Đề thi lọc tuyển thi thành phố 2024 - 2025 - Trần Đại Nghĩa

Câu 1

Đề bài:

- Tìm số chia hết cho 10 gần n nhất.v

Trường THPT Chuyên Trần Đại Nghĩa
Tổ Tin học

KIỂM TRA CHỌN ĐỘI TUYỂN CHÍNH THỨC
ĐỘI TUYỂN TIN 9 - NĂM 2024.2025
Ngày kiểm tra: 04.01.2025
Thời gian: 120'

Học sinh lập trình giải các bài tập sau, dùng ngôn ngữ Pascal, C++.

Học sinh lưu bài làm theo quy định sau:

1. Lưu tất cả các file code vào D:\ STT_HotenHS (họ tên không dấu)
2. Đọc dữ liệu từ tệp và ghi kết quả vào tệp.
3. Chấm bài bằng phần mềm Themis phiên bản mới nhất.

Bài 1: **BAI1.*** (4đ)

An có một số nguyên *N* không âm. Cậu ta muốn làm tròn nó đến số nguyên gần nhất mà chia hết cho *10*. Nếu *N* chia hết cho *10*, An coi như nó đã được làm tròn.

Cho biết số *N*, hãy tìm số nguyên mà An muốn làm tròn.

Input: Số nguyên *N* mà An muốn làm tròn.

Output: Xuất ra kết quả là số nguyên đã làm tròn theo cách của An (xuất ra bất kỳ đáp án nào nếu đáp án không duy nhất).

Ràng buộc: $0 \le N \le 10^9$;

BAI1.INP	BAI1.OUT
5	0 (10 cũng đúng)
113	110
1000000000	1000000000
5432359	5432360

Bài 2 **BAI2.* (4đ)**

Trên trục tọa độ Ox, cho N điểm khác nhau P_1, P_2, .., P_N. Đoạn thẳng AB được gọi là đoạn thẳng chia đều nếu nó được xác định bởi 3 điểm cho trước A, B, M sao cho M là trung điểm của AB.

Yêu cầu: Cho biết tọa độ của N điểm P_1, P_2, .., P_N. Hỏi có bao nhiêu đoạn thẳng chia đều được tạo ra từ các điểm đã cho?

Input: vào từ tệp văn bản **BAI2.INP** gồm:

- Dòng 1: ghi số nguyên dương N;

Code

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define endl "\n"
int main(){
    ll n;
    cin>>n;
    if(n % 10 == 0){
        cout<<n;
        return 0;
    }
    ll endnum = n % 10;

    if(endnum <= 5){
        cout<<n - endnum;
    }
    if(endnum > 5){
        cout<<n + (10-endnum);
    }
    return 0;
}
```

Câu 2

- Dòng 2: ghi lần lượt các số $x_1, x_2, .., x_N$ ($|x_i| \leq 10^5$) tương ứng là tọa độ của các điểm $P_1, P_2, .., P_N$.

Output: ghi ra tệp văn bản **BAI2.OUT** gồm một số duy nhất là kết quả tìm được của bài toán.

106

Ví dụ:

BAI2.INP	BAI2.OUT
5 3 -1 2 5 4	3

Ghi chú: Các số thực trong bộ test được so sánh là bằng nhau nếu trị tuyệt đối hiệu giữa chúng < 10^{-10}

Bài 3 **BAI3.* (4đ)**

Hai số nguyên dương A và B được gọi là một cặp số tương đồng nếu như chúng có chung tập các ước nguyên tố.

Ví dụ: 12 và 18 là cặp số tương đồng vì có chung tập ước nguyên tố là $\{2, 3\}$.

Yêu cầu: Cho trước hai số nguyên dương L và R, hãy đếm số lượng cặp tương đồng A và B mà $L \leq A < B \leq R$.

Input: vào từ file văn bản **BAI3.INP**

- Gồm một dòng duy nhất chứa hai số nguyên dương L và R ($L < R \leq 10^6$).

Output: ghi ra file văn bản **BAI3.OUT** một số nguyên duy nhất là kết quả bài toán.

Ví dụ:

BAI3.INP	BAI3.OUT	Giải thích
1 10	4	Có 4 cặp số tương đồng đó là: {2 , 4} , {2 , 8} , {3 , 9} , {4 , 8

Ràng buộc:

- Có 60% số test ứng với 60% số điểm của bài có $R - L \leq 1000$
- 40% số test còn lại ứng với 40% số điểm của bài không giới hạn gì thêm.

Đề bài:

Cho N điểm trên tọa độ Ox. Tìm số lượng cặp điểm (A, B) sao cho tồn tại một điểm M là trung điểm của A và B.

Input

