

实验 10 Count Sketch 的实现

一、实验要求

以“data.txt”作为数据流输入的源文件，自定义误差参数和哈希函数，实现 Count Sketch。

二、实验内容

(一) 参数说明

```
#define random(a,b) (rand()%(b-a+1))+a; //生成随机数，范围为[a,b]

const int NUM = 1000; //数字的个数
const double eps = 0.01; //误差参数
const double delta = 0.00001; //δ
const int prime1 = 379; //质数 1
const int prime2 = 509; //质数 2
const int big_prime = 1e9+7; //超大的质数
int hashSize; //一个哈希表的大小
int d; //一共有 d 个哈希函数
int **hashTable; //储存哈希的结果值
int **gTable; //储存 g 哈希函数的值
//哈希函数的形式  $h(x) = a * x + b$ 
vector<int> a; //储存不同哈希函数的系数值
vector<int> b; //储存不同哈希函数的常数值
```

(二) 核心代码说明

1. g 哈希函数的生成

```
void Process_g(){
cout << "process g hash function...." << endl;
    gTable = new int *[d]; //为 g 哈希函数动态分配内存
    for(int i = 0; i < d; i++){
        gTable[i] = new int[NUM];
    }
    for(int i = 0; i < d; i++){ //以 1/2 的概率生成 g 的值
        for(int j = 0; j < NUM; j++){
            int r = random(1, 10000);
            if(r <= 5000){
                gTable[i][j] = -1;
            }
        }
    }
}
```

```

        }else{
            gTable[i][j] = 1;
        }
    }
}
}
}

```

2. 不同哈希函数的生成

```

void Produce_hash(){
cout << "process hash function..." << endl;

    for(int i = 0; i < d; i++){//生成 d 个不同的哈希函数
        int temp_a = rand()%big_prime;//生成不同的系数
        int temp_b = rand()%big_prime;//生成不同的常数
        a.push_back(temp_a);
        b.push_back(temp_b);
    }
}

```

3. 更新哈希表数组的值

```

void Count_Sketch(int x, int y){//更新哈希表数组
    for(int i = 0; i < d; i++){
//哈希函数，找到哈希表数组相应的位置 index
        long long int index = (long long int)a[i] *
(long long int)x%big_prime + (long long int)b[i];
        index = (index % big_prime) % hashSize;
        hashTable[i][(int)index] += gTable[i][x] * y;//更新哈希表数组中对应的值
    }
}

```

4. 查询某一个数的频率

```


int Find_fi(int n){//查询某一个数的频率
    int ans[d];
    for(int i = 0; i < d; i++){
        long long int index = (long long int)a[i] * (long long int)n%big_prime + (long long int)b[i];
        index = (index % big_prime) % hashSize;//找到对应的索引
        ans[i] = hashTable[i][(int)index] * gTable[i][n];
    }
    sort(ans, ans+d);//通过排序寻找中位数
    return ans[(d-1)/2];
}

```

三、实验结果

(一) count_sketch.cpp

输入数字，返回该数字的频率值。每个数字的结果输出到“result.txt”中。

 "C:\大三上\学习\算法分析与设计\上机\10-Count Sketch\count_sketch.exe"

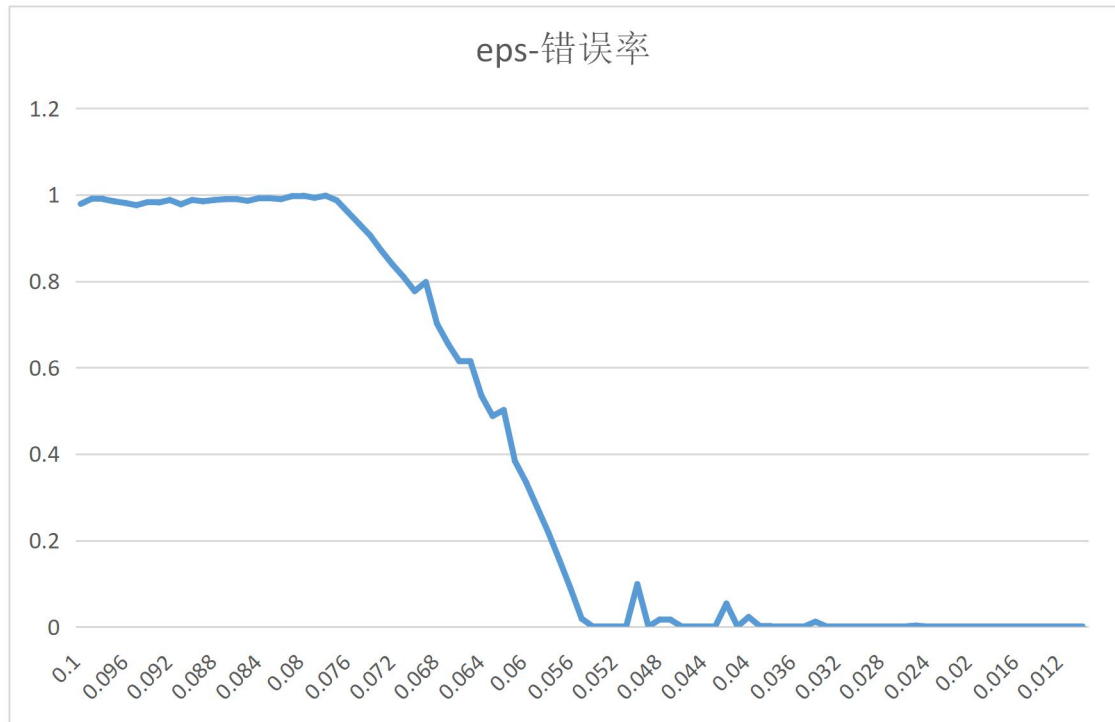
```
hashSize: 30000 d: 11
process g hash function...
process hash function...
read finish!
please enter the number you want to query:(-1 for exit)
9
the frequency of this element is:30
please enter the number you want to query:(-1 for exit)
45
the frequency of this element is:24
please enter the number you want to query:(-1 for exit)
999
the frequency of this element is:29
please enter the number you want to query:(-1 for exit)
555
the frequency of this element is:22
please enter the number you want to query:(-1 for exit)
10000
Wrong number! Please enter again!
please enter the number you want to query:(-1 for exit)
-4
Wrong number! Please enter again!
please enter the number you want to query:(-1 for exit)
-1

Process returned 0 (0x0)   execution time : 22.740 s
Press any key to continue.
```

(二) c_s.cpp

在原程序的基础上，设定不同的误差参数 ϵ 的值，与真实频率（frequency.txt）比较不同误差参数下的错误率（ $Num_{f_i \neq \hat{f}_i} / 1000$ ）。结果输出到“cs_eps.txt”中。

其中横坐标为 ϵ 的值，从左到右依次变小；纵坐标为错误率。



可以看到，当 **eps** 变小时，对数据流频率估计的错误率也在变小。在 **eps** 属于 0.08~0.05 范围内，错误率在明显下降。大约当 **eps**<0.04 时，错误率几乎稳定且趋于 0.