the discovery. The French cleric Nicole Oresme used constructions similar to Cartesian coordinates well before the time of Descartes & Fermat.

– Tính từ *Cartesian* ám chỉ nhà toán học & triết gia người Pháp René Descartes, người đã công bố ý tưởng này vào năm 1637 khi ông đang cư trú tại Hà Lan. Nó được phát hiện độc lập bởi Pierre de Fermat, người cũng làm việc trong không gian 3 chiều, mặc dù Fermat không công bố khám phá này. Giáo sĩ người Pháp Nicole Oresme đã sử dụng các cấu trúc tương tự như tọa độ Descartes từ rất lâu trước thời của Descartes & Fermat.

Both Descartes & Fermat used a single axis in their treatments & have a variable length measured in reference to this axis. The concept of using a pair of axes was introduced later, after Descartes' *La Géométrie* was translated into Latin in 1649 by Frans van Schooten & his students. These commentators introduced several concepts while trying to clarify the ideas contained in Descartes's work.

– Cả Descartes & Fermat đều sử dụng một trục duy nhất trong các phương pháp xử lý của họ & có chiều dài biến đổi được đo theo trục này. Khái niệm sử dụng một cặp trục được giới thiệu sau đó, sau khi Descartes' *La Géométrie* được dịch sang tiếng Latin vào năm 1649 bởi Frans van Schooten & các học trò của ông. Những nhà bình luận này đã giới thiệu một số khái niệm trong khi cố gắng làm rõ các ý tưởng có trong tác phẩm của Descartes.

The development of the Cartesian coordinate system would play a fundamental role in the development of the calculus by Isaac Newton & Gottfried Wilhelm Leibniz. The 2-coordinate description of the plane was later generalized into the concept of vector spaces.

– Sự phát triển của hệ tọa độ Descartes sẽ đóng vai trò cơ bản trong sự phát triển phép tính của Isaac Newton & Gottfried Wilhelm Leibniz. Mô tả 2 tọa độ của mặt phẳng sau đó được khái quát thành khái niệm không gian vectơ.

Many other coordinate systems have been developed since Descartes, e.g. the polar coordinates for the plane, & the spherical & cylindrical coordinates for 3D space.

– Nhiều hệ tọa độ khác đã được phát triển kể từ thời Descartes, e.g., tọa độ cực cho mặt phẳng, & tọa độ hình cầu & tọa độ hình trụ cho không gian 3D.

15.1.1.2 Cartesian formulae for the plane – Công thức Descartes cho mặt phẳng

15.1.1.2.1 Distance between 2 points – Khoảng cách giữa 2 điểm. The Euclidean distance between 2 points of the plane with Cartesian coordinates $A_1(x_1, y_1), A_2(x_2, y_2) \in \mathbb{R}^2$ is $d(A_1, A_2) = d((x_1, y_1), (x_2, y_2)) := \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. This is the Cartesian version of Pythagoras's theorem. In 3D space, the distance between points $A_1(x_1, y_1, z_1), A_2(x_2, y_2, z_2) \in \mathbb{R}^3$ is $d(A_1, A_2) = d((x_1, y_1, z_1), (x_2, y_2, z_2)) := \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$, which can be obtained by 2 consecutive application of Pythagoras's theorem.

15.1.1.2.2 Euclidean transformations – Các phép biến đổi Euclidean. The *Euclidean transformations* or *Euclidean motions* are the (bijective) mappings of points of the Euclidean plane to themselves which preserve distances between points. There are

4 types of these mappings (also called *isometries*): translations, rotations, reflections, & glide reflections.

– *Phép biến đổi Euclidean* hoặc *Chuyển động Euclidean* là phép ánh xạ (song ánh) của các điểm trên mặt phẳng Euclidean với chính chúng, phép ánh xạ này bảo toàn khoảng cách giữa các điểm. Có 4 loại phép ánh xạ này (còn gọi là *đẳng cự*): phép tịnh tiến, phép quay, phép phản xạ, & phản xạ trượt.

1. **Translation – Phép tịnh tiến.** Translating a set of points of the plane, preserving the distances & directions between them, is equivalent to adding a fixed pair of numbers (a, b) to the Cartesian coordinates of every point in the set. I.e., if the original coordinates of a point are (x, y), after the translation they will be $(x',') = (x + a, y + b)$.

   – Việc tịnh tiến một tập hợp các điểm của mặt phẳng, bảo toàn khoảng cách & hướng giữa chúng, tương đương với việc thêm một cặp số cố định (a, b) vào tọa độ Descartes của mọi điểm trong tập hợp. Tức là, nếu tọa độ ban đầu của một điểm là (x, y), sau khi tịnh tiến, chúng sẽ là $(x',') = (x + a, y + b)$.

2. **Rotation – Phép quay.** To rotate a figure counterclockwise around the origin by some angle θ is equivalent to replacing every point with coordinates (x, y) by the point with coordinates (x', y'), where $x' = x\cos\theta - y\sin\theta, y' = x\sin\theta + y\cos\theta$.

   – Để quay một hình ngược chiều kim đồng hồ quanh gốc tọa độ θ tương đương với việc thay thế mọi điểm có tọa độ (x, y) bằng điểm có tọa độ (x', y'), trong đó $x' = x\cos\theta - y\sin\theta, y' = x\sin\theta + y\cos\theta$. Thus $(x', y') = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$.

3. **Reflection – Phép phản xạ.** If (x, y) are the Cartesian coordinates of a point, then $(-x, y)$ are the coordinates of its reflection across the 2nd coordinate axis (the y-axis), as if that line were a mirror. Likewise, $(x, -y)$ are the coordinates of its reflection across the 1st coordinate axis (the x-axis). In more generality, reflection across a line through the origin making an angle θ with the x-axis, is equivalent to replacing every point with coordinates (x, y) by the point with coordinates $(x', y') = (x\cos 2\theta + y\sin 2\theta, x\sin 2\theta - y\cos 2\theta)$.

   – Nếu (x, y) là tọa độ Descartes của một điểm, thì $(-x, y)$ là tọa độ của phép phản chiếu của nó qua trục tọa độ thứ 2 (trục y), như thể đường thẳng đó là một tấm gương. Tương tự như vậy, $(x, -y)$ là tọa độ của phép phản chiếu của nó qua trục tọa độ thứ nhất (trục x). Nói chung hơn, phép phản chiếu qua một đường thẳng đi qua gốc tọa độ tạo thành một góc θ với trục x, tương đương với việc thay thế mọi điểm có tọa độ (x, y) bằng điểm có tọa độ $(x', y') = (x\cos 2\theta + y\sin 2\theta, x\sin 2\theta - y\cos 2\theta)$.

4. **Glide reflection – Phản xạ lướt.** A glide reflection is the composition of a reflection across a line followed by a translation in the direction of that line. It can be seen that the order of these operations does not matter (the translation can come 1st, followed by the reflection).

   – Phản xạ lướt là sự kết hợp của phản xạ qua một đường thẳng theo sau là phép tịnh tiến theo hướng của đường thẳng đó. Có thể thấy rằng thứ tự của các phép toán này không quan trọng (phép tịnh tiến có thể đứng thứ nhất, sau đó là phép phản xạ).

5. **General matrix form of the transformations – Dạng ma trận tổng quát của các phép biến đổi.** All affine transformations of the plane can be described in a uniform way by using matrices. For this purpose, the coordinates (x, y) of a point are commonly represented as the column matrix $\binom{x}{y}$. The result (x', y') of applying an affine transformation to a point (x, y) is given by the formula

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = A\begin{pmatrix} x \\ y \end{pmatrix} + b \text{ where } A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in M_2,\ b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

   I.e.,

$$x' = xA_{11} + yA_{11} + b_1,\ y' = xA_{21} + yA_{22} + b_2.$$

   Among the affine transformations, the Euclidean transformations are characterized by the fact that the matrix A is orthogonal, i.e., its columns are orthogonal vectors of Eucldiean norm 1, or explicitly,

$$A_{11}A_{12} + A_{21}A_{22} = 0,\ A_{11}^2 + A_{21}^2 = A_{12}^2 + A_{22}^2 = 1 \Leftrightarrow AA^\top = I.$$

   If these conditions do not hold, the formula describes a more general affine transformation.

   Theorem 2. *(a) The transformation is a translation iff $A = I$. The transformation is a rotation around some point iff A is a rotation matrix, i.e., it is orthogonal & $A_{11}A_{22} - A_{21}A_{12} = 1$. A reflection or glide reflection is obtained when $A_{11}A_{22} - A21A_{12} = -1$. (b) Assuming that translations are not used, i.e., $b_1 = b_2 = 0$, transformations can be composed by simply multiplying the associated transformation matrices. In the general case, it is useful to use the augmented matrix of the transformations, i.e., to rewrite the transformation formula*

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = A'\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \text{ where } A' = \begin{pmatrix} A_{11} & A_{12} & b_1 \\ A_{21} & A_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

   With this trick, the composition of affine transformations is obtained by multiplying the augmented matrices.

15.1.1.2.3 Affine transformation – Phép biến hình affine. Affine transformations of the Euclidean plane are transformations that map lines to lines, but may change distances & angles. They can be represented with augmented matrices:

$$\begin{pmatrix} A_{11} & A_{21} & b_1 \\ A_{12} & A_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}.$$

The Euclidean transformations are the affine transformations s.t. the 2×2 matrix of the A_{ij} is orthogonal.

– Phép biến đổi afin của mặt phẳng Euclid là phép biến đổi ánh xạ các đường thẳng thành các đường thẳng, nhưng có thể thay đổi khoảng cách & góc. Chúng có thể được biểu diễn bằng các ma trận tăng cường:

$$\begin{pmatrix} A_{11} & A_{21} & b_1 \\ A_{12} & A_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}.$$

Phép biến đổi Euclid là phép biến đổi afin vì ma trận 2×2 của A_{ij} là trực giao.

The augmented matrix that represents the composition of 2 affine transformations is obtained by multiplying their augmented matrices. Some affine transformations that are not Euclidean transformations have received specific names.

– Ma trận tăng cường biểu diễn thành phần của 2 phép biến đổi afin thu được bằng cách nhân các ma trận tăng cường của chúng. Một số phép biến đổi afin không phải là phép biến đổi Euclid đã nhận được tên gọi cụ thể.

- **Scaling – Tỷ lệ.** An example of an affine transformation which is not Euclidean is given by scaling. To make a figure larger or smaller is equivalent to multiplying the Cartesian coordinates of every point by the same positive number $m \in (0, \infty)$. If (x, y) are the coordinates of a point on the original figure, the corresponding point on the scaled figure has coordinates $(x', y') = (mx, my)$. If $m > 1$, the figure becomes larger; if $m \in (0, 1)$, it becomes smaller.

  – 1 ví dụ về phép biến đổi afin không phải Euclidean được đưa ra bằng cách chia tỷ lệ. Để làm cho một hình lớn hơn hoặc nhỏ hơn tương đương với việc nhân tọa độ Descartes của mọi điểm với cùng một số dương $m \in (0, \infty)$. Nếu (x, y) là tọa độ của một điểm trên hình gốc, thì điểm tương ứng trên hình đã chia tỷ lệ có tọa độ $(x', y') = (mx, my)$. Nếu $m > 1$, hình trở nên lớn hơn; nếu $m \in (0, 1)$, nó trở nên nhỏ hơn.

- **Shearing – Cắt.** A shearing transformation will push the top of a square sideways to form a parallelogram. Horizontal shearing is defined by $(x', y') = (x + sy, y)$. Shearing can also be applied vertically: $(x', y') = (x, sx + y)$.

  – Phép biến đổi cắt sẽ đẩy đỉnh của hình vuông sang một bên để tạo thành hình bình hành. Cắt ngang được định nghĩa bởi $(x', y') = (x + sy, y)$. Cắt cũng có thể được áp dụng theo chiều dọc: $(x', y') = (x, sx + y)$.

