

# 1 附 1 C++

```
/*_____
Name: Sandpile-Model-N
Description: This Program Get The Sandpile Model -When To Arrive
            At Steady State.
Author: ztao1491@gmail.com
Date: 2017/04/04
Version: 1.0.1
Github: github.com/ztao1490/Networks/blob/master/Exercise/Net2_ex1
        /sandpile-model.cpp
Cpp-Style Form : http://zh-google-styleguide.readthedocs.io/en/
                latest/google-cpp-styleguide/
_____*/

#include <iostream>
#include <fstream>
#define random(a,b) (rand()%(b-a+1)+a)

using namespace std;

int main()
{
    // 随机加沙的坐标
    int sandpile_x, sandpile_y;
    // 定义两个数组进行对称交换——使得其能同时更新—一个判断雪崩一个计
    算更新
    int sandpile_A[10][10], sandpile_B[10][10];
    // 记录沙堆的高度, 寻求稳定态 N=20000 开始趋于稳定
    int data_N[1000000];

    // 初始化数组每格沙子为0
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            sandpile_A[i][j] = 0;
            sandpile_B[i][j] = 0;
        }
    }
}
```

```

}

// 每次加 sandpile_number 粒沙子重复次数
for (int Add = 0; Add < 1000000; Add++) {

    // 实验开始—————

    // 随机添加沙粒 sandpile_number
    // for (int N = 0; N < sandpile_number; N++) {
    // srand(time(NULL)); // 初始化随机数发生器
    sandpile_x = random(0,9); // [a,b] rand()%(b-a+1)+a;
    sandpile_y = random(0,9);
    // sandpile_x = 0;
    // sandpile_y = 0;
    sandpile_A[sandpile_x][sandpile_y]++; // 也可乘以2或更多
    sandpile_B[sandpile_x][sandpile_y]++;
    //}

    // 弛豫时间T
    for (unsigned long T = 0; T < 1000000000000000; T++) {
        // 判断是否存在雪崩 Number == 1000 则不存在 否则有
        int Number = 0;

        // 遍历所有格子判断是否雪崩
        for (int x = 0; x < 10; x++) {
            for (int y = 0; y < 10; y++) {
                // 格子x,y存在雪崩
                if (sandpile_A[x][y] > 3) {
                    // 处理边界情况
                    if (x == 0 ) {
                        if (y == 0 ) {
                            sandpile_B[x][y] = sandpile_B[x][y] - 4;
                            sandpile_B[x][y+1] = sandpile_B[x][y+1] + 1;
                            sandpile_B[x+1][y] = sandpile_B[x+1][y] + 1;

```

```

    }
    else if (y == 9) {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y-1] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x+1][y] = sandpile_B[x
            +1][y] + 1;
    }
    else {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y+1] = sandpile_B[x
            ][y+1] + 1;
        sandpile_B[x][y-1] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x+1][y] = sandpile_B[x
            +1][y] + 1;
    }
}
else if (x == 9 ) {
    if (y == 0) {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y+1] = sandpile_B[x
            ][y+1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x
            -1][y] + 1;
    }
    else if (y == 9) {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y-1] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x
            -1][y] + 1;
    }
}

```

```

else {
    sandpile_B[x][y] = sandpile_B[x][
        y] - 4;
    sandpile_B[x][y-1] = sandpile_B[x
        ][y-1] + 1;
    sandpile_B[x][y+1] = sandpile_B[x
        ][y+1] + 1;
    sandpile_B[x-1][y] = sandpile_B[x
        -1][y] + 1;
}
}
else if (y == 0 && x != 0 && x != 9) {
    sandpile_B[x][y] = sandpile_B[x][
        y] - 4;
    sandpile_B[x][y+1] = sandpile_B[x
        ][y+1] + 1;
    sandpile_B[x-1][y] = sandpile_B[x
        -1][y] + 1;
    sandpile_B[x+1][y] = sandpile_B[x
        +1][y] + 1;
}
else if (y == 9 && x != 0 && x != 9) {
    sandpile_B[x][y] = sandpile_B[x][
        y] - 4;
    sandpile_B[x][y-1] = sandpile_B[x
        ][y-1] + 1;
    sandpile_B[x-1][y] = sandpile_B[x
        -1][y] + 1;
    sandpile_B[x+1][y] = sandpile_B[x
        +1][y] + 1;
}
else {
    sandpile_B[x][y] = sandpile_B[x][y] -
        4;
    sandpile_B[x][y+1] = sandpile_B[x][y
        +1] + 1;
    sandpile_B[x][y-1] = sandpile_B[x][y

