Problem & Solution: Elementary Computer Science
Bài Tập & Lời Giải: Tin Học Sơ Cấp

Nguyễn Quản Bá Hồng*

Ngày 4 tháng 6 năm 2023

Tóm tắt nội dung

1 bộ sưu tập các bài toán chọn lọc từ cơ bản đến nâng cao cho Tin học sơ cấp. Phiên bản mới nhất của tài liệu này được lưu trữ ở link sau: GitHub/NQBH/hobby/elementary computer science/problem[1].

Mục lục

*Independent Researcher, Ben Tre City, Vietnam
e-mail: nguyenquanbahong@gmail.com; website: https://nqbh.github.io.

[1] URL: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/problem/NQBH_elementary_computer_science_problem.pdf.

1 Notes on Commands

1.1 Notes on C/C++ commands

Template:

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

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    // Input & Output
    freopen("prob.inp", "r", stdin);
    freopen("prob.out", "w", stdout);
}
// Compile in Terminal:
// g++ -std=c++11 -O2 -Wall test.cpp -o test
```

1.2 Notes on Pascal commands

1.3 Notes on Python commands

1. Để sử dụng các hàm toán học trong Python, cần import thư viện `math` vào chương trình: `from math import *`
2. Để mở file dữ liệu vào `prob.inp` chỉ để đọc dữ liệu & mở file dữ liệu ra `prob.out` để thay đổi dữ liệu trong file: `file = open("prob.inp")` & `file2 = open("prob.out", "w")`.
3. Sắp xếp trong Python có thể thực hiện 1 cách đơn giản nhờ phương thức `sort()` hoặc `sorted()`. Cú pháp:

   ```
   list.sort(reverse = True|False, key = myFunc)
   list.sorted(reverse = True|False, key = myFunc)
   ```

4. Trong Python, để lấy giá trị ngẫu nhiên, sử dụng phương thức `ranint(a, b)` với `a, b` là giới hạn của giá trị cần lấy ngẫu nhiên.

2 Problems in Elementary Mathematics – Bài Toán Tin Học Trong Toán Học Sơ Cấp

Bài toán 1 (Even vs. odd)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để xét tính chẵn lẻ của* $n \in \mathbb{Z}$ *được nhập từ bàn phím.*

- Pascal script: GitHub/NQBH/hobby/elementary computer science/Pascal/even vs. odd.
- Python script: GitHub/NQBH/hobby/elementary computer science/Python/even vs. odd.

```
try:
    n = int(input())
    if n % 2 == 0 :
        print('even')
    else:
        print('odd')
except:
    print('Not be an integer.')
```

```
n = int(input())
if n % 2 == 0:
    print(n, " is even")
else:
    print(n, " is odd")
```

Bài toán 2 (Divisible by)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để kiểm tra liệu* $a \vdots b$ *hay không, với* $a, b \in \mathbb{Z}$ *được nhập từ bàn phím & in ra số dư* r *trong phép chia* a *cho* b*.*

- Pascal script: GitHub/NQBH/hobby/elementary computer science/Pascal/divisible by.
- Python script: GitHub/NQBH/hobby/elementary computer science/Python/divisible by.

```
try:
    a = int(input('nhap a='))
    b = int(input('nhap b='))
    if a%b == 0 :
        print(str(a)+' chia het cho '+str(b))
    else:
        print(str(a)+' khong chia het cho '+str(b))
except:
    print('Day khong phai la 1 so nguyen')
```

Bài toán 3 (Triangle). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để liệu* a, b, c *có phải là độ dài của: (a) 1 tam giác. (b) 1 tam giác nhọn. (c) 1 tam giác vuông. (d) 1 tam giác tù.*

- Python script: GitHub/NQBH/hobby/elementary computer science/Python/triangle.

```
try:
    a,b,c = float(input('Nhap a = ')), float(input('Nhap b = ')), float(input('Nhap c = '))
if (a + b > c) and (a + c > b) and  (b + c > a):
    loai_tg = ''
    if (a*a + b*b == c*c) or (a*a + c*c == b*b) or (b*b + c*c == a*a):
        loai_tg = 'vuong'
    elif (a*a + b*b > c*c) and (a*a + c*c > b*b) and (b*b + c*c > a*a):
        loai_tg = 'nhon'
    else:
        loai_tg = 'tu'
    print('Day la do dai 3 canh cua 1 tam giac ' + loai_tg)
else:
    print('Day khong la do dai 3 canh cua 1 tam giac.')
except:
    print('Day khong phai la 1 so.')
```

Bài toán 4 (Polynomial equation). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để giải phương trình bậc nhất, bậc 2, bậc 3, & bậc 4 với các hệ số thực được nhập từ bàn phím.*

Bài toán 5 (Fibonacci sequence). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để xuất ra màn hình, với* $n \in \mathbb{N}$ *được nhập từ bàn phím: (a) Số Fibonacci thứ* n*. (b)* n *số Fibonacci đầu tiên.*

Bài toán 6 (1st n square roots). *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc 2 của* n *số tự nhiên đầu tiên với* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím.*

Bài toán 7 (Số chính phương – Square number). *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím có phải là số chính phương hay không.*

Bài toán 8 (1st n cube roots). *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc 3 của* n *số tự nhiên đầu tiên với* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím.*

Bài toán 9. *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím có phải là lập phương của 1 số tự nhiên hay không.*

Bài toán 10 (1st n nth roots). *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc* n *của* m *số tự nhiên đầu tiên với* $m, n \in \mathbb{N}^\star$ *được nhập từ bàn phím.*

Bài toán 11. *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số* $m \in \mathbb{Z}$ *được nhập từ bàn phím có phải là lũy thừa bậc* $n \in \mathbb{N}$ *của 1 số tự nhiên hay không với* m, n *được nhập từ bàn phím.*

2.1 Algebraic Expression – Biểu Thức Đại Số

Bài toán 12 ([Vie21], 1., p. 15, Vũng Tàu 2020). *Cho* $a, b, c \in \mathbb{N}^\star$. Yêu cầu: *Tính giá trị của biểu thức* $S = \dfrac{a^2 + b^2 + c^2}{abc} + \sqrt{abc}$.

- Dữ liệu vào: *File* `algebraic_expression.inp` *chứa 3 số nguyên dương* a, b, c*. Mỗi số trên 1 dòng.*
- Kết quả: *Ghi vào File* `algebraic_expression.out` *kết quả* S *tính được (làm tròn lấy 2 chữ số sau phần thập phân). E.g.:*

algebraic_expression.inp	algebraic_expression.out
2 1 2	4.25

Python script: GitHub/NQBH/hobby/elementary computer science/Python/algebraic expression.py[2]. Input: algebraic expression.inp. Output: algebraic expression.out.

Python:

```
from math import sqrt
def algebraic_expression(a, b, c):
    return (a**2 + b**2 + c**2)/(a*b*c) + sqrt(a*b*c) # function f(a,b,c) can be modified
a = int(input()); b = int(input()); c = int(input())
print(algebraic_expression(a,b,c))
```

C++:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    float a, b, c;
    cin >> a >> b >> c;
    printf("%.2f\n", (a*a + b*b + c*c)/(a*b*c) + sqrt(a*b*c));
}
```

Run in terminal:

```
g++ -std=c++11 -O2 -Wall algebraic_expression.cpp -o algebraic_expression
```

Lưu ý 1. *Tương tự, ta có thể tính hầu như bất kỳ hàm số $f(a,b,c)$ 3 biến a,b,c với f là 1 hàm số có thể viết được nhờ thư viện* `math` *của Python. Tổng quát hơn, ta có thể tính bất kỳ hàm số nhiều biến $f(x_1,x_2,\ldots,x_n)$ với x_i, $i=1,2,\ldots,n$, $n\in\mathbb{N}^\star$ là các biến, với f là 1 hàm số có thể viết được nhờ thư viện* `math` *của Python.*

Bài toán 13 ([Vie21], 2., p. 19, Bắc Giang 2020)**.** *Nhà An có 1 trang trại rộng lớn. Do sở thích của An nên bố An chỉ nuôi gà & chó. 1 hôm bố An đố con gái nhà mình nuôi bao nhiêu gà, bao nhiêu chó? Bố An cho biết nhà có tổng số gà & chó là x con. Do số lượng nhiều & khó đếm từng loại nên An chỉ đếm được tổng số chân của gà & chó là y chân. Giúp An trả lời câu đố.*

- Dữ liệu vào: *Đọc từ file văn bản* `toanco.inp` *gồm 2 số nguyên dương x,y trên 1 dòng. 2 số cách nhau 1 khoảng trống ($x\leq 10^5$, $y\leq 4\cdot 10^5$).*
- Kết quả: *Ghi ra file văn bản* `toanco.out` *gồm 2 số tương ứng là số gà & số chó tìm được. 2 số cách nhau 1 khoảng trống. Giả sử bài toán luôn có nghiệm.*

toanco.inp	toanco.out
36 100	22 14

Python script: GitHub/NQBH/hobby/elementary computer science/Python/toanco.py[3]. Input: toanco.inp. Output: toanco.out.

```
file_in = open("toanco.inp")
file_out = open("toanco.out", "w")
s = file_in.readline()
s = s.split()
x = int(s[0])
y = int(s[1])
a = int(2*x - y/2)
b = int(y/2 - x)
file_out.write(str(a) + " " + str(b))
file_in.close()
file_out.close()
```

Bài toán 14 ([Vie21], 4., p. 26, Quãng Ngãi 2020, Lãi suất– Interest rate)**.** *1 người gửi tiền vào ngân hàng có kỳ hạn là c tháng với lãi suất mỗi tháng là $k\%$, số tiền gửi ban đầu là A (đơn vị triệu đồng).*

- Yêu cầu: *Tính số tiền người đó nhận được sau t tháng. Biết tiền lãi mỗi tháng được cộng dồn vào tiền gốc, nếu nhận tiền trước kỳ hạn thì số tiền được tính với lãi suất không kỳ hạn là $h\%$ của số tiền ban đầu A nhân với số tháng đã gửi. Trong trường hợp rút tiền sau kỳ hạn thì số tháng sau kỳ hạn sẽ được tính với lãi suất không kỳ hạn là $h\%$ so với số tiền thu được đã qua kỳ hạn.*

[2] URL: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/algebraic_expression.py.
[3] URL: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/toanco.py.

