Olympic Tin Học Sinh Viên OLP & ACM-ICPC

Nguyễn Quản Bá Hồng*

Ngày 30 tháng 5 năm 2025

Tóm tắt nội dung

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

- *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.

Mục lục

*A Scientist & Creative Artist Wannabe. Website: https://nqbh.github.io. GitHub: https://github.com/NQBH.
E-mail: nguyenquanbahong@gmail.com, hong.nguyenquanba@umt.edu.vn. Bến Tre & Hồ Chí Minh Cities, Việt Nam.

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'.

1 Basic Competitive Programming – Lập Trình Thi Đấu Cơ Bản

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

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 đầu tiên 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 thứ 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 đầu tiên 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, 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.*

2 Practice for Simple Computing – Thực Hành Tính Toán Đơn Giản

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 2 ([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 3 (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 3 ([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 4 ([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 5 ([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, $S_1 = 0, S_n = S_{n-1} + 50 = 50(n-1) = \frac{\pi r_n^2}{2} \Rightarrow r_n = \sqrt{\frac{100(n-1)}{\pi}}, \forall n \in \mathbb{N}^\star$. Tìm n thỏa mãn $r_{n-1} < \sqrt{x^2+y^2} < r_n$, tuwong □

Problem 6 ([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 7 ([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.*

3 Olympic Tin THCS & THPT

Bài toán 4 ([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.

4 Ad Hoc Problems

- **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.

4.1 Solving Ad Hoc Problems by Mechanism Analysis

Problem 8 ([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* $1960 \leq y \leq 2160$*, 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 5 ([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* $1960 \leq y \leq 2160$*, 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 $\leq 2^b - 1$, 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

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

   n can be calculated by a loop: Initially $i := 1$, 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 6 (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 $b \in \mathbb{N}^\star$, $b \geq 2$, 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 $b \in \mathbb{N}$, $b \geq 2$ 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 $y_0 \leq y$, 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 3 (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 9 ([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 7 ([WW18], 1.1.2, p. 3, "Di 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 đầu tiên 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 đầu tiên 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++:

4.2 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 10 ([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 8 ([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 đầu tiên 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 11 ([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.*

5 VNOI

Bài toán 9 (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 $0 \leq k \leq n$. Cho $n \in \mathbb{N}$. Tính* 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* GPT(n) *tương ứng.*

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

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.

6 Recurrence Relation – Quan Hệ Hồi Quy

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

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

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

$$x_n = f_n(x_{n-1}, \ldots, x_0), \ \forall n \in \mathbb{N}^\star, \tag{2}$$

where $f_n : X^n \to X$ is a function, $\forall n \in \mathbb{N}^\star$. The relations (2) give the rule to construct the term x_n of the sequence $(x_n)_{n \ge 0}$ from the 1st term x_0: $x_1 = f_1(x_0), x_2 = f_2(x_1, x_0), \ldots$, i.e., (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.:

$$\begin{cases} u_0 \in \mathbb{F}, \\ u_n = f_n(n, u_0, u_1, \ldots, u_{n-1}), \ \forall n \in \mathbb{N}^\star, \end{cases} \tag{3}$$

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, $\forall n \in \mathbb{N}^\star$; &

$$\begin{cases} u_1 \in \mathbb{F}, \\ u_n = f_n(n, u_1, \ldots, u_{n-1}), \ \forall n \in \mathbb{N}, \ n \ge 2, \end{cases} \tag{4}$$

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 $f_n : \mathbb{N}_{\ge 2} \times \mathbb{F}^{n-1} \to \mathbb{F}$ is a scalar-valued function of n-dimensional-vector-valued argument, $\forall n \in \mathbb{N}$, $n \ge 2$.