```
5
3 -1 2 5 4
```

Output

```
3
```

Nhận xét

Tính tọa độ trung điểm

- Công thức tính tọa độ trung điểm của 2 điểm A, B trên trục Ox là: $(A + B)/2$.

Trường hợp số thực

- 2 Số thực có chênh lệch 10^{-10} thì coi như bằng nhau.

Code

Cách 1: Duyệt các cặp (i, j) và dùng map đánh dấu để kiểm tra - $O(n^2 logn)$

▶ Code C++

Cách 2 - số nguyên: Duyệt các cặp (i, j) và dùng mảng đánh dấu để kiểm tra - $O(n^2)$

- Kỹ thuật padding để tránh số âm trên mảng.

▶ Code C++

Cách 3 - Số thực: 2 con trỏ - $O(n^2)$

▶ Code C++

Câu 3

- Dòng 2: ghi lần lượt các số $x_1, x_2, .., x_N$ ($|x_i| \leq 10^5$) tương ứng là tọa độ của các điểm $P_1, P_2, .., P_N$.

Output: ghi ra tệp văn bản **BAI2.OUT** gồm một số duy nhất là kết quả tìm được của bài toán.

106

Ví dụ:

BAI2.INP	BAI2.OUT
5 3 -1 2 5 4	3

Ghi chú: Các số thực trong bộ test được so sánh là bằng nhau nếu trị tuyệt đối hiệu giữa chúng < 10^{-10}

Bài 3 **BAI3.* (4đ)**

Hai số nguyên dương A và B được gọi là một cặp số tương đồng nếu như chúng có chung tập các ước nguyên tố.

Ví dụ: 12 và 18 là cặp số tương đồng vì có chung tập ước nguyên tố là $\{2, 3\}$.

Yêu cầu: Cho trước hai số nguyên dương L và R, hãy đếm số lượng cặp tương đồng A và B mà $L \leq A < B \leq R$.

Input: vào từ file văn bản **BAI3.INP**

- Gồm một dòng duy nhất chứa hai số nguyên dương L và R ($L < R \leq 10^6$).

Output: ghi ra file văn bản **BAI3.OUT** một số nguyên duy nhất là kết quả bài toán.

Ví dụ:

BAI3.INP	BAI3.OUT	Giải thích
1 10	4	Có 4 cặp số tương đồng đó là: {2 , 4} , {2 , 8} , {3 , 9} , {4 , 8

Ràng buộc:

- Có 60% số test ứng với 60% số điểm của bài có $R - L \leq 1000$
- 40% số test còn lại ứng với 40% số điểm của bài không giới hạn gì thêm.

Bài 4 BAI4.* (4đ)

Có một lưới ô vuông có kích thước NxN được đánh chỉ số hàng từ 1 đến n (theo chiều từ trên xuống dưới) và chỉ số cột từ 1 đến N theo chiều từ trái sang phải. Tại ô (1, 1) người ta đặt một con robot tự di chuyển. Mỗi lần di chuyển, robot chỉ đi sang phải một ô hoặc đi xuống dưới 1 ô. Trong lưới ô vuông này người ta đặt một số viên đá vào trong một số ô vuông để làm vật cản.

Yêu cầu: Hãy tính xem có bao nhiêu đường đi từ ô (1, 1) đến ô (N, N) biết rằng robot không thể đi vào ô có vật cản và hai đường đi được gọi là khác nhau nếu có ít nhất một ô thuộc đường đi này nhưng không thuộc đường đi kia.

Ví dụ: xét lưới ô vuông kích thước 3 x 3 như hình vẽ sau: trong lưới ô vuông có 2 viên đá được đặt vào ô (1,3) và ô (2, 1). Với dữ kiện trên thì robot có tất cả 2 đường đi như sau:

1. (1, 1) → (1, 2) → (2, 2) → (2, 3) → (3, 3)
2. (1, 1) → (1, 2) → (2, 2) → (3, 2) → (3, 3)

Input: vào từ file văn bản **BAI4.INP**

- Dòng đầu chứa 2 số nguyên $N,\ M\ (N > M)$
- M dòng tiếp theo. Mỗi dòng ghi 2 số nguyên dương i và j là chỉ số hàng và chỉ số cột của ô đặt viên đá.

Output: Ghi ra file **BAI4.OUT**

- Ghi ra một số nguyên duy nhất là kết quả tìm được.

Ví dụ

BAI4.INP	BAI4.OUT
3 2 1 3 2 1	2

Ràng buộc:

- 30% test ứng với N ≤ 10
- 40% test ứng với 10 < N ≤ 30
- 30% test ứng với 30 < N ≤100

Đề bài:

2 số nguyên A, B gọi là tương đồng nếu như có chung tập các ước số nguyên tố.

- VD: $A = 12, B = 18$ thì A và B tương đồng vì chúng có chung tập ước số nguyên tố là $\{2, 3\}$.

Yêu cầu: Đếm số lượng cặp tương đồng (A, B) trong đoạn $[L, R]$.

- Với $L, R \leq 10^6$.

Input

