Εισαγωγή

Ζητούμενο: Το πρόβλημα μας ζητάει να βρούμε τις ελάχιστες κινήσεις που χρειάζονται για να εξισώσουμε κάποιους αριθμούς, εάν σε κάθε κίνηση μπορούμε να θέσουμε δύο αριθμούς ίσους με τον μέσο όρο τους.

Προαπαιτούμενα: Δεν χρειαζόταν κάποια ιδιαίτερη γνώση πέρα από βασικές γνώσεις προγραμματισμού.

Το πρόβλημα αυτό ήταν το πιο εύκολο του διαγωνισμού, αλλά δυσκόλεψε αρκετούς συμμετέχοντες. Υπήρξαν 8 άτομα με 100 πόντους και 8 με μερικές λύσεις. Σχεδόν όλα αυτά τα άτομα ακολούθησαν έναν τρόπο λύσης αρκετά κουραστικό στην υλοποίηση, το να αναλύσουν όλες τις περιπτώσεις. Υπήρξαν και μερικές λύσεις οι οποίες ανέλυσαν μόνο κάποιες περιπτώσεις.

Λύση (σε $O(\min(l, 4) \cdot k^2)$)

Το πρόβλημα αυτό χρειάζεται δύο βασικές παρατηρήσεις.

1. Η μόνη περίπτωση στην οποία η απάντηση είναι πάντα ΝΟ ανεξάρτητα του l, μπορεί να προκύψει όταν έχουμε τρία άτομα.
2. Σε κάθε άλλη περίπτωση χρειάζονται το πολύ 4 ώρες.

Για να αποδείξουμε τον δεύτερο ισχυρισμό, βλέπουμε ότι για ένα άτομο η διαδικασία τελειώνει αμέσως, δύο άτομα χρειάζονται το πολύ 2 ώρες, και για τρία άτομα, στην περίπτωση που είναι εφικτή η διαδικασία χρειάζεται μία ώρα. Στην περίπτωση των τεσσάρων ατόμων, ας θεωρήσουμε ότι τα διαλύματά τους έχουν μάζες a, b, c, d αντίστοιχα. Τότε με δύο κινήσεις παίρνουμε $\frac{a+b}{2}, \frac{a+b}{2}, \frac{c+d}{2}, \frac{c+d}{2}$, και με άλλες δύο έχουμε $\frac{a+b+c+d}{4}, \frac{a+b+c+d}{4}, \frac{a+b+c+d}{4}, \frac{a+b+c+d}{4}$.

Από εδώ και πέρα μπορούμε είτε να αναλύσουμε τις περιπτώσεις περαιτέρω, είτε να γράψουμε μία brute force λύση η οποία δοκιμάζει όλες τις πιθανές κινήσεις (καθώς ξέρουμε ότι είναι το πολύ 4). Αν και σε μέγεθος κώδικα οι λύσεις είναι σχετικά κοντά, η brute force λύση χρειάζεται αρκετά λιγότερη σκέψη, και είναι λιγότερο επικίνδυνη καθώς δεν είναι εύκολο να χάσουμε κάποια περίπτωση. Συνεπώς να έχετε κατά νου εάν μπορείτε να αποφεύγετε την ανάλυση περιπτώσεων, και να φέρνετε το πρόβλημα σε τέτοια μορφή ώστε να μπορεί το πρόγραμμά σας να κάνει την ανάλυση των περιπτώσεων για σας.

Παραθέτουμε τον κώδικα σε `C++`.

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

bool bruteForce(int l, int k, vector<double> arr) {
    sort(arr.begin(), arr.end());
    if(l==0) {
        return arr[0]==arr[k-1];
    }
    vector<double> tmp;
    for(int i=0; i<k; ++i) {
        for(int j=i; j<k; ++j) {
            tmp=arr;
            tmp[i]=tmp[j]=tmp[i]/2 + tmp[j]/2;

            if(bruteForce(l-1, k, tmp)) {
```

```
                return true;
            }
        }
    }
    return false;
}

int main() {
    int t, k, l;
    cin >> t;
    for(int i=0; i<t; ++i) {
        cin >> k >> l;
        vector<double> arr(k);
        l=min(4, l);
        for(int i=0; i<k; ++i) {
            cin >> arr[i];
        }
        if(bruteForce(l, k, arr)) {
            cout << "YES\n";
        }
        else {
            cout << "NO\n";
        }
    }
    return 0;
}
```

Εναλλακτική λύση με ανάλυση περιπτώσεων σε $O(k)$

Για σύγκριση θα παραθέσουμε και τον κώδικα ενός από τους διαγωνιζόμενους που έλυσαν αυτό το πρόβλημα με ανάλυση όλων των περιπτώσεων. Δεν θα αναλύσουμε τις περιπτώσεις, καθώς είναι σχετικά εμφανείς από τον κώδικα. Ο κώδικας είναι του διαγωνιζόμενου Χαρίλαου Πίπη.

```
#include <bits/stdc++.h>
using namespace std;
double ar[4];

int main() {
    int t;
    cin >> t;
    while (t--) {
        int k, l;
        cin >> k >> l;
        for (int i = 0; i < k; i++)
            cin >> ar[i];
        if (k == 1)
            puts("YES");
        else if (k == 2) {
            if (ar[0] != ar[1] && l > 0)
                ar[0] = ar[1];
            if (ar[0] == ar[1])
                puts("YES");
```

```
            else
                puts("NO");
        } else if (k == 3) {
            const double value = (ar[0] + ar[1] + ar[2]) / 3.0;
            if (ar[0] != value && ar[1] != value && ar[2] != value)
                puts("NO");
            else if (ar[0] != value || ar[1] != value || ar[2] != value)
                puts(l > 0 ? "YES" : "NO");
            else
                puts("YES");
        } else {
            double sum = ar[0] + ar[1] + ar[2] + ar[3];
            if (l >= 3)
                puts("YES");
            else if (l == 2) {
                if ((*max_element(ar, ar + 4) + *min_element(ar, ar + 4)) / 2.0 ==
sum / 4.0)
                    puts("YES");
                else
                    puts("NO");
            } else if (l == 1) {
                if (count(ar, ar + 4, sum / 4.0) >= 2)
                    puts("YES");
                else
                    puts("NO");
            } else {
                if (count(ar, ar + 4, sum / 4.0) == 4)
                    puts("YES");
                else
                    puts("NO");
            }
        }
    }
    return 0;
}
```

Γενικά συμπεράσματα

Αυτό ήταν ένα πρόβλημα που δυσκόλεψε αρκετούς, καθώς προσπάθησαν να το αντιμετωπίσουν με ανάλυση περιπτώσεων, και έχασαν χρόνο διότι δεν είναι εύκολο να βρεθούν όλες οι περιπτώσεις. Θα μπορούσαμε να συμπεράνουμε ότι αξίζει να αφιερώνουμε λίγη παραπάνω σκέψη ακόμα και στα πιο εύκολα προβλήματα αφού έχουμε βρει τη λύση, για να μπορέσουμε να βρούμε κάποια πιο εύκολη υλοποίηση η οποία θα είναι σωστή με μεγαλύτερη πιθανότητα.