- Dữ liệu vào: *Tệp văn bản* `bl2.inp` *ghi* 5 *số kỳ hạn* c *(nếu* $c = 0$ *là gửi không kỳ hạn), thời gian gửi* t*, số tiền ban đầu* A*, lãi suất có kỳ hạn* k*, lãi suất không kỳ hạn* h*, các số cách nhau 1 ký tự trắng.*
- Dữ liệu ra: *Tệp văn bản* `bl2.out` *ghi* 1 *số là số tiền nhận được (làm tròn đến* 1 *số lẻ sau dấu chấm thập phân). E.g.:*

`bl2.inp`	`bl2.out`
12 13 100 1.0 0.2	112.9
0 10 100 1.0 0.2	102.0

2.2 Number Theory – Số Học

Bài toán 15 ([Vie21], 3., p. 20, Yên Bái 2020, Tổng nguyên tố)**.** *Viết chương trình nhập vào* 2 *số nguyên* $a, b \in \mathbb{Z}$*,* $0 < a < b$*. (a) Tìm & tính tổng các số nguyên tố của dãy số từ* a *đến* b*. (b) Xuất ra màn hình các số chia hết cho* 5 *của dãy số từ* a *đến* b*. (c)* (Bội của $n \in \mathbb{N}^\star$) *Xuất ra màn hình các số chia hết cho* n *của dãy số từ* a *đến* b *với* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím. E.g., nhập* $a = 6$*,* $b = 22$*. Kết quả tổng các số nguyên tố trong dãy số từ* 6 *đến* 22*:* $7 + 11 + 13 + 17 + 19 = 67$*. Các số chia hết cho* 5 *của dãy số từ* 6 *đến* 22*:* $10, 15, 20$*.*

Bài toán 16 ([Vie21], 4., p. 22, Hải Dương 2020, Số mạnh mẽ)**.** Số mạnh mẽ *là số khi nó chia hết cho 1 số nguyên tố thì cũng chia hết cho cả bình phương của số nguyên tố đó, i.e.,* $a \in \mathbb{N}^\star$ *là số mạnh mẽ* $\Leftrightarrow (a \vdots p \Rightarrow a \vdots p^2, \forall p$*: prime). E.g.,* 25 *là số mạnh mẽ, vì nó chia hết cho số nguyên tố* 5 *& chia hết cho cả* $5^2 = 25$*. Viết chương trình liệt kê các số mạnh mẽ không vượt quá* 1000*.*

See, e.g., Wikipedia/powerful number, MathWorld/powerful number.

Bài toán 17 ([Vie21], 5., p. 23, Việt Nam 2020, Bội chính phương)**.** *Cho 1 dãy số* A *có* n *phần tử. Tìm số nguyên dương* P *nhỏ nhất thỏa mãn: a là số chính phương & a chia hết cho tất cả các phần tử của dãy số* A*.*

- Yêu cầu: *In ra phần dư của phép chia khi chia a cho* $10^9 + 7$*.*
- Dữ liệu vào: *Vào từ thiết bị theo khuôn dạng sau: Dòng đầu tiên chứa số nguyên dương* n *là số lượng phần tử của dãy số. Dòng tiếp theo chứa* n *số nguyên dương là các phần tử của dãy* A*. Các số trên 1 dòng được ghi cách nhau bởi dấu cách.*
- Kết quả: *Ghi ra thiết bị ra gồm 1 số nguyên duy nhất là kết quả của bài toán. E.g.:*

Dữ liệu vào	Dữ liệu ra
3 2 1 3	36

Bài toán 18 ([Vie21], 1., p. 25, Hải Dương 2020, Số hạnh phúc & số buồn bã – Happy- & sad numbers)**.** *Với 1 số nguyên dương bất kỳ, thay thế số đó bằng tổng bình phương các chữ số của nó & cứ lặp lại quá trình đó sẽ có các trường hợp sau xảy ra: Kết thúc bằng* 1 *– ta gọi số đó là* số hạnh phúc/happy number*. Kết thúc bằng* 0 *– ta gọi số đó là* số buồn bã/sad number*. Lặp lại vô hạn lần – số đó không hạnh phúc cũng không buồn bã. E.g., số* 44*: lần 1:* $4^2 + 4^2 = 32$*, lần 2:* $3^2 + 2^2 = 13$*, lần 3:* $1^2 + 3^2 = 10$*, lần 4:* $1^2 + 0^2 = 1$*, nên* 44 *là số hạnh phúc. Viết chương trình để kiểm tra xem ngày sinh của 1 người bất kỳ có phải là số hạnh phúc không?*

Python:

```
max_iter = 1000
def sum_digit_sqr(n):
    sum = 0
    for i in str(n):
        sum += int(i)**2
    return sum
n0 = n = int(input())
count = 0
while n > 1 and count <= max_iter:
    n = sum_digit_sqr(n)
    print(n)
    count += 1
if n == 1:
    print(n0, " is a happy number.")
elif n == 0:
    print(n0, " is a sad number.")
else:
    print(n0, " is neither a happy nor a sad number.")
```

Bài toán 19 ([Vie21], 2., p. 25, Gia Lai 2019, Phân số tối giản – Irreducible fraction)**.** *1 chuỗi được gọi là có dạng phân số nếu nó có dạng* `'tử_số/mẫu_số'`*. Viết chương trình nhập vào chuỗi có dạng phân số, sau đó xuất ra dạng tối giản của phân số đó. E.g., Chuỗi* `'12/15'` *biểu diễn cho phân số. Dạng tối giản của phân số đó là* `'3/5'`*.*

Python:

```
from math import gcd
frac = input()
frac = frac.split("/")
num = int(frac[0])
den = int(frac[1])
gcd = gcd(num, den)
num = int(num/gcd)
den = int(den/gcd)
print(str(num) + "/" + str(den))
```

Bài toán 20 (Tổng tất cả, tổng phần tử chẵn, lẻ, bình phương, lập phương, lũy thừa bậc n, căn bậc 2, 3, & căn bậc n, nghịch đảo, nghịch đảo bình phương, nghịch đảo lập phương, nghịch đảo lũy thừa bậc n, nghịch đảo căn bậc 2, 3, & nghịch đảo căn bậc n – Sums of all, odds, evens, squares, cubes, nth powers, square roots, cube roots, nth roots, reciprocals of square, of cubes, of nth powers, of square roots, of cube roots, of nth roots)**.** *Cho 1 dãy gồm n số nguyên:* $(a_i)_{i=1}^m = a_1, a_2, \ldots, a_m$, $m \in \mathbb{N}^\star$, $a_i \in \mathbb{Z}$, $\forall i = 1, 2, \ldots, m$, *mỗi số có giá trị không vượt quá* 10^9.

- Yêu cầu: *Tính tổng S tất cả các phần tử, tổng* S_{even} *các số chẵn, tổng* S_{odd} *các số lẻ, tổng* S_{sqr} *bình phương, tổng* $S_{\text{sqr,even}}$ *bình phương các số chẵn, tổng* $S_{\text{sqr,odd}}$ *bình phương các số lẻ, tổng* S_{cb} *lập phương, tổng* $S_{\text{cb,even}}$ *lập phương các số chẵn, tổng* $S_{\text{cb,odd}}$ *lập phương các số lẻ, tổng* $S_{\text{pwr},n}$ *lũy thừa bậc n, tổng* $S_{\text{pwr,even},n}$ *lũy thừa bậc n các số chẵn, tổng* $S_{\text{pwr,odd},n}$ *lũy thừa bậc n các số lẻ, tổng* S_{sqrt} *căn bậc 2, tổng* $S_{\text{sqrt,even}}$ *căn bậc 2 các số chẵn, tổng* $S_{\text{sqrt,odd}}$ *căn bậc 2 các số lẻ, tổng* S_{cbrt} *căn bậc 3, tổng* $S_{\text{cbrt,even}}$ *căn bậc 3 các số chẵn, tổng* $S_{\text{cbrt,odd}}$ *căn bậc 3 các số lẻ, tổng* $S_{\text{rt},n}$ *căn bậc n của các số, tổng* $S_{\text{rt,even},n}$ *căn bậc n của các số chẵn, tổng* $S_{\text{rt,odd},n}$ *căn bậc n của các số lẻ trong dãy* $(a_i)_{i=1}^m$.
- Dữ liệu: *Dòng đầu tiên chứa* $m \in \mathbb{N}^\star$, $1 \le m \le 10^9$. *Dòng thứ 2 chứa* $n \in \mathbb{N}^\star$. m *dòng tiếp theo, dòng thứ* $i+2$ *chứa* a_i, $\forall i = 1, 2, \ldots, m-1$.

