GỢI Ý LỜI GIẢI BÀI THI OLP TIN HỌC SINH VIÊN ICTU 2017

Dưới đây là đề bài, hướng dẫn giải, lời giải của đề thi. Code được viết bằng ngôn ngữ C, các ngôn ngữ khác làm tương tự. Tất cả các bài đều dùng nhập xuất chuẩn (nhập từ bàn phím, xuất ra màn hình).

Bài 1: Doanh thu (50 điểm)
Đề bài:

DOANHTHU
Thời gian chạy: 1 gây
Điểm: 50

Các nhà kinh tế thế giới đang làm một cuộc thống kê về doanh thu của các doanh nghiệp trên toàn cầu. Họ đi khảo sát ngẫu nhiên các doanh nghiệp của các quốc gia và ghi lại doanh thu (tính bằng $). Vấn đề là có rất nhiều doanh nghiệp được khảo sát và số tiền của nhiều doanh nghiệp cũng rất lớn (có thể đến 1000 tỷ $) làm cho các nhà kinh tế dễ nhầm lẫn. Nhiệm vụ của bạn là giúp các nhà kinh tế lọc ra doanh thu lớn nhất và doanh thu nhỏ nhất của các doanh nghiệp được khảo sát.

Đầu vào:
- Dòng thứ nhất gồm 1 số nguyên n là số lượng doanh nghiệp khảo sát ($5 <= n <= 10^5$).
- Dòng thứ hai gồm n số nguyên a_i với a_i là doanh thu của doanh nghiệp thứ i, mỗi số cách nhau bởi một dấu cách ($|a_i| <= 10^{12}$).

Đầu ra:
Gồm 2 số, mỗi số nằm trên một dòng và lần lượt là doanh thu lớn nhất và nhỏ nhất tìm được.

Ví dụ:

DOANHTHU.INP	DOANHTHU.OUT
10 1 2 8 8 3 4 11 6 6 7	11 1

Gợi ý, lời giải:
Bản chất của bài toán là nhập vào n số nguyên, tìm và in ra số lớn nhất, nhỏ nhất của chúng. Chúng ta dùng một mảng **a** để lưu các số. Dùng biến max và min để lưu giá trị lớn nhất và nhỏ nhất. Tuy nhiên cần lưu ý:

- Ban đầu gán max và min có giá trị là a[0] không nên gán là 0 hoặc bất kỳ một số nào khác.
- Kiểu dữ liệu của mảng a và max, min nên để là ***long long*** vì $|a_i| <= 10^{12}$. Các bạn có thể tham khảo giới hạn độ lớn các kiểu trên mạng. Khi dùng ***long long***, định dạng nhập xuất là ***%lld***.

Code: file name: DOANHTHU.c

```c
#include <stdio.h>

int main()
{
	int n, i;
	scanf("%d", &n);

	long long a[n], max, min;

	for (i = 0; i < n; i++)
	{
		scanf("%lld", &a[i]);
	}

	max = min = a[0];

	for(i = 0; i < n; i++)
	{
		if(max < a[i]) max = a[i];
		if(min > a[i]) min = a[i];
	}

	printf("%lld\n%lld", max, min);

	return 0;
}
```

Bài 2: Sim số đẹp (30 điểm)
Đề bài:

SIMSODEP
Thời gian chạy: 2 giây
Điểm: 30

Để chúc mừng cho sinh nhật của Cường nên bố mẹ đã quyết định mua tặng Cường một chiếc iPhone X. Tất nhiên đi kèm với điện thoại sẽ là một chiếc sim số đẹp. Cường muốn tự lựa cho mình một số đẹp theo cách của riêng mình. Bố mẹ đã đưa Cường đến một cửa hàng sim số đẹp, Cường muốn lựa một sim thỏa mãn điều kiện 5 số cuối vừa là số đối xứng vừa là một số nguyên tố.

Số đối xứng là số đọc từ trái qua phải hay phải quá trái thì vẫn là một số (ví dụ 12321). (Lưu ý bài này chỉ xét các số có 5 chữ số có đối xứng không tương ứng với 5 chữ số cuối của sim).
Số nguyên tố là số nguyên dương mà chỉ có 2 ước duy nhất là 1 và chính nó (ví dụ số 7)

Cho hai số nguyên a, b thỏa mãn $10^4 <= a < b < 10^5$
Bạn hãy giúp Cường đếm số lượng sim số đẹp thỏa mãn yêu cầu trên mà nằm trong đoạn [a,b].

Đầu vào:
Hai số nguyên a, b cách nhau bởi một dấu cách ($10^4 <= a < b < 10^5$)
Đầu ra:
Một số nguyên duy nhất là số lượng các số sim đẹp.

