データ構造とアルゴリズム コーディング課題

情報科学科 22140026 谷 知拓

Approach

- 「最悪実行時間が, O(nlog(n))」「空間計算量(メモリ消費量)が最小」となるようなアルゴリズムとして, ヒープソートを利用する.
- 言語は, PHP を用いる.

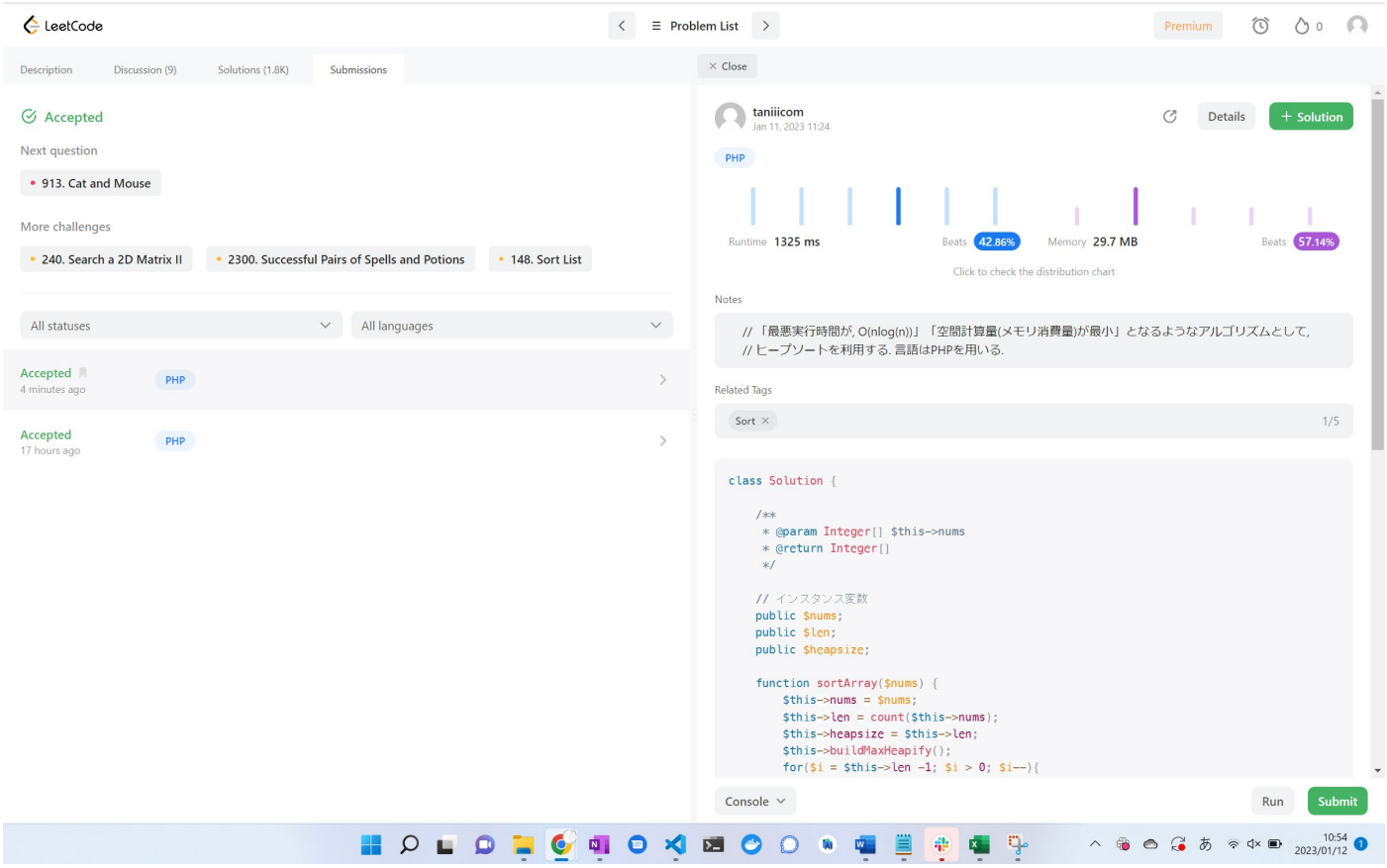
Code

```php
class Solution {

    /**
     * @param Integer[] $this->nums
     * @return Integer[]
     */

    // インスタンス変数
    public $nums;
    public $len;
    public $heapsize;

    function sortArray($nums) {
        $this->nums = $nums;
        $this->len = count($this->nums);
        $this->heapsize = $this->len;
        $this->buildMaxHeapify();
        for($i = $this->len -1; $i > 0; $i--){
            list($this->nums[0], $this->nums[$i]) = array($this->nums[$i], $this->nums[0]);
            $this->heapsize -= 1;
            $this->maxHeapify(0);
        }
        return $this->nums;
    }

    protected function maxHeapify($i) {
            $il = ($this->len > $i*2+1) ? $i*2+1 : null; // index of light-child
            $ir = ($this->len > $i*2+2) ? $i*2+2 : null; // index of light-child
            if(($il != null)&&($il < $this->heapsize)&&($this->nums[$il] > $this->nums[$i])){
                $imax = $il;  // index of max
            }else{
                $imax = $i;
            }
            if(($ir != null)&&($ir < $this->heapsize)&&($this->nums[$ir] > $this->nums[$imax])){
                $imax = $ir;
            }

            if($i != $imax){
                list($this->nums[$i], $this->nums[$imax]) = array($this->nums[$imax], $this->nums[$i]);
                $this->maxHeapify($imax);
            }
    }

    protected function buildMaxHeapify(){
        for($i = floor($this->len/2)-1; $i > -1; $i--){
            $this->maxHeapify($i);
        }
        return $this->nums;
    }

}
```

