Editorial

เฉลยและอธิบาย โจทย์วันทดสอบระบบของ Crack 'n' Code Pre-TOI18 Contest

จอมอยากบวก (plus)

Solution #1 — 5925 คะแนน

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    int Q;
    cin >> Q;

    for(int i = 0; i < Q; i++) {
        int a, b;
        cin >> a >> b;
        cout << a + b << "\n";
    }
}
```

Time Complexity: $\mathcal{O}(Q)$

Solution #2 — 21773 คะแนน (คะแนนเต็ม)

สาเหตุที่ Solution #1 ไม่ได้คะแนนเต็มเนื่องจากใช้ชนิดข้อมูลเป็น `int` ซึ่งมีช่วง $[-2^{31},\ 2^{31}-1]$ แต่ขอบเขตของค่า $a+b$ ที่เป็นไปได้อยู่ในช่วง $[-2\cdot 10^{18},\ 2\cdot 10^{18}]$ ดังนั้นจึงต้องใช้ชนิดข้อมูลเป็น `long long` หรือ `int64_t`

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    int Q;
    cin >> Q;

    for(int i = 0; i < Q; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a + b << "\n";
    }
}
```

Time Complexity: $\mathcal{O}(Q)$

ข้อผิดพลาดที่พบบ่อย หลายคนที่ติด Time Limit มักจะเกิดจากการใช้ `cin cout` ซึ่งฟังก์ชันเหล่านี้จะช้ากว่า `printf scanf` ทำให้ในข้อที่มี Input Output มาก ๆ ก็เกิดปัญหาได้ เราสามารถแก้ปัญหาความช้าได้โดยการเพิ่มสองคำสั่งนี้เข้าไปในบรรทัดแรกของ `main`

```cpp
ios_base::sync_with_stdio(false);
cin.tie(nullptr);
```

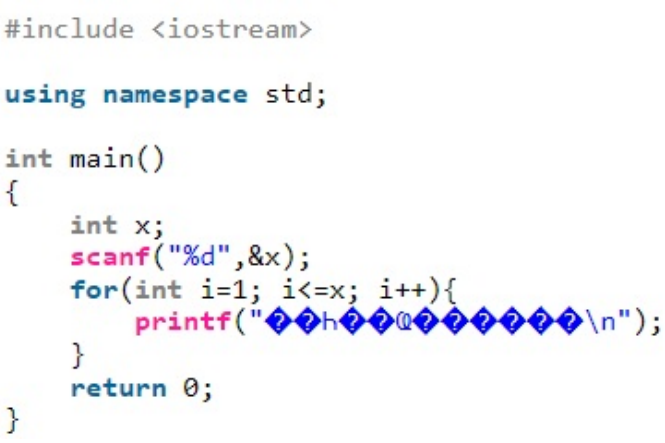
จอมน้อยส์ (jomnoiz)

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int JOMNOIZ;
    cin >> JOMNOIZ;

    for(int i = 0; i < JOMNOIZ; i++) {
        cout << "กราบท่านจอมน้อยส์\n";
    }
}
```

Time Complexity: $\mathcal{O}(JOMNOIZ)$

หมายเหตุ

- สำหรับคนที่ลองรันโค้ดนี้บนเครื่องตัวเองที่เป็น Windows อาจจะเห็นตัวประหลาดพิมพ์ออกมา เนื่องจาก Windows นั้นใช้ UTF-16 เป็นวิธี encoding พื้นฐาน แต่หากใครใช้ระบบปฏิบัติการที่เป็น UTF-8 ได้แก่ ตระกูล Linux ทั้งหลาย ก็จะเห็นคำว่า "กราบท่านจอมน้อยส์" ออกมาตามปกติ
- สำหรับคนที่ใช้โปรแกรมเช่น Dev-C++ หรือ CodeBlocks บน **Windows** อาจจะมีปัญหาได้ เนื่องจากโปรแกรมพวกนี้**ไม่ได้เซฟไฟล์อย่างถูกต้อง**
- ขอเน้นย้ำอีกครั้งว่าวันจริงไม่มีข้อไหนให้พิมพ์ภาษาไทย ดังนั้นขอให้ผู้เข้าแข่งขันวางใจได้



สิ่งที่แอดมินเห็นจากเกรดเดอร์

เต็ม 100 (tem100)

วิธีทำของข้อนี้ได้ถูกอธิบายไว้ในโจทย์แล้ว

Time Complexity: $\mathcal{O}(1)$

ยกเลิก 1112 (cancel_1112)

ข้อนี้หากลองอ่านดูก็จะพบว่าเป็น All-Pairs Shortest Path ซึ่งวิธีที่สามารถแก้ข้อนี้ได้ ยกตัวอย่างเช่น

- Floyd-Warshall Algorithm (แนะนำคลิป คุณภาพที่คนแต่งดูก่อนแข่ง TOI17 => https://www.youtube.com/watch?v=oNI0rf2P9gE)
- Dijkstra Algorithm N รอบ

Official Solution (Floyd-Warshall Algorithm)

