TÊN: HUỲNH CÔNG TÍNH

LỚP: DCT124C6

MSSV: 3124411315

MÃ HACKERANK NỘP: congtinh1932006

SỐ ĐIỂM NỘP: 50/50

Câu 1:

A screenshot of a test

AI-generated content may be incorrect.

PHẦN Ý TƯỞNG:

Ý tưởng giải bài toán "Bộ Ba"

1. Hiểu đề

* Chuỗi chỉ có chữ cái in hoa (A–Z).
* "Bộ ba" = 3 ký tự liên tiếp nhau trong chuỗi.
* Ta cần tìm: bộ ba nào xuất hiện nhiều nhất, và nó xuất hiện bao nhiêu lần.
* Nhưng đề chỉ yêu cầu in ra số lần xuất hiện nhiều nhất, không cần in ra bộ ba đó.

2. Cách làm từng bước

Bước 1: Xác định tất cả bộ ba trong chuỗi

* Nếu chuỗi dài n, thì có đúng (n - 2) bộ ba.
* Ví dụ: "AAAAAD" (n=6) → có (6-2)=4 bộ ba:
  + i=0: "AAA"
  + i=1: "AAA"
  + i=2: "AAA"
  + i=3: "AAD"

Bước 2: Đếm số lần xuất hiện mỗi bộ ba

* Ta dùng 1 cái "sổ đếm" (hash table / map).
* Mỗi khi gặp bộ ba nào → tăng đếm cho bộ ba đó.

Bước 3: Tìm số lớn nhất

* Sau khi đã đếm hết, duyệt qua bảng đếm để lấy giá trị max.
* Đó chính là kết quả.

3. Giải tay ví dụ

Ví dụ 1: AAAAAD

* Chuỗi = 6 ký tự → có 4 bộ ba.
* Liệt kê:
  + "AAA" (i=0)
  + "AAA" (i=1)
  + "AAA" (i=2)
  + "AAD" (i=3)
* Bảng đếm:
  + "AAA": 3 lần
  + "AAD": 1 lần
* Bộ ba xuất hiện nhiều nhất là "AAA" với 3 lần.

=> Kết quả: 3

Ví dụ 2: BCDEGHDJL

* Chuỗi = 9 ký tự → có 7 bộ ba.
* Liệt kê:
  + "BCD"
  + "CDE"
  + "DEG"
  + "EGH"
  + "GHD"
  + "HDJ"
  + "DJL"
* Bảng đếm: mỗi bộ ba xuất hiện 1 lần.
* Max = 1  
  => Kết quả: 1

4. Tổng kết ý tưởng

* B1: Với mỗi chuỗi → tách tất cả bộ ba.
* B2: Đếm số lần xuất hiện từng bộ ba.
* B3: Tìm số lần lớn nhất → in ra.

Code c++

#include <iostream>

#include <string>

using namespace std;

int main() {

ios::sync\_with\_stdio(false);

cin.tie(nullptr);

int T;

cin >> T;

while (T--) {

string s;

cin >> s;

int n = s.size();

if (n < 3) {

cout << 0 << "\n";

continue;

}

int freq[26\*26\*26] = {0};

int ans = 0;

for (int i = 0; i <= n - 3; i++) {

int x = s[i] - 'A';

int y = s[i+1] - 'A';

int z = s[i+2] - 'A';

int idx = x\*26\*26 + y\*26 + z;

freq[idx]++;

if (freq[idx] > ans) ans = freq[idx];

}

cout << ans << "\n";

}

return 0;

}

Câu 2:

A screenshot of a questionnaire

AI-generated content may be incorrect.

PHẦN Ý TƯỞNG:

Bước 1: Điều kiện cần

* Tất cả các phần tử phải nằm trong đoạn [1..N].  
  Nếu có số nhỏ hơn 1 hoặc lớn hơn N → chắc chắn sai.

Bước 2: Điều kiện đủ

* Mỗi số từ 1 đến N xuất hiện đúng 1 lần.  
  Nếu một số bị thiếu hoặc trùng lặp → sai.

3. Cách làm

1. Nhập N và dãy A[1..N].
2. Tạo một mảng đánh dấu check[1..N] (ban đầu = 0).
3. Duyệt từng phần tử Ai:
   * Nếu Ai < 1 hoặc Ai > N → in "NO" ngay.
   * Nếu check[Ai] đã bằng 1 → nghĩa là Ai bị trùng → "NO".
   * Ngược lại check[Ai] = 1.
4. Sau khi duyệt xong → nếu tất cả hợp lệ → "YES".