15.1.1.3 Orientation & handedness ⋆ – Định hướng & thuận tay ⋆

15.1.2 Problems: Computational elementary geometry

Bài toán 36 (Bao phủ hình chữ nhật nguyên 2D nhỏ nhất)**.** (2.5 điểm) *Viết chương trình Python để tính chu vi, diện tích, & độ dài đường chéo hình chữ nhật nhỏ nhất "chứa trọn" $n \in \mathbb{N}^\star$ điểm $A_1(x_1, y_1), A_2(x_2, y_2), \ldots, A_n(x_n, y_n) \in \mathbb{R}^2$ với tọa độ nguyên $x_i, y_i \in \mathbb{Z}$, $\forall i = 1, \ldots, n$, cho trước trong mặt phẳng 2 chiều, "chứa trọn" ở đây nghĩa là các điểm chỉ được nằm bên trong hình chữ nhật, không được nằm trên cạnh hình chữ nhật.*

Input. *Dòng 1 chứa số nguyên $n \in \mathbb{N}^\star$: * số điểm trong mặt phẳng 2 chiều. n dòng tiếp theo, mỗi dòng chứa 2 số nguyên $x_i, y_i \in \mathbb{Z}$: hoành độ & tung độ của điểm thứ i $A_i(x_i, y_i)$.*

Output. *In ra chu vi, diện tích, & đường chéo của hình chữ nhật thỏa mãn nhờ gọi lại hàm của Bài 1 (hoặc tự viết lại nếu muốn).*

Sample.

min_rectangle.inp	min_rectangle.out
4 1 0 -2 2 -1 3 4 2	$P = 26$ $S = 40$ $d = 9.43398113206$

Solution. Python:

```
n = int(input()) # number of 2D points -- số điểm trên mặt phẳng
x_min = y_min = 1e9
x_max = y_max = -1e9
```

```
        for i in range(n):
        x, y = map(int, input().split())
        if x < x_min:
        x_min = x
        if x > x_max:
        x_max = x
        if y < y_min:
        y_min = y
        if y > y_max:
        y_max = y
        a, b = x_max - x_min + 2, y_max - y_min + 2
        print('P =', 2 * (a + b))
        print('S =', a * b)
        print('d =', sqrt(a * a + b * b))
```

□

Bài toán 37 (Bao phủ hình hộp chữ nhật nguyên 3D nhỏ nhất)**.** (3.5 điểm) *Viết chương trình Python để tính diện tích toàn phần, thể tích, & độ dài đường chéo hình hộp chữ nhật nhỏ nhất chứa* $n \in \mathbb{N}^\star$ *điểm* $A_1(x_1, y_1, z_1), A_2(x_2, y_2, z_2), \ldots, A_n(x_n, y_n, z_n) \in \mathbb{R}^3$ *với tọa độ nguyên* $x_i, y_i \in \mathbb{Z}$, $\forall i = 1, \ldots, n$, *cho trước trong không gian 3 chiều, ở đây các điểm có thể nằm bên trong hoặc nằm trên cạnh hình hộp chữ nhật.*

Input. *Dòng 1 chứa số nguyên* $n \in \mathbb{N}^\star$*: số điểm trong không gian 3 chiều.* n *dòng tiếp theo, mỗi dòng chứa 3 số nguyên* $x_i, y_i, z_i \in \mathbb{Z}$*: hoành độ, tung độ, & cao độ của điểm thứ* i $A_i(x_i, y_i, z_i)$.

Output. *In ra diện tích toàn phần, thể tích, & độ dài đường chéo hình hộp chữ nhật thỏa mãn, biết với hình hộp chữ nhật có kích thước* $a \times b \times c$ *thì* $S_{\text{tp}} = 2(ab + bc + ca), V = abc, d = \sqrt{a^2 + b^2 + c^2}$.

Sample.

min_rectangular_cuboid.inp	min_rectangular_cuboid.out
4 1 0 0 0 1 0 -1 0 2 2 0 1	$S_{\text{tp}} = 22$ $V = 6$ $d = 3.74165738677$

Solution. Python:

```
        n = int(input()) # number of 3D points
        x_min = y_min = z_min = 1e9
        x_max = y_max = z_max = -1e9
        for i in range(n):
        x, y, z = map(int, input().split())
        if x < x_min:
        x_min = x
        if x > x_max:
        x_max = x
        if y < y_min:
        y_min = y
        if y > y_max:
        y_max = y
        if z < z_min:
        z_min = z
        if z > z_max:
        z_max = z
        a, b, c = x_max - x_min, y_max - y_min, z_max - z_min
        print('S_tp =', 2 * (a * b + b * c + c * a))
        print('V =', a * b * c)
        print('d =', sqrt(a * a + b * b + c * c))
```

□

Bài toán 38 ([Dàm+19b], 8.1, segment in triangle – tìm độ dài đoạn thẳng nằm trong tam giác)**.** *Trên mặt phẳng cho ΔABC & đoạn thẳng DE. Tính độ dài của phần đoạn thẳng DE nằm trong ΔABC.*

Input. *Dòng 1 có 6 số thực $x_A, y_A, x_B, y_B, x_C, y_C$ lần lượt từng cặp là tọa độ tương ứng của 3 đỉnh A, B, C. Dòng 2 là 4 số thực x_D, y_D, x_E, y_E lần lượt từng cặp là tọa độ 2 điểm D, E.*

Output. *In 1 số thực duy nhất là độ dài phần đoạn thẳng DE nằm trong ΔABC.*

Sample.

segment_in_triangle.inp	segment_in_triangle.out
0 2 2 4 4 2 0 2 3 3	3.16228

Bài toán 39 ([Dàm+19b], 8.2, common area of 2 triangles – tìm diện tích phần chung của 2 tam giác)**.** *Trên mặt phẳng cho 2 tam giác. Tìm diện tích phần chung của 2 tam giác.*

Input. *Gồm 2 dòng, mỗi dòng có 6 số thực lần lượt từng cặp là tọa độ tương ứng của 3 đỉnh của 1 tam giác. Các tọa độ có giá trị tuyệt đối $\leq 10^3$.*

Output. *In ra duy nhất 1 số thực là diện tích phần chung của 2 tam gaics này.*

Constraints. $n \in [2 \cdot 10^5], m \in [100], x_i \in \{0, 1, \ldots, m\}$.

Sample.

common_area_triangle.inp	common_area_triangle.out
0 6 8 6 4 0 4 8 8 2 0 2	16

Problem 137 (CSES Problem Set/point location test)**.** *There is a line that goes through the points $p_1 = (x_1, y_2), p_2(x_2, y_2)$. There is also a point $p_3 = (x_3, y_3)$. Determine whether p_3 is located on the left or right side of the line or if it touches the line when we are looking from p_1 to p_2.*

Input. *The 1st input line has an integer $t \in \mathbb{N}^\star$: the number of tests. After this, there are t lines that describe the tests. Each line has 6 integers $x_1, y_1, x_2, y_2, x_3, y_3 \in \mathbb{Z}$.*

Output. *For each test, print* `LEFT, RIGHT` *or* `TOUCH`.

Constraints. $t \in [10^5], x_i, y_i \in \overline{-10^9, 10^9}$, $\forall i \in [3]$, $x_1 \neq x_2$ *or* $y_1 \neq y_2$ *(i.e.,* $(x_1, x_1) \neq (x_2, y_2)$*).*

Sample.

point_location_test.inp	point_location_test.out
3 1 1 5 3 2 3 0 2 1 1 5 3 4 1 1 1 5 3 3 2	LEFT RIGHT TOUCH

Solution. C++:

1. PVT's C++: point location test

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long

void solve() {
    ll x1, y1, x2, y2, x3, y3;
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
    ll dot = (x2 - x1) * (x3 - x1) + (y2 - y1) * (y3 - y1);
    ll det = (x2 - x1) * (y3 - y1) - (x3 - x1) * (y2 - y1);
    double angle = atan2(det, dot);
    if (angle == 0 || angle == M_PI || angle == -M_PI) cout << "TOUCH\n";
    else if (angle > 0) cout << "LEFT\n";
    else cout << "RIGHT\n";
```

```
}

int main() {
    ios_base::sync_with_stdio(false); cin.tie(NULL);
    ll t; cin >> t;
    while (t--)
    solve();
}
```

2. NHT's C++: point location test

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;

int t;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> t;
    while (t--) {
        ll x1, y1, x2, y2, x3, y3;
        cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
        ll area = (x2 - x1) * (y3 - y1) - (y2 - y1) * (x3 - x1);
        if (area > 0)
        cout << "LEFT\n";
        else if (area < 0)
        cout << "RIGHT\n";
        else
        cout << "TOUCH\n";
    }
}
```

□

Problem 138 (CSES Problem Set/line segment intersection)**.** *There are* 2 *line segments: the 1st goes through the points* $(x_1, y_1), (x_2, y_2)$*, & the 2nd goes through the points* $(x_3, y_3), (x_4, y_4)$*. Determine if the line segments intersect, i.e., they have at least* 1 *common point.*

Input. *The 1st input line has an integer* $t \in \mathbb{N}^\star$*: the number of tests. After this, there are* t *lines that describe the tests. Each line has* 8 *integers* $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \in \mathbb{Z}$*.*

Output. *For each test, print* `TEST` *if the line segments intersect &* `NO` *otherwise.*

Constraints. $t \in [10^5], x_i, y_i \in \overline{-10^9, 10^9}$, $\forall i \in [4]$, $(x_1, x_1) \neq (x_2, y_2)$, $(x_3, x_3) \neq (x_4, y_4)$.

Sample.

line_segment_intersection.inp	line_segment_intersection.out
5	NO
1 1 5 3 1 2 4 3	YES
1 1 5 3 1 1 4 3	YES
1 1 5 3 2 3 4 1	YES
1 1 5 3 2 4 4 1	YES
1 1 5 3 3 2 7 4	

Solution. C++:

1. VNTA's C++: line segment intersection: https://github.com/NQBH/advanced_STEM_beyond/blob/main/OLP_ICPC/C%2B%2B/VNTA_line_segment_intersection.cpp.

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long

struct P {
    ll x;
    ll y;
};

int ori(P a, P b, P c) {
    ll v = (b.y - a.y) * (c.x - b.x) - (b.x - a.x) * (c.y - b.y);
    if (v == 0) return 0;
    return (v > 0) ? 1 : 2;
}

bool onSeg(P a, P b, P c) {
    return (b.x <= max(a.x, c.x) &&
    b.x >= min(a.x, c.x) &&
    b.y <= max(a.y, c.y) &&
    b.y >= min(a.y, c.y));
}

bool check(P a, P b, P c, P d) {
    int o1 = ori(a, b, c);
    int o2 = ori(a, b, d);
    int o3 = ori(c, d, a);
    int o4 = ori(c, d, b);

    if (o1 != o2 && o3 != o4) return true;

    if (o1 == 0 && onSeg(a, c, b)) return true;
    if (o2 == 0 && onSeg(a, d, b)) return true;
    if (o3 == 0 && onSeg(c, a, d)) return true;
    if (o4 == 0 && onSeg(c, b, d)) return true;

