เฉลยโจทย์การแข่งขัน

Gean Dev

1 A - Gean

ทำได้โดยการเก็บรูปตัว G ไว้ และทำการแสดงอักขระละ $N \times N$ รอบ

```cpp
#include <iostream>
#include <string>

using namespace std;

string s[5] = {
     ".###.",
     "#....",
     "#..##",
     "#...#",
     ".###."
};

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     int n;
     cin >> n;
     for(int i = 0; i < 5; i++) {
          for(int row = 0; row < n; row++) {
               for(int j = 0; j < 5; j++) {
                    for(int col = 0; col < n; col++) {
                         cout << s[i][j];
                    }
               }
               cout << "\n";
          }
     }
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(N^2)$

2 B - Inversion Count

2.1 ปัญหาย่อยที่ 1 (30 คะแนน)

ในปัญหาย่อยนี้จะมี $N \leq 3\,000$

ทำได้โดยการลองจับคู่ทุกคู่ i, j ที่เป็นไปได้ และนับจำนวนคู่ที่สอดคล้องกับเงื่อนไขของโจทย์

```cpp
#include <iostream>

using namespace std;

const int MAX_N = 100005;

int n;
int a[MAX_N];
int ans = 0;

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n;
     for(int i = 0; i < n; i++) {
          cin >> a[i];
     }
     for(int i = 0; i < n; i++) {
          for(int j = i + 1; j < n; j++) {
               if(a[i] > a[j]) {
                    ans++;
               }
          }
     }
     cout << ans;
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(N^2)$ ซึ่งเร็วพอที่จะผ่านในปัญหาย่อยนี้

2.2 ปัญหาย่อยที่ 2 (30 คะแนน)

ในปัญหาย่อยนี้จะมี $A_i \leq 2$

สังเกตว่าในปัญหาย่อยนี้ จะเกิด inversion ก็ต่อเมื่อ $i < j, A[i] = 2$ และ $A[j] = 1$ ดังนั้นเรา

สามารถทำได้โดยการไล่ทุกค่า j ทั้งหมดที่มี $A[j] = 1$ และนับจำนวน i ที่ $i < j$ และ $A[i] = 2$ ซึ่งจะใช้ตัวแปร cnt ในการนับจำนวน i ดังกล่าว

เนื่องจากคำตอบอาจมีค่าเยอะมากจึงจะต้องใช้ตัวแปรประเภท 64-bit (long long ในภาษา C และ C++) ในการเก็บคำตอบ

```cpp
#include <iostream>

using namespace std;

const int MAX_N = 100005;

int n;
int a[MAX_N];
int cnt = 0;
long long ans = 0;

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n;
     for(int i = 0; i < n; i++) {
          cin >> a[i];
     }
     for(int j = 0; j < n; j++) {
          if(a[j] == 1) {
               ans += cnt;
          } else {
               cnt++;
          }
     }
     cout << ans;
}
```

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

2.3 ปัญหาย่อยที่ 3 (40 คะแนน)

ทำได้ด้วยการนำแนวคิดในปัญหาย่อยที่ 2 มาปรับแต่งเล็กน้อย โดยสำหรับแต่ละค่า j จะนับจำนวนตำแน่ง i ทั้งหมดที่ $i < j$ และ $A[i] > A[j]$ แทน

```cpp
#include <iostream>

```

```cpp
using namespace std;

const int MAX_N = 100005;

int n;
int a[MAX_N];
int cnt[11] = {};
long long ans = 0;

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n;
     for(int i = 0; i < n; i++) {
          cin >> a[i];
     }
     for(int j = 0; j < n; j++) {
          for(int val = a[j] + 1; val <= 10; val++) {
               ans += cnt[val];
          }
          cnt[a[j]]++;
     }
     cout << ans;
}
```

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

3 C - Max In Stack

3.1 ปัญหาย่อยที่ 1 (14 คะแนน)

ในปัญหาย่อยนี้จะมี $Q \leq 1000$

ทำได้โดยการทำตามที่โจทย์กำหนดโดยตรง

```c
#include <stdio.h>
#include <string.h>

int s[100001], sz, q;
char op[5];

int main(int argc, char *argv[]) {
     scanf("%d", &q);
     while (q--) {
          scanf("%s", op);
          if (0 == strcmp(op, "Push")) {
               scanf("%d", &s[sz]);
               ++sz;
          } else {
               --sz;
          }
          int mx = s[0];
          for (int i = 1; i < sz; ++i) {
               if (s[i] > mx) {
                    mx = s[i];
               }
          }
          printf("%d\n", mx);
     }
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(Q^2)$

3.2 ปัญหาย่อยที่ 2 (86 คะแนน)

ในปัญหาย่อยนี้จะมี $Q \leq 100000$

สังเกตว่าทุกครั้งที่ทำคำสั่ง Push หาก x น้อยกว่าค่ามากสุดในกองซ้อนก่อนทำ แล้วสามารถเปลี่ยน

x เป็นค่ามากสุดดังกล่าว และคำตอบที่ได้จะยังเหมือนเดิม

นอกจากนี้ในทุกขณะ ค่าบนสุดบนกองซ้อนจะเป็นค่ามากสุด ดังนั้นสามารถนำค่าบนสุดมาตอบได้เลย