```
1 10
```

Output

```
4
```

Các công cụ cần thiết

Công cụ 1: Tập ước số nguyên tố - Gọi là hàm `primes_divisor(n)`

Ý nghĩa: `primes_divisor(n)` trả về tập ước số nguyên tố của n.

- `primes_divisor(12) = {2, 3}` .

Một số cách làm có thể sử dụng với các độ phức tạp thời gian khác nhau:

- *Cách 1*: Duyệt từ $2 \rightarrow n$. Độ phức tạp $O(n)$.
  - Nếu n chia hết cho i thì thêm i vào tập ước số nguyên tố của n.
  - Tiếp tục chia n cho i cho đến khi n không chia hết cho i.
  - Độ phức tạp: Nếu n là số nguyên tố thì vòng lặp sẽ cần chạy tới n. Như vậy, độ phức tạp sẽ là $O(n)$.
- *Cách 2*: Độ phức tạp $O(\sqrt{n})$.
  - Duyệt từ $2 \rightarrow \sqrt{n}$.
    - Nếu n chia hết cho i thì thêm i.
    - Tiếp tục chia n cho i cho đến khi n không chia hết cho i.
  - Khi kết thúc vòng lặp, nếu $n > 1$ thì thêm n vào tập ước số nguyên tố của n (ban đầu).
    - Suy nghĩ và tự chứng minh.

    - Tham khảo ở đây: VNOI - Mục: 'Phân tích số nguyên lớn hơn'
    - Ví dụ: 12 = 2 * 2 * 3.
        - Ban đầu chạy tới 2, thì 12 chia hết cho 2, thêm 2 vào tập ước số nguyên tố của 12.
            - Chia 12 cho 2, ta được 6.
            - Tiếp tục chia 6 cho 2, ta được 3.
            - Lúc này, $i = 2$ và $n = 3$.
        - Ở vòng lặp tiếp theo, $i = 3$, kiểm tra thấy: $i > \sqrt{n} (n = 3)$ nên dừng vòng lặp.
        - Lúc này, $n = 3 > 1$ nên thêm 3 vào tập ước số nguyên tố của 12.
- Cách 3: Sàng nguyên tố + Phân tích số nguyên tố $(O(\log n))$.
    - Sàng nguyên tố để tìm ước nguyên tố nhỏ nhất của n.
        - $snt[x] = y$ thì y là ước nguyên tố nhỏ nhất của x.
    - Phân tích số nguyên tố:
        - Duyệt từ $x \to 1$.
        - Nếu $snt[x] = y$ thì thêm y vào tập ước số nguyên tố của n.
        - Gán $x = x/y$.
        - Lặp lại cho đến khi $x = 1$.
    - Tham khảo ở đây: Phân tích số nguyên tố - Mục: Phân tích số nguyên nhỏ

Nhận xét

Trâu bò

Các bước làm như sau:

1. Chạy mỗi cặp số (a, b) trong đoạn $[L, R]$.
2. Nếu `primes_divisor(a) == primes_divisor(b)` thì tăng biến đếm lên 1.

Độ phức tạp: $O((R - L)^2 \times primes\_divisor)$.

Tối ưu

Câu hỏi 1: Giả sử (a, b) và (b, c) tương đồng thì (a, c) có tương đồng không?

- Đáp án: Có. Vì ...
- Như vậy, ta có thể thực hiện việc gom nhóm các số tương đồng lại với nhau. Sau đó, bất kỳ 2 số nào trong cùng 1 nhóm thì chúng tương đồng với nhau.
    - Lúc này, nhóm có k phần tử thì số cặp tương đồng sẽ là $k \times (k - 1)/2$.
- Ví dụ: $2, 4, 8$ đều có ước nguyên tố là 2. Thì:
    - $2, 4, 8$ sẽ tương đồng với nhau.

Câu hỏi 2: Làm sao để gom nhóm các số tương đồng lại với nhau?

- Đáp án: Tự suy nghĩ đê

Như vậy các bước làm như sau:

1. Gom các số từ $L \rightarrow R$ vào các nhóm.
2. Duyệt qua từng nhóm và tính số cặp tương đồng trong nhóm đó.
3. Cộng kết quả lại với nhau.

Ví dụ, 1 -> 10:

- Nhóm 1: 2, 4, 8 => {2} => Đánh index 1
- Nhóm 2: 3, 9 => {3} => Đánh index 2
- Nhóm 3: 5 => {5} => Đánh index 3
- Nhóm 4: 6 => {2, 3} => Đánh index 4
- Nhóm 5: 7 => {7}
- Nhóm 6: 10 => {2, 5}

Code

▶ Code C++

Câu 4

Bài 4 BAI4.* (4đ)