    return false;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int t; cin >> t;
    while (t--) {
        P p1, p2, p3, p4;
        cin >> p1.x >> p1.y >> p2.x >> p2.y >> p3.x >> p3.y >> p4.x >> p4.y;
        if (check(p1, p2, p3, p4)) cout << "YES\n";
        else cout << "NO\n";
    }
}
```

□

Problem 139 (CSES Problem Set/polygon area)**.** *Calculate the area of a given polygon. The polygon consists of n vertices (x_i, y_i), $\forall i \in [n]$. The vertices (x_i, y_i) & (x_{i+1}, y_{i+1}) are adjacent for $i \in [n-1]$, & the vertices (x_1, y_1) & (x_n, y_n) are also adjacent.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of vertices. After this, there are n lines that describe the vertices. The ith such line has 2 integers $x_i, y_i \in \mathbb{Z}$. You may assume that the polygon is simple, i.e., it does not intersect itself.*

Output. *Print 1 integer: $2a \in \mathbb{Z}$ where the area of the polygon is a (is ensures that the result is an integer).*

Constraints. $n \in \overline{3, 1000}, x_i, y_i \in \overline{-10^9, 10^9}$, $\forall i \in [n]$.

Sample.

polygon_area.inp	polygon_area.out
4 1 1 4 2 3 5 1 4	16

Problem 140 (CSES Problem Set/point in polygon). *You are given a polygon of $n \in \mathbb{N}^\star$ vertices & a list of $m \in \mathbb{N}^\star$ points. Determine for each point if it is inside, outside or on the boundary of the polygon. The polygon consists of n vertices (x_i, y_i), $\forall i \in [n]$. The vertices (x_i, y_i) & (x_{i+1}, y_{i+1}) are adjacent for $i \in [n-1]$, & the vertices (x_1, y_1) & (x_n, y_n) are also adjacent.*

Input. *The 1st input line has 2 integer $n, m \in \mathbb{N}^\star$: the number of vertices in the polygon & the number of points. After this, there are n lines that describe the polygon. The ith such line has 2 integers x_i, y_i. You may assume that the polygon is simple, i.e., it does not intersect itself. Finally, there are m lines that describe the points. Each line has 2 integers $x, y \in \mathbb{Z}$.*

Output. *For each point, print* `INSIDE`, `OUTSIDE` *or* `BOUNDARY`.

Constraints. $M, n \in \overline{3, 1000}, m \in [1000], x, y, x_i, y_i \in \overline{-10^9, 10^9}$, $\forall i \in [n]$.

Sample.

point_in_polygon.inp	point_in_polygon.out
4 3 1 1 4 2 3 5 1 4 2 3 3 1 1 3	INSIDE OUTSIDE BOUNDARY

Problem 141 (CSES Problem Set/polygon lattice points). *Given a polygon , calculate the number of lattice points inside the polygon & on its boundary. A* lattice point *is a point whose coordinates are integers. The polygon consists of $n \in \mathbb{N}^\star$ vertices (x_i, y_i), $\forall i \in [n]$. The vertices (x_i, y_i) & (x_{i+1}, y_{i+1}) are adjacent for $i \in [n-1]$, & the vertices (x_1, y_1) & (x_n, y_n) are also adjacent.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of vertices. After this, there are n lines that describe the vertices. The ith such line has 2 integers x_i, y_i. You may assume that the polygon is simple, i.e., it does not intersect itself.*

Output. *Print 2 integers: the number of lattice points inside the polygon & on its boundary.*

Constraints. $n \in \overline{3, 10^5}, x_i, y_i \in \overline{-10^9, 10^9}$, $\forall i \in [n]$.

Sample.

polygon_lattice_point.inp	polygon_lattice_point.out
4 1 1 5 3 3 5 1 4	6 8

Problem 142 (CSES Problem Set/minimum Euclidean distance). *Given a set of points in 2D plane, find the minimum Euclidean distance between 2 distinct points. The Euclidean distance of points $(x_1, y_1), (x_2, y_2)$ is $d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of points. After this, there are n lines that describe the points. Each line has 2 integers $x, y \in \mathbb{Z}$: the coordinates of a point. You may assume that each point is distinct.*

Output. *Print 1 integer: d^2 where d is the minimum Euclidean distance (this ensures that the result is an integer).*

Constraints. $n \in \overline{2, 2 \cdot 10^5}, x, y \in \overline{-10^9, 10^9}$.

Sample.

minimum_Euclidean_distance.inp	minimum_Euclidean_distance.out
4 2 1 4 4 1 2 6 3	2

Problem 143 (CSES Problem Set/convex hull). *Given a set of $n \in \mathbb{N}^\star$ points in the 2D plane, determine the convex hull of the points.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of points. After this, there are n lines that describe the points. Each line has 2 integers $x, y \in \mathbb{Z}$: the coordinates of a point. You may assume that each point is distinct, & the area of the hull is positive.*

Output. *1st print an integer $k \in \mathbb{N}$: the number of points in the convex hull. After this, print k lines that describe the points. You can print the points in any order. Print all points that lie on the convex hull.*

Constraints. $n \in \overline{3, 2 \cdot 10^5}, x, y \in \overline{-10^9, 10^9}$.

Sample.

.inp	.out
6 2 1 2 5 3 3 4 3 4 4 6 3	4 2 1 2 5 4 4 6 3

Problem 144 (CSES Problem Set/maximum Manhattan distance). *A set is initially empty & $n \in \mathbb{N}^\star$ points are added to it. Calculate the maximum Manhattan distance of 2 points after each addition.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of points. The following n lines describe the points. Each line has 2 integers $x, y \in \mathbb{Z}$. You can assume that each point is distinct.*

Output. *After each addition, print the maximum distance.*

Constraints. $n \in \overline{2, 2 \cdot 10^5}, x, y \in \overline{-10^9, 10^9}$.

Sample.

maximum_Manhattan_distance.inp	maximum_Manhattan_distance.out
5 1 1 3 2 2 4 2 1 4 5	0 3 4 4 7

Problem 145 (CSES Problem Set/all Manhattan distances). *Given a set of points, calculate the sum of all Manhattan distances between 2 point pairs.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of points. The following n lines describe the points. Each line has 2 integers $x, y \in \mathbb{Z}$. You can assume that each point is distinct.*

Output. *Print the sum of all Manhattan distances.*

Constraints. $n \in \overline{2, 2 \cdot 10^5}, x, y \in \overline{-10^9, 10^9}$.

Sample.

all_Manhattan_distance.inp	all_Manhattan_distance.out
5 1 1 3 2 2 4 2 1 4 5	36

Problem 146 (CSES Problem Set/intersection points)**.** *Given $n \in \mathbb{N}^\star$ horizontal & vertical line segments, calculate the number of their intersection points. You can assume that no parallel line segments intersect, & no endpoint of a line segment is an intersection point.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of line segments. Then there are n lines describing the line segments. Each line has 4 integers: x_1, y_1, x_2, y_2: a line segment begins at point (x_1, y_1) & ends at point (x_2, y_2).*

Output. *Print the number of intersection points.*

Constraints. $n \in [10^5], x_1, x_2, y_1, y_2 \in \overline{-10^6, 10^6}$, $(x_1, y_1) \neq (x_2, y_2)$.

Sample.

intersection_point.inp	intersection_point.out
3 2 3 7 3 3 1 3 5 6 2 6 6	2

Problem 147 (CSES Problem Set/line segments trace I)**.** *There are $n \in \mathbb{N}^\star$ line segments whose endpoints have integer coordinates. The left x-coordinate of each segment is 0 & the right x-coordinate is $m \in \mathbb{N}^\star$. The slope of each segment is an integer. For each x-coordinate $0, 1, \ldots, m$, find the maximum point in any line segment.*

Input. *The 1st input line has 2 integer $n, m \in \mathbb{N}^\star$: the number of line segments & the maximum x-coordinate. The next n lines describe the line segments. Each line has 2 integers y_1, y_2: there is a line segment between points $(0, y_1)$ & (m, y_2).*

Output. *Print $m + 1$ integers: the maximum points for $x = 0, 1, \ldots, m$.*

Constraints. $m, n \in [10^5], m \in [100], y_1, y_2 \in \overline{0, 10^9}$.

Sample.

line_segment_trace_I.inp	line_segment_trace_I.out
4 5 1 6 7 2 5 5 10 0	10 8 6 5 5 6

Problem 148 (CSES Problem Set/line segments trace II)**.** *There are $n \in \mathbb{N}^\star$ line segments whose endpoints have integer coordinates. Each x-coordinate is between 0 & m. The slope of each segment is an integer. For each x-coordinate $0, 1, \ldots, m$, find the maximum point in any line segment. If there is no segment at some point, the maximum is -1.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of line segments & the maximum x-coordinate. The next n lines describe the line segments. Each line has 4 integers x_1, y_1, x_2, y_2: there is a line segment between points $(x_1, y_1), (x_2, y_2)$.*

Output. *Print $m + 1$ integers: the maximum points for $x = 0, 1, \ldots, m$.*

Constraints. $m, n \in [10^5], 0 \leq x_1 < x_2 \leq m, y_1, y_2 \in \overline{0, 10^9}$.

Sample.

line_segment_trace_II.inp	line_segment_trace_II.out
4 5 1 1 3 3 1 2 4 2 2 4 5 7 2 8 5 2	-1 2 8 6 6 7

Problem 149 (CSES Problem Set/lines & queries I). *Efficiently process the following types of queries:*

1. *Add a line $ax + b$.*

2. *Find the maximum point in any line at position x.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of queries. The following n lines describe the queries. The format of each line is either* `1 a b` *or* `2 x`*. You may assume that the 1st query is of type 1.*

Output. *Print the answer for each query of type 2.*

Constraints. $n \in [2 \cdot 10^5], a, b \in \overline{-10^9, 10^9}, x \in \overline{0, 10^5}$.

Sample.

line_query_I.inp	line_query_I.out
6 1 1 2 2 1 2 3 1 0 4 2 1 2 3	3 5 4 5

Problem 150 (CSES Problem Set/lines & queries II). *Efficiently process the following types of queries:*

1. *Add a line $ax + b$ that is active in range $[l, r]$.*

2. *Find the maximum point in any active line at position x.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of queries. The following n lines describe the queries. The format of each line is either* `1 a b l r` *or* `2 x`*.*

Output. *Print the answer for each query of type 2. If no line is active, print* `NO`*.*

Constraints. $n \in [2 \cdot 10^5], a, b \in \overline{-10^9, 10^9}, x, l, r \in \overline{0, 10^5}$.

Sample.

line_query_II.inp	line_query_II.out
6 1 1 2 1 3 2 3 2 4 1 0 4 1 5 2 3 2 4	5 NO 5 4

Problem 151 (CSES Problem Set/area of rectangles). *Given $n \in \mathbb{N}^\star$, determine the total area of their union.*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the number of rectangles. After that, there are n lines describing the rectangles. Each line has 4 integers x_1, y_1, x_2, y_2: a rectangle begins at point (x_1, y_1) & ends at point (x_2, y_2).*

Output. *Print the total area covered by the rectangles.*

Constraints. $n \in [10^5], m \in [100], x_1, x_2, y_1, y_2 \in \overline{-10^6, 10^6}$.

Sample.

area_rectangle.inp	area_rectangle.out
3 1 3 4 5 3 1 7 4 5 3 8 6	24

Problem 152 (CSES Problem Set/robot path)**.** *You are given a description of a robot's path. The robot begins at point* $(0, 0)$ *& performs* $n \in \mathbb{N}^\star$ *commands. Each command moves the robot some distance up, down, left, or right. The robot will stop when it has performed all commands, or immediately when it returns to a point that it has already visited. Calculate the total distance the robot moves.*

Input. *The 1st input line has an integer* $n \in \mathbb{N}^\star$*: the number of commands. After that, there are* n *lines describing the commands. each line has a character* d *&* *an integer* $x \in \mathbb{N}^\star$*: the robot moves the distance* x *to the direction* d*. Each direction is* `U`*,* `D`*,* `L`*,* `R` *(up, down, left, right).*

Output. *Print the total distance the robot moves.*

Constraints. $n \in [10^5], x \in [10^6]$.

Sample.

robot_path.inp	robot_path.out
5 U 2 R 3 D 1 L 5 U 2	9

Problem 153 (IMO2007P6)**.** *Let* $n \in \mathbb{N}^\star$*. Consider* $S = \{(x, y, z); x, y, z \in \overline{0, n},\ x + y + z > 0\}$ *as a set of* $(n + 1)^3 - 1$ *points in 3D space. Determine the smallest possible number of planes, the union of which contains* S *but does not include* $(0, 0, 0)$*.*

Bài toán 40 ([VL24], 6., p. 10, IMO2007P6)**.** *Cho* $n \in \mathbb{N}^\star$*. Xét* $S = \{(x, y, z); x, y, z \in \overline{0, n},\ x + y + z > 0\}$ *là 1 tập hợp gồm* $(n+1)^3 - 1$ *điểm trong không gian* 3*-chiều. Xác định số nhỏ nhất có thể các mặt phẳng mà hợp của chúng chứa tất cả các điểm của* S *nhưng không chứa điểm* $(0, 0, 0)$*.*

Chương 16

Number Theory – Lý Thuyết Số

Contents

16.1 Divisor – Ước Số

Problem 154 (IMO2007P5). *Let $a, b \in \mathbb{N}^\star$. Show that if $4ab - 1 | (4a^2 - 1)^2$, then $a = b$.*

Bài toán 41 (CP version of IMO2007P5). *Cho $m, n \in \mathbb{N}^\star$. Đếm số phần tử của tập hợp*

$$S = \{(a, b) \in [m] \times [n]; 4ab - 1 | (4a^2 - 1)^2\} = \{(a, b) \in [m] \times [n]; (4a^2 - 1)^2 \vdots 4ab - 1\}.$$

Input. *Chỉ 1 dòng chứa $m, n \in \mathbb{N}^\star$.*

Output. $|S|$.

Sample.

IMO2007_P5.inp	IMO2007_P5.out
5 3	3
10 15	10

Solution. Theo lời giải bài toán IMO2007P5 trước đó, đáp số bài toán này đơn giản chỉ là $\min\{m, n\}$. Nhưng đó là về mặt chứng minh toán học, sau đây ta sẽ dùng vòng lặp để kiểm tra.

C++:

1. DPAK's C++: IMO2007P5:

