Second Min

(1 sec, 512mb)

จงเขียนฟังก์ชัน int get_second_min(stack<int> &s) โดยรับประกันว่า s เป็น stack ที่มีข้อมูลจำนวนเต็มแบบ int ที่มีข้อมูลอย่างน้อย 2 ตัวที่แตกต่างกัน ฟังก์ชันนี้จะต้องคืนค่าข้อมูลที่อยู่ใน s ที่มีค่าน้อยที่สุดเป็นลำดับที่ 2 (กล่าวคือ คืนข้อมูลที่มีค่าน้อยที่สุดที่มากกว่าข้อมูลที่น้อยที่สุดใน s) โดยมีข้อกำหนดเพิ่มเติมคือ เมื่อจบการทำงานของฟังก์ชันแล้ว แล้ว ค่าใน s จะต้องเหมือนกับตอนเริ่มเรียกใช้งาน อย่างไรก็ตาม ระหว่างการทำงานของฟังก์ชัน เราสามารถปรับเปลี่ยน s อย่างไรก็ได้

ข้อบังคับ

ในโจทย์ข้อนี้จะมี code เริ่มต้นมาให้แล้ว (แสดงอยู่ด้านล่างของโจทย์) ให้นิสิตเขียนโปรแกรมเพิ่มเติมลงไปในฟังก์ชัน get_second_min เท่านั้นโดยห้ามแก้ไขส่วนอื่น ๆ นอกจากนี้ ในฟังก์ชัน get_second_min นั้น ห้ามเรียกฟังก์ชันใด ๆ ที่มีการอ่านเขียนข้อมูลจากคีย์บอร์ดหรือจอภาพโดยเด็ดขาด (เช่น ห้ามเรียกใช้ cin, cout, scanf, printf, ฯลฯ) และห้ามสร้างตัวแปรแบบ static (ถ้าไม่รู้จักว่า static คืออะไร ก็ไม่ต้องกังวล) grader จะไม่ทำการตรวจสอบในเรื่องนี้ระหว่างการสอบ แต่จะมีการตรวจสอบอีกทีในภายหลัง หากเรียกใช้จะได้ 0 คะแนนทันที

อธิบายฟังก์ชัน main

สำหรับแต่ละ test case นั้น ฟังก์ชัน main จะทำงานโดยรับข้อมูลสองบรรทัดดังรูปแบบต่อไปนี้

บรรทัดแรกรับค่า n ซึ่งบอกจำนวนข้อมูล

บรรทัดสองรับตัวเลข n ตัว กำหนดให้ข้อมูลนี้คือ v[0] ถึง v[n-1]

หลังจากนั้น main จะทำการเรียก get_second_min จำนวน n-1 รอบ โดยที่ในรอบที่ i (เมื่อ i มีค่าตั้งแต่ 1 ถึง n-1) จะทำการเรียก get_second_min(s) โดยให้ s มีข้อมูลประกอบด้วย v[0] ถึง v[i] (โดยที่จะเรียกก็ต่อเมื่อข้อมูลดังกล่าวมีตัวเลขอย่างน้อย 2 ตัวที่แตกต่างกัน)

หลังจากการเรียก get_second_min ในแต่ละรอบจะมีการพิมพ์ค่าผลลัพธ์ของ get_second_min ขนาดของ s และค่าใน s ออกมา (ค่าที่พิมพ์ออกมาจากซ้ายไปขวาจะเรียงจาก top of stack ไปจนถึงตัวสุดท้าย)

การให้คะแนน

ในโจทย์ข้อนี้จะมีการตรวจผลการทำงานอยู่ 2 อย่าง คือ 1) ฟังก์ชันนี้คืนค่าได้ถูกต้องหรือไม่ และ 2) ค่าของ stack<int> s หลังจากจบการทำงานมีข้อมูลเหมือนกับก่อนเริ่มทำงานหรือไม่ คะแนนในแต่ละ test case จะขึ้นอยู่กับผลของการตรวจทั้งสองอย่างดังนี้

การตรวจ 1) ได้ผลเป็นไม่ถูกต้อง ได้ 0 คะแนน

การตรวจ 1) ได้ผลถูกต้อง แต่ 2) ไม่ถูกต้อง ได้ 6 คะแนน (ผลการตรวจจะเป็น “s”)

การตรวจ 1) และ 2) ได้ผลถูกต้องทั้งคู่ ได้ 10 คะแนน (ถือว่าเต็มใน test case นั้น)

โดยเราจะถือว่า 1) หรือ 2) จะถูกต้องก็ต่อเมื่อการเรียก get_second_min ทุกครั้งของ test case ดังกล่าวทำงานถูกต้อง

ชุดข้อมูลทดสอบ

- 20% ของข้อมูลทดสอบจะมี n <= 3
- 30% ของข้อมูลทดสอบจะมี n <= 1,000 และไม่มีตัวเลขซ้ำกันเลย
- 50% ของข้อมูลทดสอบจะมี n <= 3,500

คำแนะนำ

ถ้าหากติดปัญหาในข้อนี้ ให้พยายามทำข้อนี้โดยสนใจเฉพาะการตรวจ 1) ก่อน แล้วค่อยพยายามทำการตรวจ 2) ในภายหลัง

ตัวอย่าง

ข้อมูลนำเข้า	ข้อมูลส่งออก
5 4 5 1 3 2	--use v[1] to v[0] -- result is 5 size of s is 2 member of s are 4 5 --use v[2] to v[0] -- result is 4 size of s is 3 member of s are 4 5 1 --use v[3] to v[0] -- result is 3 size of s is 4 member of s are 4 5 1 3 --use v[4] to v[0] -- result is 2 size of s is 5 member of s are 4 5 1 3 2

โค้ดตั้งต้น (อยู่ในหน้าถัดไป)

```
#include <algorithm>
#include <iostream>
#include <stack>
#include <queue>
#include <vector>
#include <set>
#include <map>
#include <list>

using namespace std;

int get_second_min(stack<int> &s) {
    //write only in this function, do not declare static
}

int main()
{
    ios_base::sync_with_stdio(false);cin.tie(0);
    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0;i < n;i++) {
        cin >> v[i];
    }

    //repeat n-1 times
    for (int last = 1;last < n;last+=1) {
        stack<int> s;
        //build s;
        bool distinct = false;
        for (int i = last;i >= 0;i--) {
            s.push(v[i]);
            if (v[i] != v[0]) distinct = true;
        }

        cout << "--use v[" << last << "] to v[0] --" << "\n";
        if (distinct) {
            //call get_second_min if we have at least 2 distinct value
            int answer = get_second_min(s);

            //print result and s
            cout << "result is " << answer << "\n";
            cout << "size of s is " << s.size() << "\n" << "member of s are ";
            while(s.size() > 0) {
               cout << s.top() << " ";
               s.pop();
            }
            cout << "\n";
        } else {
            cout << "skip because s has only one value\n\n\n";
        }
    }
    return 0;
}
```