

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ЛАБОРАТОРНАЯ РАБОТА № 4**

на тему «Методы Монте-Карло»

Выполнил обучающийся гр. ВКБ33

Короп Анастасия

Проверил

Доцент, Савельев Василий Александрович

Ростов-на-Дону

2022

**Задание 1**

Использование вероятностного (стохастического) программирования требует эффективного источника случайных чисел. Создание полноценного источника случайных чисел — задача не простая и, как правило, такой источник получается слишком медленным для практического применения.

Поэтому источник (генератор) случайных чисел (ГСЧ) заменяют на генератор псевдослучайных чисел (ГПСЧ). Однако это оказалось связано с серьезными проблемами. Наиболее часто встречающиеся недостатки ГПСЧ:

* Короткий цикл — числа быстро начинают повторяться
* Явление Марсалья — случайные точки на плоскости — в пространстве заполняют какие-то плоскости, а не равномерно распределяются по телу
* последовательность предсказуема

Были предложены тесты для генераторов. Наиболее известны тесты DieHard, его дальнейшее развитие dieharder, и тест TestU01. Они проверяют пригодность генератора для задач стохастического программирования.

Отдельно стоит вопрос о пригодности ГПСЧ для использования в криптографических приложениях. Требования к криптографическим ГПСЧ сформулированы в рекомендациях NIST SP800-22. Но в данном задании нам не нухны криптографические генераторы.

Необходимо проверить с помощью одного из перечисленных тестов встроенный генератор случайных чисел. Если он проходит тест — использовать его в дальнейшем. В случае неудачи — найти быстрый ГПСЧ (например, генератор Марсалья-МакЛарена или «Вихрь Мерсенна») реализовать (или адаптировать) для используемой системы программирования. Протестировать и его.

Код программы:

|  |
| --- |
| from sys import argv  import graphlib as gr  if len(argv)>1:  if argv[1]!='/?':  filename=argv[1]  else:  print('Бросаем точки')  exit()  if len(argv)>4:  if argv[2]=='/b':  is\_bin=True  else:  is\_bin=False  start=int(argv[3])  fin=int(argv[4])  else:  is\_bin=False  start=0  fin=3  else:  is\_bin=False  start=0  fin=3  filename='input.txt'  def prima(cpoint,tpoint,rebrs,length=0,dellst=[],lens=[],path=''):  if length==0:  lens=[]  path+=str(cpoint)+'-'  if cpoint==tpoint:  lens.append(length)  #print(Брошено точек',path[:-1],':',length)  return None  for num in dellst:  rebrs.pop(num)  dellst=[]  for c,d in enumerate(rebrs):  if (cpoint in rebrs[c]):  dellst.append(c)  dellst.reverse()  for num in dellst:  if rebrs[num][0]!=cpoint:  nextp=rebrs[num][0]  else:  nextp=rebrs[num][1]  sqr\_count(nextp,tpoint,rebrs[:],length+rebrs[num][2],dellst,lens,path)  if not lens:  otv='{} {} -1'.format(cpoint,tpoint)  else:  otv=min(lens)  lens=''  return otv  graph=prima(filename,is\_bin)  length=sqr\_count(start,fin,graph)  print(length) |

Тест:

class Test3(unittest.TestCase):

def test\_1(self):

self.assertEqual(sqr\_count(0,3,graph), 20)

def test\_2(self):

self.assertEqual(sqr\_count(2,3,graph), 11)

def test\_3(self):

self.assertEqual(sqr\_count(3,0,graph), 20)

def test\_4(self):

self.assertEqual(sqr\_count(3,2,graph), 11)

def test\_5(self):

self.assertEqual(sqr\_count(2,4,graph), '2 4 -1')

def test\_6(self):

self.assertEqual(sqr\_count(4,2,graph), '4 2 -1')

def test\_7(self):

self.assertEqual(sqr\_count(5,6,graph), '5 6 -1')

def test\_8(self):

self.assertEqual(sqr\_count(6,5,graph), '6 5 -1')

def test\_9(self):

self.assertEqual(sqr\_count(1,3,graph), 15)

def test\_10(self):

self.assertEqual(sqr\_count(3,1,graph), 15)

if \_\_name\_\_ == "\_\_main\_\_":

graph=gr.load\_graph(filename,is\_bin)

unittest.main()

Итог прохождения теста:



**Задание 2 «Приближенное вычисление площади фигуры»**

Методы Монте-Карло или методы статистических испытаний – это группа численных методов, основанных на воспроизведении большого числа реализаций случайного процесса. Таким образом, суть метода заключается в статистическом моделировании случайных процессов, численном моделировании реализаций случайных процессов и оценивании параметров по реализациям случайных процессов методами математической статистики.

Под численным статистическим моделированием обычно понимают реализацию с помощью компьютера вероятностной модели некоторого объекта с целью оценивания изучаемых интегральных характеристик на основе закона больших чисел.

Свое экзотическое название метод получил от города Монте-Карло (княжество Монако), который известен благодаря своему казино, поскольку именно рулетка является одним из самых широко известных генераторов случайных чисел.

Применим метод статических испытаний или метод Монте-Карло к задаче вычисления площади геометрической фигуры на плоскости.

Метод заключается в следующем. Поместим данную фигуру в квадрат и будем наугад бросать точки в этот квадрат. Будем исходить из того, что чем больше площадь фигуры, тем чаще в нее будут попадать точки. Таким образом, при большом числе N точек, наугад выбранных внутри квадрата, доля точек, содержащихся в данной фигуре k, приближенно равна отношению площади этой фигуры и площади квадрата.

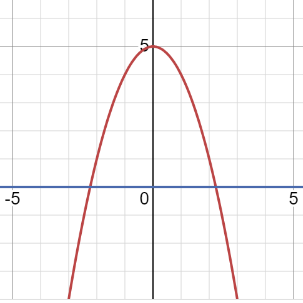
Код программы:

|  |
| --- |
| import os.path as osp, numpy as np  def cruscall(file,is\_bin=False):  if is\_bin:  graph=open(file,'rb')  fsize=osp.getsize(file)  reblist=[]  for e in range(int(fsize/12)):  temp=[]  for k in range(3):  temp.append(int.from\_bytes(graph.read(4),'little'))  reblist.append(temp)  else:  graph=open(file,'r')  reblist=[[int(k) for k in e.split()] for e in graph.readlines()]  graph.close()  return reblist  def save\_graph(graph,file\_name,is\_bin=False):  if is\_bin:  newf=open(file\_name,'wb')  for e in graph:  newf.write(e[0].to\_bytes(4, byteorder='little'))  newf.write(e[1].to\_bytes(4, byteorder='little'))  newf.write(e[2].to\_bytes(4, byteorder='little'))  else:  newf=open(file\_name,'w')  for e in graph:  newf.write(' \n'.format(e[0],e[1],e[2]))  newf.close() |

**Задание 2 (5 вариант)**

Найти площадь фигуры, ограниченной линиями:

y=0



Вывод программы:

 integral_(-sqrt(5))^sqrt(5) (5 - x^2) dx = (20 sqrt(5))\/3~~14.9071