Giải. Công thức toán học tính các tổng:

$$S \coloneqq \sum_{i=1}^{m} a_i = a_1 + a_2 + \cdots + a_m,\ S_{\text{even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i,\ S_{\text{odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i,$$

$$S_{\text{sqr}} \coloneqq \sum_{i=1}^{m} a_i^2 = a_1^2 + a_2^2 + \cdots + a_m^2,\ S_{\text{sqr,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^2,\ S_{\text{sqr,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^2,$$

$$S_{\text{cb}} \coloneqq \sum_{i=1}^{m} a_i^3 = a_1^3 + a_2^3 + \cdots + a_m^3,\ S_{\text{cb,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^3,\ S_{\text{odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^3,$$

$$S_{\text{pwr},n} \coloneqq \sum_{i=1}^{m} a_i^n = a_1^n + a_2^n + \cdots + a_m^n,\ S_{\text{pwr,even},n} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^n,\ S_{\text{pwr,odd},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^n,\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{sqrt}} \coloneqq \sum_{i=1}^{m} \sqrt{a_i} = \sqrt{a_1} + \sqrt{a_2} + \cdots + \sqrt{a_m},\ S_{\text{sqrt,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt{a_i},\ S_{\text{sqrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt{a_i},$$

$$S_{\text{cbrt}} \coloneqq \sum_{i=1}^{m} \sqrt[3]{a_i} = \sqrt[3]{a_1} + \sqrt[3]{a_2} + \cdots + \sqrt[3]{a_m},\ S_{\text{cbrt,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt[3]{a_i},\ S_{\text{cbrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt[3]{a_i},$$

$$S_{\text{rt},n} \coloneqq \sum_{i=1}^{m} \sqrt[n]{a_i} = \sqrt[n]{a_1} + \sqrt[n]{a_2} + \cdots + \sqrt[n]{a_m},\ S_{\text{rt,even},n} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt[n]{a_i},\ S_{\text{rt,odd},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt[n]{a_i},\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{rcpc}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_m},\ S_{\text{rcpc,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i},\ S_{\text{rcpc,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i},$$

$$S_{\text{rcpc,sqr}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^2} = \frac{1}{a_1^2} + \frac{1}{a_2^2} + \cdots + \frac{1}{a_m^2},\ S_{\text{rcpc,sqr,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i^2},\ S_{\text{rcpc,sqr,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^2},$$

$$S_{\text{rcpc,cb}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^3} = \frac{1}{a_1^3} + \frac{1}{a_2^3} + \cdots + \frac{1}{a_m^3},\ S_{\text{rcpc,cb,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i^3},\ S_{\text{rcpc,cb,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^3},$$

$$S_{\text{rcpc,pwr},n} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^n} = \frac{1}{a_1^n} + \frac{1}{a_2^n} + \cdots + \frac{1}{a_m^n},\ S_{\text{rcpc,even,pwr},n} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i^n},\ S_{\text{rcpc,odd,pwr},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^n},\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{rcpc,sqrt}} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt{a_i}} = \frac{1}{\sqrt{a_1}} + \frac{1}{\sqrt{a_2}} + \cdots + \frac{1}{\sqrt{a_m}},\ S_{\text{rcpc,sqrt,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{\sqrt{a_i}},\ S_{\text{rcpc,sqrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt{a_i}},$$

$$S_{\text{rcpc,cbrt}} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt[3]{a_i}} = \frac{1}{\sqrt[3]{a_1}} + \frac{1}{\sqrt[3]{a_2}} + \cdots + \frac{1}{\sqrt[3]{a_m}},\ S_{\text{rcpc,cbrt,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{\sqrt[3]{a_i}},\ S_{\text{rcpc,cbrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt[3]{a_i}},$$

$$S_{\text{rcpc,rt},n} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt[n]{a_i}} = \frac{1}{\sqrt[n]{a_1}} + \frac{1}{\sqrt[n]{a_2}} + \cdots + \frac{1}{\sqrt[n]{a_m}},\ S_{\text{rcpc,even,rt},n} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{\sqrt[n]{a_i}},\ S_{\text{rcpc,odd,rt},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt[n]{a_i}},\ \forall n \in \mathbb{N}^\star.$$

Dựa vào các công thức này, sử dụng vòng lặp `for` hoặc `while` để tính các tổng này. □

Nhận xét 1. *Nếu chỉ tính tổng S_{odd} các số lẻ của dãy $(a_i)_{i=1}^n \subset \mathbb{Z}$ thì ta có bài toán* [Vie21, 3., p. 25, Tây Ninh 2019].

Nhận xét 2 (Mở rộng $\mathbb{Z}$ ra $\mathbb{R}, \mathbb{C}$)**.** *Các tổng $S, S_{\text{sqr}}, S_{\text{cb}}, S_{\text{pwr},n}, S_{\text{sqrt}}, S_{\text{cbrt}}, S_{\text{rt},n}, S_{\text{rcpc}}$ (i.e., các tổng không có liên quan đến tính chẵn lẻ) vẫn có thể áp dụng cho các dãy số thực thay vì chỉ cho dãy số nguyên, i.e., áp dụng cho $(a_i)_{i=1}^n \subset \mathbb{R}$, $a_i \in \mathbb{R}$, $\forall i = 1, 2, \ldots, n$, thay vì chỉ cho $(a_i)_{i=1}^n \subset \mathbb{R}$, $a_i \in \mathbb{Z}$, $\forall i = 1, 2, \ldots, n$, thậm chí có thể áp dụng cho các dãy số phức $(a_i)_{i=1}^n \subset \mathbb{C}$, $a_i \in \mathbb{C}$, $\forall i = 1, 2, \ldots, n$.*

Nhận xét 3 (Mở rộng từ dãy hữu hạn dãy vô hạn & chuỗi)**.** *Bài toán trên có thể mở rộng từ dãy hữu hạn (finite sequence) ra dãy vô hạn (infinite sequence) các số nguyên $(a_n)_{i=1}^\infty \subset \mathbb{Z}$, dãy vô hạn các số thực $(a_n)_{i=1}^\infty \subset \mathbb{R}$, & dãy vô hạn các số phức $(a_n)_{i=1}^\infty \subset \mathbb{C}$, cũng như các chuỗi (series) "xác định" (i.e., có giới hạn để có thể tính được) $S \coloneqq \sum_{i=1}^\infty a_i \in \overline{\mathbb{R}}$. Dương nhiên, 1 chương trình máy tính chỉ có thể lặp (e.g., `for, while` loops) hữu hạn lần chứ không thể lặp vô hạn lần (infinite loop error) nên ta chỉ có thể tính tổng riêng S_m của 1 chuỗi S xác định để xấp xỉ chuỗi S tới 1 độ chính xác (tolerance) nào đó (tolerance thường có dạng 10^{-N} với $N \in \mathbb{N}^\star$ thích hợp), e.g.,*

$$S_m \coloneqq \sum_{i=1}^{m} a_i \to S \coloneqq \sum_{i=1}^{\infty} a_i \text{ as } n \to \infty, \text{ i.e. } \lim_{m\to\infty} S_m = S \text{ if } S \in \overline{\mathbb{R}},$$

trong đó $\overline{\mathbb{R}} \coloneqq \mathbb{R} \cup \{\pm\infty\}$ ký hiệu tập số thực mở rộng bao gồm tập số thực, âm- & dương vô cực.

Bài toán 21 ([Vie21], 5., p. 26, Nghệ An 2019, Giả thuyết Goldbach cho số nguyên tố – Goldbach conjecture for primes)**.** *Cho 1 số chẵn $k \in \mathbb{N}$, $2 \leq k \leq 1000$, tìm 2 số nguyên tố sao cho tổng của chúng bằng số chẵn k đã cho.*

- Yêu cầu: *Viết chương trình* Pascal, Python, C/C++ *để trả lời câu hỏi.*
- Dữ liệu vào: *Tệp văn bản* `prime.inp`*: Dòng đầu tiên chứa $n \in \mathbb{N}^\star$ tương ứng số test. n dòng tiếp theo, mỗi dòng chứa 1 số k, i.e., k_i, $i = 1, 2, \ldots, n$.*
- Dữ liệu ra: *Tệp văn bản* `prime.out` *gồm n dòng tương ứng n kết quả. Mỗi kết quả hiển thị tổng 2 số nguyên tố bằng số k nhập vào. E.g.:*

`prime.inp`	`prime.out`
2	$8 = 5 + 3$
8	$24 = 19 + 5$
24	

Python:

```
from math import sqrt
from math import floor
def is_prime(x):
    if x == 1:
        return 0
    for i in range (2, int(sqrt(x)) + 1):
        if x % i == 0:
            return 0
    return 1

t = int(input())
for i in range(t):
    n = int(input())
    if n % 2 != 0 and is_prime(n - 2) == 1:
        print(n, " = 2 + ", n - 2)
    else:
        for j in range(3, floor(n/2)):
            if is_prime(j) == 1 and is_prime(n - j) == 1:
                print(n, " = ", j, " + ", n - j)
                # break # if want only 1 representation
```