```

```

        -1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x-1][
            y] + 1;
        sandpile_B[x+1][y] = sandpile_B[x+1][
            y] + 1;
    }
    } else Number++;
}
}

// 将B同步到A再对A进行遍历
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        sandpile_A[i][j] = sandpile_B[i][j];
    }
}

// 当没有雪崩或雪崩结束时跳出循环
if (Number == 100) break;
}

// 实验结束—————

// 计算沙粒的总和
int Sum_Sand = 0;
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        Sum_Sand = Sum_Sand + sandpile_A[i][j];
    }
}

data_N[Add] = Sum_Sand;
}

// 分别将数组写入Txt文件
ofstream file3("sandpile-model_N.txt");
if (file3.is_open())
{

```

```

        for (int i = 0; i < 1000000; ++i) {
            file3 << data_N[i] << "\n";
        }
        file3.close();
    } else cout << "Unable to open file";

    //Python 处理后期数据

    return 0;
}

```

## 2 附 2 C++

```

/*_____
Name: Sandpile-Model
Description: This Program Simulate Sandpile Model Since It Has
            Self-Organized Criticality .
Author: ztao1991@gmail.com
Date: 2017/04/04
Version: 1.0.1
Github: github.com/ztao1991/Networks/blob/master/Exercise/Net2_ex1
        /sandpile-model.cpp
Cpp-Style Form : http://zh-google-styleguide.readthedocs.io/en/
                latest/google-cpp-styleguide/
_____*/

#include <iostream>
#include <vector>
#include <stdlib.h>
#include <time.h>
#include <fstream>
#include <numeric>
// #include <algorithm>
#include <map>
// #include <unordered_map>
#define random(a, b) (rand()%(b-a+1)+a)

```

```

using namespace std;

int main()
{
    srand((unsigned int) time(NULL));
    // 记录其随T的变化当存在雪崩时则为1否则为0
    // std::vector< int > height;
    // 每次增加N-1粒沙子
    // int sandpile_number = 2;
    // 随机加沙的坐标
    int sandpile_x, sandpile_y;
    // 定义两个数组进行对称交换——使得其能同时更新——一个判断雪崩一个计算更新
    int sandpile_A[10][10], sandpile_B[10][10];
    // 扩散状态——当雪崩扩散至此时为1状态标记否则为0
    // 存储不同规模下雪崩大小的数据
    std::vector<int> data_s;
    // 记录每次实验T的大小
    std::vector<int> data_t;

    // 初始化数组每格沙子为0
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            sandpile_A[i][j] = 0;
            sandpile_B[i][j] = 0;
        }
    }

    // 每次加 sandpile_number 粒沙子重复次数
    for (int Add = 0; Add < 1000000; Add++) {

        // 记录雪崩的规模
        int Avalanche = 0;
        // 弛豫记录时间长度
        int Time = 0;

        // 实验开始————