LeetCode 上での結果

- 「Accepted」受理された.
- Memory: Beats 57.14%

- Runtime Beats 42.86%



考察と感想

- Time Complexity: O(nlog(n)) のソートとして, HEAPSORT もしくは MERGESORT を思いついたが, Space Complexity: Smallest の条件より, HEAPSORT は 1, MERGESORT は n であるので, HEAPSORT を選択した.

- 時間計算量: 最悪実行時間 O(nlog(n))
- 空間計算量: 1
- In-place である(内部ソートである)
- 安定ではない

- 実際に LeetCode 上の受理結果をみてみると, 時間計算量に関しては, 今回の Submit された入力値の場合では, MERGESORT がより速いソリューションとなっている. 一方, 空間計算量に関しては, HEAPSORT が非常に優秀な結果となっている. (Runtime と Memory のどちらも, 最速/最小として記録されているソリューションは組み込み関数を用いているため, PHP 組み込み関数は複数のソートを組み合わせた, もしくは入力値により最適化するような, 複合ソートを用いているのではないかと考える)

- 今回 PHP でソートアルゴリズムを書いたことで気付けた発見として, これまで Python において, なぜ list 型の代入が参照渡しになるのかということが疑問だったが, これは 1 つには今回のソートのような, 1 つのオブジェクトを様々な場所(様々なスコープ内)から参照, 操作する利便性のためなんじゃないかと思った. PHP では, 代入(引数で渡す場合も含めて)はコピーとして扱われるので, 別関数内で list(正確には array)を変更しても, 関数間のスコープに阻まれるため, 当然のように, global な変数を使う, もしくは都

度 return で返し上書きする必要があると思っていたが, Python では, list 型変数は，オブジェクトの参照 ID が渡されているので，オブジェクト自体を操作することになり，複数関数を行き来する場合でもスコープに阻まれることなく操作ができるということに気付いた. Python がスクリプト言語としての使用を強く想定しており，かつオブジェクト指向に強く根差して作られているゆえかと思った. PHP においては，object 型の変数を, array の代わりに利用しているコードをよく見たことがあり, object 型の場合は代入や引数で渡す場合に「参照渡し」となるため, array 型(PHP では配列と連想配列は同じ型)と object 型を使い分けられるため，記述方針(手続き型に近い書き方かオブジェクト指向に近い書き方かのような感じ?)により選択できるのだと考えた.