LaTeX:附录的插入与使用

Yushu @ LinCol

Date: August 18, 2019

1 Introduction

A First appendix

Here are simulation programmes we used in our model as follow.

Input matlab source:

./code/mcmthesis–matlab1.m

```
function [t,seat,aisle]=OI6Sim(n,target,seated)
pab=rand(1,n);
for i=1:n
    if pab(i)<0.4
        aisleTime(i)=0;
    else
        aisleTime(i)=trirnd(3.2,7.1,38.7);
    end
end
```

B Second appendix

some more text **Input C++ source:**

./code/mcmthesis–sudoku.cpp

```
//=======================================
// Name         : Sudoku.cpp
// Author       : wzlf11
// Version      : a.0
// Copyright    : Your copyright notice
// Description : Sudoku in C++.
//=======================================

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;
```

```
14
15 int table[9][9];
16
17 int main() {
18
19     for(int i = 0; i < 9; i++){
20         table[0][i] = i + 1;
21     }
22
23     srand((unsigned int)time(NULL));
24
25     shuffle((int *)&table[0], 9);
26
27     while(!put_line(1))
28     {
29         shuffle((int *)&table[0], 9);
30     }
31
32     for(int x = 0; x < 9; x++){
33         for(int y = 0; y < 9; y++){
34             cout << table[x][y] << "␣";
35         }
36
37         cout << endl;
38     }
39
40     return 0;
41 }
```

./code/PSO.py

```
1 # -*- coding: cp936 -*-
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import copy
5 from math import *
```

```
from random import *

x = np.linspace(0.0, 4.0)
y = np.cos(0 * x) - np.cos(3 * x) * np.exp(-x)

plt.subplot(1, 1, 1)
plt.plot(x, y, '-')

def F(x): return 1.-cos(3.*x)*exp(-x)
```

```
int main(int argc, char ** argv) {
printf("Hello world!\n"); // 显示结果
return 0;
}
```

C 伪代码的插入

引用伪代码中的内容：REM, 2

Algorithm 1: disjoint decomposition

Input: A bitmap Im of size $w \times l$

Output: A partition of the bitmap

special treatment of the first line;

for $i \leftarrow 2$ **to** l **do**

  special treatment of the first element of line i;

  for $j \leftarrow 2$ **to** w **do**

    left $\leftarrow$ `FindCompress(`$Im[i, j-1]$`);`

    up $\leftarrow$ `FindCompress(`$Im[i-1,]$`);`

    this $\leftarrow$ `FindCompress(`$Im[i, j]$`);`

    if left *compatible with* this **then** `// O(left,this)==1`

      if left $<$ this **then** `Union(left,this);`

      else `Union(this,left);`

    if up *compatible with* this **then** `// O(up,this)==1`

      if up $<$ this **then** `Union(up,this);`

      `// this is put under up to keep tree as flat as possible`

      else `Union(this,up);`

  foreach *element e of the line i* **do** `FindCompress(`p`);`

Algorithm 2: IntervalRestriction

Data: $G = (X, U)$ such that G^{tc} is an order.

Result: $G\ß = (X, V)$ with $V \subseteq U$ such that $G\ß^{tc}$ is an interval order.

begin

$V \longleftarrow U$;

$S \longleftarrow \emptyset$;

for $x \in X$ **do**

$NbSuccInS(x) \longleftarrow 0$;

$NbPredInMin(x) \longleftarrow 0$;

$NbPredNotInMin(x) \longleftarrow |ImPred(x)|$;

for $x \in X$ **do**

if $NbPredInMin(x) = 0$ **and** $NbPredNotInMin(x) = 0$ **then**

$AppendToMin(x)$

1 **while** $S \neq \emptyset$ **do**

REM remove x from the list of T of maximal index;

2 **while** $|S \cap ImSucc(x)| \neq |S|$ **do**

for $y \in S - ImSucc(x)$ **do**

{ remove from V all the arcs zy : };

for $z \in ImPred(y) \cap Min$ **do**

remove the arc zy from V;

$NbSuccInS(z) \longleftarrow NbSuccInS(z) - 1$;

move z in T to the list preceding its present list;

{i.e. If $z \in T[k]$, move z from $T[k]$ to $T[k-1]$};

$NbPredInMin(y) \longleftarrow 0$;

$NbPredNotInMin(y) \longleftarrow 0$;

$S \longleftarrow S - \{y\}$;

$AppendToMin(y)$;

$RemoveFromMin(x)$;