МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ МЕТАЛЛУРГИИ, МАШИНОСТРОЕНИЯ И ТРАНСПОРТА
ВЫСШАЯ ШКОЛА АВТОМАТИЗАЦИИ И РОБОТОТЕХНИКИ

КУРСОВАЯ РАБОТА

Дисциплина: Объектно-ориентированное программирование

Запаздывающие генераторы Фибоначчи

Разработал:
ст. гр. 3331506/80401 Сердюков С. Ю.

Преподаватель: Ананьевский М.С.

Санкт-Петербург

2020

Введение

Запаздывающие генераторы Фибоначчи (англ. Lagged Fibonacci Generator (LFG)) —это метод генерации псевдослучайных чисел.

Запаздывающие генераторы Фибоначчи примечательны тем, что их можно использовать статистических алгоритмах, требующих высокого разрешения.

Эти генераторы приобрели популярность из-за того, что выполнения арифметических операций с вещественными числами сравнялась со скоростью целочисленной арифметики, а LFG естественно реализуются в вещественной арифметике.

История

Последовательность Фибоначчи может быть описана рекуррентным соотношением:

Последовательность Фибоначчи может быть описана рекуррентным соотношением :

$$S_n = S_{n-1} + S_{n-2}$$

Следовательно, новый член представляет собой сумму двух последних членов в последовательности. Это может быть обобщено на последовательность:

$$S_n \equiv S_{n-a} \star S_{n-b} \pmod{m},\ 0 < a < b$$

В этом случае новый термин представляет собой некоторую комбинацию любых двух предыдущих членов. m обычно является степенью 2 (m = 2). Оператор ⋆ обозначает общую двоичную операцию. Это может быть сложение, вычитание, умножение или побитовый оператор исключающего ИЛИ (XOR). Теория генератора этого типа довольно сложна, и простого выбора случайных значений для j и k может быть недостаточно. Эти генераторы также очень чувствительны к инициализации.

Генераторы этого типа используют k слов состояния (они «запоминают» последние k значений).

Очевидными преимуществами данного алгоритма являются его быстрота, поскольку он не требует умножения чисел, а также, длина периода, однако, случайность, полученных с помощью него чисел, мало исследована.

Числа S_n – называют последовательностью Фибоначчи с запаздыванием, а числа a и b – запаздыванием.

Свойства

Лаги a и b — «магические» и их не следует выбирать произвольно. Рекомендуются следующие значения лагов: (a,b)=(55,24), (17,5) или (97,33). Качество получаемых случайных чисел зависит от значения константы, a чем оно больше, тем выше размерность пространства, в котором сохраняется равномерность случайных векторов, образованных из полученных случайных чисел. В то же время, с увеличением величины константы a увеличивается объём используемой алгоритмом памяти.

Значения (a,b)=(17,5) можно рекомендовать для простых приложений, не использующих векторы высокой размерности со случайными компонентами. Значения (a,b)=(55,24) позволяют получать числа, удовлетворительные для большинства алгоритмов, требовательных к качеству случайных чисел. Значения (a,b)=(97,33) позволяют получать очень качественные случайные числа и используются в алгоритмах, работающих со случайными векторами высокой размерности. Описанный фибоначчиев генератор случайных чисел (с лагами 20 и 5) используется в широко известной системе Matlab.

В настоящее время подобрано достаточно много пар чисел a и b, приведём некоторые из них:

(24,55),(38,89),(37,100),(30,127),(83,258),(107,378),(273,607),(1029,2281),(576,3217),(4178,9689),...}(24,55),(38,89),(37,100),(30,127),(83,258),(107,378),(273,607),(1029,2281),(576,3217),(4178,9689),...

Постановка задачи

Нам нужно получить псевдослучайное значение. При чем, если нам нужна последовательность случайных чисел, то она должна обладать хорошими статистическими свойствами.

Известны разные схемы использования метода Фибоначчи с запаздыванием. Один из широко распространѐнных фибоначчиевых датчиков основан на следующей рекуррентной формуле:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{если } k_{i-a} \geq k_{i-b} \\ k_{i-a} - k_{i-b} + 1, & \text{если } k_{i-a} < k_{i-b} \end{cases},$$

k_i, где — вещественные числа из диапазона [0,1), a и b — целые положительные числа, параметры генератора, называемые лагами. Для работы фибоначчиеву генератору требуется знать max(a,b) предыдущих сгенерированных случайных чисел. При программной реализации для хранения сгенерированных случайных чисел необходим некоторый объем памяти, зависящих от параметров a и b.

Такая формула будет выдавать нам случайные числа в диапазоне [0,1), нам же нужны целые положительные числа в несколько разрядов. Модернизируем эту формулу в следующую:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{если } k_{i-a} \geq k_{i-b} \\ k_{i-b} - k_{i-a}, & \text{если } k_{i-a} < k_{i-b} \end{cases},$$

k_i, где теперь целые положительные числа. Для полученной формулы есть следующий алгоритм:

1. Запрашиваем у пользователя параметры a и b, и кол-во желаемых случайных величин (Amount).
2. Создаем массив Arr[] , размер которого будет равен max(a,b)+Amount.
3. Для работы этой формулы необходимо участок массива от 0 до max(a,b), заполнить случайными величинами. Как эти величины получены, не имеет значения. В данном случае будем использовать встроенную функцию rand().

4. Выполняем цикл от i=0, до i=max(a,b)+Amount, увеличивая i на единицу.

5. В каждой итерации цикла выполняем проверку:

Если Arr[i-a]≥Arr[i-b] ,

- следующий эл-т массива равен Arr[i-a]−Arr[i-b],
- в противном случае Arr[i-b]−Arr[i-a].