```
#include <iostream>
using namespace std;
int m, n;
const double oo = 1e9 + 7;

int main() {
    cin >> m >> n;
    int cnt = 0;
    for (int a = 1; a <= m; ++a) {
        int num = 4 * a * a - 1;
        num = num * num;
        for (int b = 1; b <= n; ++b) {
            int divisor = 4 * a * b - 1;
            if (num % divisor == 0) ++cnt;
        }
```

```
    }
    cout << cnt;
}
```

2. TQS's & DNDK's C++: IMO2007P5:

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

int main() {
    int m, n, count = 0;
    cin >> m >> n;
    for (int i = 1; i <= m; ++i)
        for (int j = 1; j <= n; ++j)
            if (((4 * i * i - 1) * (4 * i * i - 1)) % (4 * i * j - 1) == 0) ++count;
    cout << count;
}
```

Remark 7. *Dù code này không cần dùng tới 2 biến phụ* `num, divisor` *như code của DPAK, nhưng sẽ tính nhiều lần hơn do đoạn code* `(4 * i * i - 1) * (4 * i * i - 1)` *lặp lại phép tính* `(4 * i * i - 1)` *2 lần (unnecessary duplication).*

□

Problem 155 (IMO2008P3)**.** *Prove that there exists infinitely many positive integers n s.t. n^2+1 has a prime divisor which is $> 2n + \sqrt{2n}$.*

Bài toán 42 ([VL24], p. 12, IMO2008P3)**.** *Chứng minh tồn tại vô hạn $n \in \mathbb{N}^\star$ sao cho n^2+1 có ước nguyên tố lớn hơn $2n+\sqrt{2n}$.*

Bài toán 43 (CP version of IMO2008P3)**.** *Đếm số số $n \in \mathbb{N}^\star$ sao cho n^2+1 có ước nguyên tố lớn hơn $2n+\sqrt{2n}$ trong phạm vi $n \in [a,b]$ với $a, b \in \mathbb{N}^\star$ cho trước.*

Input. *2 số $a, b \in \mathbb{N}^\star$.*

Output. *Số số $n \in [a,b]$ thỏa mãn.*

Solution. C++:

1. DPAK's C++: IMO2008P3:

```
#include <iostream>
#include <cmath>
using namespace std;
long long a, b;
const double oo = 1e9 + 7;

int main() {
    long long cnt = 0;
    cin >> a >> b;
    for (long long i = a; i <= b; ++i) {
        long long new_num = i * i + 1;
        double limit = 2.0 * i + sqrt(2 * i);
        long long temp = new_num;
        bool ok = false;
        for (long long j = 2; j * j <= new_num; ++j)
        if (temp % j == 0) {
            while (temp % j == 0) temp /= j;
            if (j > limit) {
                ok = true;
                break;
            }
        }
        if (!ok && temp > 1 && temp > limit) ok = true;
        if (ok) ++cnt;
    }
```

```
26      cout << cnt;
27  }
```

2. DNDK's C++: IMO2008P3:

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   bool check(long long n) {
6       long long x = n * n + 1, X = x;
7       long long y = 2 * n + sqrtl(2 * n);
8       for (long long i = 2; i * i <= X; ++i)
9           if (x % i == 0) {
10              while (x % i == 0) x /= i;
11              if (i > y) return true;
12          }
13      if (x > 1 and x > y) return true;
14      return false;
15  }
16
17  int main() {
18      long long a, b;
19      cin >> a >> b;
20      long long count = 0;
21      for (long long i = a; i <= b; ++i)
22          if (check(i)) ++count;
23      cout << count;
24  }
```

□

Problem 156 (IMO2011P1). *Given any set $A = \{a_1, a_2, a_3, a_4\}$ of 4 distinct positive integers, we denote $s_A := a_1 + a_2 + a_3 + a_4$. Let n_A denote the number of pairs (i, j) with $1 \le i < j \le 4$ for which $a_i + a_j | s_A$. Find all sets A of 4 distinct positive integers which achieve the largest possible value of n_A.*

Bài toán 44 ([VL24], 1., p. 17, IMO2011P1). *Cho tập hợp $A = \{a_1, a_2, a_3, a_4\}$ gồm 4 số nguyên phân biệt, ta ký hiệu tổng $a_1 + a_2 + a_3 + a_4$ bởi s_A. Giả sử n_A là số các cặp (i, j) với $1 \le i < j \le 4$ sao cho $a_i + a_j \vdots s_A$. Tìm tất cả các tập hợp A gồm 4 số nguyên dương phân biệt mà với chúng n_A đạt được* GTLN *có thể.*

Bài toán 45. *Viết thuật toán & chương trình* C/C++, Pascal, Python *để minh hoạ IMO2011P1.*

16.2 Primorial

In mathematics, & more particular in number theory, *primorial*, denoted by $p_n\# : \mathbb{N} \to \mathbb{N}$ similar to the factorial function, but rather than successively multiplying positive integers, the function only multiplies prime numbers. The name "primorial", coined by Harvey Dubner, draws an analogy to *primes* similar to the way the name "factorial" relates to *factors*.

– Trong toán học, & cụ thể hơn trong lý thuyết số, *primorial*, được ký hiệu là $p_n\# : \mathbb{N} \to \mathbb{N}$ tương tự như hàm giai thừa, nhưng thay vì nhân liên tiếp các số nguyên dương, hàm này chỉ nhân các số nguyên tố. Tên "primorial", do Harvey Dubner đặt ra, có sự tương tự với *primes* tương tự như cách tên "factorial" liên quan đến các thừa số.

Definition 5 (Primorial for prime numbers). *For the nth prime number p_n, the primorial $p_n\#$ is defined as the product of the 1st n primes: $p_n\# = \prod_{i=1}^{n} p_i$, where p_i is the ith prime number.*

The 1st few primorials $p_n\#$ are $1, 2, 6, 30, 210, 2310, 30030, 510510, 9699690, \ldots$ (sequence A002110 in the OEIS). Asymptotically, primorials $p_n\#$ grow according to $p_n\# = e^{(1+o(1))n \log n}$.

Definition 6. *The primorial $n\#$ for $n \in \mathbb{N}^\star$ is the product of the primes that are not greater than n, i.e.,*

$$n\# = \prod_{p \le n,\, p \text{ prime}} p = \prod_{i=1}^{\pi(n)} p_i = p_{\pi(n)}\#,$$

where $\pi(n)$ is the prime-counting function (sequence A000720 in the OEIS), which gives the number of primes $\leq n$. This is equivalent to:

$$n\# = \begin{cases} 1 & \text{if } n = 0, 1, \\ (n-1)\# \times n & \text{if } n \text{ is prime}, \\ (n-1)\# & \text{if } n \text{ is composite}. \end{cases}$$

Example 3. $12\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$. *Since $\pi(12) = 5$, this can be calculated as $12\# = p_{\pi(12)}\# = p_5\# = 2310$.*

Consider the 1st 12 values of $n\#$: $1, 2, 6, 6, 30, 30, 210, 210, 210, 210, 2310, 2310$. We see that for composite n every term $n\#$ simply duplicates the preceding term $(n-1)\#$, as given in the definition. Since 12 is a composite number, $12\# = p_5\# = 11\#$.

Primorials are related to the 1st Chebyshev function, written $\vartheta(n)$ or $\theta(n)$ according to $\ln(n\#) = \vartheta(n)$. Since $\vartheta(n)$ asymptotically approaches n for large values of n, primorials therefore grow according to $n\# = e^{(1+o(1))n}$. The idea of multiplying all known primes occur in some proofs of the infinitude of the prime numbers, where it is used to derive the existence of another prime.

Theorem 3 (Characteristics of primorials)**.** *(a) Let p, q be 2 adjacent prime numbers. Given any $n \in \mathbb{N}$, where $p \leq n < q$, $n\# = p\#$.*

(b) The fact that the binomial coefficient $\binom{2n}{n}$ is divisible by every prime between $n+1$ & $2n$, together with the inequality $\binom{2n}{n} \leq 2^n$, allows to derive the upper bound $n\# \leq 4^n$.

16.3 Divisor function – Hàm ước số

Resources – Tài nguyên.

1. Wikipedia/divisor function.

In mathematics, & specifically in number theory, a *divisor function* is an arithmetic function related to the divisors of an integer. When referred to as *the* divisor function, it counts the *number of divisors of an integer* (including 1 & the number itself). It appears in a number of remarkable identities, including relationships on the Riemann zeta function & the Eisenstein series of modular forms. Divisor functions were studied by Ramanujan, who gave a number of important congruences & identities; these are treated separately in Wikipedia/Ramanujan's sum.

Definition 7. *The* sum of positive divisors function $\sigma_z(n)$*, for a real or complex number z, is defined as the sum of the zth powers of the positive divisors of n. It can be expressed in sigma notation as*

$$\sigma_z(n) = \sum_{d|n} d^z, \ \forall n \in \mathbb{N}^\star.$$

The notation $d(n), \nu(n), \tau(n)$ (for the German *Teiler* = divisors) are also used to denote $\sigma_0(n)$, or the *number-of-divisors function* (OEIS: A000005). When $z = 1$, the function is called the *sigma function* or *sum-of-divisors function*, & the subscript is often omitted, so $\sigma(n)$ is the same as $\sigma_1(n)$ (OEIS: A000203).

The aliquot sum $s(n)$ of n is the sum of the proper divisors (i.e., the divisors excluding n itself, OEIS: A001065), & equals $\sigma_1(n) - n$; the aliquot sequence of n is formed by repeatedly applying the aliquot sum function.

Example 4. *The number of divisors of 12 is $\sigma_0(12) = 6$, the sum of all the divisors of 12 is $\sigma_1(12) = 1+2+3+4+6+12 = 28$, & the aliquot sum $s(12) = 1+2+3+4+6 = 16$. $\sigma_{-1}(n)$ is sometimes called the abundancy index of n, & we have $\sigma_{-1}(12) = 1^{-1} + 2^{-1} + 3^{-1} + 4^{-1} + 6^{-1} + 12^{-1} = \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{12} = \frac{12+6+4+3+2+1}{12} = \frac{28}{12} = \frac{7}{3} = \frac{\sigma_1(12)}{12}$.*

Theorem 4. *For any prime number p: (a) $\sigma_0(p) = 2, \sigma_1(p) = p+1$. (b) $\sigma_0(p^n) = n+1, , \sigma_1(p^n) = \sum_{i=0}^{n} p^i = \dfrac{p^{n+1}-1}{p-1}$, $\forall n \in \mathbb{N}$. (c) $\sigma_0(p_n\#) = 2^n$ where $p_n\#$ denotes the primorial*

Chương 17

Advanced Techniques – Các Kỹ Thuật Nâng Cao

Problem 157 (CSES Problem Set/meet in the middle). *You are given an array of $n \in \mathbb{N}^\star$ numbers. In how many ways can you choose a subset of the numbers with sum $x \in \mathbb{N}^\star$?*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the array size & the required sum. The 2nd line has n integers $t_1, \ldots, t_n$: the numbers in the array.*

Output. *Print the number of ways you can create the sum x.*

Constraints. $n \in [40], x \in [10^9], t_i \in [10^9], \forall i \in [n]$.

Sample.

meet_middle.inp	meet_middle.out
4 5 1 2 3 2	3

Problem 158 (CSES Problem Set/Hamming distance). *The Hamming distance between 2 strings a, b of equal length is the number of positions where the strings differ. You are given $n \in \mathbb{N}^\star$ bit strings, each of length $k \in \mathbb{N}^\star$ & your task is to calculate the minimum Hamming distance between 2 strings.*

Input. *The 1st input line has 2 integer $n \in \mathbb{N}^\star$: the number of bit strings & their length. Then there are n lines each consisting of 1 bit string of length k.*

Output. *Print the minimum Hamming distance between 2 strings.*

Constraints. $2 \leq n \leq 2 \cdot 10^4, k \in [30]$.

Sample.

Hamming_distance.inp	Hamming_distance.out
3 5 2 0 2	3

C++:

1. NHT's C++: Hamming distance.

