เฉลย

Thailand Online Competitive Programming Contest 2021

โจทย์ทดสอบระบบ

วันที่ 10 พฤศจิกายน 2021

ID โจทย์	ชื่อโจทย์	Time	Memory	คะแนนชุดทดสอบย่อย	รวม (คะแนน)
plus	บวกเลข	1 s	256 MB	30 70	100
sort	เรียงลำดับเลข	1 s	256 MB	25 75	100
sssp	วิถีที่สั้นที่สุดจากแหล่งต้นทางเดียว	1 s	256 MB	15 35 50	100

บวกเลข

1 second, 256 megabytes

Subtask 1 ($-10^9 \leq a, b \leq 10^9$)

รับข้อมูลนำเข้ามาในรูปของตัวแปร Integer เนื่องจากข้อมูลนำเข้ามีค่าอยู่ระหว่าง -2^{30} ถึง $2^{30}-1$ ทำให้ผลบวกของข้อมูลนำเข้ามีค่าอยู่ในช่วง -2^{31} ถึง $2^{31}-1$ ซึ่งไม่เกิน limit ของตัวแปร Integer จึงสามารถใช้ตัวแปร Integer เก็บทั้งข้อมูลนำเข้าและข้อมูลส่งออกได้

Time Complexity: $O(1)$

Subtask 2 ($-10^{18} \leq a, b \leq 10^{18}$)

เนื่องจากข้อมูลนำเข้ามีโอกาสอยู่นอกช่วง limit ของ Integer ดังนั้นผลบวกของข้อมูลนำเข้าก็มีโอกาสอยู่นอกช่วง limit ของ Integer เช่นกัน ทำให้ต้องใช้ตัวแปร Long (`long long` ในภาษา C) ในการเก็บข้อมูลแทน เพราะมีช่วง limit เป็น -2^{63} ถึง $2^{63}-1$

Time Complexity: $O(1)$

Solution Code:

```cpp
#include <bits/stdc++.h>

#define long long long

using namespace std;

int main() {
    long a, b;
    scanf("%lld %lld", &a, &b);
    printf("%lld\n", a + b);

    return 0;
}
```

เรียงลำดับตัวเลข

1 second, 256 megabytes

Subtask 1 ($N \leq 3 \cdot 10^3$)

สามารถใช้อัลกอริทึมใดก็ได้ในการเรียงค่าภายในอาร์เรย์ที่ใช้เวลาการทำงานไม่ช้าไปกว่า $O(N^2)$ ยกตัวอย่างเช่น Bubble Sort Algorithm เป็นต้น

Time Complexity: $O(N^2)$

Subtask 2 ($N \leq 10^5$)

จำเป็นต้องใช้อัลกอริทึมในการเรียงค่าภายในอาร์เรย์ที่ใช้เวลาการทำงานไม่ช้าไปกว่า $O(N \log N)$ ยกตัวอย่างเช่น Merge Sort Algorithm เป็นต้น

Time Complexity: $O(N \log N)$

Solution Code:

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    int n;
    vector<int> vec;
    scanf("%d", &n);
    for(int i = 1, a; i <= n; i++) {
        scanf("%d", &a);
        vec.emplace_back(a);
    }
    sort(vec.begin(), vec.end());
    for(int x : vec) printf("%d ", x);
    printf("\n");

    return 0;
}
```

วิถีที่สั้นที่สุดจากแหล่งต้นทางเดียว

1 second, 256 megabytes

สำหรับข้อนี้ เนื่องจากระยะทางของแต่ละเส้นทางมีค่ามากถึง 10^9 ทำให้คำตอบของข้อนี้มีโอกาสเกิน limit ของตัวแปรประเภท Integer (เช่น กรณีที่เส้นทางที่สั้นที่สุดจำเป็นต้องผ่านทุกเส้นทางในเมือง) ทำให้ต้องใช้ตัวแปรประเภท Long แทน

Subtask 1 ($N \leq 200$)

สามารถใช้อัลกอริทึมใดก็ได้ในการหา Shortest Path บน Connected Undirected Simple Graph ที่ใช้เวลาการทำงานไม่ช้าไปกว่า $O(N^3)$ ยกตัวอย่างเช่น Floyd-Warshall Algorithm เป็นต้น

Time Complexity: $O(N^3)$

Subtask 2 ($M = N - 1$)

สำหรับ Subtask นี้ สังเกตว่ากราฟจะมีสมบัติเป็น Tree เสมอ ทำให้สำหรับคู่โหนด u, v ใด ๆ จะมีเพียงเส้นทางเดียวเท่านั้นที่เชื่อมระหว่าง 2 โหนดนี้ ทำให้เส้นทางนั้นเป็น Shortest Path จาก u ไป v และ v ไป u เสมอ จึงสามารถใช้ Graph Transversal แบบใดก็ได้ (Depth First Search หรือ Breadth First Search) ในการหาระยะทางจากโหนด 1 ไปยังโหนดอื่น ๆ

Time Complexity: $O(N)$

Subtask 3 ($N \leq M \leq 10^5$)

สำหรับ Subtask นี้ จำเป็นต้องใช้อัลกอริทึม Single Source Shortest Path ที่ใช้เวลาไม่ช้าไปกว่า $O(M + N \log N)$ ยกตัวอย่างเช่น Dijkstra's Algorithm เป็นต้น

Time Complexity: $O(M + N \log N)$

Solution Code:

```cpp
#include <bits/stdc++.h>

#define long long long
#define pii pair<long, long>
#define x first
#define y second

using namespace std;

const int N = 1e5 + 5;

int n, m;
long dp[N];
vector<pii> g[N];
priority_queue<pii, vector<pii>, greater<pii> > Q;

int main() {
    fill_n(dp, N, 1e18);

    scanf("%d %d", &n, &m);
    for(int i = 1, a, b, c; i <= m; i++) {
        scanf("%d %d %d", &a, &b, &c);
        g[a].emplace_back(b, c);
        g[b].emplace_back(a, c);
    }
    Q.emplace(dp[1] = 0, 1);
    while(!Q.empty()) {
        pii u = Q.top(); Q.pop();
        if(dp[u.y] != u.x) continue;
        for(pii v : g[u.y])
            if(u.x + v.y < dp[v.x])
                Q.emplace(dp[v.x] = u.x + v.y, v.x);
    }
    for(int i = 1; i <= n; i++) printf("%lld ", dp[i]);
    printf("\n");

    return 0;
}
```