Реализация

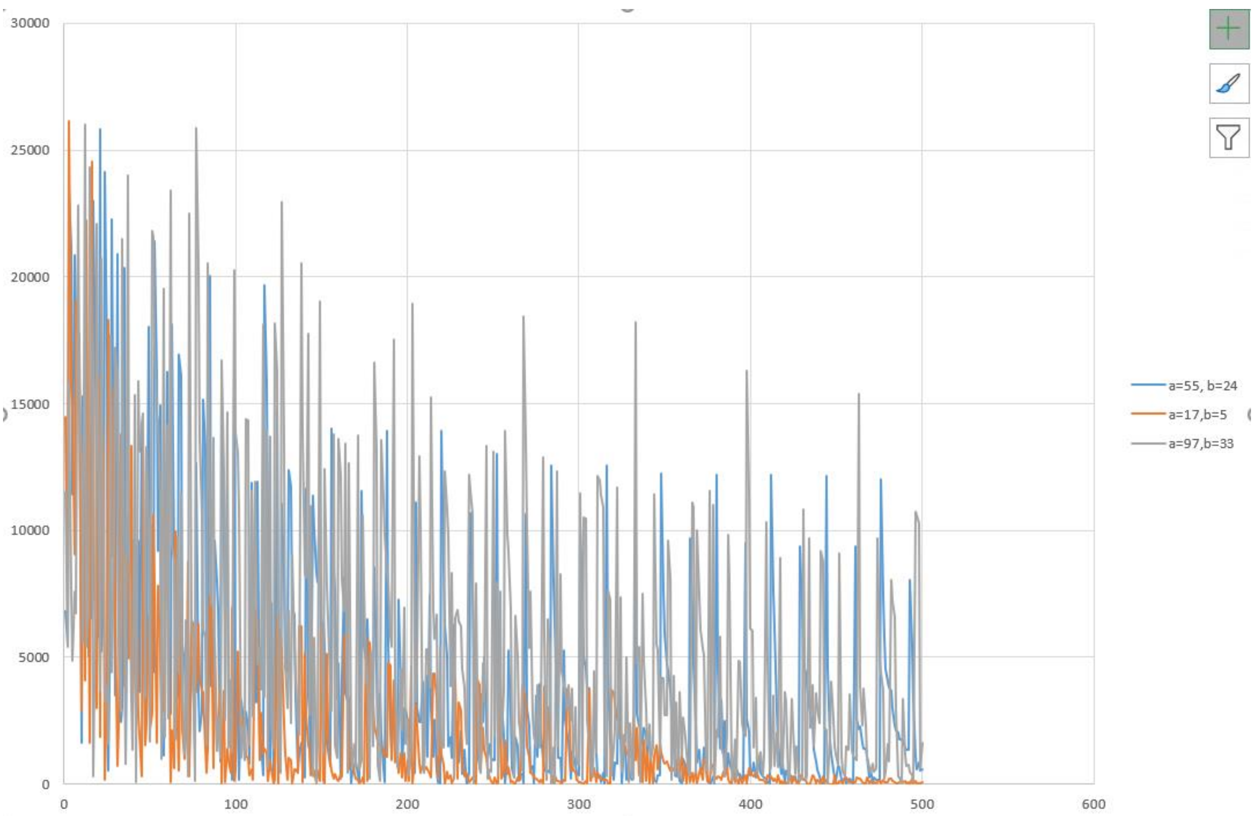
```cpp
#include <iostream>
#include <vector>
#include <iterator>
using namespace std;

typedef vector <int> Mass;

int main()
{
    unsigned int a = 0, b = 0;
    int Amount = 0;

    Mass::iterator LagA;                    //создаем все итераторы
    Mass::iterator LagB;
    Mass::iterator Lag;

    std::cout<<" Lag a = ";                 //запрос параметров
    std::cin>>a;
    std::cout<<" Lag b = ";
    std::cin>>b;
    std::cout<<" Amount of numbers ";
    std::cin>>Amount;

    Mass massive(1);                        //создаем массив
    massive.resize( max(a,b)+Amount+1);

    LagA=massive.begin();                   //инициализируем лаги и сдвигаем на значения а и б
    advance( LagA,a);
    LagB=massive.begin();
    advance( LagB,b);
    if(LagA>LagB) Lag=LagA;                 //доп. итератор для корректной работы цикла
    else Lag=LagB;

    Mass::iterator index;                   //основной итератор движения по массиву
    index=massive.begin();

    srand(0);                     //заполняем первую часть массива случайными значениями
    while(index<=Lag){
        *index = rand();
        index++;
    }

    for (; index != massive.end(); index++,LagA++,LagB++)
    {
        if (*LagA >= *LagB)                 //получаем последовательность методом Фибаначчи с запаздываем
        {
            *index = *LagA - *LagB;
        } else *index = *LagB -*LagA;
        std::cout<<*index<< std::endl;
    }
}
```

```
 Lag a = 55
  Lag b =24
  Amount of numbers10
 6808
5485
23396
21156
11431
20847
18401
17811
16971
1616
```

Анализ алгоритма

Сложность алгоритма можно оценить так: O(n)= max(a,b) +Amount +1.

Ниже приведён график полученных 500 значений при 3-х разных параметрах a и b.



На графики мы наблюдаем, что при повышении значений лагов a и b, алгоритм лучше рандомизирует числа.

Список литературы

1. Параметризация параллельных мультипликативных генераторов Фибоначчи с запаздыванием , М. Масканьи, А.Сринивасан
2. «Генераторы однородных случайных чисел для суперкомпьютеров» , Ричард Брен t, 1992