Εισαγωγή

Ζητούμενο: Το πρόβλημα μας ζητάει να υλοποιήσουμε τον αλγόριθμο Misra-Gries, ο οποίος βρίσκει γρήγορα τους $(p-1)$ heavy hitters με την προϋπόθεση ότι εμφανίζονται τουλάχιστον σε $1/p$ θέσεις. Ο αλγόριθμος λειτουργεί διατηρώντας $(p-1)$ ζεύγη μεταβλητών. Για κάθε ζεύγος, η πρώτη μεταβλητή αντιστοιχεί σε ένα στοιχείο που είναι πιθανός heavy hitter, ενώ η δεύτερη αντιστοιχεί σε έναν μετρητή **ο οποίος δεν αντιστοιχεί στην πραγματική συχνότητα του heavy hitter**. Κάθε φορά που συναντάμε ένα νέο στοιχείο, εάν υπάρχει στο σύνολο των μεταβλητών μας, αυξάνουμε τον αντίστοιχο μετρητή. Εάν δεν υπάρχει, και το σύνολο των μεταβλητών είναι μικρότερο από $(p-1)$, τότε το εισάγουμε σε νέο ζεύγος, και θέτουμε τον αντίστοιχο μετρητή σε 1. Εάν το σύνολο των μεταβλητών είναι ίσο με $(p-1)$, τότε μειώνουμε όλους τους μετρητές και αφαιρούμε όσα ζεύγη έχουν μετρητές ίσους με 0. Η απλή υλοποίηση αυτού του αλγορίθμου απαιτεί χρόνο $O(n \cdot p)$, αλλά με τη χρήση δομών δεδομένων μπορούμε να τον υλοποιήσουμε σε χρόνο $O(n \log p)$.

Προαπαιτούμενα: Για το πρόβλημα αυτό χρειάζεται βασική γνώση δομών δεδομένων. Στην λύση που παραθέτουμε χρησιμοποιούμε τη βιβλιοθήκη `STL` της `C++`, αλλά δεν είναι ιδιαίτερα δύσκολο να αντικαταστήσουμε την δομή που χρησιμοποιούμε με κάποια που είναι εύκολο να υλοποιηθεί σε άλλες γλώσσες.

Το πρόβλημα αυτό έλυσαν πλήρως 5 άτομα, αλλά 13 το έλυσαν μερικώς. Οι περισσότεροι από όσους το έλυσαν μερικώς είχαν σωστές απαντήσεις αλλά σε λάθος σειρά, κυρίως διότι δεν υλοποίησαν τον αλγόριθμο Misra-Gries, αλλά χρησιμοποίησαν κάποια άλλη μέθοδο. Η παγίδα του προβλήματος ήταν ότι στο τέλος του αλγορίθμου Misra-Gries οι μετρητές των heavy hitters δεν έχουν καμία αντιστοιχία με τις πραγματικές συχνότητες, οπότε δεν είναι εύκολο να βρούμε την επιθυμητή σειρά χωρίς να υλοποιήσουμε τον αλγόριθμο.

Η υλοποίησή του αλγορίθμου είναι αρκετά απλή, και όλοι οι διαγωνιζόμενοι με σωστές λύσεις ακολούθησαν ακριβώς τον ίδιο τρόπο.

Λύση σε $O(n \log p)$

Όταν το σύνολο των heavy hitters είναι γεμάτο, αντί να μειώσουμε όλους τους μετρητές, μπορούμε να αυξήσουμε έναν μετρητή b. Επομένως, οι μετρητές που διατηρούμε δεν θα είναι ακριβείς, αλλά θα είναι μετατοπισμένοι κατά b. Κάθε φορά που αυξάνουμε το b θα πρέπει να αφαιρέσουμε από το σύνολό μας όλα τα στοιχεία με μετρητές ίσους με b. Για να το κάνουμε αυτό, μπορούμε να έχουμε μία δομή που να διατηρεί τα στοιχεία που αντιστοιχούν σε κάθε τιμή μετρητή.

Ακολουθεί η υλοποίηση σε `C++`. Ο κώδικας είναι του διαγωνιζόμενου Χαρίλαου Πίπη καθώς θεωρήσαμε ότι είχε καθαρή υλοποίηση και σωστή χρήση της γλώσσας.

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

map<int, int> cnt;
set<int> freq[100010];
int lazy;

int main() {
```

```
    int n, p;
    scanf("%d %d", &n, &p);
    for (int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        if (cnt.count(x)) {
            freq[cnt[x]].erase(x);
            cnt[x]++;
            freq[cnt[x]].insert(x);
        }
        else if ((int)cnt.size() == p - 1) {
            lazy++;
            for (auto it : freq[lazy]) {
                cnt.erase(it);
            }
            freq[lazy].clear();
        }
        else {
            cnt[x] = 1 + lazy;
            freq[cnt[x]].insert(x);
        }
    }
    vector<pair<int, int> > vec;
    for (auto it : cnt) {
        vec.push_back({it.second - lazy, it.first});
    }
    sort(vec.begin(), vec.end());
    for (auto it : vec) {
        printf("%d\n", it.second);
    }
    return 0;
}
```

Απόδειξη της ορθότητας του αλγορίθμου Misra-Gries

Το παρακάτω δεν χρειάζεται για διαγωνισμούς, αλλά το παραθέτουμε για όσους ενδιαφέρονται για αποδείξεις.

Μπορούμε να διατυπώσουμε τον αλγόριθμο Misra-Gries λίγο διαφορετικά έτσι ώστε να γίνει ευκολότερη η ανάλυση. Έστω f_i ο αριθμός εμφανίσεων του αριθμού i στα δεδομένα. Ο αλγόριθμος Misra-Gries προσπαθεί να προσεγγίσει αυτόν τον αριθμό με τους μετρητές των μεταβλητών, έστω $\tilde{f}_i$. Αντί να διατηρούμε ένα σύνολο, θέτουμε $\tilde{f}_i = 0$ για κάθε i, και όταν συναντάμε έναν αριθμό j, αυξάνουμε το $\tilde{f}_j$. Έπειτα, εάν υπάρχουν παραπάνω από $(p-1)$ θετικά $\tilde{f}_i$, τότε μειώνουμε όλα τα θετικά $\tilde{f}_i$ κατά 1. Παρατηρούμε ότι αυτός ο αλγόριθμος είναι ισοδύναμος με αυτόν που ορίσαμε παραπάνω.

Θέλουμε να αποδείξουμε ότι $\tilde{f}_i \geq f_i - n/p$, δηλαδή ότι μειώσαμε την μεταβλητή $\tilde{f}_i$ το πολύ n/p φορές. Κάθε φορά που μειώνουμε μεταβλητές, γνωρίζουμε ότι η δομή μας περιέχει p στοιχεία. Συνεπώς κάθε φορά που μειώνουμε μεταβλητές "αφαιρούμε" p στοιχεία. Προφανώς, μπορούμε να αφαιρέσουμε το πολύ n στοιχεία, άρα η διαδικασία αυτή μπορεί να γίνει το πολύ n/p φορές.

Επομένως, εάν για κάποιο στοιχείο j ισχύει ότι $f_j > n/p$, τότε στο τέλος του αλγορίθμου θα ισχύει ότι $\tilde{f}_j > 0$, άρα το j θα βρίσκεται στην τελική δομή.