Có một lưới ô vuông có kích thước NxN được đánh chỉ số hàng từ 1 đến n (theo chiều từ trên xuống dưới) và chỉ số cột từ 1 đến N theo chiều từ trái sang phải. Tại ô (1, 1) người ta đặt một con robot tự di chuyển. Mỗi lần di chuyển, robot chỉ đi sang phải một ô hoặc đi xuống dưới 1 ô. Trong lưới ô vuông này người ta đặt một số viên đá vào trong một số ô vuông để làm vật cản.

Yêu cầu: Hãy tính xem có bao nhiêu đường đi từ ô (1, 1) đến ô (N, N) biết rằng robot không thể đi vào ô có vật cản và hai đường đi được gọi là khác nhau nếu có ít nhất một ô thuộc đường đi này nhưng không thuộc đường đi kia.

Ví dụ: xét lưới ô vuông kích thước 3 x 3 như hình vẽ sau: trong lưới ô vuông có 2 viên đá được đặt vào ô (1,3) và ô (2, 1). Với dữ kiện trên thì robot có tất cả 2 đường đi như sau:

1. (1, 1) → (1, 2) → (2, 2) → (2, 3) → (3, 3)
2. (1, 1) → (1, 2) → (2, 2) → (3, 2) → (3, 3)

Input: vào từ file văn bản **BAI4.INP**

- Dòng đầu chứa 2 số nguyên $N,\ M\ (N > M)$
- M dòng tiếp theo. Mỗi dòng ghi 2 số nguyên dương i và j là chỉ số hàng và chỉ số cột của ô đặt viên đá.

Output: Ghi ra file **BAI4.OUT**

- Ghi ra một số nguyên duy nhất là kết quả tìm được.

Ví dụ

BAI4.INP	BAI4.OUT
3 2 1 3 2 1	2

Ràng buộc:

- 30% test ứng với N ≤ 10
- 40% test ứng với 10 < N ≤ 30
- 30% test ứng với 30 < N ≤100

Đề bài:

Bảng $N \times N$ và mỗi ô có thể bị chặn. Tìm số cách đi từ ô $(1, 1)$ đến ô (N, N) mà không đi qua ô nào bị chặn.

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define endl "\n"

void init() {
    freopen("BAI4.INP", "r", stdin);
    freopen("BAI4.OUT", "w", stdout);
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
}

string cong(string a, string b) {
    ll len_a = a.size();
    ll len_b = b.size();
    ll len = max(len_a, len_b);
    vector<int> res(len + 1, 0);
    ll carry = 0;

    while (a.size() < len) a = "0" + a;
    while (b.size() < len) b = "0" + b;

    for (ll i = len - 1; i >= 0; i--) {
        int sum = (a[i] - '0') + (b[i] - '0') + carry;
        res[i + 1] = sum % 10;
        carry = sum / 10;
    }
    res[0] = carry;

    string result = "";
    bool leading_zero = true;
    for (int i = 0; i <= len; i++) {
        if (res[i] == 0 && leading_zero) continue;
        leading_zero = false;
        result += (res[i] + '0');
    }
    if (leading_zero) result = "0";
    return result;
}
```

```cpp
int main() {
    //init();
    int n, m;
    cin >> n >> m;
    vector<vector<string>> a(n + 1, vector<string>(n + 1, "1"));
    vector<vector<string>> f(n + 1, vector<string>(n + 1, "0"));


    for (int i = 1; i <= m; i++) {
        int a1, b;
        cin >> a1 >> b;
        a[a1][b] = "0";
    }
    f[1][1] = a[1][1];

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (a[i][j] == "0") {
                f[i][j] = "0";
            } else {
                if (i > 1) f[i][j] = cong(f[i][j], f[i - 1][j]);
                if (j > 1) f[i][j] = cong(f[i][j], f[i][j - 1]);
            }
        }
    }

    cout << f[n][n] << endl;

    return 0;
}
```