```

```

// 随机添加沙粒 sandpile_number
// for (int N = 0; N < sandpile_number; N++) {
// srand(time(NULL)); // 初始化随机数发生器

sandpile_x = random(0,9); // [a,b] rand()%(b-a+1)+a;
sandpile_y = random(0,9);
// sandpile_x = 0;
// sandpile_y = 0;
sandpile_A[sandpile_x][sandpile_y]++; // 也可乘以2或更多
sandpile_B[sandpile_x][sandpile_y]++;
//}

// 此前应该判断到达稳定态与否再统计N=20100

// 弛豫时间T
for (unsigned long T = 0; T < 100000000000000; T++) {
    // 判断是否存在雪崩 Number == 100 则不存在 否则有
    int Number = 0;

    // 遍历所有格子判断是否雪崩
    for (int x = 0; x < 10; x++) {
        for (int y = 0; y < 10; y++) {
            // 格子x,y存在雪崩
            if (sandpile_A[x][y] > 3) {
                // 处理边界情况
                if (x == 0 ) {
                    if (y == 0 ) {
                        sandpile_B[x][y] = sandpile_B[x][y] - 4;
                        sandpile_B[x][y+1] = sandpile_B[x][y+1] + 1;
                        sandpile_B[x+1][y] = sandpile_B[x+1][y] + 1;
                    }
                    else if(y == 9) {

```



```

        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y-1] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x+1][y] = sandpile_B[x
            +1][y] + 1;

    }

    else {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y+1] = sandpile_B[x
            ][y+1] + 1;
        sandpile_B[x][y-2] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x+1][y] = sandpile_B[x
            +1][y] + 1;

    }
}

else if (x == 9) {
    if (y == 0) {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y+1] = sandpile_B[x
            ][y+1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x
            -1][y] + 1;

    }
    else if (y == 9) {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y-1] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x
            -1][y] + 1;
    }
}

```

```

    }
    else {
        sandpile_B[x][y] = sandpile_B[x][
            y] - 4;
        sandpile_B[x][y-1] = sandpile_B[x
            ][y-1] + 1;
        sandpile_B[x][y+1] = sandpile_B[x
            ][y+1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x
            -1][y] + 1;
    }
}
else if (y == 0 && x != 0 && x != 9) {
    sandpile_B[x][y] = sandpile_B[x][y] -
        4;
    sandpile_B[x][y+1] = sandpile_B[x][y
        +1] + 1;
    sandpile_B[x-1][y] = sandpile_B[x-1][
        y] + 1;
    sandpile_B[x+1][y] = sandpile_B[x+1][
        y] + 1;
}
else if (y == 9 && x != 0 && x != 9) {
    sandpile_B[x][y] = sandpile_B[x][y] -
        4;
    sandpile_B[x][y-1] = sandpile_B[x][y
        -1] + 1;
    sandpile_B[x-1][y] = sandpile_B[x-1][
        y] + 1;
    sandpile_B[x+1][y] = sandpile_B[x+1][
        y] + 1;
}
else {
    sandpile_B[x][y] = sandpile_B[x][y] -

```

```

        4;
        sandpile_B[x][y+1] = sandpile_B[x][y
            +1] + 1;
        sandpile_B[x][y-1] = sandpile_B[x][y
            -1] + 1;
        sandpile_B[x-1][y] = sandpile_B[x-1][
            y] + 1;
        sandpile_B[x+1][y] = sandpile_B[x+1][
            y] + 1;
    }
    } else Number++;
}
}

// 将B同步到A再对A进行遍历
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        sandpile_A[i][j] = sandpile_B[i][j];
    }
}

// 当没有雪崩或雪崩结束时跳出循环
if (Number == 100) {
    Time = T - 1;
    break;
} else Avalanche = Avalanche + 100 - Number;
}

// 实验结束—————

// 到达稳定态时开始计算统计 Sandpile Steady State
if (Add > 200000) {
    // Save Avalanche
    data_s.push_back(Avalanche);
    // Save Time
    data_t.push_back(Time);
}