4. Giải tay ví dụ

Input:

|  |
| --- |
| 3 |
| 2 1 3 |

* N = 3
* Dãy = [2, 1, 3]

👉 Kiểm tra:

* 2 nằm trong [1..3], chưa xuất hiện → đánh dấu check[2] = 1.
* 1 nằm trong [1..3], chưa xuất hiện → check[1] = 1.
* 3 nằm trong [1..3], chưa xuất hiện → check[3] = 1.

Kết quả: tất cả {1,2,3} đều xuất hiện đúng 1 lần.  
👉 Output: "YES"

Ví dụ khác:

Input:

|  |
| --- |
| 3 |
| 1 1 2 |

* N=3
* Dãy = [1,1,2]

👉 Kiểm tra:

* 1 → ok, check[1]=1
* 1 lần nữa → check[1] đã =1 ⇒ bị trùng ⇒ "NO".

Code C++:

#include <iostream>

using namespace std;

int main() {

ios::sync\_with\_stdio(false);

cin.tie(nullptr);

int N;

cin >> N;

int A[1005];

int check[1005] = {0};

bool ok = true;

for (int i = 0; i < N; i++) {

cin >> A[i];

if (A[i] < 1 || A[i] > N) {

ok = false;

} else {

if (check[A[i]] == 1) {

ok = false;

}

check[A[i]] = 1;

}

}

if (ok) cout << "YES\n";

else cout << "NO\n";

return 0;

}

Câu 3:

A screenshot of a computer

AI-generated content may be incorrect.

Ý tưởng chi tiết

1. Sinh ra dãy số nguyên tố ≤ N
   * Vì khi phân tích N!, ta chỉ quan tâm các số nguyên tố ≤ N (vì trong N! không thể xuất hiện thừa số nguyên tố > N).
   * Ví dụ: với N=10, dãy nguyên tố là [2,3,5,7].
2. Cách tính số mũ của một số nguyên tố p trong N!

A screenshot of a black background with white text

AI-generated content may be incorrect.

1. Xuất kết quả
   * Với mỗi N nhập vào → liệt kê số mũ ứng với dãy số nguyên tố.
   * Output chỉ liệt kê đến số nguyên tố cuối cùng có số mũ > 0 (không in số 0 thừa ở cuối).

🔹 Làm bằng tay

|  |
| --- |
| Input: |
| 10 |

Bước 1: Các số nguyên tố ≤ 10

[2, 3, 5, 7]

Bước 2: Tính số mũ

* Với p=2:  
  ⌊10/2⌋+⌊10/4⌋+⌊10/8⌋=5+2+1=8
* Với p=3:  
  ⌊10/3⌋+⌊10/9⌋=3+1=4
* Với p=5:  
  ⌊10/5⌋=2
* Với p=7:  
  ⌊10/7⌋=1

Bước 3: Ghi kết quả

|  |
| --- |
| Output: |
| 8 4 2 1 |

Code C++:

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

vector<int> sieve(int n) {

vector<bool> isPrime(n+1, true);

vector<int> primes;

isPrime[0] = isPrime[1] = false;

for (int i = 2; i <= n; i++) {

if (isPrime[i]) {

primes.push\_back(i);

for (int j = i\*i; j <= n; j += i)

isPrime[j] = false;

}

}

return primes;

}

int countExp(int n, int p) {

int exp = 0;

while (n > 0) {

n /= p;

exp += n;

}

return exp;

}

int main() {

int N;

while (cin >> N) {

vector<int> primes = sieve(N);

vector<int> exps;

for (int p : primes) {

int e = countExp(N, p);

exps.push\_back(e);

}

for (int i = 0; i < exps.size(); i++) {

if (i > 0) cout << " ";

cout << exps[i];

}

cout << endl;

}

return 0;

}

Câu 4:

A black and white text

AI-generated content may be incorrect.

Ý tưởng chi tiết

1. Khi nào thì có 0 tận cùng trong N!?
   * Một chữ số 0 tận cùng xuất hiện khi tích chứa thừa số 10.
   * Mà 10 = 2 × 5.
2. Trong N! có bao nhiêu thừa số 2 và 5?
   * Rõ ràng, số lượng thừa số 2 luôn nhiều hơn thừa số 5 (vì số chẵn xuất hiện nhiều hơn bội của 5).
   * Do đó, số lượng số 0 tận cùng của N! = số lần xuất hiện thừa số 5 trong N!.
3. Cách tính số lần xuất hiện của 5 trong N!

A black background with white text

AI-generated content may be incorrect.

1. Dừng vòng lặp
   * Vì N≤2⋅10 ^ 9 chỉ cần tính đến 5 ^ 13=12207031255

🔹 Làm thử bằng tay với ví dụ

Ví dụ 1: N = 5

Z(5)=⌊5/5⌋=1

* Output = 1

Ví dụ 2: N=10N = 10N=10

Z(10)=⌊10/5⌋+⌊10/25⌋=2+0=2

* Output = 2

Code C++:

#include <iostream>

using namespace std;

long long countZeros(long long n) {

long long cnt = 0;

for (long long p = 5; p <= n; p \*= 5) {

cnt += n / p;

}

return cnt;

}

int main() {

long long N;

while (cin >> N) {

cout << countZeros(N) << endl;

}

return 0;

}

Câu 5:

A white paper with black text

AI-generated content may be incorrect.