Câu 5

Bài 5 BAI5.* (4đ)

Phép nén của một số nguyên dương K là gán số đó thành số nguyên dương nhỏ nhất mà K không chia hết. Ví dụ phép nén số của 120 là 7, vì 7 là số nguyên dương nhỏ nhất mà 120 không chia hết. Độ nén của một số là số lượng phép nén để số đó thành số 2. Ví dụ số 120 ta có: 120 →7→2 vậy độ nén của 120 là 2.

Yêu cầu: Cho hai số nguyên dương A, B ($2 < A < B < 10^{18}$). Tính tổng độ nén của các số lớn hơn hoặc bằng A và nhỏ hơn hoặc bằng B.

Input: vào từ file BAI5.INP, gồm một dòng duy nhất chứa hai số A và B.

Ouput: ghi ra file BAI5.OUT gồm số nguyên duy nhất là kết quả của bài toán.

Ví dụ:

BAI5.INP	BAI5.OUT	Giải thích
3 7	8	Độ nén của 3 là 1; Độ nén của 4 là 2; Độ nén của 5 là 1; Độ nén của 6 là 3; Độ nén của 7 là 1; Tổng độ nén là: 8.

Ràng buộc: 60% số test tương ứng 60% số điểm của bài có $B<10^6$

Đề bài:

Độ nén của 1 số N là:

1. Số lần để biến N thành 2.
2. Mỗi lần biến đổi, N sẽ biến thành số nguyên dương nhỏ nhất mà N không chia hết.

Nhận xét

Nhận xét 1: Biến đổi bài toán về 1 chiều.

- Cơ bản:

$$nen(L \to R) = nen(1 \to R) - nen(1 \to L-1)$$

- Bài toán: tính $nen(1 \to R)$ với $R \leq 10^{18}$.

Nhận xét 2 (Lời phán xử cho bài toán): Biến đổi khoảng bài toán về đoạn nhỏ hơn.

- Dù ban đầu, đoạn cần xử lý là rất lớn $1 \to 10^{18}$. Nhưng ta thấy rằng:
  - Mỗi số sẽ được nén không quá nhiều lần.
  - Sau lần nén đầu tiên thì các số từ $1 \to 10^{18}$ sẽ biến thành các số trong đoạn $2 \to 100$. (Thật ra là nhỏ hơn 100 khá nhiều)
- Và sau khi nén qua lần đầu tiên, các số về những giá trị giống nhau thì chi phí nén tiếp theo sẽ là như nhau. Vì vậy ta có thể giảm đoạn cần xử lý xuống như sau.
- Gọi:
  - $f[i]$ là số lượng số trong đoạn $1 \to R$ mà sau lần nén đầu tiên sẽ biến thành i.
  - $nen[i]$ là số lần nén của số i.
- Lúc này kết quả bài toán sẽ là:

$$nen(1 \to R) = \sum_{i=1}^{100} f[i] \times (nen[i] + 1)$$

Câu hỏi quan trọng: Làm sao để tính $f[i]$?

Nhận xét 3: Nếu sau 1 lần nén, X trở thành y thì ta có kết luận gì về X?

- X không chia hết cho y.
- X chia hết cho $1, 2, 3, ..., y-1$. (Nhận xét này quan trọng)

Nhận xét 4: X chia hết cho $1, 2, 3, ..., y-1$ thì ta khai thác được điều gì nữa?

- X sẽ chia hết cho $LCM(1, 2, 3, ..., y - 1)$.
- Lưu ý: X lúc này cũng sẽ chia hết cho $LCM(1, 2, 3, ..., y - 2)$ nên lúc tìm $f[i]$ phải hết sức cẩn thận.

Gọi:

- $g[i]$ là số lượng số trong đoạn $1 \rightarrow R$ mà chia hết cho $LCM(1, 2, 3, ..., i)$.

Nhận xét 5: Cách tính $f[i]$ là số lượng số chia hết cho $LCM(1, 2, 3, ..., i - 1)$ nhưng lại không chia hết cho i.

- $f[i] = g[i - 1] - g[i]$.
- Tại sao lại đúng? (Tự suy nghĩ xem)

Code

Code trâu

▶ Code C++

Code tối ưu

▶ Code C++

Trình chấm

▶ Code C++