```c
#include <stdio.h>
#include <string.h>

int s[100001], sz, q;
char op[5];

int main(int argc, char *argv[]) {
     scanf("%d", &q);
     while (q--) {
          scanf("%s", op);
          if (0 == strcmp(op, "Push")) {
               scanf("%d", &s[sz]);
               if (s[sz - 1] > s[sz]) {
                    s[sz] = s[sz - 1];
               }
               ++sz;
          } else {
               --sz;
          }
          printf("%d\n", s[sz - 1]);
     }
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(\mathrm{Q})$

4 D - Count In Subarray

4.1 ปัญหาย่อยที่ 1 (30 คะแนน)

ในปัญหาย่อยนี้จะมี $N, Q \leq 3\,000$

ทำได้โดยการไล่นับตัวเลขในช่วง l, r ได้โดยตรง

```cpp
#include <iostream>

using namespace std;

const int MAX_N = 100005;

int n, q;
int a[MAX_N];

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n >> q;
     for(int i = 1; i <= n; i++) {
          cin >> a[i];
     }
     while(q--) {
          int l, r, x;
          cin >> x >> l >> r;
          int ans = 0;
          for(int i = l ; i <= r; i++) {
               if(a[i] == x) {
                    ans++;
               }
          }
          cout << ans << "\n";
     }
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(N \cdot Q)$

4.2 ปัญหาย่อยที่ 2 (30 คะแนน)

ในปัญหาย่อยนี้จะมี $A_i \leq 10$

จะสร้างอาร์เรย์ B ขนาด $10 \times N$ โดย $B[v][i] = 1$ เมื่อ $A[i] = v$ และ $B[v][i] = 0$ เมื่อ $A[i] \neq v$

จากนับ สำหรับแต่ละค่า v ตั้งแต่ 1 ถึง 10 จะสร้าง Prefix Sum ของแต่ละอาร์เรย์ $B[v]$ เพื่อทำให้สามารถหาคำตอบของแต่ละคำถามได้อย่างรวดเร็ว

สามารถอ่านเพิ่มเติมเกี่ยวกับ Prefix Sum ได้ที่ https://usaco.guide/silver/prefix-sums

```cpp
#include <iostream>

using namespace std;

const int MAX_N = 100005;

int n, q;
int a[MAX_N];
int b[11][MAX_N];
int prefix[11][MAX_N];

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n >> q;
     for(int i = 1; i <= n; i++) {
          cin >> a[i];
     }
     for(int v = 1; v <= 10; v++) {
          for(int i = 1; i <= n; i++) {
               if(a[i] == v) {
                    b[v][i] = 1;
               }
          }
          for(int i = 1; i <= n; i++) {
               prefix[v][i] = prefix[v][i - 1] + b[v][i];
          }
     }
     while(q--) {
          int l, r, x;
          cin >> x >> l >> r;
          int ans = (x <= 10 ? prefix[x][r] - prefix[x][l - 1] : 0);
          cout << ans << "\n";
     }
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(N + Q)$

4.3 ปัญหาย่อยที่ 3 (40 คะแนน)

หากเรานำโปรแกรมในปัญหาย่อยที่ 2 มาใช้ในปัญหาย่อยนี้ เนื่องจาก $A[i] \leq N$ จะทำให้โปรแกรมดังกล่าวมี Time Complexity $\mathcal{O}(N^2)$ ซึ่งจะช้าไปสำหรับปัญหาย่อยนี้

สังเกตว่าใน อาร์เรย์ B นั้นจะมีค่า 0 อยู่เป็นจำนวนมาก แต่จะมีค่า 1 อยู่เพียงแต่ N ค่า ดังนั้นเราสามารถใช้ Dynamic Array (vector ในภาษา C++) ในการเก็บตำแน่งของแต่ละเลยได้

