GO FOR IT Q5

s336651 大垣慶介

2012年2月13日

q5_elevator_4_search2.py のドキュメント

```
================================================================
q5_elevator_4_search2.py
----------------------------------------------------------------
arguments
    |- 1.input.csv
output
    |- Out1,Out2,Out3,Out4,Out5,Out6,Out7,Out8,Out9
    |- ....
        |-Out1 - エレベータの識別番号です。(1-)
        |-Out2 - エレベータの動作時刻です。開始してからの経過秒数で表します。(0-)
        |-Out3 - エレベータが動作した階です。(1-10)
        |-Out4 - エレベータの動作です。('B開': 'E閉':)
        |-Out5 - 入力データの識別番号1
        |-Out6 - 入力データの識別番号2
        |-Out7 - 入力データの識別番号3
        |-Out8 - 入力データの識別番号4
        |-Out9 - 入力データの識別番号5
----------------------------------------------------------------
for Python 2.7
K.Ogaki(ogaki@iis.u-tokyo.ac.jp)
2012/02/13
----------------------------------------------------------------
This program is released under GPL.
http://www.opensource.jp/gpl/gpl.ja.html.euc-jp
================================================================
```

表 1: 環境

言語	python2.7
OS	Linux(Ubuntu10.10)
CPU	Intel(R) Core(TM) i5 CPU M 480 @ 2.67GHz
メモリ	3780088 kB

仕様

問 i-iv に使用するファイルと実行コマンド (bash 環境) はそれぞれ

- i: q5_elevator_1.py input_i.csv
- ii: q5_elevator_1.py input_i.csv¿5_1.out; q5_elevator_2_check.py input_i.csv 5_1.out
- iii: q5_elevator_3_search.py¿5_3.out; q5_elevator_2_check.py input_iii_iv.csv 5_3.out
- iii: q5_elevator_4_search2.py¿5_4.out; q5_elevator_2_check.py input_iii_iv.csv 5_4.out

である。

アルゴリズム

iii : 1 台エレベーターでの探索

かなり探索の自由度が高いので、簡単のために、

- 客が待っている階に来たら必ず止まる (素通りはしない)
- 客は申告順に載せる (申告順に降ろすわけではない)

としたもとで、どのような動きをするエレベーターが最適かを考えた。

```
## --- pseudocode --- ##
direction = sum ([ 1.0/(destination\_i-nowfloor) foreach passangers])
```

この擬似コードのように、毎秒の行き先 (上 or 下) を、各乗客の目的地からの距離の逆数の和で決定した。全乗客は等しく合計時間に影響するという問題条件から、既に載せている乗客については近い順に降ろした方が効率的、という発想である。

iv : 2 台エレベーターでの探索

iii のアルゴリズムをベースに、2 台のエレベーターそれぞれ、“次に載せる乗客” をどのように選ぶかに主眼を置いた。

```
## --- pseudocode --- ##
nextcallindex = 2
def assignindexs(elevators, calldata):
    while( (not elevators[0].callindexs) or (not elevators[1].callindexs) ):
        if( distance1 > distance2 ):
            elevators[1].callindexs.append(nextcallindex)
        else:
            elevators[0].callindexs.append(nextcallindex)
        nextcallindex+=1
```

各エレベーターは次に載せる乗客の queue(:elevator.callindexs) を持っていて、2 台のエレベータのどちらかが乗客を降ろすたびに上の assignindexs() を実行して callindexs を更新する。次の待っている客 :nextcallindex を、その時点で近い方のエレベーターに割り当てる単純なアルゴリズムである。

結果

次の出力の章の最後の 3 つのプログラムを比較する。i,iii,iv 用の 3 つのプログラムを比べると、iii 用のプログラムでは i 用に比べて、同時に搬送する人数などが改善されている。また、iv 用のプログラムではエレベーターが 2 台になった時に、この評価基準では効率が 2 倍以上になっていることが分かる。

出力

```
$ ./q5_elevator_1.py input_i.csv
1,0,1,E,0,0,0,0,0
1,0,1,B,0,0,0,0,0
1,5,1,E,1,0,0,0,0
1,13,5,B,1,0,0,0,0
1,18,5,E,0,0,0,0,0
1,24,2,B,0,0,0,0,0
1,29,2,E,2,0,0,0,0
1,37,6,B,2,0,0,0,0
1,42,6,E,0,0,0,0,0
1,46,4,B,0,0,0,0,0
1,80,4,E,3,0,0,0,0
1,82,5,B,3,0,0,0,0
1,87,5,E,0,0,0,0,0
1,89,4,B,0,0,0,0,0
1,94,4,E,4,0,0,0,0
1,96,5,B,4,0,0,0,0
1,101,5,E,0,0,0,0,0

$ ./q5_elevator_1.py input_i.csv>5_1.out;./q5_elevator_2_check.py input_i.csv 5
    _1.out
Is Correct Move? :  True
Sum of All Cost is :  28

$ ./q5_elevator_1.py input_iii_iv.csv>5_3_test.out;./q5_elevator_2_check.py
    input_iii_iv.csv 5_3_test.out
Is Correct Move? :  True
Sum of All Cost is :  72936

$ ./q5_elevator_3_search.py input_iii_iv.csv>5_3.out;./q5_elevator_2_check.py
    input_iii_iv.csv 5_3.out
Is Correct Move? :  True
Sum of All Cost is :  60212

$ ./q5_elevator_4_search2.py input_iii_iv.csv>5_4.out;./q5_elevator_2_check.py
    input_iii_iv.csv 5_4.out
```

```
Is Correct Move? :  True
Sum of All Cost is :  26892
```