Question 4. *What define uniquely (3)?*

Answer. The solutions $\{u_n\}_{n=0}^{\infty}$ defined by (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 (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*

$$\boxed{\begin{cases} u_0 \in X, \\ u_n = \varphi(n, u_{n-1}), \ \forall n \in \mathbb{N}^\star, \end{cases}} \tag{5}$$

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*

$$\boxed{\begin{cases} u_0, u_1, \ldots, u_{k-1} \in X, \\ \qquad\qquad u_n = \varphi(n, k, u_{n-1}, u_{n-2}, \ldots, u_{n-k}), \ \forall n \in \mathbb{N}, \ n \geq k, \end{cases}} \tag{6}$$

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 (3) or (4); meanwhile the implicit recurrence relations are the recurrence relations that can be given as*

$$\begin{cases} u_0 \in \mathbb{F}, \\ f_n(n, u_0, u_1, \ldots, u_{n-1}, u_n) = 0, \ \forall n \in \mathbb{N}^\star, \end{cases} \tag{7}$$

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, $\forall n \in \mathbb{N}^\star$; &

$$\begin{cases} u_1 \in \mathbb{F}, \\ f_n(n, u_1, \ldots, u_{n-1}, u_n) = 0, \ \forall n \in \mathbb{N}, \ n \geq 2, \end{cases} \tag{8}$$

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 (7) & (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 (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*

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

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?)

6.1 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* $a_1, \dots, a_n, b$*:*

$$y_n = \sum_{i=1}^{k} a_i y_{n-i} + b, \tag{9}$$

or equivalently as

$$y_{n+k} = \sum_{i=1}^{n} a_i y_{n+k-i} + b, \tag{10}$$

7 Dynamic Programming – Quy Hoạch Động

Resource – Tài nguyên.

1. Wikipedia/dynamic programming.
2. [Thư+21b]. Trần Đan 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 10 ([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 11 (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 $i = 0, 1, \ldots, n$, 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:

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

với $\alpha, \beta \in \mathbb{C}$, $a, b : \mathbb{N}_{\geq 2} \to \mathbb{C}$ 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 12 ([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 13 ([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 $a_i \in [10^9]$, $1 \leq i \leq n$.*
- *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 14 ([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 15 ([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 16 ([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 đầu tiên 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 17 ([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 18 ([Tru23a], Vĩnh Phúc HSG9 2022–2023, 2: lật ký tự, p. 210)**.** *Cho xâu S gồm $n \in \mathbb{N}^\star$ ký tự `.` `#` 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 $1 \le n \le 2000$*
- *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:*

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

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

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

Bài toán 19 ([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 20 ([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 $n, k \in \mathbb{N}$, $0 \le k \le n$, 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 21 ([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 22 ([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 23 ([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 24 ([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 12 (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 13 (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 14 (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 15 (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 16 (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 $27 \to 20 \to 18 \to 10 \to 9 \to 0$.

Problem 17 (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 18 (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 19 (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 20 (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 21 (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 22 (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 23 (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 24 (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 25 (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 26 (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 27 (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 28 (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 29 (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 30 (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 31 (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 32 (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 33 (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 one integer: the answer to the problem.*

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

Sample.

counting_number.inp	counting_number.out
123	321

Problem 34 (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 one 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 $[2], [1], [3], [2, 3], [1, 3]$.

8 Combinatorial Optimization – Tối Ưu Tổ Hợp

Bài toán 25 ([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
$$\sum_{i \in I} s_i \le S, \ \sum_{i \in I} \nu_i \to \max.$$
- Formulation of an optimization problem.
$$\max_{I \subset [n]} \sum_{i \in I} \nu_i \text{ subject to } \sum_{i \in I} s_i \le S.$$

9 CSES Problem Set

Link: https://cses.fi/problemset/.

9.1 Introductory Problems

Problem 35 (CSES)**.** *There are n concert tickets available, each with a certain price. Then, m 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 nearest possible price such that it does not exceed the maximum price.*

Input. *1st input line contains $n, m \in \mathbb{N}$: number of tickets & 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, ticket cannot be purchased again. If a customer cannot get any ticket, print -1.*

Constraints. $1 \le m, n \le 2 \cdot 10^5$, $1 \le h_i, t_i \le 10^9$.

9.2 Graph Algorithms

9.3 Range Queries

9.4 Mathematics

Problem 36 (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 37 (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: $S \coloneqq \sum_{i=1}^n a_i \in \{n, \ldots, 6n\}$.

9.5 String Algorithms

9.6 Geometry

9.7 Advanced Techniques

9.8 Additional Problems

10 Miscellaneous

10.1 Contributors

1. Võ Ngọc Trâm Anh. C++ codes.
2. Đặng Phúc An Khang. 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. Phan Vĩnh Tiến. C++ codes.

10.2 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.

10.3 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

[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.

[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.

[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.