```
#include <bits/stdc++.h>
using namespace std;

const int maxN = 2e4;
int N, ans, b[maxN];

int scanBinary() {
    char c;
    int res = 0;
    while ((c = getchar()) != '\n') {
        res <<= 1;
        res += (c - '0') & 1;
    }
```

```
    return res;
}

int main() {
    scanf("%d %d ", &N, &ans);
    for (int i = 0; i < N; i++)
    b[i] = scanBinary();

    for (int i = 0; i < N; i++)
    for (int j = i + 1; j < N; j++)
    ans = min(ans, __builtin_popcount(b[i] ^ b[j]));

    printf("%d\n", ans);
}
```

Problem 159 (CSES Problem Set/corner subgrid check). *You are given a grid of letters. Find subgrids whose height & width is at least 2 & all the corners have the same letter. For each letter, check if there is a valid subgrid whose corners have that letter.*

Input. *The 1st input line has 2 integers $n, k \in \mathbb{N}^\star$: the size of the grid & the number of letters. The letters are the 1st k uppercase letters. After this, there are n lines that describe the grid. Each line has n letters.*

Output. *Print k lines: for each letter,* `YES` *if there is a valid subgrid &* `NO` *otherwise.*

Constraints. $n \in [3000], k \in [26]$.

Sample.

corner_subgrid_check.inp	corner_subgrid_check.out
4 5	YES
AAAA	YES
CBBC	NO
CBBE	NO
AAAA	NO

Problem 160 (CSES Problem Set/corner subgrid count). *You are given an $n \times n$ grid whose each square is either black or white. A subgrid is called* beautiful *if its height & width is at least 2 & all of its corners are black. How many beautiful subgrids are there within the given grid?*

Input. *The 1st input line has an integer $n \in \mathbb{N}^\star$: the size of the grid. Then there are n lines describing the grid: 1 means that a square is black & 0 means it is white.*

Output. *Print the number of beautiful subgrids.*

Constraints. $n \in [3000]$.

Sample.

corner_subgrid_count.inp	corner_subgrid_count.out
5	4
00010	
11111	
00110	
11001	
00010	

Problem 161 (CSES Problem Set/reachable nodes). *A directed acyclic graph consists of $n \in \mathbb{N}^\star$ nodes & $m \in \mathbb{N}^\star$ edges. The nodes are numbered $1, 2, \ldots, n$. Calculate for each node the number of nodes you can reach from that node (including the node itself).*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of nodes & edges. Then there are m lines describing the edges. Each line has 2 distinct integers $a, b \in \mathbb{N}^\star$: there is an edge from node a to node b.*

Output. *Print n integers: for each node the number of reachable nodes.*

Constraints. $n \in [5 \cdot 10^4], m \in [10^5]$.

Sample.

reachable_node.inp	reachable_node.out
5 6 1 2 1 3 1 4 2 3 3 5 4 5	5 3 2 2 1

Problem 162 (CSES Problem Set/reachability queries). *A directed graph consists of $n \in \mathbb{N}^\star$ nodes & $m \in \mathbb{N}^\star$ edges. The edges are numbered $1, 2, \ldots, n$. Answer q queries of the form "can you reach node b from node a?"*

Input. *The 1st input line has 3 integers $n, m, q \in \mathbb{N}^\star$: the number of nodes, edges, & queries. Then there are m lines describing the edges. Each line has 2 distinct integers a, b: there is an edge from node a to node b. Finally there are q lines describing the queries. Each line consists of 2 integers a, b: "can you reach node b from node a?"*

Output. *Print the answer for each query: either* `YES` *or* `NO`.

Constraints. $n \in [5 \cdot 10^4], m \in [100], m, q \in [10^5]$.

Sample.

reachability_query.inp	reachability_query.out
4 4 3 1 2 2 3 3 1 4 3 1 3 1 4 4 1	YES NO YES

Problem 163 (CSES Problem Set/cut & paste). *Given a string, process operations where you cut a substring & paste it to the end of the string. What is the final string after all the operations?*

Input. *The 1st input line has 2 integers $n.m \in \mathbb{N}^\star$: the length of the string & the number of operations. The characters of the string are numbered $1, 2, \ldots, n$. The next line has a string of length n that consists of characters* `A-Z`. *Finally, there are m lines that describe the operations. Each line has 2 integers $a, b \in \mathbb{N}^\star$: you cut a substring from position a to position b.*

Output. *Print the final string after all the operations.*

Constraints. $m, n \in [2 \cdot 10^5], 1 \leq a \leq b \leq n$.

Sample.

cut_paste.inp	cut_paste.out
7 2 AYBABTU 3 5 3 5	AYABTUB

Problem 164 (CSES Problem Set/substring reversals). *Given a string, process operations where you reverse a substring of the string. What is the final string after all the operations?*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the length of the string & the number of operations. The characters of the string are numbered $1, 2, \ldots, n$. The next line has a string of length n that consists of characters* `A-Z`. *Finally, there are m lines that describe the operations. Each line has 2 integers a, b: you reverse a substring from position a to position b.*

Output. *Print the final string after all the operations.*

Constraints. $m, n \in [2 \cdot 10^5], 1 \leq a \leq b \leq n$.

Sample.

substring_reversal.inp	substring_reversal.out
7 2 AYBABTU 3 4 4 7	AYAUTBB

Problem 165 (CSES Problem Set/reversals & sums). *Given an array of $n \in \mathbb{N}^\star$ integers, you have to process following operations:*

1. *Reverse a subarray*
2. *Calculate the sum of values in a subarray.*

Input. *The 1st input line has 2 integer $n, m \in \mathbb{N}^\star$: the size of the array & the number of operations. The array elements are numbered $1, 2, \ldots, n$. The next line as n integers $x_1, x_2, \ldots, x_n$: the contents of the array. Finally, there are m lines that describe the operations. Each line has 3 integers t, a, b. If $t = 1$, you should reverse a subarray from a to b. If $t = 2$, you should calculate the sum of values from a to b.*

Output. *Print the answer to each operation where $t = 2$.*

Constraints. $n \in [2 \cdot 10^5], m \in [10^5], x_i \in \overline{0, 10^9}, 1 \le a \le b \le n$.

Sample.

reversal_sum.inp	reversal_sum.out
8 3 2 1 3 4 5 3 4 4 2 2 4 1 3 6 2 2 4	8 9

Problem 166 (CSES Problem Set/necessary roads). *There are $n \in \mathbb{N}^\star$ cities & $m \in \mathbb{N}^\star$ roads between them. There is a route between any 2 cities. A road is called* necessary *if there is no route between some 2 cities after removing that road. Find all necessary roads.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & roads. The cities are numbered $1, 2, \ldots, n$. After this, there are m lines that describe the roads. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a road between cities a & b. There is at most 1 road between 2 cities, & every road connects 2 distinct cities.*

Output. *1st print an integer: the number of necessary roads. After that, print k lines that describe the roads. You may print the roads in any order.*

Constraints. $n \in \overline{2, 10^5}, m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

necessary_road.inp	necessary_road.out
5 5 1 2 1 4 2 4 3 5 4 5	2 3 5 4 5

Problem 167 (CSES Problem Set/necessary cities). *There are $n \in \mathbb{N}^\star$ cities & $m \in \mathbb{N}^\star$ roads between them. There is a route between any 2 cities. A city is called* necessary *if there is no route between some other 2 cities after removing that city (& adjacent roads). Find all necessary cities.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of cities & roads. The cities are numbered $1, 2, \ldots, n$. After this, there are m lines that describe the roads. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is a road between cities a & b. There is at most 1 road between 2 cities, & every road connects 2 distinct cities.*

Output. *1st print an integer: the number of necessary cities. After that, print a list of k cities. You may print the cities in any order.*

Constraints. $n \in \overline{2, 10^5}, m \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

necessary_city.inp	necessary_city.out
5 5 1 2 1 4 2 4 3 5 4 5	2 4 5

Problem 168 (CSES Problem Set/Eulerian subgraphs). *You are given an undirected graph that has $n \in \mathbb{N}^\star$ nodes & $m \in \mathbb{N}^\star$ edges. We consider subgraphs that have all nodes of the original graph & some of its edges. A subgraph is called Eulerian if each node has even degree. Count the number of Eulerian subgraphs modulo $10^9 + 7$.*

Input. *The 1st input line has 2 integers $n, m \in \mathbb{N}^\star$: the number of nodes & edges. The nodes are numbered $1, 2, \ldots, n$. After this, there are m lines that describe the edges. Each line has 2 integers $a, b \in \mathbb{N}^\star$: there is an edge between nodes a & b. There is at most 1 edge between 2 nodes, & each edge connects 2 distinct nodes.*

Output. *Print the number of Eulerian subgraphs modulo $10^9 + 7$.*

Constraints. $n \in [10^5], m \in [100], x_i \in \{0, 1, \ldots, m\}$.

Sample.

Eulerian_subgraph.inp	Eulerian_subgraph.out
4 3 1 2 1 3 2 3	2

Explanation. *You can either keep or remove all edges, so there are 2 possible Eulerian subgraphs.*

Problem 169 (CSES Problem Set/monster game I). *You are playing a game that consists of $n \in \mathbb{N}^\star$ levels. Each level has a monster. On levels $1, 2, \ldots, n-1$, you can either kill or escape the monster. However, on level n you must kill the final monster to win the game. Killing a monster takes sf time where s is the monster's strength & f is your skill factor (lower skill factor is better). After killing a monster, you get a new skill factor. What is the minimum total time in which you can win the game?*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the number of levels & your initial skill factor. The 2nd line has n integers $s_1, s_2, \ldots, s_n \in \mathbb{N}^\star$: each monster's strength. The 3rd line has n integers $f_1, f_2, \ldots, f_n n \in \mathbb{N}^\star$: your new skill factor after killing a monster.*

Output. *Print 1 integer: the minimum total time to win the game.*

Constraints. $n \in [2 \cdot 10^5], x \in [10^6], 1 \le s_1 \le s_2 \le \cdots \le s_n \le 10^6, x \ge f_1 \ge f_2 \ge \cdots \ge f_n \ge 1$.