Bài toán 22 ([Vie21], 6., p. 27, Tây Ninh 2019, Số hoàn hảo – Perfect number)**.** Số hoàn hảo *là 1 số tự nhiên mà tổng tất cả các ước tự nhiên thực sự của nó bằng chính nó. Trong đó ước thực sự của 1 số là các ước dương không bằng số đó. Lập trình nhập vào 1 số tự nhiên có* 2 *chữ số bất kỳ. In ra màn hình thông báo số vừa nhập có phải là số hoàn hảo hay không? Nếu là số hoàn hảo thì in tất cả các ước nguyên dương của số đó (i.e., bao gồm tất cả các ước tự nhiên thực sự & chính số đó).*

Python:

```
def is_perfect_number(n):
    sum_proper_divisor = 0
    list_proper_divisor = []
    for i in range(1, round(n/2 + 1)):
        if n % i == 0:
            list_proper_divisor.append(i)
            sum_proper_divisor += i
    if sum_proper_divisor == n:
        list_proper_divisor.append(n)
        print(n, " is a perfect number.")
        print("List of of all divisor of ", n, ":", *list_proper_divisor)
    else:
        print(n, " is not a perfect number.")

n = int(input())
is_perfect_number(n)
```

Bài toán 23 ([Vie21], 7., p. 27, Đồng Nai 2020, Số may mắn – Lucky number)**.** *Để động viên thành tích học tập xuất sắc của các em học sinh lớp 6-3 trong năm học 2019–2020, thầy giáo chủ nhiệm đã chuẩn bị các món quà được đánh số từ* 1 *đến* n*. Sau đó thầy giáo sẽ cho các em lên bốc thăm để nhận món quà may mắn của mình. Đầu tiên thầy giáo sẽ ghi tất cả số nguyên lẻ từ* 1 *đến* n*, sau đó sẽ ghi tất cả các số nguyên chẵn từ* 2 *đến* n *(theo thứ tự tăng dần) để tạo thành 1 dãy số phần thưởng. Mỗi bạn sẽ bốc thăm 1 số* k *ứng với con số của món quà mình đạt được.*

- Yêu cầu: *In số của món quà học sinh đạt được.*
- Dữ liệu vào: *Dòng duy nhất ghi số nguyên* n *&* k*,* $1 \leq k \leq n \leq 1000$*.*
- Dữ liệu ra: *In số của món quà học sinh đạt được:*

lucky_number.inp	lucky_number.out
10 6	2

Python:

```
from math import floor
def luck_number(n, k):
    if k <= floor((n + 1)/2):
        print(2*k - 1)
    else:
        print(2*(k - round((n + 1)/2) + 1))

nk = input()
nk = nk.split()
n = int(nk[0])
k = int(nk[1])
luck_number(n, k)
"""
# Print all
for i in range(1, k + 1):
    luck_number(n, i)
"""
```

Bài toán 24 ([Vie21], 8., p. 27, Ninh Bình 2019, Ước chung lớn nhất ƯCLN – greatest common divisor gcd)**.** *Nhập vào* 3 *số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm* ƯCLN *của* 3 *số này. E.g., nhập vào* 3 *số:* $4, 6, 12$ *thì kết quả* ƯCLN *là* 2*.*

Python:

```
from math import gcd
def gcd3(a, b, c):
    return gcd(a, b, c)

a = int(input())
b = int(input())
c = int(input())
print(gcd3(a, b, c))
```

Bài toán 25 (Bội chung nhỏ nhất BCNN – least common multiplier lcd)**.** *Nhập vào $n \in \mathbb{N}^\star$ số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm* ƯCLN *&* BCNN *của n số này.*

Python:

```
from math import lcm
def lcm3(a, b, c):
   return lcm(a, b, c)

a = int(input())
b = int(input())
c = int(input())
print(lcm3(a, b, c))
```

We are interested in various types of numbers whose beautiful properties are studied in number theory.

2.2.1 Square number

Definition 1 (Square number)**.** *In mathematics, a* square number *or* perfect square *is an integer that is the square of an integer, i.e., it is the product of some integer with itself $n \cdot n = n^2$, for some $n \in \mathbb{Z}$.*

The set of all square numbers is given by $A = \{n^2 | n \in \mathbb{Z}\} = \{n^2 | n \in \mathbb{N}\} = \{0^2, 1^2, 2^2, 3^2, \ldots\}$.

"The usual notation for the square of a number $n \in \mathbb{Z}$ is not the product $n \cdot n$, but the equivalent exponentiation n^2, usually pronounced as "n squared". The name *square* number comes from the name of the shape. The unit of area is defined as the area of a unit square 1×1. Hence, a square with side length n has area n^2. If a square number is represented by n points, the points can be arranged in rows as a square each side of which has the same number of points as the square root of n, thus, square numbers are a type of figurate numbers (other examples being cube numbers & triangular numbers).

In the real number system $\mathbb{R}$, square numbers are nonnegative. A nonnegative integer is a square number when its square root is again an integer, e.g., $\sqrt{9} = 3$, so 9 is a square number. A positive integer has no square divisors except 1 is called square-free.

For a nonnegative integer $n \in \mathbb{N} \equiv \mathbb{Z}_{\geq 0}$, the nth square number is n^2, with $0^2 = 0$ being the 0th one. The concept of square can be extended to some other number systems. If rational numbers are included, then a square is the ratio of 2 square integers, &, conversely, the ratio of 2 square integers is a square $\frac{m^2}{n^2} = \left(\frac{m}{n}\right)^2$, $\forall m, n \in \mathbb{Z}$, $n \neq 0$.

Starting with 1, there are $\lfloor \sqrt{m} \rfloor$ square numbers up to & including m, where the expression $\lfloor x \rfloor$ represents the floor of the number $x \in \mathbb{R}$." – Wikipedia/square number

Example 1 (Set of squares: $\{n^2 | n \in \mathbb{N}, n \leq 60\}$)**.** *"The squares (sequence A000290 in the OEIS) smaller than $60^2 = 3600$ are:* $0^2 = 0$, $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 16$, $5^2 = 25$, $6^2 = 36$, $7^2 = 49$, $8^2 = 64$. $9^2 = 81$, $10^2 = 100$, $11^2 = 121$, $12^2 = 144$, $13^2 = 169$, $14^2 = 196$, $15^2 = 225$, $16^2 = 256$, $17^2 = 289$, $18^2 = 324$, $19^2 = 361$, $20^2 = 400$, $21^2 = 441$, $22^2 = 484$, $23^2 = 529$, $24^2 = 576$, $25^2 = 625$, $26^2 = 676$, $27^2 = 729$, $28^2 = 784$, $29^2 = 841$, $30^2 = 900$, $31^2 = 961$, $32^2 = 1024$, $33^2 = 1089$, $34^2 = 1156$, $35^2 = 1225$, $36^2 = 1296$, $37^2 = 1369$, $38^2 = 1444$, $39^2 = 1521$, $40^2 = 1600$, $41^2 = 1681$, $42^2 = 1764$, $43^2 = 1849$, $44^2 = 1936$, $45^2 = 2025$, $46^2 = 2116$, $47^2 = 2209$, $48^2 = 2304$, $49^2 = 2401$, $50^2 = 2500$, $51^2 = 2601$, $52^2 = 2704$, $53^2 = 2809$, $54^2 = 2916$, $55^2 = 3025$, $56^2 = 3136$, $57^2 = 3249$, $58^2 = 3364$, $59^2 = 3481$.

The difference between any perfect square & its predecessor is given by the identity $n^2 - (n-1)^2 = 2n - 1$, $\forall n \in \mathbb{Z}$. Equivalently, it is possible to count square numbers by adding together the last square, the last square's root, & the current root, i.e., $n^2 = (n-1)^2 + (n-1) + n$, $\forall n \in \mathbb{Z}$." – Wikipedia/square number/examples

"The numebr $m \in \mathbb{N}$ is a square iff one can arrange m points in a square. The expression for the nth square number is n^2. This is also equal to the sum of the 1st n odd numbers, where a square results from the previous one by adding an odd number of points. The formula follows: $n^2 = \sum_{i=1}^{n}(2i-1)$, $\forall n \in \mathbb{N}^\star$.

Example 2 (Square number n^2 = sum of 1st n odds)**.** $1^2 = 1$, $2^2 = 4 = 1 + 3$, $3^2 = 1 + 3 + 5$, $4^2 = 16 = 1 + 3 + 5 + 7$, $5^2 = 25 = 1+3+5+7+9$, $6^2 = 36 = 1+3+5+7+9+11$, $7^2 = 49 = 1+3+5+7+9+11+13$, $8^2 = 64 = 1+3+5+7+9+11+13+15$, $9^2 = 81 = 1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17$, $10^2 = 100 = 1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19$.

There are several recursive methods for computing square numbers. E.g., the nth square number can be computed from the previous square by $n^2 = (n-1)^2 + (n-1) + n = (n-1)^2 + (2n-1)$. Alternatively, the nth square number can be calculated from the previous two by douling the $(n-1)$th square, subtracting the $(n-2)$th square number, & adding 2, because $n^2 = 2(n-1)^2 - (n-2)^2 + 2$, e.g., $2 \cdot 5^2 - 4^2 + 2 = 2 \cdot 25 - 16 + 2 = 50 - 16 + 2 = 36 = 6^2$.

