算法设计与实现作业 1

窦嘉伟 518021911160

1

各层塔高为 k 的概率为 P（h=k）=（1/2)^k-1 * 1/2 =(1/2)^k

于是有 期望 E（h）=1/（1-2）=2 期望塔高为 2

对于期望层数：首先我们有 P（塔高为 h）=（$1/2$）$^{\mathrm{h}-1} * (1-1/2) = (1/2)^{\mathrm{h}}$

以及 P（塔高<=h）=1 $- (1/2)^{\mathrm{h}}$

所以期望层数为 $\sum_{\mathrm{H}=1}^{\infty}$（$1/2^{\mathrm{H}}$）$^{\mathrm{n}-1} \times \mathrm{H} \times (1/2)^{\mathrm{H}}$

第 0 层中关键码在第 k 层仍出现的概率是（1/2）^k

于是第 k 层有 n*（1/2）^k 个元素

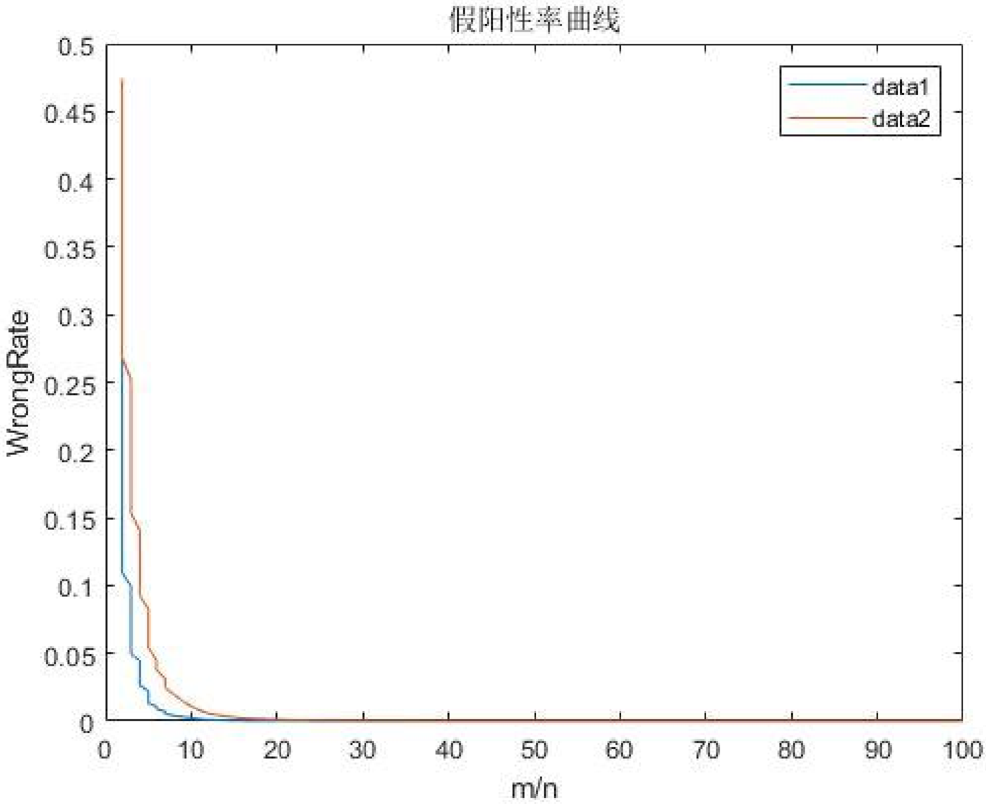
总元素有 n*（1+1/2+1/4+1/8+…）<2n

故空间复杂度为 O（n）

2 经过案例测试，得到结论，两个布隆选择器相比一个布隆选择器效率更高，假阳性曲线如图所示；

其中红色曲线为方案 ii，使用一个 bloom 选择器，在 n 较小是，二者几乎没有差异，当 n 接近二分之一 m 时，二者差异显著。

方案设计，在测试时，使用整形作为输入。m 设定为 200000，n 为 100000，每次插入 2000 组数据为间隔，每次间隔随机生成 400000 万组数据作为测试。



4 组哈希函数的选择如下

```
unsigned int Hash1(unsigned int x,unsigned int m){
        x = ~x + (x << 15); // x = (x << 15) - x - 1;
    x = x ^ (x >> 12);
    x = x + (x << 2);
    x = x ^ (x >> 4);
    x = x * 2057; // x = (x + (x << 3)) + (x << 11);
    x = x ^ (x >> 16);
    return x%m;

}
unsigned int Hash2(unsigned int x,unsigned int m){
    x = (~x) + (x << 21); // x = (x << 21) - x - 1;
    x = x ^ (x >> 24);
    x = (x + (x << 3)) + (x << 8); // x * 265
    x = x ^ (x >> 14);
    x = (x + (x << 2)) + (x << 4); // x * 21
    x = x ^ (x >> 28);
    x = x + (x << 31);
    return x%m;

}
```

```
unsigned int Hash3(unsigned int x,unsigned int m){
	x = (x+0x7ed55d16) + (x<<12);
	x = (x^0xc761c23c) ^ (x>>19);
	x = (x+0x165667b1) + (x<<5);
	x = (x+0xd3a2646c) ^ (x<<9);
	x =(x+0xfd7046c5) + (x<<3); // <<和 +的组合是可逆的
	x = (x^0xb55a4f09) ^ (x>>16);
	return x%m;
}
unsigned int Hash4(unsigned int x,unsigned int m){
	x = (x+0xfd7046c5) + (x<<3);	// <<和+ 是可逆
	x = (x+0xfd7046c5) + (x>>3);	// >>和+ 不保证是可逆的
	x = (x^0xb55a4f09) ^ (x<<16); // ^ 和<< 是可逆
	x = (x^0xb55a4f09) ^ (x>>16);
	return x%m;
}
```

生成随机测试数据时，与插入数据的区间做了区别，以防止生成以插入的数据使错误率虚高。