Sample.

monster_game_I.inp	monster_game_I.out
5 100 20 30 30 50 90 90 60 20 20 10	4800

Explanation. *The best way to play is to kill the 3rd & 5th monsters.*

Problem 170 (CSES Problem Set/monster game II). *You are playing a game that consists of $n \in \mathbb{N}^\star$ levels. Each level has a monster. On levels $1, 2, \ldots, n-1$, you can either kill or escape the monster. However, on level n you must kill the final monster to win the game. Killing a monster takes sf time where s is the monster's strength & f is your skill facto. After killing a monster, you get a new skill factor (lower skill factor is better). What is the minimum total time in which you can win the game?*

Input. *The 1st input line has 2 integers $n, x \in \mathbb{N}^\star$: the number of levels & your initial skill factor. The 2nd line has n integers $s_1, s_2, \ldots, s_n \in \mathbb{N}^\star$: each monster's strength. The 3rd line has n integers $f_1, f_2, \ldots, f_n n \in \mathbb{N}^\star$: your new skill factor after killing a monster.*

Output. *Print 1 integer: the minimum total time to win the game.*

Constraints. $n \in [2 \cdot 10^5], x \in [10^6], x \in [10^6], s_i, f_i \in [10^6]$.

Sample.

monster_game_II.inp	monster_game_II.out
5 100 50 20 30 90 30 60 20 20 10 90	2600

Explanation. *The best way to play is to kill the 2nd & 5th monsters.*

Problem 171 (CSES Problem Set/subarray squares)**.** *Given an array of $n \in \mathbb{N}^\star$ elements, divide into $k \in \mathbb{N}^\star$ subarrays. The cost of each subarray is the square of the sum of the values in the subarray. What is the minimum total cost if you act optimally?*

Input. *The 1st input line has 2 integers $n, k \in \mathbb{N}^\star$: the array elements & the number of subarrays. The array elements are numbered $1, 2, \ldots, n$. The 2nd line has n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*

Output. *Print 1 integer: the minimum total cost.*

Constraints. $1 \leq k \leq n \leq 3000, x_i \in [1o^5]$, $\forall i \in [n]$.

Sample.

subarray_square.inp	subarray_square.out
8 3 2 3 1 2 2 3 4 1	110

Explanation. *An optimal solution is $[2, 3, 1], [2, 2, 3], [4, 1]$, whose cost is $(2 + 3 + 1)^2 + (2 + 2 + 3)^2 + (4 + 1)^2 = 110$.*

Problem 172 (CSES Problem Set/houses & schools)**.** *There are $n \in \mathbb{N}^\star$ houses on a street, numbered $1, 2, \ldots, n$. The distance of houses a, b is $|a - b|$. You know the number of children in each house. Establish k schools in such a way that each school is in some house. Then, each child goes to the nearest school. What is the minimum total walking distance of the children if you act optimally?*

Input. *The 1st input line has 2 integers $n, k \in \mathbb{N}^\star$: the number of houses & the number of schools. The houses are numbered $1, 2, \ldots, n$. After this, there are n integers $c_1, c_2, \ldots, c_n$: the number of children in each house.*

Output. *Print the minimum total distance.*

Constraints. $1 \leq k \leq n \leq 3000, c_i \in [10^9]$, $\forall i \in [n]$.

Sample.

house_school.inp	house_school.out
6 2 2 7 1 4 6 4	11

Explanation. *Houses $2, 5$ will have schools.*

Problem 173 (CSES Problem Set/Knuth division)**.** *Given an array of n numbers, divide it into n subarrays, each of which has a single element. On each move, you may choose any subarray & split it into 2 subarrays. The cost of such a move is the sum of values in the chosen subarray. What is the minium total cost if you act optimally?*

Input. *The 1st input line has an integers $n \in \mathbb{N}^\star$: the array size. The array elements are numbered $1, 2, \ldots, n$. The 2nd line has n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*

Output. *Print 1 integer: the minimum total cost.*

Constraints. $n \in [5000], x_i \in [10^9]$.

Sample.

Knuth_division.inp	Knuth_division.out
5 2 7 3 2 5	43

Problem 174 (CSES Problem Set/apples & bananas)**.** *There are $n \in \mathbb{N}^\star$ apples & $m \in \mathbb{N}^\star$ bananas, & each of them has an integer weight between $1, \ldots, k$. Calculate, for each weight w between $2, \ldots, 2k$, the number of ways we can choose an apple & a banana whose combined weight is w.*

Input. *The 1st input line contains 3 integers $k, n, m \in \mathbb{N}^\star$: the number K, the number of apples & the number of bananas. The next line contains n integers $a_1, a_2, \ldots, a_n$: weight of each apple. The last line contains m integers $b_1, b_2, \ldots, b_m$: weight of each banana.*

Output. *For each integer w between $2, \ldots, 2k$, print the number of ways to choose an apple & a banana whose combined weight is w.*

Constraints. $m, n, k \in [2 \cdot 10^5], a_i, b_j \in [k], \forall i \in [n], \forall j \in [m]$.

Sample.

apple_banana.inp	apple_banana.out
5 3 4 5 2 5 4 3 2 3	0 0 1 2 1 2 4 2 0

Explanation. *E.g., for $w = 8$ there are 4 different ways: we can pick an apple of weight 5 into 2 different ways & a banana of weight 3 in 2 different ways.*

Problem 175 (CSES Problem Set/1 bit positions). *You are given a binary string of length n. Calculate, for every k between $1, \ldots, n-1$, the number of ways we can choose 2 positions i, j s.t. $i - j = k$ & there is a 1-bit at both positions.*

Input. *The only input line has a string that consists only of characters $0, 1$.*

Output. *For every distance k between $1, \ldots, n-1$ print the number of ways we can choose 2 such positions.*

Constraints. $n \in \overline{2, 2 \cdot 10^5}$.

Sample.

one_bit_position.inp	one_bit_position.out
1001011010	1 2 3 0 2 1 0 1 0

Problem 176 (CSES Problem Set/signal processing). *You are given 2 integer sequences: a signal & a mask. Process the signal by moving the mask through the signal from left to right. At each mask position calculate the sum of products of aligned signal & mask values in the part where the signal & the mask overlap.*

Input. *The 1st input line consists of 2 integers $n, m \in \mathbb{N}^\star$: the length of the signal & the length of the mask. The next line consists of n integers $a_1, a_2, \ldots, a_n$ defining the signal. The last line consists of m integers $b_1, b_2, \ldots, b_m$ defining the mask.*

Output. *Print $n + m - 1$ integer: the sum of products of aligned values at each mask position from left to right.*

Constraints. $m, n \in [2 \cdot 10^5], a_i, b_i \in [100]$.

Sample.

signal_processing.inp	signal_processing.out
5 3 1 3 2 1 4 1 2 3	3 11 13 10 16 9 4

Explanation. *E.g., at the 2nd mask position the sum of aligned products is $2 \cdot 1 + 3 \cdot 3 = 11$.*

Problem 177 (CSES Problem Set/new roads queries). *There are $n \in \mathbb{N}^\star$ cities in Byteland but no roads between them. However, each day, a new road will be built. There will be a total of $m \in \mathbb{N}^\star$ roads. Process $q \in \mathbb{N}^\star$ queries of the form: "after how many days can we travel from city a to city b for the 1st time?"*

Input. *The 1st input line has 3 integers $n, m, q \in \mathbb{N}^\star$: the number of cities, roads, & queries. The cities are numbered $1, 2, \ldots, n$. After this, there are m lines that describe the roads in the order they are built. Each line has 2 integers a, b: there will be a road between cities a & b. Finally, there are q lines that describe the queries. Each line has 2 integers a, b: we want to travel from city a to city b.*

Output. *For each query, print the number of days, or -1 if it is never possible.*

Constraints. $m, n, q \in [2 \cdot 10^5], a, b \in [n]$.

Sample.

new_roads_query.inp	new_roads_query.out
5 4 3 1 2 2 3 1 3 2 5 1 3 3 4 3 5	2 −1 4

Problem 178 (CSES Problem Set/dynamic connectivity). *Consider an undirected graph that consists of $n \in \mathbb{N}^\star$ nodes & $m \in \mathbb{N}^\star$ edges. There are 2 types of events that can happen:*

1. A new edge is created between nodes a, b.

2. An existing edge between nodes a, b is removed.

Report the number of components after every event.

Input. *The 1st input line has 3 integers $n, m, k \in \mathbb{N}^\star$: the number of nodes, edges, & events. After this there are m lines describing the edges. Each line has 2 integers a, b: there is an edge between nodes a, b. There is at most 1 edge between any pair of nodes. Then there are k lines describing the events. Each line has the form* `t a b` *where $t = 1$ (create a new edge) or $t = 2$ (remove an edge). A new edge is always created between 2 nodes that do not already have an edge between them, & only existing edges can get removed.*

Output. *Print $k+1$ integers: 1st the number of components before the 1st event, & after this the new number of components after each event.*

Constraints. $n \in \overline{2, 10^5}, m, k \in [10^5], a, b \in [n]$.

Sample.

dynamic_connectivity.inp	dynamic_connectivity.out
5 3 3 1 4 2 3 3 5 1 2 5 2 3 5 1 1 2	2 2 2 1

Problem 179 (CSES Problem Set/parcel delivery). *There are $n \in \mathbb{N}^\star$ cities & $m \in \mathbb{N}^\star$ routes through which parcels can be carried from 1 city to another city. For each route, you know the maximum number of parcels & the cost of a single parcel. You want to send k parcels from Syrjälä to Lehmälä. What is the cheapest way to do that?*

Input. *The 1st input line has 3 integers $n, m, k \in \mathbb{N}^\star$: the number of cities, routes, & parcels. The cities are numbered $1, 2, \ldots, n$. City 1 is Syrjälä & city n is Lehmälä. After this, there are m lines that describe the routes. Each line has 4 integers a, b, r, c: there is a route from city a to city b, at most r parcels can be carried through the route, & the cost of each parcel is c.*

Output. *Print 1 integer: the minimum total cost or -1 if there are no solutions.*

Constraints. $2 \le n \le 500, m \in [1000], k \in [100], a, b \in [n], r, c \in [1000]$.

Sample.

parcel_delivery.inp	parcel_delivery.out
4 5 3 1 2 5 100 1 3 10 50 1 4 7 500 2 4 8 350 3 4 2 100	750

Explanation. *1 parcel is delivered through route* $1 \to 2 \to 4$ *(cost* $1 \cdot 450 = 450$*) & 2 parcels are delivered through route* $1 \to 3 \to 4$ *(cost* $2 \cdot 150 = 300$*).*