The square minus 1 of a number $m \in \mathbb{R}$ is always the product of $m-1$ & $m+1$, i.e., $m^2-1=(m-1)(m+1)$, $\forall m \in \mathbb{R}$, e.g., since $7^2=49$, one has $6 \cdot 8=48$. Since a prime number has factors of only 1 & itself, & since $m=2$ is the only nonzero value of m to give a factor of 1 on the RHS of the equation, it follows that 3 is the only prime number 1 less than a square, i.e., $3=2^2-11$.

More generally, the difference of the squares of 2 numbers is the product of their sum & their difference, i.e., $a^2-b^2=(a+b)(a-b)$. This is the difference-of-squares formula, which can be usful for mental arithmetic, e.g., $47 \cdot 53$ can be easily computed as $50^2-3^2=2500-9=2491$. A square number is also the sum of 2 consecutive triangular numbers. The sum of 2 consecutive square numbers is a centered square number. Every odd square is also a centered octagonal number.

Another property of a square number is that (except 0) it has an odd number of positive divisors, while other natural numbers have an even number of positive divisors. An integer root is the only divisor that pairs up with itself to yield the square number, while other divisors come in pairs.

Lagrange's 4-square theorem states that any positive integer can be written as the sum of 4 or fewer perfect squares. 3 squares are not sufficient for numbers of the form $4^k(8m+7)$. A positive integer can be represented as a sum of 2 squares precisely if its prime factorization contains no odd powers of primes of the form $4k+3$. This is generalized by Waring's problem.

The sum of the n 1st square numbers is

$$\boxed{\sum_{i=1}^{n} i^2=1^2+2^2+\cdots+n^2=\frac{n(n+1)(2n+1)}{6},\ \forall n \in \mathbb{N}^\star.}$$

" – Wikipedia/square number/properties

2.2.2 Square-free integer

Definition 2 (Square-free integer)**.** *In mathematics, a* square-free integer *(or* squarefree integer*) is an integer which is divisible by no square number other than* 1*, i.e., its prime factorization has exactly* 1 *factor for each prime that appears in it.*

Example 3. $10=2 \cdot 5$ *is square-free, but* $18=2 \cdot 3^2$ *is not, because* $18 \vdots 3^2$.

Example 4. *The smallest positive square-free numbers are* 1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 33, 34, 35, 37, 38, 39, ... *(sequence A005117 in the OEIS).*

"Every positive integer $n \in \mathbb{N}^\star$ can be factored in a unique way as $n=\prod_{i=1}^{k} q_i^k$, where the q_i different from one are square-free integers that are pairwise coprime. This is called the *square-free factorization* of n.

To construct the square-free factorization, let $n=\prod_{j=1}^{h} p_j^{e_j}$ be the prime factorization of n, where the p_j are distinct prime numbers. Then the factors of the square-free factorization are defined as $q_i=\prod_{j:e_j=i} p_j$.

An integer is square-free iff $q_i=1$ for all $i>1$. An integer greater than 1 is the kth power of another integer iff k is a divisor of all i such that $q_i \neq 1$.

The use of the square-free factorization of integers is limited by the fact that its computation is as difficult as the computation of the prime factorization. More precisely every known algorithm for computing a square-free factorization computes also the prime factorization. This is a notable difference with the case of polynomials for which the same definitions can be given, but, in this case, the square-free factorization is not only easier to compute than the complete factorization, but it is the 1st step of all standard factorization algorithms." – Wikipedia/square-free integer/square-free factorization

2.2.3 Figurate number

2.2.4 Cube number

2.2.5 Triangular number

2.2.6 Powerful number

Definition 3 (Powerful number)**.** *A* powerful number *is a positive integer* $n \in \mathbb{N}^\star$ *such that for every prime number* p *dividing* n, p^2 *also divides* n*. Equivalently, a powerful number is the product of a square & a cube, i..e, a number* n *of the form* $n=a^2b^3$*, where* $a, b \in \mathbb{N}^\star$*. Powerful numbers are also known as* squareful, square-full, *or* 2-full.

Example 5. *The following is a list of all powerful numbers between* 1 *&* 1000*:* 1, 4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 72, 81, 100, 108, 121, 125, 128, 144, 169, 196, 200, 216, 225, 243, 256, 288, 289, 324, 343, 361, 392, 400, 432, 441, 484, 500, 512, 529, 576, 625, 648, 675, 676, 729, 784, 800, 841, 864, 900, 961, 968, 972, 1000, ... *(sequence A001694 in the OEIS).*

Definition 4 (k-powerful number)**.** *A* k-powerful number *(or* k-ful number*, or* k-full number*) is an integer all of whose prime factors have exponents a least* k*.*

2.2.7 Highly powerful number

2.2.8 Achilles number

"An Achilles number is a positive integer that is powerful (in the sense that each prime factor occurs with exponent >1) but imperfect (in the sense that the number is not a perfect power)." – Wolfram MathWorld/Achilles number

Definition 5 (Achilles number). *"An* Achilles number *is a number that is powerful but not a perfect power.*

A positive integer $n \in \mathbb{N}^\star$ is a powerful number if, for every prime factor p of n, p^2 is also a divisor. I.e., every prime factor appears at least squared in the factorization. All Achilles numbers are powerful. However, not all powerful numbers are Achilles numbers: only those that cannot be represented as m^k, where m, k are positive integers greater than 1.

Achilles numbers were named by Henry Bottomley after Achilles, a hero of the Trojan war, who was also powerful but imperfect. *Strong Achilles numbers* are Achilles numbers whose Euler totients are also Achilles numbers." – Wikipedia/Achilles number

"A number $n = \prod_{i=1}^{k} p_i^{a_i} = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$ is powerful if $\min\{a_1, a_2, \ldots, a_k\} \geq 2$. If in addition $\gcd(a_1, a_2, \ldots, a_k) = 1$ the number is an Achilles number.

Example 6. *The Achilles numbers up to 5000 are:* 72, 108, 200, 288, 392, 432, 500, 648, 675, 800, 864, 968, 972, 1125, 1152, 1323, 1352, 1372, 1568, 1800, 1944, 2000, 2312, 2592, 2700, 2888, 3087, 3200, 3267, 3456, 3528, 3872, 3888, 4000, 4232, 4500, 4563, 4608, 5000 *(sequence A052486 in the OEIS).*

The smallest pair of consecutive Achilles numbers is: $5425069447 = 7^3 \cdot 41^2 \cdot 97^2$, $5425069448 = 2^3 \cdot 26041^2$." – Wikipedia/Achilles number/sequence of Achilles numbers

Example 7. *"*108 *is a powerful number. Its prime factorization is* $2^2 \cdot 3^3$*, & thus its prime factors are* 2 *&* 3*. Both* $2^2 = 4$ *&* $3^2 = 9$ *are divisors of* 108*. However,* 108 *cannot be represented as* m^k*, where* m *&* k *are positive integers greater than* 1*, so* 108 *is an Achilles number.*

Example 8. 360 *is not an Achilles number because it is not powerful. 1 of its prime factors is* 5 *but* 360 *is not divisible by* $5^2 = 25$*.*

Example 9. 784 *is not an Achilles number. It is a powerful number, because not only are* 2 *&* 7 *its only prime factors, but also* $2^2 = 4$ *&* $7^2 = 49$ *are divisors of it. Nonetheless, it is a perfect power:* $784 = 2^4 \cdot 7^2 = (2^2)^2 \cdot 7^2 = (2^2 \cdot 7)^2 = 28^2$*. So it is not an Achilles number.*

Example 10. $500 = 2^2 \cdot 5^3$ *is a strong Achilles number as its Euler totient of* $200 = 2^3 \cdot 5^2$ *is also an Achilles number."* – *Wikipedia/Achilles number/examples*

Problem 1 (Project Euler, Problem 302: Strong Achilles Number). *A positive integer* n *is* powerful *if* p^2 *is a divisor of* n *for every prime factor* p *in* n*. A positive integer* n *is a* perfect power *if* n *can be expressed as a power of another positive integer. A positive integer* n *is an* Achilles number *if* n *is powerful but not a perfect power. E.g.,* 864 *&* 1800 *are Achilles numbers:* $864 = 2^5 \cdot 3^3$*, &* $1800 = 2^3 \cdot 3^2 \cdot 5^2$*. We will call a positive integer* S *a* strong Achilles number *if both* S *&* S *its Euler's totient function* $\varphi(S)$ *are Achilles numbers. E.g.,* 864 *is a strong Achilles number:* $\varphi(864) = 288 = 2^5 \cdot 3^2$*. However,* 1800 *isn't a strong Achilles number because:* $\varphi(1800) = 480 = 2^5 \cdot 3 \cdot 5$*. There are* 7 *strong Achilles numbers below* 10^4 *&* 656 *below* 10^8*. How may strong Achilles numbers are there below* 10^{18}*?*

2.2.9 Perfect power

Definition 6 (Perfect power). *"In mathematics, a* perfect power *is a natural number that is a product of equal natural factors, or, in other words, an integer that can be expressed as a square or a higher integer power of another integer greater than* 1*. More formally,* n *is a* perfect power *if there exist natural numbers* $m > 1$*, &* $k > 1$ *such that* $m^k = n$*. In this case,* n *may be called a* perfect kth power*. If* $k = 2$ *or* $k = 3$*, then* n *is called a perfect square or perfect cube, respectively.*

Sometimes, 0 & 1 are also considered perfect power: $0^k = 0$, $\forall k \in (0, \infty)$, $1^k = 1$, $\forall k \in \mathbb{R}$." – Wikipedia/perfect power

"A sequence of perfect powers can be generated by iterating through the possible values for m & k. The 1st few ascending perfect powers in numerical order (showing duplicate powers) are (sequence A072103 in the OEIS): $2^2 = 4$, $2^3 = 8$, $3^2 = 9$, $2^4 = 16$, $4^2 = 16$, $5^2 = 25$, $3^3 = 27$, $2^5 = 32$, $6^2 = 36$, $7^2 = 49$, $2^6 = 64$, $4^3 = 64$, $8^2 = 64$, ...

