本日の課題 (1)

Gnome（ノーム）ソートアルゴリズムについて，以下のWikipediaの記事を参考にして，JavsScriptでこのソートアルゴリズムを実現せよ．具体的には 関数 kadai(lst)を定義せよ．lstは数のリストであり，関数の返り値はlstを小さい方から順番に大きくなるように並べたリストになるようにせよ．

http://ja.wikipedia.org/wiki/%E3%83%8E%E3%83%BC%E3%83%A0%E3%82%BD%E3%83%BC%E3%83%88

この関数を用いた場合，almost sortedなデータ（ほとんどソートされているが，一部順番が崩れているようなデータ）について高速にソートされることを確認せよ．
almost sortedな数のリストを生成するプログラムはつぎのスライドに示す．

本日の課題 (2)

almost sortedな数のリストを生成する関数

```
function almost_sorted(n){
    var lst = []
    var N = Math.floor(n * 0.8)
    for (var i = 0; i < n; i++){
        lst.push(Math.random())
    }
    lst.sort(function (x, y){
        if (x < y) return -1
        else if (x > y) return 1
        else return 0
    } )
    for (var k = 0; k < N; k++){
        var i = Math.floor(Math.random() * (n - 5))
        var d = Math.floor(Math.random() * 5)
        var temp = lst[i]
        lst[i] = lst[i + d]
        lst[i + d] = temp
    }
    return lst
}
```

```
almost_sorted(20) =
[0.330844901990497 2,0.3024965223249045 4,0.1205683620239865 7,0.111728016151234 18,0.203901077965716 1,0.365640476419254 46,0.42568469103496687,0.4968298584874332,0.4853874045544003,0.565355013569610 7,0.5730692039281771,0.6564342010933445,0.6806016400656708,0.7906112778012458,0.7461100455669576,0.8206033959039577,0.8264918353558333,0.8305824371613544,0.894450389966517 4,0.9060165336873575]
```

解答例 (1)

ホームページの説明からわかるように，極めて単純な構造のアルゴリズムである．プログラム例は以下のとおり．

```
function kadai(lst){
    lst = lst.slice(0)
    var n = lst.length
    var i = 1
    while (i < n){
        if (lst[i - 1] <= lst[i]) i += 1
        else {
            [lst[i - 1], lst[i]] = [lst[i], lst[i - 1]]
            i -= 1
            if (i == 0) i += 1
        }
    }
    return lst
}
```

解答例 (2)

同じ個数の通常のランダムなデータとほとんどソートされているデータをノームソートでソートするのに要する時間を比較する.

```
function make_data(n){
    var s = []
    for (var i = 0; i < n; i++)
        s.push(Math.random())
    return s
}

function work(){
    var len = 10000
    var data = make_data(len)
    var datb = almost_sorted(len)
    var t1 = new Date()
    var dat1 = kadai(data)
    var t2 = new Date()
    var dat2 = kadai(datb)
    var t3 = new Date()
    puts(dat1[0] + " : " + dat1[len - 1])
    puts(dat2[0] + " : " + dat2[len - 1])
    puts("Time = " + (t2 - t1) + " ms")
    puts("Time = " + (t3 - t2) + " ms")
}

work()
```

```
0.00001800193019763085 5 : 0.999955148333930 3
0.00004282049330739 39 : 0.99981094878146 8
Time = 7211 ms
Time = 7 ms
```

効果は顕著で1000倍ほど実行時間が異なる. このソートアルゴリズムは一般のデータについてはそれほど高速にソートできないが, ある程度ソートされたデータに対しては高速に動く.