Problem 180 (CSES Problem Set/task assignement)**.** *A company has* $n \in \mathbb{N}^\star$ *employees & there are* n *tasks that need to be done. We know for each employee the cost of carrying out each task. Every employee should be assigned to exactly* 1 *task. What is the minium total cost if we assign the task's optimally & how could they be assigned?*

Input. *The 1st input line has* 1 *integer* $n \in \mathbb{N}^\star$*: the number of employees & the number of tasks that need to be done. After this, there are* n *lines each consisting of* n *integers. The* i*th line consists of integers* $c_{i1}, c_{i2}, \ldots, c_{in}$*: the cost of each task when it is assigned to the* i*th employee.*

Output. *1st print the minimum total cost. Then print* n *lines each consisting of* 2 *integers* a, b*: you assign the* b*th task to the* a*th employee. If there are multiple solutions you can print any of them.*

Constraints. $n \in [200], c_{ij} \in [1000]$.

Sample.

task_assignement.inp	task_assignement.out
4	33
17 8 16 9	1 4
7 15 12 19	2 1
6 9 10 11	3 3
14 7 13 10	4 2

Explanation. *The minium total cost is* 33*. We can reach this by assigning employee 1 task 4, employee 2 task 1, employee 3 task 3, & employee 4 task 2. This will cost* $9 + 7 + 10 + 7 = 33$.

Problem 181 (CSES Problem Set/distinct routes II)**.** *A game consists of* $n \in \mathbb{N}^\star$ *rooms &* $m \in \mathbb{N}^\star$ *teleporters. At the beginning of each day, you start in room* 1 *& you have to reach room* n*. You can use each teleporter at most once during the game. You want to play the game for exactly* k *days. Every time you use any teleporter you have to pay* 1 *coin. What is the minimum number of coins you have to pay during* k *days if you play optimally?*

Input. *The 1st input line has* 3 *integers* $n, m, k \in \mathbb{N}^\star$*: the number of rooms, the number of teleporters & the number of days you play the game. The rooms are numbered* $1, 2, \ldots, n$*. After this, there are* m *lines describing the teleporters. Each line has* 2 *integers* a, b*: there is a teleporter from room* a *to room* b*. There are no* 2 *teleporters whose starting & ending room are the same.*

Output. *1st print* 1 *integer: the minimum number of coins you have to pay if you pay optimally. Then, print* k *route descriptions according to the example. You can print any valid solution. If it is not possible to play the game for* k *days, print only* -1.

Constraints. $2 \le n \le 500, m \in [1000], 1 \le k \le n - 1, 1 \le a, b \le n$.

Sample.

distinct_routes_II.inp	distinct_routes_II.out
8 10 2	6
1 2	4
1 3	1 2 4 8
2 5	4
2 4	1 3 5 8
3 5	
3 6	
4 8	
5 8	
6 7	
7 8	

Chương 18

Sliding Window Problems – Các Bài Toán Về Cửa Sổ Trượt

The sliding window technique is 1 of common techniques in digital image processing & Computer Vision.

Chương 19

Interactive Problems – Các Bài Toán Tương Tác

Chương 20

Bitwise Operations – Các Phép Toán Bitwise

Chương 21

Construction Problems – Các Bài Toán Xây Dựng

Chương 22

Advanced Graph Problems – Các Bài Toán Đồ Thị Nâng Cao

Chương 23

Counting Problems – Các Bài Toán Đếm

Problem 182 (IMO2008P5)**.** *Let $n, k \in \mathbb{N}^\star$, $k \geq n$, $k - n \vdots 2$. Let $2n$ lamps labeled $1, 2, \ldots, 2n$ be given, each of which can be either on or off. Initially all the lamps are off. We consider sequences of* steps*: at each step 1 of the lamps is switched (from on to off or from off to on). Let N be the number of such sequences consisting of k steps & resulting in the state where lamps 1 through n are all on, & laps $n+1$ through $2n$ are all off. Let M be the number of such sequences consisting of k steps, resulting in the state where lamps 1 through n are all on, & lamps $n+1$ through $2n$ are all off, but where none of the lamps $n+1$ through $2n$ is ever switched on. Determine the ratio $\frac{M}{N}$.*

Bài toán 46 ([VL24], p. 12, IMO2008P5)**.** *Giả sử $n, k \in \mathbb{N}^\star$, $k \geq n$, $k - n \vdots 2$. Cho $2n$ bóng đèn được đánh số từ 1 đến $2n$; mỗi bóng có thể sáng hoặc tắt. Tại thời điểm ban đầu mỗi bóng đều tắt. Xét các dãy gồm các bước: tại mỗi bước, công tắc của 1 trong các bóng đèn được bật (từ sáng chuyển thành tắt hoặc từ tắt chuyển thành sáng). Giả sử N là số các dãy mà mỗi dãy gồm k bước & kết thúc ở trạng thái: các bóng đèn từ 1 đến n sáng, các bóng từ $n+1$ đến $2n$ tắt. Giả sử M là số các dãy mà mỗi dãy gồm k bước & cũng kết thúc ở trạng thái: các bóng đèn từ 1 đến n sáng, các bóng từ $n+1$ đến $2n$ tắt, nhưng trong quá trình đó không 1 công tắc nào của các bóng từ $n+1$ đến $2n$ được bật. Tính tỷ số $\frac{M}{N}$.*

Input. *Chỉ chứa 1 dòng gồm 2 số $n, k \in \mathbb{N}^\star$, $k \geq n$, $k - n \vdots 2$ (viết* `if else` *để kiểm tra điều kiện này).*

Output. *Tỷ số $\frac{M}{N}$.*

Chương 24

Additional Problems

Chương 25

Combinatorial Optimization – Tối Ưu Tổ Hợp

Contents

25.1 Some combinatorics problems in Mathematical Olympiads – Vài bài toán tổ hợp trong các kỳ thi Olympic Toán

Problem 183 (IMO2009P6). *Let $a_1, a_2, \ldots, a_n$ be distinct positive integers & let M be a set of $n-1$ positive integers not containing $s = \sum_{i=1}^{n} a_i$. A grasshopper is to jump along the real axis, starting at the point 0 & making n jumps to the right with lengths $a_1, a_2, \ldots, a_n$ in some order. Prove that the order can be chosen in such a way that the grasshopper never lands on any point in M.*

Bài toán 47 ([VL24], 6., p. 14, IMO2009P6). *Giả sử $a_1, a_2, \ldots, a_n$ là $n \in \mathbb{N}^\star$ số nguyên dương khác nhau từng cặp & M là tập hợp gồm $n-1$ số nguyên dương không chứa số $s = \sum_{i=1}^{n} a_i$. 1 con châu chấu nhảy dọc theo trục thực, xuất phát từ điểm 0 & tiến hành n bước nhảy về bên phải với độ dài n bước nhảy là $a_1, a_2, \ldots, a_n$ theo 1 thứ tự nào đó. Chứng minh con châu chấu có thể chọn thứ tự các bước nhảy sao cho nó không bao giờ nhảy lên bất kỳ điểm nào thuộc M.*

Bài toán 48 (CP version of IMO2009P6). *Giả sử $a_1, a_2, \ldots, a_n$ là $n \in \mathbb{N}^\star$ số nguyên dương khác nhau từng cặp & M là tập hợp gồm $n-1$ số nguyên dương $b_1, b_2, \ldots, b_{n-1}\mathbb{N}^\star$ không chứa số $s = \sum_{i=1}^{n} a_i$. 1 con châu chấu nhảy dọc theo trục thực, xuất phát từ điểm 0 & tiến hành n bước nhảy về bên phải với độ dài n bước nhảy là $a_1, a_2, \ldots, a_n$ theo 1 thứ tự nào đó. Đếm số cách để con châu chấu có thể chọn thứ tự các bước nhảy sao cho nó không bao giờ nhảy lên bất kỳ điểm nào thuộc M & liệt kê cụ thể các bước nhảy của từng cách đó.*

Input. *Dòng 1 chứa 1 số nguyên dương $n \in \mathbb{N}^\star$. Dòng 2 chứa n số nguyên dương khác nhau từng cặp $a_1, a_2, \ldots, a_n \in \mathbb{N}^\star$. Dòng 3 chứa $n-1$ số nguyên dương $b_1, b_2, \ldots, b_{n-1}\mathbb{N}^\star$ là $n-1$ phần tử của tập M (dùng lệnh* `if else` *để kiểm tra xem số $s = \sum_{i=1}^{n} a_i$ có thuộc M hay không).*

Output. *Dòng 1 chứa số cách N nhảy thỏa mãn. N dòng tiếp theo liệt kê cụ thể các bước nhảy của từng cách đó.*

Note 1. *Hình như bài IMO2009P6 là 1 trong các bài IMO khó nhất về mặt Toán học. Thư xem CP version coi khó tương đương không.*

Problem 184 (IMO2010P5). *In each of 6 boxes $B_1, B_2, B_3, B_4, B_5, B_6$ there is initially 1 coin. There are 2 types of operation allowed:*

1. *Type 1: Choose a nonempty box B_i with $i \in [5]$. Remove 1 coin from B_i & add 2 coins to B_{i+1}.*
2. *Type 2: Choose a nonempty box B_i with $i \in [4]$. Remove 1 coin from B_i & exchange the contents of (possibly empty) boxes B_{i+1} & B_{i+2}.*

Determine whether there is a finite sequence of such operations that results in boxes

$$B_1, B_2, B_3, B_4, B_5$$

being empty & box B_6 containing exactly $2010^{2010^{2010}}$ coins. (Note that $a^{b^c} = a^{(b^c)}$.)

Bài toán 49 ([VL24], 5., p. 16, IMO2010P5)**.** *Mỗi 1 hộp trong* 6 *hộp* B_1, B_2, B_3, B_4, B_5 *ban đầu chứa* 1 *đồng xu. Cho phép tiến hành* 2 *loại thao tác:*

1. *Chọn 1 hộp không rỗng* B_i *với* $i \in [5]$*. Lấy* 1 *đồng xu ra khỏi* B_i *& bỏ thêm* 2 *đồng xu vào* B_{i+1}*.*
2. *Chọn 1 hộp không rỗng* B_i *với* $i \in [4]$*. Lấy* 1 *đồng xu ra khỏi* B_i *& tráo đổi số đồng xu đựng trong các hộp (có thể rỗng)* B_{i+1}, B_{i+2} *cho nhau.*

Tồn tại hay không 1 *dãy hữu hạn thao tác như trên sao cho đi đến kết quả cuối cùng là* 5 *hộp* B_1, B_2, B_3, B_4, B_5 *đều rỗng, còn hộp* B_6 *đựng đúng* $2010^{2010^{2010}}$ *đồng xu?*

Bài toán 50 (CP version of IMO2010P5)**.** *Mỗi 1 hộp trong* 6 *hộp* $B_1, B_2, B_3, B_4, B_5, B_6$ *ban đầu chứa* 1 *đồng xu. Cho phép tiến hành* 2 *loại thao tác:*

1. *Chọn 1 hộp không rỗng* B_i *với* $i \in [5]$*. Lấy* 1 *đồng xu ra khỏi* B_i *& bỏ thêm* 2 *đồng xu vào* B_{i+1}*.*
2. *Chọn 1 hộp không rỗng* B_i *với* $i \in [4]$*. Lấy* 1 *đồng xu ra khỏi* B_i *& tráo đổi số đồng xu đựng trong các hộp (có thể rỗng)* B_{i+1}, B_{i+2} *cho nhau.*

Tồn tại hay không 1 *dãy hữu hạn thao tác như trên sao cho đi đến kết quả cuối cùng là trạng thái* B_i *chứa* a_i *đồng xu,* $\forall i \in [6]$*.*

Input. 6 *số nguyên* $a_1, a_2, a_3, a_4, a_5, a_6$*.*

Output. NO *nếu không thể đi đến trạng thái cuối* $\{|B_i|\}_{i=1}^6 = \{a_i\}_{i=1}^6$*.* YES *nếu có thể đi đến trạng thái cuối* $\{|B_i|\}_{i=1}^6 = \{a_i\}_{i=1}^6$*, sau đó in ra 1 dãy hữu hạn thao tác thỏa mãn (bất cứ dãy thao tác thỏa mãn nào cũng được, tính duy nhất không/ chưa (?) được đảm bảo ở đây).*

Bài toán 51 (Generalized CP version of IMO2010P5)**.** *(a) Mỗi 1 hộp trong* $n \in \mathbb{N}^\star$ *hộp* B_i *ban đầu chứa* $a_i \in \mathbb{N}$ *đồng xu,* $\forall i \in [n]$*. Cho phép tiến hành* 2 *loại thao tác:*

1. *Chọn 1 hộp không rỗng* B_i *với* $i \in [n-1]$*. Lấy* 1 *đồng xu ra khỏi* B_i *& bỏ thêm* b_i *đồng xu vào* B_{i+1}*.*
2. *Chọn 1 hộp không rỗng* B_i *với* $i \in [n-2]$*. Lấy* 1 *đồng xu ra khỏi* B_i *& tráo đổi số đồng xu đựng trong các hộp (có thể rỗng)* B_{i+1}, B_{i+2} *cho nhau.*

Tồn tại hay không 1 *dãy hữu hạn thao tác như trên sao cho đi đến kết quả cuối cùng là trạng thái* B_i *chứa* c_i *đồng xu,* $\forall i \in [n]$*.*

