@stakemura　stakemuraさんの問題に挑戦中！

プログラムの高速化に挑戦してみよう

次々と入力されるデータをリアルタイムに処理する、”ストリームデータ処理”と呼ばれる技術があります。

リアルタイム性が要求される処理の代表格とも言えるのが異常検知。

データ上の異常はなるべく速く発見する必要があることから、

夜間などにまとめてデータを処理する”バッチ処理”とは区別されて行われることが多々あります。

それでは、ストリームデータ処理において、どのような実装が求められるでしょうか？

以下のようなプログラムを考えてみたいと思います。

・標準入力からN個（1≦N≦10000）の整数（ただしいずれも偶数）が　改行(¥n)区切りで与えられる

・新しい整数を受け取るたびに、それまでの数の中央値を求め、同様に改行(¥n)区切りで標準出力に返す

・１秒以内にN個分の中央値を出力すること

なお、本設問における中央値とは、

「有限個のデータを小さい順に並べたとき中央に位置する値」のことであり、

データが偶数個の場合は、「中央に近い2つの値の算術平均」を用いてください。

本設問では、入力として与えられる数値はすべて偶数という前提があるため、中央値は整数で表現できると仮定して構いません。

また出力の際は、小数及び小数点は外すようお願いします。以下、入力と出力例になります。

```
8
6
> 2
> 5
> 6
```

ちなみに、なぜ中央値か？

それは外れ値の発見を目的とする異常検知において特に重宝されるためです。

例えば外れ値検出の指標の１つに、中央絶対偏差（Median Absolute Deviation）というものがあり、

これは計算過程において、対象となる入力データの中央値を用います。

ただし今回は、あくまでも中央値を出力することが目的となります。

そして最後の1秒以内というのが本設問のポイントです。

”ストリームデータ処理”では処理内容の性質上、高速なプログラムが求められるわけですが、

この1秒というのはCodeIQの自動採点プログラムの制約でもあります。

そもそも、中央値を単に求めるなら、配列をソートして真ん中の値を参照するだけです。

しかし、N=10000として、10000回配列をソートするとして果たしてトータルの計算時間は、1秒以内に収まるでしょうか？

選ぶ言語や実装によって変わってくるかと思いますが、C++で実装された以下のプログラム（見苦しい点がございますが、ご容赦ください）では、残念ながらN=10000の入力データを与えるとCodeIQ sandboxでは計算時間オーバーで弾かれてしまいました。

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

template<typename T>
T get_median(const std::vector<T> &v)
{
    std::size_t vh = v.size() >> 1;
    if ((v.size() & 1) == 1)
        return v[vh];
    else
        return (v[vh - 1] + v[vh]) / T(2);
}

int main()
{
    std::vector<int> v;
    v.reserve(10000);

    int median = 0;
    std::string row;
    while (std::getline(std::cin, row))
    {
        std::istringstream stream(row);
        int n;
        stream >> n;

        v.push_back(n);
        std::sort(v.begin(), v.end());
        median = get_median(v);

        std::cout << median << "¥n";
    }

    return 0;
}
```

さて、アルゴリズムやデータ構造を見直すなどしてもう少し高速化することはできないものでしょうか？

結論から言うと、上記プログラムはわずかな修正で1秒の壁を破ることが出来る（実測値で0.03秒前後）のですが、

そのことには拘らず、開発に用いる言語は何を選択しても大丈夫です。

是非挑戦してみてください。

【解答方法】

■挑戦言語は下記のプログラム言語選択で選択可能なものであれば何でもＯＫです！

1)自分の書いたプログラム言語を選択

2)解答欄にソースコードを記入

3)送信前に「提出前に確認」ボタンをクリック（構文エラーがないかどうかチェックできます）

4)「解答コードは正常に実行されました」というメッセージを確認の上、「解答を送信」ボタンで解答してください。

■この問題にはテストケースが3つ用意されています。そちらに通れば正解です！

【採点について】

・採点は「ideone」を使ってプログラムを実行し、標準入力および標準出力のテストケースと照合して正誤を判定します

・各言語の標準入力と標準出力はこちらを参考にしてください

・標準入力の最終行の改行はあり／なし両方に対応してください

※なおCodeIQで使用しているideoneは企業版のため、webで公開されているコンシューマー版ideoneとは

対応言語・バージョン・挙動が異なる場合があります。

企業版ideoneの対応バージョンは、「提出前チェック」の結果とともに表示されます。

▲

あなたの解答

```
if __name__ == "__main__":

    try:
        values = []
        while True:
            # 標準入力受付
            values.append(int(input()))
            # ソート
            values.sort()

            if len(values) % 2 == 0 :
                print(int((values[len(values) // 2 -1] + values[len(values) // 2]) / 2))
            else :
                print(values[len(values) // 2])

    except EOFError:
        pass
```

ページ上部へ

公式Facebookページ

公式Twitterアカウント

CodeIQ

CodeIQ MAGAZINE

CodeIQ JOBS

Read CodeIQ on RSS

CodeIQとは

CodeIQ（コードアイキュー）とは、自分の実力を知りたいITエンジニア向けの、実務スキル評価サービスです。

プレスリリース

CodeIQ

CodeIQ Magazine

CodeIQ JOBS

CodeIQご利用にあたって

利用規約

プライバシーポリシー

お問い合わせ

エンジニア採用ご担当者様へ

関連サイト

ATND

CODE VS

サンカク

カクシン

リクナビNEXT

リクナビNEXT ITキャリア

リクナビNEXT Tech総研

リクルートグループサイトへ