The sum of the reciprocals of the perfect powers (including duplicates such as 3^4 & 9^2, both of which equal 81) is 1: $\sum_{m=2}^{\infty} \sum_{k=2}^{\infty} \frac{1}{m^k} = 1$, which can be proved as follows:

$$\sum_{m=2}^{\infty} \sum_{k=2}^{\infty} \frac{1}{m^k} = \sum_{m=2}^{\infty} \frac{1}{m^2} \sum_{k=0}^{\infty} \frac{1}{m^k} = \sum_{m=2}^{\infty} \frac{1}{m^2} \cdot \frac{m}{m-1} = \sum_{m=2}^{\infty} \frac{1}{m(m-1)} = \sum_{m=2}^{\infty} \left(\frac{1}{m-1} - \frac{1}{m} \right) = 1.$$

Example 11. *The 1st perfect powers without duplicates are: (sometimes* 0 *&* 1*),* 4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 81, 100, 121, 125, 128, 144, 169, 196, 216, 225, 243, 256, 289, 324, 343, 361, 400, 441, 484, 512, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1000, 1024, ... *(sequence A001597 in the OEIS).*

3 Character & String – Xâu & Chuỗi

Bài toán 26 ([Vie21], 1., p. 28, Tây Ninh 2019, Số đảo ngược – Reversed number). *Tìm số đảo ngược* y *của 1 số* $x \in \mathbb{Z}$ *biết* y *gồm các chữ số của* x *& viết theo thứ tự ngược lại. Xuất ra kết quả là số* $y \mod 19$. Dữ liệu: $x \in \mathbb{N}^\star$. Kết quả: $y \mod 19$ *với* y *là số đảo ngược của* x*.*

`reversed_number.inp`	`reversed_number.out`	Giải thích
123	17	Đảo ngược của 123 là 321 & $321 \mod 19 = 17$

Bài toán 27 ([Vie21], 2., pp. 28–29, Bắc Giang 2020, Nén xâu – String compression). *Viết chương trình nhập vào 1 xâu ký tự chỉ gồm các chữ cái Tiếng Anh, chữ số, dấu cách, & dấu gạch nối.*

- Yêu cầu: *Nén các ký tự liên tiếp giống nhau thành số lượng ký tự & ký tự đó, rồi đưa ra xâu sau khi nén.*
- Dữ liệu vào: *Đọc từ file văn bản* `string_compression.inp` *gồm 1 xâu S có số lượng ký tự không quá* 255 *ký tự.*
- Dữ liệu ra: *Đưa ra file văn bản* `string_compression.out` *gồm xâu S sau khi nén.*

`string_compression.inp`	`lstring_compression.out`
aaaababb cc	4a1b1a2b4 2c

Bài toán 28 ([Vie21], 3., p. 30, Hậu Giang 2020, Tính nhân – Multiplication). *Viết chương trình nhập vào 2 số nguyên dương* $a, b \in \mathbb{N}^\star$. *Sau đó thực hiện nhân* $a \times b$ *như cách nhân bằng tay thông thường ở tiểu học. E.g., nhập vào thừa số thứ 1:* 125*, nhập vào thừa số thứ 2:* 15. Dữ liệu ra:

```
  125
x
   15
-----
  625
 125
-----
 1875
```

Problem 2 ([DV21], pp. 42–43, Anagrams). *A word* `w` *is an* anagram *of a word* `v` *if a permutation of the letters transforming* `w` *into* `v` *exists. Given a set of* n *words of length at most* k*, we would like to detect all possible anagrams.*

- Sample Input. *below the car is a rat drinking cider and bending its elbow while this thing is an arc that can act like a cat which cried during the night caused by pain in its bowel.*
- Sample Output. `{bowel below elbow}, {arc car}, {night thing}, {cried cider}, {act act}`

4 1D Array & List – Mảng 1 Chiều & Danh Sách

5 Matrix – Ma Trận

6 Arrangement – Sắp Xếp

See, e.g., [Knu98, Chap. 5: Sorting], [Vie21, Chap. II, Sect. Dạng bài sắp xếp].

Bài toán 29 ([Vie21], 1., p. 83, Bắc Giang 2019, Dãy số – Sequence). *Sử dụng hàm Randomize để khởi tạo dãy số ngẫu nhiên từ* 0 *đến* 9 *gồm* $n \in \mathbb{N}^\star$ *phần tử,* $0 < n \leq 100$*, kết quả ghi ra tệp* `random.out`*, mỗi phần tử cách nhau* 1 *dấu cách.*

- Yêu cầu: *Viết chương trình đọc dữ liệu từ tệp* `random.inp`*, sau đó sắp xếp lại các phần tử theo chiều tăng dần, đồng thời cho biết số lần xuất hiện của mỗi phần tử trong dãy số đã được khởi tạo.*
- Dữ liệu ra: *Ghi ra tệp* `random.out` *gồm* 11 *dòng: Dòng thứ nhất là dãy các phần tử đã được sắp xếp. Dòng thứ* 2 *đến dòng thứ* 11 *tương ứng chữ số ghi tổng số lần xuất hiện của* $0, 1, \ldots, 9$. *E.g.:*

`random_sequence.inp`	`random_sequence.out`
0 5 2 0 1 6 7 8 7 3 1	0 0 1 1 2 3 5 6 7 7 8 2 2 1 1 0 1 1 2 1 0

Bài toán 30 ([Vie21], 2., p. 85, Vị Thanh, Hậu Giang 2019, Dãy số không giảm – Nondecreasing sequence)**.** *Nhập từ bàn phím 3 số nguyên dương* a_1, a_2, a_3, $100 < a_1, a_2, a_3 < 10^5$. *Dãy số b được sinh ra bằng cách ghép từng số nguyên dương đã nhập lần lượt với 2 số còn lại, e.g.,* $a_1 = 234$, $a_2 = 123$, $a_3 = 345$ *ta tìm được* $b_1 = 234123$, $b_2 = 234345$, $b_3 = 123234$, $b_4 = 123345$, $b_5 = 345234$. *Sắp xếp các số trong dãy số b thành dãy không giảm & xuất ra màn hình, các số cách nhau 1 khoảng trắng. E.g.:*

nondecreasing_sequence.inp	nondecreasing_sequence.out
$a_1 = 234$ $a_2 = 123$ $a_3 = 345$	123234 123345 234123 234345 345123 345234

Bài toán 31 ([Vie21], 1., p. 87, Sorting ascending)**.** *Cho vào m dãy số nguyên* $(a_{i,j})_{j=1}^{n_i}$, $n_i \in \mathbb{N}^\star$, $n_i \leq 100$, $\forall i = 1, 2, \ldots, m$, $|a_{ij}| < 32000$, $\forall i = 1, 2, \ldots, m$, $\forall j = 1, 2, \ldots, \max\{n_i | i = 1, 2, \ldots, m\}$.

- Yêu cầu: *Sắp xếp từng dãy số trên theo thứ tự tăng dần.*
- Dữ liệu vào: *Trong m dòng, mỗi dòng là dãy số, bắt đầu là 1 số nguyên n là số lượng các phần tử của dãy số,* $1 \leq n \leq 100$, *n số nguyên tiếp theo là giá trị các phần tử của dãy.*
- Dữ liệu ra: *Ghi ra m dòng là m dãy số đã được sắp xếp theo thứ tự tăng dần. E.g.:*

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

7 Algorithm – Thuật Toán

7.1 Recursion algorithm – Thuật toán đệ quy

Đệ quy (recursion) là phương pháp dùng trong các chương trình máy tính trong đó có 1 hàm tự gọi chính nó.

Bài toán 32 ([Vie21], p. 91, Giai thừa – Factorial $n!$)**.** *Tính* $n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdots (n-1)n$.

- Input. *Dòng đầu là số lượng test. Mỗi dòng tiếp theo gồm 1 số* $n \in \mathbb{N}$.
- Output. *Với mỗi test, in ra n! theo mẫu:*

factorial.inp	factorial.out
2	3! = 6
3	4! = 24
4	

Bài toán 33 ([Vie21], 1., p. 92, Hàm $f91$ của McCarthy – McCarthy's $f91$ function)**.** *McCarthy là 1 nhà khoa học máy tính nổi tiếng, ông đã định nghĩa hàm đệ quy* $f91 : \mathbb{Z} \to \mathbb{Z}$ *như sau:*

$$f_{91}(n) = \begin{cases} f_{91}(f_{91}(n+11)), & \text{if } n \leq 100, \\ n - 1, & \text{if } n \geq 101. \end{cases}$$

Viết 1 chương trình tính toán hàm McCarthy's f91.