Input. *Dòng 1 chứa 1 số nguyên dương* $n \in \mathbb{N}^\star$*. Dòng 2 chứa* n *số nguyên* $a_1, a_2, \ldots, a_n$*. Dòng 3 chứa* $n-1$ *số nguyên dương* $c_1, c_2, \ldots, c_{n-1}$*.*

Output. *In ra* NO *nếu không thể đi đến trạng thái cuối* $\{|B_i|\}_{i=1}^n = \{c_i\}_{i=1}^n$*. In ra* YES *nếu có thể đi đến trạng thái cuối* $\{|B_i|\}_{i=1}^6 = \{c_i\}_{i=1}^n$*, sau đó in ra 1 dãy hữu hạn thao tác thỏa mãn (bất cứ dãy thao tác thỏa mãn nào cũng được, tính duy nhất không/ chưa (?) được đảm bảo ở đây). (b) Có thể mở rộng từ* 2 *thao tác đã mô tả cho vài thao tác khác phức tạp hơn được không?*

Problem 185 (IMO2011P4)**.** *Let* $n \in \mathbb{N}^\star$*. We are given a balance & n weights of weight* $2^0, 2^1, \ldots, 2^{n-1}$*. We are to place each of the n weights on the balance, 1 after another, in such a way that the right pan is never heavier than the left pan. At each step we choose* 1 *of the weights that has not yet been placed on the balance, & place it on either the left pan or the right pan, until all of the weights have been placed. Determine the number of ways in which this can be done.*

Bài toán 52 ([VL24], 5., p. 16, IMO2010P5)**.** *Cho* $n \in \mathbb{N}^\star$*. Cho 1 cái cân* 2 *đĩa & n quả cân với trọng lượng là* $2^0, 2^1, \ldots, 2^{n-1}$*. Ta muốn đặt lên cái cân mỗi* 1 *trong n quả cân, lần lượt từng quả 1, theo cách để đảm bảo đĩa cân bên phải không bao giờ nặng hơn đĩa cân bên trái. Ở mỗi bước ta chọn* 1 *trong các quả cân chưa được đặt lên cân, rồi đặt nó hoặc vào đĩa bên trái, hoặc vào đĩa bên phải, cho đến khi tất cả các quả cân đều đã được đặt trên cân. Đếm số cách để thực hiện được mục tiêu.*

Problem 186 (IMO2012P3)**.** *The* liar's guessing game *is a game played between* 2 *players* A, B*. The rules of the game depend on* 2 *positive integers* $k, n \in \mathbb{N}^\star$ *which are known to both players. At the start of the game A chooses integers* x, N *with* $x \in [N]$*. Player A keeps x secret, & truthfully tells N to player B. Player B now tries to obtain information about x by asking player A questions as follows: each question consists of B specifying an arbitrary set S of positive integers (possibly one specified in some previous question), & asking A whether x belongs to S. Player B may asks as many such questions as he wishes. After each question, player A must immediately it with* yes *or* no*, but is allowed to lie as many times as she wants; the only restriction is that, among any* $k+1$ *consecutive answers, at least* 1 *answer must be truthful. After B has asked as many questions as he wants, he must specify a set X of at most n positive integers. If* $x \in X$*, then B wins; otherwise, he loses. Prove: (a) If* $n \geq 2^k$*, then B can guarantee a win. (b) For all sufficiently large k, there exists an integer* $n \geq 1.99^k$ *s.t. B cannot guarantee a win.*

Bài toán 53 ([VL24], IMO2012P3)**.** *Trò chơi* nói dối & đoán *là 1 trò chơi giữa* 2 *người chơi A & B. Luật chơi dựa trên số số nguyên dương* $k, n \in \mathbb{N}^\star$ *mà cả* 2 *người chơi đều được biết trước. Khi bắt đầu trò chơi, A chọn các số nguyên* x, N *với* $x \in [N]$*. Người chơi A giữ bí mật số* x*, & thông báo số* N *1 cách trung thực cho người chơi B. Bây giờ B tìm cách nhận thông tin về số* x *bằng cách hỏi người chơi A các câu hỏi: B xác định 1 tập hợp S các số nguyên dương tùy ý & hỏi A liệu* x *có thuộc* S*. Người chơi B có thể hỏi bao nhiêu câu hỏi cũng được, & cũng có thể hỏi lại* 1 *câu hỏi nhiều lần, vào bất cứ lúc nào mà anh ta muốn. Người chơi A phải trả lời mỗi câu hỏi của B ngay lập tức bằng* đúng *hoặc* sai*, nhưng anh ta được phép nói dối nhiều lần 1 cách tùy ý. Sự hạn chế duy nhất ở đây là trong bất kỳ* $k+1$ *câu trả lời liên tiếp nào cũng phải có ít nhất* 1 *câu trả lời thật. Sau khi B đã đặt nhiều câu hỏi như anh ta muốn, anh ta phải xác định 1 tập hợp* X *có không quá* n *số nguyên dương. Nếu* $x \in X$ *thì B thắng cuộc; còn nếu ngược lại, anh ta thua cuộc. Chứng minh: (a) Nếu* $n \geq 2^k$*, thì B có thể đảm bảo thắng cuộc. (b) Với tất cả số* k *đủ lớn, tồn tại* $n \geq 1.99^k$ *sao cho B không thể thắng cuộc.*

Remark 8. *For this kind of lie-&-guess game, watch, e.g., Tomodachi Game (2022.*

25.2 Knapsack problem – Bài toán xếp balô

Bài toán 54 ([Thà13], p. 25, xếp balô, MAX-KNAPSACK)**.** *Cho 1 lô hàng hóa gồm các gói hàng, mỗi gói đều có khối lượng cùng với giá trị cụ thể, & cho 1 chiếc balô. Chọn từ lô này vài gói hàng nào đó & xếp đầy vào balô, nhưng không được quá, sao cho thu được 1* GTLN *có thể.*

Đây là 1 bài toán tối ưu tổ hợp quen thuộc, được ký hiệu là MAX-KNAPSACK & được phát biểu bằng ngôn ngữ Toán học dưới dạng tổng quát:

- Input. Cho 2 dãy số nguyên dương $\{s_i\}_{i=1}^N \cup \{S\} = s_1, \ldots, s_n, S \in \mathbb{N}^\star$ & $\{\nu_i\}_{i=1}^n = \nu_1, \ldots, \nu_n$.
- Task. Tìm 1 tập con $I \subset [n]$ thỏa
$$\sum_{i \in I} s_i \leq S, \ \sum_{i \in I} \nu_i \to \max.$$
- Formulation of an optimization problem.
$$\max_{I \subset [n]} \sum_{i \in I} \nu_i \text{ subject to } \sum_{i \in I} s_i \leq S.$$

25.3 Traveling salesman problem – Bài toán người bán hàng du lịch

Chương 26

Miscellaneous

Contents

26.1 Contributors

In alphabetical order:

1. Võ Ngọc Trâm Anh [VNTA]. C++ codes.
2. Đặng Phúc An Khang [DPAK]. C++ codes.
   - Đặng Phúc An Khang. *Combinatorics & Number Theory in Competitive Programming – Tổ Hợp & Lý Thuyết Số trong Lập Trình Thi Đấu.*
   - Đặng Phúc An Khang. *Hướng Đến Kỳ Thi Olympic Tin học Sinh Viên Toàn Quốc & ICPC 2025.* URL: https://github.com/GrootTheDeveloper/OLP-ICPC/blob/master/2025/COMPETITIVE_REPORT.pdf.
3. Nguyễn Lê Anh Khoa. C++ codes.
4. Anh/thầy Lê Phúc Lữ. Tặng NQBH quyển [VL24] để tác giả có thể sáng tạo các bài toán với phiên bản lập trình thi đấu (Competitive Programming version) tương ứng các bài VMO, VN TST, IMO tương ứng.
5. Phan Vĩnh Tiến. C++ codes.

26.2 Donate or Buy Me Coffee

Donate (but do not donut) or buy me some coffee via NQBH's bank account information at https://github.com/NQBH/publication/blob/master/bank/NQBH_bank_account_information.

26.3 See also

1. *Vietnamese Mathematical Olympiad for High School- & College Students (VMC) – Olympic Toán Học Học Sinh & Sinh Viên Toàn Quốc.*

   PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/VMC/NQBH_VMC.pdf.

   TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/VMC/NQBH_VMC.tex.

   - Codes:
     - C++ code: https://github.com/NQBH/advanced_STEM_beyond/tree/main/VMC/C++.
     - Python code: https://github.com/NQBH/advanced_STEM_beyond/tree/main/VMC/Python.
   - Resource: https://github.com/NQBH/advanced_STEM_beyond/tree/main/VMC/resource.
   - Figures: https://github.com/NQBH/advanced_STEM_beyond/tree/main/VMC/figure.

Tài liệu tham khảo

[AB20] Dorin Andrica and Ovidiu Bagdasar. *Recurrent sequences—key results, applications, and problems*. Problem Books in Mathematics. Springer, Cham, [2020] ©2020, pp. xiv+402. ISBN: 978-3-030-51502-7; 978-3-030-51501-0. DOI: 10.1007/978-3-030-51502-7. URL: https://doi.org/10.1007/978-3-030-51502-7.

[Ber05] Dimitri P. Bertsekas. *Dynamic programming and optimal control. Vol. I*. Third. Athena Scientific, Belmont, MA, 2005, pp. xvi+543. ISBN: 1-886529-26-4.

[Ber07] Dimitri P. Bertsekas. *Dynamic programming and optimal control. Vol. II*. Third. Athena Scientific, Belmont, MA, 2007, p. 445.

[Ber12] Dimitri P. Bertsekas. *Dynamic programming and optimal control. Vol. II. Approximate dynamic programming*. Fourth. Athena Scientific, Belmont, MA, 2012, pp. xvii+694. ISBN: 978-1-886529-44-1; 1-886529-44-2; 1-886529-08-6.

[Ber17] Dimitri P. Bertsekas. *Dynamic programming and optimal control. Vol. I*. Fourth. Athena Scientific, Belmont, MA, 2017, pp. xix+555. ISBN: 978-1-886529-43-4; 1-886529-43-4; 1-886529-08-6.

[Đàm+19a] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài Liệu Chuyên Tin Học Bài Tập Quyển 3*. Tái bản lần 2. Nhà Xuất Bản Giáo Dục Việt Nam, 2019, p. 159.

[Đàm+19b] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài Liệu Chuyên Tin Học Quyển 3*. Tái bản lần 3. Nhà Xuất Bản Giáo Dục Việt Nam, 2019, p. 171.

[Đức22] Nguyễn Tiến Đức. *Tuyển Tập 200 Bài Tập Lập Trình Bằng Ngôn Ngữ Python*. Nhà Xuất Bản Đại Học Thái Nguyên, 2022, p. 327.

[Laa20] Antti Laaksonen. *Guide to Competitive Programming: Learning & Improving Algorithms Through Contests*. 2nd edition. Undergraduate Topics in Computer Science. Springer, 2020, pp. xv+309.

[Laa24] Antti Laaksonen. *Guide to Competitive Programming: Learning & Improving Algorithms Through Contests*. 3rd edition. Undergraduate Topics in Computer Science. Springer, 2024, pp. xviii+349.

[Thà13] Lê Công Thành. *Lý Thuyết Độ Phức Tạp Tính Toán*. Nhà Xuất Bản Khoa Học Tự Nhiên & Công Nghệ, 2013, p. 370.

[Thư+21a] Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, and Trần Minh Triết. *Nhập Môn Lập Trình*. Nhà Xuất Bản Khoa Học & Kỹ Thuật, 2021, p. 427.

[Thư+21b] Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, Trần Minh Triết, and Đặng Bình Phương. *Kỹ Thuật Lập Trình*. Nhà Xuất Bản Khoa Học & Kỹ Thuật, 2021, p. 526.

[Tru23a] Vương Thành Trung. *Tuyển Tập Đề Thi Học Sinh Giỏi Cấp Tỉnh Trung Học Cơ Sở & Đề Thi Vào Lớp 10 Chuyên Tin Môn Tin Học*. Nhà Xuất Bản Dân Trí, 2023, p. 220.

[Tru23b] Vương Thành Trung. *Tuyển Tập Đề Thi Học Sinh Giỏi Cấp Tỉnh Trung Học Phổ Thông Tin Học*. Tài liệu lưu hành nội bộ, 2023, p. 235.

[TTK21] Trần Đan Thư, Đinh Bá Tiến, and Nguyễn Tấn Trần Minh Khang. *Phương Pháp Lập Trình Hướng Đối Tượng*. Nhà Xuất Bản Khoa Học & Kỹ Thuật, 2021, p. 401.

[VL24] Lê Anh Vinh and Lê Phúc Lữ. *Tuyển Tập Đề Thi Học Sinh Giỏi Toán Quốc Gia & Chọn Đội Tuyển Quốc Tế Môn Toán*. Nhà Xuất Bản Giáo Dục Việt Nam, 2024, pp. x+588.

[WW16] Yonghui Wu and Jiande Wang. *Data Structure Practice for Collegiate Programming Contests & Education*. 1st edition. CRC Press, 2016, p. 496.

[WW18] Yonghui Wu and Jiande Wang. *Algorithm Design Practice for Collegiate Programming Contests & Education*. 1st edition. CRC Press, 2018, p. 692.