Ví dụ

SIMSODEP.INP	SIMSODEP.OUT
11111 33333	32

Gợi ý, lời giải:
Bản chất bài toán chỉ là đếm số lượng các số vừa là số nguyên tố, vừa là số đối xứng trong đoạn [a,b] ($10^4 <= a < b < 10^5$).
Do vậy các bạn chỉ cần tìm hiểu cách kiểm tra 1 số có là số nguyên tố không và kiểm tra 1 số có đối xứng không là được. Sau đó duyệt từng số từ a đến b. Lưu ý là xét trong đoạn [a, b] nên ta phải xét cả a và b.

Cách kiểm tra một số n có là số nguyên tố không:

- Nếu n < 2 thì không phải là số nguyên tố.
- Nếu n > 2. Duyệt từ 2 đến căn bậc hai của n, nếu n chia hết cho bấy kỳ một số nào trong đoạn này thì n không phải là số nguyên tố, nếu sau khi duyệt mà nó không chia hết cho số nào thì n là số nguyên tố.

Cách kiểm tra n (có 5 chữ số) có đối xứng không.
Giả sử n gồm 5 chữ số abcde.
Lấy a so sánh với e, Lấy b so sánh với d. Nếu a = e và b = d thì n là số đối xứng. Ngược lại thì không phải. Ta có:
a = n / 10000.
e = n % 10.
Sau khi so sánh thấy a = e ta loại bỏ a và e ra khỏi n để lấy bcd bằng cách:
n = (n%10000) / 10 (Trong đó: % 10000 để lấy bcde, chia tiếp cho 10 để lấy bcd). Từ đây có:
b = n / 100
d = n % 10
("%" là phép chia lấy dư, do n là số nguyên, 10, 100, 10000 cũng là số nguyên nên "/" là phép chia lấy phần nguyên).

Code: File name: SIMSODEP.c

```c
#include <stdio.h>    // scanf, printf
#include <math.h> // sqrt

/* Kiem tra so n la so nguyen to khong
      Tra ve 1 neu la so nguyen to.  Tra ve 0 neu khong la so nguyen to
*/

int ktNT (int n) {
      if(n < 2) return 0;
      int m = sqrt(n);
      for(int i = 2; i <= m; i++ ) {
            if(n % i == 0) return 0;
      }
      return 1;
}

/* Kiem tra so n co la so doi xung khong (n co 5 chu so)
      Tra ve 1 neu la so doi xung.  Tra ve 0 neu khong la so doi xung
*/

int ktDX(int n) {
      if(n % 10 == n / 10000) {
            n = (n%10000) / 10;
            if(n % 10 == n / 100) {
                  return 1;
            }
      }
      return 0;
}
```

```
int main()
{
	int a, b, i, dem = 0;
	scanf("%d%d", &a, &b);
	for(i = a; i <= b; i++) {
		if(ktNT(i) && ktDX(i)) {
			dem++;
		}
	}
	printf("%d", dem);
	return 0;
}
```

Bài 3: Robot (20 điểm)
Đề bài:

ROBOT

Thời gian chạy: 1 gây
Điểm: 20

Trường ĐH CNTT&TT đang lập đội thi Robocon 2018. Luật của Robocon năm 2018 là hãy làm cho robot chạy nhanh nhất từ điểm xuất phát tới đích, tuy nhiên theo một số quy tắc sau:

Cho một lưới các ô vuông ghi các số nguyên từ 0 đến 100. Các số này là thời gian (giây) phải dừng lại của robot khi đi vào ô đó. Tức nếu robot đi vào ô có giá trị 5 thì phải dừng lại 5 giây. Robot luôn phải đi hướng từ trái sang phải tức là có thể đi chéo lên góc phải, chéo xuống góc phải, hoặc thẳng sang phải. Mỗi lần đi chỉ được đi sang một ô.

Ban đầu, robot có thể xuất phát từ bất cứ ô nào của cột đầu tiên và phải đi theo quy tắc trên đến bất cứ một ô nào của cột cuối cùng thì dừng lại.

Đội thi đang mắc một vấn đề là làm sao tìm được đường đi để tổng thời gian về đích của robot là ngắn nhất (Không tính thời gian đi chuyển, chỉ tính thời gian bị dừng lại tại các ô).

Đầu vào:

- Dòng thứ nhất gồm 2 số nguyên dương n, m lần lượt là số hàng và số cột của lưới.
- n dòng tiếp theo, mỗi dòng chứa m số (mỗi số có giá trị từ 0 đến 100).

Đầu ra:

- Một số duy nhất là thời gian ngắn nhất robot có thể đi từ trái qua phải.

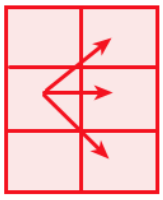
Ví dụ:

ROBOT.INP	ROBOT.OUT
4 3 4 5 7 5 2 1 6 5 3 5 6 9	7

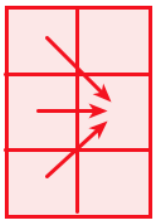
Giải thích: Với lưới ô vuông 4x3 như trên, robot có thể đi như sau để được tổng thời gian dừng là bé nhất: A[0][0] → A[1][1] → A[1][2] , do vậy tổng thời gian dừng là 4 + 2 + 1 = 7.