```

```

        data_s.shrink_to_fit();
        data_t.shrink_to_fit();
    }
}

// 统计S的频率
map<int, int> data_s_s;
for (size_t i = 0; i < data_s.size(); ++i) {
    ++data_s_s[data_s[i]];
}

// 统计T的频率
map<int, int> data_t_t;
for (size_t i = 0; i < data_t.size(); ++i) {
    ++data_t_t[data_t[i]];
}

// 分别将数组写入Txt文件
ofstream file1("sandpile-model_S.txt");
if (file1.is_open())
{
    using iterator = map< int, int >::iterator;
    for ( iterator iter = data_s_s.begin(); iter != data_s_s.
        end(); ++iter )
        file1 << iter->first << ":" << iter->second << '\n';
    // cout << "-" << data_p[i];
    file1.close();
} else cout << "Unable to open file";

ofstream file2("sandpile-model_T.txt");
if (file2.is_open())
{
    using iterator = map< int, int >::iterator;
    for ( iterator iter = data_t_t.begin(); iter != data_t_t.
        end(); ++iter )
        file2 << iter->first << ":" << iter->second << '\n';
    file2.close();
}

```

```

    } else cout << "Unable to open file";

    //Python 处理后期数据

    return 0;
}

```

### 3 附 3 Python

```

import numpy as np
import matplotlib.pyplot as plt

lines3 = open("sandpile-model_N.txt").readlines()
for i in range(len(lines3)):
    lines3[i] = lines3[i].strip('\n')

S3 = range(0,1000000,1)
P3 = []

for i in range(len(lines3)):
    P3.append(int(lines3[i]))

P3 =np.asarray(P3)

plt.title(r"$Sandpile-Model:N \sim T$")
plt.xlabel(r"$T$")
plt.ylabel(r"$N$")
plt.plot(S3,P3, '.b')
#plt.plot(S,P /100000)
# plt.plot(506652, 20629, 'or')
plt.savefig("sandpile_model_N.png")
plt.show()

#-----

import numpy as np

```

```

import matplotlib.pyplot as plt

#lines3 = open("sandpile-model_N.txt").readlines()
#for i in range(len(lines3)):
#    lines3[i] = lines3[i].strip('\n')
#
#S3 = range (0,1020678,1)
#P3 = []
#
#for i in range(len(lines3)):
#    P3.append(int(lines3[i]))
#
#P3 =np.asarray(P3)
#
#plt.title(r"$Sandpile-Model:N \sim T$")
#plt.xlabel(r"$T$")
#plt.ylabel(r"$N$")
#plt.plot(S3,P3, '.b')
##plt.plot(S,P /99999)
#plt.plot(506652, 20629, 'or')
#plt.savefig("sandpile_model_N.png")
#plt.show()

lines = open("sandpile-model_S.txt").readlines()
for i in range(len(lines)):
    lines[i] = lines[i].strip('\n')
    lines[i] = lines[i].split(':')

# lines

# print(lines[0][1])
#?lines[0][1]

S = []
P = []

for i in range(len(lines)):

```

```

        S.append(int(lines[i][0]))
        P.append(int(lines[i][1]))

S = np.asarray(S)
P = np.asarray(P)

# for i in range(len(P)):
#     P[i] = P[i] / 100000
# print(P)

plt.title(r"$Sandpile-Model:D(S) \sim S^{-\alpha}$")
plt.xlabel(r"$S$")
plt.ylabel(r"$D(S)$")
plt.loglog(S,P / 600000,'.b')
#plt.plot(S,P /100000)
plt.savefig("sandpile_model_S.png")
plt.show()
# #S = np.array[S]

lines2 = open("sandpile-model_T.txt").readlines()
for i in range(len(lines2)):
    lines2[i] = lines2[i].strip('\n')
    lines2[i] = lines2[i].split(':')

#lines2

#print(lines2[0][1])
#?lines[0][1]

S2 = []
P2 = []

for i in range(len(lines2)):
    S2.append(int(lines2[i][0]))
    P2.append(int(lines2[i][1]))

S2 = np.asarray(S2)

```

```

P2 = np.asarray(P2)

# for i in range(len(P)):
#     P[i] = P[i] / 100000
# print(P)

plt.title(r"$Sandpile-Model:D(S) \sim T^{-\beta}$")
plt.xlabel(r"$T$")
plt.ylabel(r"$D(S)$")
plt.loglog(S2,P2 /1000000, '.b')
# plt.plot(S,P /100000)
plt.savefig("sandpile_model_T.png")
plt.show()
# #S = np.array[S]

```