เมื่อเรามีตำแหน่งของแต่ละเลขแล้ว ในการตอบคำถาม เราจะต้องหาตำแหน่งของเลข x ที่น้อยที่สุดที่มากกว่าหรือเท่ากับ l (แทนตำแหน่งนี้ด้วย l_index) และมากที่สุดที่มากกว่า r (แทนตำแหน่งนี้ด้วย r_index) คำตอบของคำถามนี้จะมีค่าเท่ากับ $r\_index - l\_index$

ในการจะหาค่าของ l_index และ r_index ได้อย่างมีประสิทธิภาพ สามารถทำได้โดยการ Binary Search หรือ ใช้ฟังก์ชั่น lower_bound และ upper_bound ในภาษา C++ ซึ่งจะทำงานใน $\mathcal{O}(\log(N))$

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

const int MAX_N = 100005;

int n, q;
int a[MAX_N];
vector<int> position[MAX_N];

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n >> q;
     for(int i = 1; i <= n; i++) {
          cin >> a[i];
     }
     for(int i = 1; i <= n; i++) {
          position[a[i]].push_back(i);
     }
     while(q--) {
          int l, r, x;
          cin >> x >> l >> r;
          int l_index = lower_bound(position[x].begin(),
```

```
    position[x].end(), l) - position[x].begin();
26          int r_index = upper_bound(position[x].begin(),
    position[x].end(), r) - position[x].begin();
27          cout << r_index - l_index << "\n";
28      }
29  }
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(\mathrm{N} + \mathrm{Q}\log(\mathrm{N}))$

5 E - Exceed K

5.1 ปัญหาย่อยที่ 1 (50 คะแนน)

ในปัญหาย่อยนี้จะมี $N \leq 5\,000$

วิธีการทำในปัญหาย่อยนี้สามารถทำได้โดยการกำหนดจุดซ้ายสุด l และ ขวาสุด r ของช่วงที่พิจารณา (อาจใช้ Loop มาช่วย) แล้วทำการนับจำนวนของจำนวนเต็มแต่ละแบบ เมื่อจำนวนเต็มใดมากกว่า K แล้ว แสดงว่านี่คือช่วงที่ต้องการนับจำนวน

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

using namespace std;
using ll = long long;

const int MxN = 100010;

int a[MxN], cnt[2];

signed main(int argc, char *argv[]) {
  cin.tie(nullptr)->ios::sync_with_stdio(false);
  int n, k;
  cin >> n >> k;
  ll answer = 0;
  for(int i=1; i<=n; ++i) {
    cin >> a[i];
  }
  for(int l=1; l<=n; ++l) {
    cnt[0] = cnt[1] = 0;
    for(int r=l; r<=n; ++r) {
      cnt[a[r]] += 1;
      if(cnt[0] > k || cnt[1] > k) {
        answer++;
      }
    }
  }
  cout << answer << "\n";
  return 0;
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(N^2)$

5.2 ปัญหาย่อยที่ 2 (50 คะแนน)

สังเกตว่า หากช่วง $[l, r]$ มีจำนวนใดจำนวนหนึ่งมากกว่า K จำนวน แล้วจะทำให้ช่วง $[l - 1, r]$ มีจำนวนใดจำนวนหนึ่งมากกว่า K จำนวนด้วย ดังนั้น หากกำหนดขอบด้านขวาของช่วงเป็น r แล้วให้ l เป็นตำแหน่งขวาสุดที่ทำให้ช่วง $[l, r]$ มีจำนวนใดจำนวนหนึ่งมากกว่า K จำนวน จะได้ว่าจะจำนวนช่วงทั้งหมดที่มีจำนวนใดจำนวนหนึ่งมากกว่า K จำนวน เมื่อกำหนดขอบด้านขวาเป็น r จะมีอยู่ l ช่วง

ในการหา l สามารถทำได้โดยใช้ queue 2 อันในการเก็บตำแหน่งของเลข 0 และ 1 ได้ดังโปรแกรมนี้

```cpp
#include <iostream>
#include <queue>

using namespace std;

const int MAX_N = 100005;

int n, k;
int a[MAX_N];
queue<int> q[2];
long long ans = 0;

int main() {
     cin.tie(nullptr)->sync_with_stdio(false);
     cin >> n >> k;
     for(int i = 1; i <= n; i++) {
          cin >> a[i];
     }
     for(int r = 1; r <= n; r++) {
          q[a[r]].push(r);
          int l = 0;
          for(int v = 0; v < 2; v++) {
               if((int) q[v].size() > k + 1) {
                    q[v].pop();
               }
               if((int) q[v].size() > k) {
                    l = max(l, q[v].front());
               }
          }
          ans += l;
     }
     cout << ans;
}
```

โปรแกรมดังกล่าวจะมี Time Complexity $\mathcal{O}(\mathrm{N})$