- Input. *File input chứa 1 dãy các số nguyên dương, mỗi số không quá* 1000. *Mỗi số trên 1 dòng.*
- Output. *Hiện ra theo định dạng trong ví dụ sau:*

McCarthy_f91_function.inp	McCarthy_f91_function.out
500	f91(500) = 490
91	f91(91) = 91

See, e.g., Wikipedia/McCarthy 91 function.

Bài toán 34 ([Vie21], 2., p. 93, Chỉnh hợp – Arrangement A_n^k)**.** *Tìm tất cả các chỉnh hợp chập k của n phần tử từ* 1 *đến n,* $0 < k \leq n < 10$.

- Input. *File input chứa 1 dòng gồm* $n, k \in \mathbb{N}$.

- Output. *In ra số chỉnh hợp tìm được & liệt kê các chỉnh hợp, giữa các test là 1 dòng trắng, e.g.,*

arrangement.inp	arrangement.out	arrangement.inp	arrangement.out
1 3	3 11 13 33	2 3	6 1 2 1 3 2 1 2 3 3 1 3 2

Bài toán 35 ([Vie21], 3., p. 93, Hoán vị – Permutation $P_n = n!$)**.** *Viết chương trình in ra tất cả các hoán vị của $n \in \mathbb{N}^\star$ được nhập từ bàn phím, e.g.,*

permutation.inp	permutation.out
2	1 2 2 1
3	1 2 3 1 3 2 2 1 3 2 3 1 3 2 1 3 1 2

Bài toán 36 ([Vie21], 5., p. 94, Tổ hợp – Combinatoric C_n^k)**.** *Tìm tất cả các tổ hợp chập k của n phần tử từ 1 đến n, $0 < k \leq n < 10$.*

- Input. *File input có dòng đầu gồm 1 số tự nhiên* ***ntest*** *là số lượng test của file* `combinatoric.inp`*. Mỗi test 1 dòng gồm 2 số $n, k \in \mathbb{Z}$.*
- Output. *Với mỗi test, in ra số tổ hợp tìm được & liệt kê các tổ hợp, giữa các test là 1 dòng trắng. E.g.:*

combinatoric.inp	combinatoric.out
2 1 3 2 3	3 1 2 3 3 1 2 1 3 2 3

Bài toán 37 ([Vie21], 4., pp. 93–94, Trò chơi số học – Number theory game)**.** *1 trò chơi phổ biến của trẻ em với bảng $n \times n$ ô $2 \leq n \leq 5$. Trong mỗi ô chứa 1 số có 1 chữ số từ 1 tới 9. Với 1 số số cho trước, điền các số còn lại vào ô sao cho tổng các hàng ngang bằng nhau & bằng tổng các hàng dọc.*

- Input. `number_theory_game.inp`*: Số đầu tiên là 1 số nguyên* ***ntest*** *là số lượng test của bài, mỗi test gồm có: Dòng đầu tiên là số n. Tiếp theo là ma trận $n \times n$ trong đó số 0 là ô trắng cần điền.*
- Output. `number_theory_game.out`*: Với mỗi test, nếu có thể tìm ra đáp án thỏa mãn quy luật của bảng thì in ra ma trận $n \times n$ 1 cách điền bất kỳ. Nếu không có kết quả nào in ra 1 chuỗi:* `"cannot find."`*. Giữa các test cách nhau 1 dòng trắng. E.g.:*

number_theory_game.inp	number_theory_game.out
1 3 1 0 5 0 3 0 3 0 4	1 4 5 6 3 1 3 3 4

7.2 Search algorithm – Thuật toán tìm kiếm

Tìm kiếm & sắp xếp là 2 hoạt động hằng ngày ta thường sử dụng trong các ứng dụng tin học.
See, e.g., [Knu98, Chap. 6: Searching].

8 Problem in Elementary Physics – Bài Toán Tin Học Trong Vật Lý Sơ Cấp

9 Problem in Elementary Chemistry – Bài Toán Tin Học Trong Hóa Học Sơ Cấp

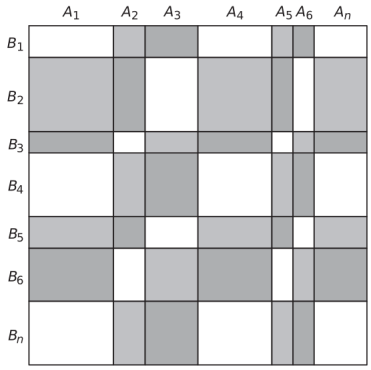
10 Miscellaneous

Bài toán 38 ([BTC10], 1., p. 5, Connect)**.** *Cho n số nguyên dương $a_1, a_2, \ldots, a_n$, $n \in \mathbb{N}$, $1 < n \leq 100$, $0 < a_i \leq 10^9$, $\forall i = 1, 2, \ldots, n$. Từ các số nguyên này người ta tạo ra 1 số nguyên mới bằng cách kết nối tất cả các số đã cho viết liên tiếp nhau. E.g., với $n = 4$ & các số $12, 34, 567, 890$ ta có thể tạo ra các số mới như sau:* 1234567890, 3456789012, 8905673412, … *Dễ thấy có $4! = 24$ cách tạo mới như vậy. Trong trường hợp này, số lớn nhất có thể tạo thành là* 8905673412.

- Yêu cầu: *Cho n & các số $a_1, a_2, \ldots, a_n$. Xác định số lớn nhất có thể kết nối được theo quy tắc trên.*
- Dữ liệu vào: *Cho trong file văn bản* `connect.inp` *gồm $n + 1$ dòng. Dòng đầu tiên ghi số nguyên n. Trong các dòng còn lại, dòng thứ $i + 1$ ghi số a_i.*
- Dữ liệu ra: *Ghi vào file văn bản* `connect.out` *số lớn nhất được kết nối thành từ các số ban đầu. E.g.:*

`connect.inp`	`connect.out`
4 12 34 567 890	8905673412

Problem 3 (Frosting on the Cake)**.** *Iskander the Baker is decorating a huge cake by covering the rectangular surface of the cake with frosting. For this purpose, he mixes frosting sugar with lemon juice & beetle juice, in order to produce 3 kinds of frosting: yellow, pink, & white. These colors are identified by the number* 0 *for yellow,* 1 *for pink, & 2 for white. To obtain a nice pattern, he partitions the cake surface into vertical stripes of width $A_1, A_2, \ldots, A_n$ centimeters, & horizontal stripes of height $B_1, B_2, \ldots, B_n$ centimeters, for some positive integer $n \in \mathbb{N}^\star$. These stripes split the cake surface into $n \times n$ rectangles. The intersection of vertical stripe i & horizontal stripe j has color number $(i + j) \mod 3$ for all $1 \leq i, j \leq n$, e.g.:*



Hình 1: An instance of the problem *Frosting on the Cake.*

To prepare the frosting, Iskander wants to know the total surface in square centimeters to be colored for each of the 3 colors, & asks for your help.

- Input. *The input consists of the following integers: on the 1st line: the integer $n \in \mathbb{N}^\star$, on the 2nd line: the values of $A_1, A_2, \ldots, A_n, n$ integers separated by single spaces, on the 3rd line: the values of $B_1, B_2, \ldots, B_n, n$ integers separated by single spaces.*
- Limits. *The input satisfies $3 \leq n \leq 100000$ & $1 \leq A_i, B_i \leq 10000$, $i = 1, 2, \ldots, n$.*
- Output. *The output should consist of 3 integers separated by single spaces, representing the total area for each color* $0, 1, 2$.

Solution. Followed [DV21, Sect. 1.8, pp. 39–41], we want to find a solution that runs in time $O(n)$ or possibly $O(n \log n)$, which rules out the naive solution which loops over all n^2 grid cells & accumulates their areas in variables corresponding to each color. Permuting columns or rows preserves the total area of each color. Hence, we can reduce to the $n = 3$ case, by simply summing the values of each color class $A_{3k}, A_{3k+1}, A_{3k+2}$. Then the answer per color class is just the sum of the areas of 3 rectangles. The tricky part relies in not mixing up the colors. □

```
def read_ints(): return [int(x) for x in input().split()]
def cat(l): return tuple(sum(l[n::3]) for n in [1, 2, 0])
input() # n
A = cat(read_ints())
B = cat(read_ints())
print("{} {} {}".format(B[2] * A[0] + B[0] * A[2] + B[1] * A[1],
                        B[2] * A[1] + B[0] * A[0] + B[1] * A[2],
                        B[2] * A[2] + B[0] * A[1] + B[1] * A[0]))
```

10.1 Google Kickstart Round A 2020

Watch YouTube/William Lin/Winning Google Kickstart Round A 2020.