```
#include <bits/stdc++.h>
using namespace std;

int N, M;
int64_t pop[505];
int64_t floyd[505][505];
const int64_t floyd_INF = 1e15;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> N >> M;

    for (int i = 1; i <= N; i++) {
        cin >> pop[i];
    }

    // * Init
    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= N; j++) {
            if (i == j) {
                floyd[i][j] = 0;
            } else {
                floyd[i][j] = floyd_INF;
            }
        }
    }

    for (int i = 1; i <= M; i++) {
        int u, v;
        int64_t l;
        cin >> u >> v >> l;
        assert(floyd[u][v] == floyd_INF);
```

```
        assert(floyd[v][u] == floyd_INF);
        floyd[u][v] = l;
        floyd[v][u] = l;
    }

    // * Floyd-Warshall
    for (int m = 1; m <= N; m++) {
        for (int i = 1; i <= N; i++) {
            for (int j = 1; j <= N; j++) {
                if (i == j || i == m)
                    continue;

                floyd[i][j] = min(floyd[i][j], floyd[i][m] + floyd[m]
[j]);
            }
        }
    }

    // * Find Best Location
    int bestloc = 0;
    int64_t bestcost = floyd_INF;
    for (int i = 1; i <= N; i++) {
        int64_t totalcost = 0;
        for (int j = 1; j <= N; j++) {
            totalcost += floyd[i][j] * pop[j];
        }

        if (totalcost < bestcost) {
            bestcost = totalcost;
            bestloc = i;
        }
    }

    cout << bestloc << " " << bestcost << "\n";
}
```

Time Complexity: $\mathcal{O}(N^3)$
Space Complexity: $\mathcal{O}(N^2)$

ข้อผิดพลาดที่พบบ่อย ใช้ Data Type เป็น `int32` (`int`) ซึ่งค่า p_i, l_i ในข้อนี้ค่อนข้างสูง จะทำให้หลุดขอบเขตของ `int32` ได้ วิธีแก้คือใช้ `int64` (`long long`)

การล้างแค้นของโมกข์ (mok_revenge)

เก็บ Subtask #4 — $P = 1$

เนื่องจากเราสามารถตัดต้นสนที่ตำแหน่งเดียวเท่านั้น จำนวนครั้งที่เราจะตัดก็คือจำนวนสมาชิกของ $\{x_i\}$ หรือก็คือจำนวนตำแหน่งต้นสนที่แตกต่างกัน

Time Complexity: $\mathcal{O}(N)$

เก็บ Subtask #5 — $N = 2$

เนื่องจาก $N = 2$ เราเพียงแค่เช็คว่า $x_1 - x_0 > P$ หรือใหม่ หากใช่ตอบ 2 ไม่ใช่ให้ตอบ 1

Time Complexity: $\mathcal{O}(N)$ หรือ $\mathcal{O}(1)$

Solution #1 ~~(Official Solution)~~ — 72 คะแนน

เราสามารถใช้ Sliding Window เพื่อหาวิธีการใช้กระบวนท่าที่ดีที่สุดได้