Gợi ý, lời giải:
Bài này bản chất là tìm đường đi sao cho từ một ô bất kỳ cột đầu tiên đên ô bất kỳ cột cuối cùng mà tổng đường đi là bé nhất.

Giả sử ta tìm và tính được tất cả các ô tại cột cuối cùng là bé nhất có thể khi đi từ một ô nào đó ở cột đầu tiên đến cột cuối cùng. Khi đó kết quả chính là ô bé nhất trong các ô ở cột cuối cùng. Theo quy tắc từ một ô, ta có thể đi tới 3 ô ở cột tiếp theo như sau:



Do vậy, ta cũng có thể suy ra là để đi tới một ô nào đó, ta có thể xuất phát từ một trong 3 ô ở cột trước.



Giả sử ta đi tới ô [i][j] thì ô này sẽ có thể đi từ 3 ô cột trước là:
[i-1][j-1], [i][j-1], [i+1][j-1]. Để ô [i][j] bé nhất, ta chỉ cần cộng giá trị của nó với ô bé nhất trong 3 ô kia là được. Cứ làm theo quy tắc đó, ta có thể tính lần lượt các của các cột từ trái sang phải.

Ta tạo ra ma trận b để chứa kết quả đường đi. Vậy lưới ô vuông như ví dụ. ta tính được như sau: (ma trận a bên trái, ma trận b bên phải).

4	5	7
5	2	1
6	5	3
5	6	9

4	9	13
5	6	7
6	10	9
5	11	19

- Cột đầu tiên của ma trận b ta giữ nguyên như cột đầu tiên của ma trận a.
- Để đi đến ô b[0][1], ta có thể đi từ b[0][0] hoặc b[1][0], nhưng để b[0][1] bé nhất thì phải chọn đi từ b[0][0] khi đó b[0][1] = a[0][1] + b[0][0] = 5 + 4 = 9.
- Để đi đến b[1][1] ta có thể đi từ b[0][0] hoặc b[1][0] hoặc b[2][0], nhưng để b[1][1] bé nhất ta phải chọn đi từ b[0][0], khi đó b[1][1] = a[1][1] + b[0][0] = 2 + 4 = 6.

- Tương tự với các ô khác, ta sẽ có được bảng thứ 2 như trên.

Thuật toán này gọi là Quy hoạch động. Ngoài ra các bạn có thể sử dụng các thuật toán duyệt đồ thị đếm làm bài này cũng được nhưng sẽ phức tạp hơn chút.

Để code bài này, chúng ta cần lưu ý là đi đến ô [i][j] có thể đi từ 3 ô là [i-1][j-1], [i][j-1], [i+1][j-1], do vậy cẩn thận với i – 1, j – 1. Chúng ta thêm các ô bao quanh để tránh bị sai sót. Do số lớn nhất của 1 ô là 100, do vậy tổng lớn nhất của tất cả các ô là 10^6, chúng ta tìm nhỏ nhất nên chỉ cần để các ô bao quanh là 10^6 +1 (M) thì trong quá trình tính toán sẽ không ảnh hưởng gì đến kết quả bé nhất.

M	M	M
4	9	13
5	6	7
6	10	9
5	11	19
M	M	M

File name: ROBOT.c

```
#include <stdio.h>

#define M  1E6 + 1

/* Tim so lon nha trong 2 so */
int min2 (int a, int b)
{
	return a < b ? a : b;
}

/* Tim so lon nha trong 3 so */
int min3 (int a, int b, int c)
{
	return min2 ( a, min2 (b, c) );
}

int main()
{
	int m, n, i, j, s;
	int a[102][100];
	int b[102][100];

	scanf("%d%d", &m, &n);
```

```
	for(i = 1; i <= m; i++)
		for(j = 0; j < n; j++)
			scanf("%d", &a[i][j]);

	// cho cot dau cua b bang cot dau cua a
	for (i = 1; i <= m; i++)
		b[i][0] = a[i][0];

	// hang dau va hang cuoi cua b nhan gia tri lon nhat M
	for(j = 0; j < n; j++)
	{
		b[0][j] = M;
		b[m+1][j] = M;
	}

	// Tinh toan ma tran b.
	for(j = 1; j < n; j++)
		for(i = 1; i <= m; i++)
			b[i][j] = a[i][j] + min3( b[i-1][j-1], b[i][j-1], b[i+1][j-1] );

	// duyet cot cuoi cung cua ma tran b tim min trong cot cuoi.
	s = M;
	for(i=1; i<= m; i++)
		if(s > b[i][n-1])
			s = b[i][n-1];

	printf("%d", s);

	return 0;
}
```