1. Phân tích yêu cầu

* Có 3 dãy số nguyên:
  + Dãy x có nx phần tử
  + Dãy y có ny phần tử
  + Dãy z có nz phần tử
* Nhiệm vụ: tìm các phần tử chung của cả ba dãy.
* Kết quả:
  + Dòng 1: số lượng phần tử chung.
  + Dòng 2: các phần tử chung theo thứ tự tăng dần, không trùng lặp.

2. Cách tiếp cận

Cách 1: Dùng tập hợp (set)

* Ý tưởng:
  + Cho vào setX chứa tất cả phần tử của dãy x.
  + Cho vào setY chứa tất cả phần tử của dãy y.
  + Cho vào setZ chứa tất cả phần tử của dãy z.
  + Lấy giao 3 tập hợp này: common = setX ∩ setY ∩ setZ.
* Ưu điểm: đơn giản, dễ cài đặt.
* Sau khi lấy giao, sắp xếp kết quả để in ra.

Cách 2: Dùng mảng đánh dấu (frequency array)

* Vì giới hạn giá trị phần tử ≤ 10000.
* Ta có thể tạo mảng freq[10001] để đếm.
* Thuật toán:
  + Với mỗi phần tử trong dãy x: freq[val]++ nhưng chỉ cộng một lần (dùng set phụ để loại trùng).
  + Với dãy y: tiếp tục freq[val]++.
  + Với dãy z: tiếp tục freq[val]++.
  + Sau đó duyệt freq, phần tử nào có freq[val] == 3 thì đó là phần tử chung.
* Ưu điểm: nhanh, không cần thao tác giao phức tạp.

3. Thuật toán chi tiết (chọn Cách 2 vì trực quan)

1. Đọc vào nx, sau đó đọc dãy x. Dùng setX để giữ phần tử duy nhất trong dãy x.
2. Tương tự cho dãy y (setY) và dãy z (setZ).
3. Khai báo mảng freq[10001] = {0}.
   * Với từng phần tử trong setX, tăng freq[val] thêm 1.
   * Với từng phần tử trong setY, tăng freq[val] thêm 1.
   * Với từng phần tử trong setZ, tăng freq[val] thêm 1.
4. Duyệt i từ 0 đến 10000: nếu freq[i] == 3 thì đó là phần tử chung.
5. Gom lại các phần tử chung → đưa vào mảng kết quả.
6. In ra số lượng và danh sách phần tử theo thứ tự tăng dần (mảng đã duyệt từ nhỏ đến lớn nên tự động tăng dần).

4. Ví dụ minh họa (tự chạy bằng tay)

|  |
| --- |
| Input: |
| 5  1 2 5 4 3  4  5 6 1 4  5  3 5 3 5 1 |

* Dãy X = {1,2,5,4,3}
* Dãy Y = {5,6,1,4}
* Dãy Z = {3,5,1}

Bảng freq sau khi xử lý:

* freq[1] = 3 (có trong X, Y, Z)
* freq[5] = 3 (có trong X, Y, Z)
* freq[2] = 1, freq[3] = 2, freq[4] = 2, freq[6] = 1

👉 Phần tử chung = {1, 5}

|  |
| --- |
| Output: |
| 2 |
| 1 5 |

5. Độ phức tạp

* Thời gian: O(nx + ny + nz + 10000) ≈ O(10000) → rất nhanh.
* Bộ nhớ: mảng freq kích thước 10001 + 3 set nhỏ.

Code C++:

#include <iostream>

#include <set>

#include <vector>

using namespace std;

int main() {

int nx, ny, nz;

cin >> nx;

set<int> setX, setY, setZ;

for (int i = 0; i < nx; i++) {

int val; cin >> val;

setX.insert(val);

}

cin >> ny;

for (int i = 0; i < ny; i++) {

int val; cin >> val;

setY.insert(val);

}

cin >> nz;

for (int i = 0; i < nz; i++) {

int val; cin >> val;

setZ.insert(val);

}

const int MAXV = 10000;

int freq[MAXV + 1] = {0};

for (int val : setX) freq[val]++;

for (int val : setY) freq[val]++;

for (int val : setZ) freq[val]++;

vector<int> result;

for (int i = 0; i <= MAXV; i++) {

if (freq[i] == 3) result.push\_back(i);

}

cout << result.size() << "\n";

for (int v : result) cout << v << " ";

cout << "\n";

return 0;

}