随机化算法 - changyuanchn的专栏 - CSDN博客

0） 引论

随机是很有用的一个东西，先不去管什么随机化算法，至少随机数是个很好的东西，就像掷骰子，总可以帮组我们决定一些犹豫不决的并且无关紧要的事。在机器学习中，一般我们都是要在整个数据集中随机抽取一定的数据做训练，另外一些做测试，这样结果才能有说服力，这里也将用到了随机数。因此下面我们首先来讲解一下伪随机数发生器。

1） 伪随机数发生器

真正意义上的随机数是很难产生的，大多数的随机数都是伪随机数，伪随机数是可以计算出来的，并且它有一个周期。但是由于伪随机数拥有随机数的大部分统计性质，因此对于一般的应用，伪随机数就可以用于解决问题了，当然密码学除外，这个是要真的随机数的。

伪随机数发生器的原理是：

$$x_{i+1} = Ax_i \bmod M$$

，其中M为质数，并且一般取1<=x0

举个例子，如果M=11，A=7，x0=1，那么产生的伪随机数为：7,5,2,3,10,4,6,9,8,1,7,5,.......

这个随机数列的周期为10.

当然如果用上面的数计算随机数列的话，效果相当不好，因为周期太小，得不到几个随机数，因此我们要选择更好的数据，一般来说M要大，这样才能得到更长的周期。同时也要注意A的选取，因为A也会影响周期。

一般来说我们采用M=(2^31)-1 = 2147483647，这个是一个31位的质数，A=48271，这个A能使M得到一个完全周期

[cpp]view plaincopy

```cpp
static unsigned long Seed = 1;
define A 48271L
define M 2147483647
double Random(void)
{
    Seed = (A*Seed)%M;
return (double) Seed/M;
}
```

上面的代码产生的是（0,1）之间的随机数。这段代码由于乘法可能溢出，因此这个只具有理论意义，实际编程需要做一些改进

上面的算法产生的随机数还是有一个问题的，它无法产生重复的随机数。例如5,5,3,7,1这样的数列。

还有很多的随机数的算法利用的是下面的公式：

$$x_{i+1} = (Ax_i + C) \bmod 2^B$$

，这里C为奇数，同时如果数据选择不好的话，很有可能得到周期很短的随机数，例如

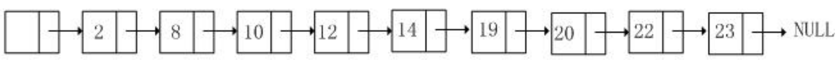
$$x_{i+1} = (48271x_i + 1) \bmod (2^{31} - 1)$$

，如果我们去Seed=179424105的话，那么随机数的周期为1，也就失去了随机的意义。

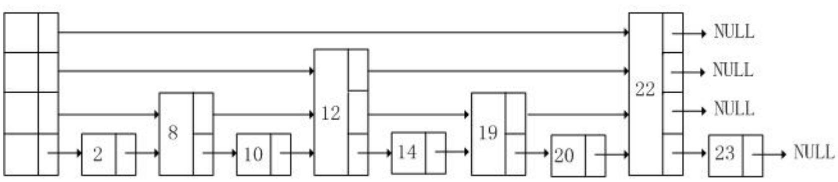
$$(48271 \times 179424105 + 1) \bmod (2^{31} - 1) = 179424105$$

2） 跳跃表

跳跃表是链表的一种变体，跳跃表可以使查找与插入的平均时间为O(logN)，而链表的时间为O(N)。这里我们指的是已排序的链表。如下图所示：



a. 简单的链表

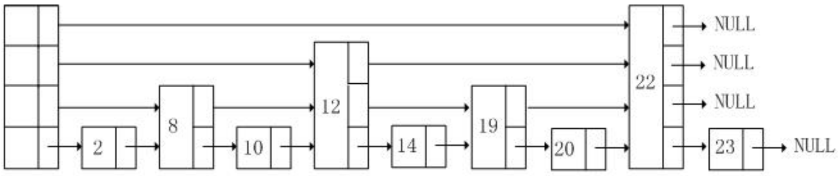


b. 跳跃链表

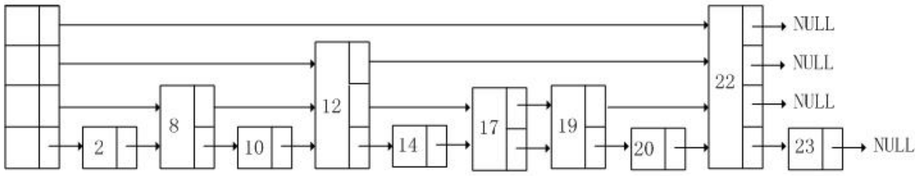
跳跃链表的k-level是指有k个指针的节点。

跳跃链表的查找是沿着节点的level进行查找，知道碰到大于查找值的点，那么我们降一个level查找，知道找到节点，例如查找节点19，首先从head查找到null,然后降一个level返回继续从head查找到12，然后查找到22，返回到12，降低一个level，查找到19。

跳跃链表的插入：首先查找到合适的位置，然后插入节点，节点的level是由随机算法产生。这是唯一用到随机算法的地方。如下图所示：



a. 跳跃链表



b. 跳跃链表插入新的值17

3） 素性检测

素性检测就是检测一个数是否为素数，当然这个数要是一个很大的数，这是数论中的一个重要的问题，有很多种方法，这里我们用到的费马小定理，利用这个定理判定一个数不是素数，那么这个数一定不是素数；而如果定理判定一个数是素数，那么有很大的概率可以判定这个数是素数，因此我们可以通过引入随机化算法来提高这个概率。

费马小定理：

如果 P 是一个素数，并且 $0<A<P$，那么 $A^{P-1} \equiv 1 (\bmod P)$

也就是说A^(P-1)与P同余。

这个定理存在的问题就是：如果这个定理判定一个数不是素数，那么这个数一定不是素数；而如果定理判定一个数是素数，那么有很大的概率可以判定这个数是素数

也就是说存在一些数，满足这个条件，但是这个数不是素数，这样的数集为Carmichael数集，其中最小的数为561。然而这个数集中的数出现的概率不大，因此我们可以通过一定的方法来提高这个概率。

引入下面的定理：

如果 P 是一个素数，并且 $0<X<P$，那么 $X^2 \equiv 1 (\bmod P)$ 仅有的两个解为 $X = 1, P-1$

因此如果在计算A^(P-1)（mod P）的任何时刻违背了上面的定理，那么P就不是素数。我们可以通过随机化算法随机选取A，比如说重复100次，如果均判定P为素数，那么P就有很大的可能是素数。因此这一方法可以以任意小的概率判定你某一个数为素数。