Problem 4 (Google Kickstart Round A 2020, Allocation)**.** *There are n houses for sale. The ith house costs a_i dollars to buy. You have a budget of b dollars to spend. What is the maximum number of houses you can buy?*

- Input. *The 1st line of the input gives the number of test cases, t. t test cases follow. Each test case begins with a single line containing the 2 integers n, b. The 2nd line contains n integers. The ith integer is a_i, the cost of the ith house.*
- Output. *For each test case, output 1 line containing* `Case #x: y`*, where* `x` *is the test case number (starting from* 1*) &* `y` *is the maximum number of houses you can buy.*
- Limits: *Time limit:* 15 s/test set. *Memory limit:* 1GB. $1 \leq t \leq 100$, $1 \leq b \leq 10^5$, $1 \leq a_i \leq 1000$, $\forall i = 1, 2, \ldots, n$. *Test set 1:* $1 \leq n \leq 100$. *Test set 2:* $1 \leq n \leq 10^5$.
- Sample.

allocation.inp	allocation.out
3 4 100 20 90 40 90 4 50 30 30 10 10 3 300 999 999 999	Case #1: 2 Case #2: 3 Case #3: 0

```
#include <bits/stdc++.h>
using namespace std;
int n, b, a[100000];

void solve() {
    cout << "Enter n: ";
    cin >>  n;
    cout << "Enter b: ";
    cin >>  b;
    cout << "Enter array a: ";
    for(int i = 0; i < n; ++i)
    cin >> a[i];
    sort(a, a + n);
    int ans = 0;
    for (int i = 0; i < n; ++i) {
        if (b >= a[i]) {
            b -= a[i];
            ++ ans;
        }
    }
    cout << ans << "\n";
}

int main() {
    int t, i = 1;
    cout << "Enter t: ";
    cin >> t;
    while (t--) {
        cout << "Case #" << i << ": ";
```

```
        solve();
        ++i;
    }
}
```

Problem 5 (Google Kickstart Round A 2020, Plates)**.** *Dr. Patel has n stacks of plates. Each stack contains k plates. Each plate has a positive beauty value, describing how beautiful it looks. Dr. Patel would like to take exactly p plates to use for dinner tonight. If he would like to take a plate in a stack, he must also take all of the plates above it in that stack as well. Help Dr. Patel pick the p plates that would maximize the total sum of beauty values.*

- Input. *The 1st line of the input gives the number of test cases, t. t test cases follow. Each test case begins with a line containing the 3 integers n, k, p. Then, n lines follow. The ith line contains k integers, describing the beauty values of each stack of plates from top to bottom.*
- Output. *For each test case, output 1 line containing* `Case #x: y`*, where* **`x`** *is the test case number (starting from 1) & **`y`** is the maximum total sum of beauty values that Dr. Patel could pick.*
- Limits: *Time limit:* 20 s/test set. *Memory limit:* 1GB. $\leq t \leq 100$, $1 \leq k \leq 30$, $1 \leq p \leq nk$. *The beauty values are between* 1 *&* 100*, inclusive. Test set 1:* $1 \leq n \leq 3$. *Test set 2:* $1 \leq n \leq 50$.
- Sample.

plate.inp	plate.out
2	Case #1: 250
2 4 5	Case #2: 180
10 10 100 30	
80 50 10 50	
3 2 3	
80 80	
15 50	
20 10	

11 CSES Problem Set

Problem 6 (Weird Algorithm)**.** *Consider an algorithm that takes as input a positive integer $n \in \mathbb{N}^\star$. If n is even, the algorithm divides it by 2, & if n is odd, the algorithm multiplies it by 3 & adds 1. The algorithms repeats this, until n is 1. E.g., the sequence for $n = 3$ is as follows: $3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$. Your task is to simulate the execution of the algorithm for a given value of n.*

- Input. *The only input line contains an integer $n \in \mathbb{Z}$.*
- Output. *Print a line that contains all values of n during the algorithm.*
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

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

Source: CSES Problem Set/weird algorithm, & [Laa20, Sect. 1.3, pp. 5–7].

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

int main() {
    long long n;
    cin >> n;
    while (1) {
        cout << n << " ";
        if (n == 1) break;
        if (n%2 == 0) n /= 2;
        else n = n*3 + 1;
    }
    cout << "\n";
}
```

Remark 1 (Collatz conjecture). *The above algorithm terminates* $\forall n \in \mathbb{N}^\star$.

Problem 7 (Missing Number). *You are given all numbers between* $1, 2, \ldots, n$ *except one. Your task is to find the missing number.*

- Input. *The 1st input line contains a positive 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. $2 \leq n \leq 2 \cdot 10^5$.
- Sample.

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

Source: CSES Problem Set/missing number.

```
file_in = open("missing_number.inp")
file_out = open("missing_number.out", "w")
n = int(file_in.readline())
data = file_in.readline()
A = data.split()
A = [int(i) for i in A]
A.sort()
for i in range(n-1):
    if i + 1 != A[i]:
        file_out.write(str(i + 1))
        break
file_in.close()
file_out.close()
```

Problem 8 (Repetitions). *You are given a DNA sequence: a string consisting of characters A, C, G, & T. Your task is to 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. $1 \leq n \leq 10^6$.
- Sample.

repetition.inp	repetition.out
ATTCGGGA	3

Source: CSES problem Set/repetition.

```
DNA = input()
DNA_count = []
count = 1
char = DNA[0]
for i in range(1, len(DNA)):
    if DNA[i] == DNA[i-1]:
        count = count + 1
    else:
        DNA_count.append(count)
        count = 1
DNA_count.append(count)
print(max(DNA_count))
```

Problem 9 (Non-decreasing Array). *You are given an array of* n *integers. You want to modify the array so that it is non-decreasing, 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*: the size of the array. The 2nd line contains* n *integers* $x_1, x_2, \ldots, x_n$*: the contents of the array.*

- Output. *Print the minimum number of moves.*
- Constraints. $1 \le n \le 2 \cdot 10^5$, $1 \le x_i \le 10^9$.
- Sample.

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

Source: CSES problem Set/increasing array.

```
n = int(input())
A = input()
A = A.split()
A = [int(i) for i in A]
ans = 0
for i in range(1, len(A)):
    if A[i] < A[i - 1]:
        ans = ans + A[i-1] - A[i]
        A[i] = A[i - 1]
print(ans)
```

Problem 10 (Permutations). *A permutation of integers* $1, 2, \ldots, n$ *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 only input line contains an integer* n.
- 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. $1 \le n \le 10^6$.
- Sample.

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

```
n = int(input())
A = [1]
if n == 1:
    print(1)
elif n == 2 or n == 3:
    print("NO SOLUTION")
elif n == 4:
    print("2 4 1 3")
else:
    for i in range(1, n):
        tmp = A[i-1] + 2
        if tmp <= n:
            A.append(tmp)
        else:
            A.append(2)
    print(*A)
```

Problem 11 (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

Your task is to find out the number in row y *&* *column* x.

- Input. *The 1st input line contains an integer t: the number of tests. After this, there are t lines, each containing integers y, x.*
- Output. *For each test, print the number in row y and column x.*
- Constraints. $1 \le t \le 10^5$, $1 \le x, y \le 10^9$.
- Sample.

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

Resources

[Đàm+09a; Đàm+09b; Đàm+11; Knu97; Vie21; Vie22].

Tài liệu

[BTC10] BTC. *Tuyển Tập Đề Thi Olympic 30 Tháng 4, Lần Thứ XVI - 2010 Tin học*. Nhà Xuất Bản Đại Học Sư Phạm, 2010, p. 285.

[Đàm+09a] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài Liệu Giáo Khoa Chuyên Tin, quyển 1*. Nhà Xuất Bản Giáo Dục Việt Nam, 2009, p. 219.

[Đàm+09b] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài Liệu Giáo Khoa Chuyên Tin, quyển 2*. Nhà Xuất Bản Giáo Dục Việt Nam, 2009, p. 240.

[Đàm+11] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài Liệu Giáo Khoa Chuyên Tin, quyển 3*. Nhà Xuất Bản Giáo Dục Việt Nam, 2011, p. 170.

[DV21] Christoph Dürr and Jill-Jênn Vie. *Competitive Programming in Python: 128 Algorithms to Develop Your Coding Skills*. Translated by Greg Gibbons & Danièle Gibbons. Cambridge University Press, 2021, pp. x+254.

[Knu97] Donald Ervin Knuth. *The Art of Computer Programming. Volume 1: Fundamental Algorithms*. 3rd edition. Addison-Wesley Professional, 1997, pp. xx+652.

[Knu98] Donald Ervin Knuth. *The Art of Computer Programming. Volume 3: Sorting and Searching*. 2nd edition. Addison-Wesley Professional, 1998, pp. xiii+782.

[Laa20] Antti Laaksonen. *Guide to Competitive Programming: Learning & Improving Algorithms Through Contests*. 2nd edition. Undergraduate Topics in Computer Science. Springer, 2020, pp. xv+309.

[Vie21] Học Viện VietSTEM. *Sách Luyện Thi Hội Thi Tin Học Trẻ với Python Bảng B: Thi Kỹ Năng Lập Trình Cấp Trung Học Cơ Sở*. Nhà Xuất Bản Đại Học Quốc Gia Hà Nội, 2021, p. 190.

[Vie22] Học Viện VietSTEM. *Lập Trình với Python: Hành Trang Cho Tương Lai*. Nhà Xuất Bản Đại Học Quốc Gia Hà Nội, 2022, p. 224.