实验报告

2015210874 王庆 计研 156

算法思想

随机生成一亿个自然数：

方法一：使用 C++11 的标准类 std::random_device rd; std::mt19937_64 mt(rd());生成随机自然数（这些自然数都很大，范围在 0-18446744073709551615 之间）。

```
void produceTestfile(char* filename, int testnum) {
	FILE* fileStream;
	fileStream = fopen(filename,"wb+");
	unsigned long long maxNum = 18446744073709551615ll;

	std::random_device rd;
	std::mt19937_64 mt(rd());

	unsigned long long randomNum;
	for (int i = 0; i < testnum; i++)
	{
		randomNum = mt()%maxNum;
		fseek(fileStream, 8*i, SEEK_SET);
		fwrite((char*)&randomNum, 8, sizeof(char), fileStream);
	}
```

```
        fclose(fileStream);
}
```

方法二：使用 rand()生成较小的随机数(0-65536)

```
void produceTestfile2(char* filename, int testnum) {
        FILE* fileStream;
        fileStream = fopen(filename,"wb+");

        int randomNum;
        for (int i = 0; i < testnum; i++)
        {
                randomNum = rand();   //65534
                fseek(fileStream, 8*i, SEEK_SET);
                fwrite((char*)&randomNum, 8, sizeof(char), fileStream);
        }
        fclose(fileStream);
}
```

将随机生成的自然数按照二进制的格式保存到.bat 文件中：

```
unsigned long long getTestdata(char* filename, int index) {
        FILE* fileStream;
        fileStream = fopen(filename, "rb+");
        unsigned long long randomNum;
        fseek(fileStream, 8*index, SEEK_SET);
```

```
        fread(&randomNum, 8, sizeof(char), fileStream);
        fclose(fileStream);
        return randomNum;
}
```

可以直接读出对应 index 的随机数。

位图法：

定义一个数组： unsigned long long mark[100000000/512+1];这样做，能标记 100000256 个数据。使用字节位置和位位置（字节 0~195313 ，位 0~511 ）

对于一个 unsigned long long 数据 data

字节位置： int nBytePos =data/512

位位置： int nBitPos = data& 511;

对于一亿个数进行循环，找到出现数字对应的 bit 位，并将该位置标记为 1.

算法复杂度分析:

不考虑生成随机数的时间，算法的时间复杂度和空间复杂度都为 O(n), n 为输入自然数的个数。将生成的测试例写入文件运行时间变长，如果生成随机数之后直接处理，程序的运行时间将会变短。