```cpp
#include <bits/stdc++.h>
using namespace std;

long long N, A, P;
vector<long long> pos;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> N >> P;

    pos.reserve(N);

    for (long long i = 0; i < N; i++) {
        long long x;
        cin >> x;
        if (pos[pos.size() - 1] != x)
            pos.push_back(x);
    }

    deque<pair<long long, long long>> sw;
    sw.push_back({1, pos[0]});
    long long bestwin = 1;
    for (long long i = 1; i < pos.size(); i++) {
        while (!sw.empty() && sw.front().second <= pos[i] - P)
            sw.pop_front();

        sw.push_back({bestwin + 1, pos[i]});
        bestwin = sw.front().first;
    }

    cout << bestwin << "\n";
}
```

Time Complexity: $\mathcal{O}(N)$
Space Complexity: $\mathcal{O}(N)$

วิธีนี้ได้แค่ 72 คะแนน ถึงแม้ว่าจะเป็น $\mathcal{O}(N)$ แต่เนื่องจากข้อนี้จำกัดหน่วยความจำไว้ได้โหดมาก โดยใช้ได้แค่ **16 MB** ดังนั้นเราต้องหาวิธีที่ใช้หน่วยความจำได้ดีกว่านี้

Solution #2 — 100 คะแนน

หากเรามองดี ๆ แล้ว ความจริงแล้วเราสามารถ "Greedy" ได้

พิจารณาต้นไม้ต้นแรก x_0 (recap: $x_i \leq x_{i+1}$ เสมอ) จะต้องถูกตัดแน่นอน การตัดที่ดีที่สุดคือการตัดให้ต้นไม้อื่นมีโอกาสโดนด้วย นั่นคือเลือก $x = x_0$ (เนื่องจากไม่มีต้นอื่นมี $x_i < x_0$ เลย) หลังจากตัดออกไปแล้ว เราก็จะได้ป่าสนป่าใหม่ เราสามารถเลือกตำแหน่งตัดต้นสนด้วยวิธีนี้ซ้ำไปเรื่อย ๆ ได้

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cin.tie(nullptr)->sync_with_stdio(false);

    long long N, P;
    cin >> N >> P;

    long long latestPos = INT64_MIN, ans = 0;
    for (long long i = 1; i <= N; i++) {
        long long x;
        cin >> x;

        if (x > latestPos) {
            ans++;
            latestPos = x + P - 1;
        }
    }
    cout << ans << "\n";
}
```

Time Complexity: $\mathcal{O}(N)$
Space Complexity: $\mathcal{O}(1)$

Assert Dominance (assert_dominance)

เก็บ Subtask #1 — $S[i][j] = \texttt{'X'}$

เนื่องจากทุกตำแหน่งเราสามารถวางอะไรก็ได้ ดังนั้นจึงไม่มีข้อจำกัดที่จะมาขัดไม่ให้เราวางตัว `T`

ทั้งนี้ขนาดของ `T` จะต้องมีค่าตั้งแต่ 3 ขึ้นไป และเป็นเลขคี่

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    int R, C;
    cin >> R >> C;

    int t = min(R, C);

    if (t % 2 == 0)
        t--;

    if (t < 3)
        t = 0;

    cout << t << "\n";
}
```

Time Complexity: $\mathcal{O}(1)$

Solution #1 — 15 คะแนน

ใช้ Brute Force เพื่อหาทุกรูปแบบที่เป็นไปได้

เพื่อความง่ายในการคำนวณเวลา ให้ $R \approx C = N$
Time Complexity: $\mathcal{O}(N^4)$

Solution #2 — 37 คะแนน

ใช้ Quick Sum แล้วหาตั้งแต่ขนาด N ลงมา ช่วยลดเวลาให้เหลือ $\mathcal{O}(N^3)$
(แนะนำให้ไปลองทำข้อ `toi9_fence` มาก่อน)

Solution #3 — 61 คะแนน

สังเกตว่าข้อนี้ต่างจากข้อ `toi9_fence` ตรงที่ หากมี `T` ขนาด N แล้วจะต้องมี `T` ขนาด $N-2$ แน่นอน ($N \geq 5$) ดังนั้นเราสามารถใช้ Binary Search เพื่อลดเวลาให้เหลือ $\mathcal{O}(N^2 logN)$

Solution #4 — 100 คะแนน

สำหรับแต่ละตำแหน่ง i, j ใช้วิธีการหาว่าจากตำแหน่งนี้สามารถยืดไปทางซ้าย ขวา และด้านล่างได้ไกลแค่ไหน เพื่อหาขนาด `T` ที่ใหญ่ที่สุดที่มีจุดศูนย์กลาง ณ จุด i, j การหาขนาดนี้สามารถใช้การ Preprocess ได้ ทำให้เวลารวมเป็น $\mathcal{O}(N^2)$

ได้รับแรงบันดาลใจมาจาก

- ข้อ `toi9_fence`