Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский химико-технологический университет имени Д.И. Менделеева»
Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

Выполнил студент группы КС-30 Суханова Евгения Валерьевна
Ссылка на репозиторий:
https://github.com/MUCTR-IKT-CPP/EVSuhanova_30/blob/main/Algoritms/laba10.cpp

Приняли: Пысин Максим Дмитриевич
Краснов Дмитрий Олегович

Дата сдачи: 24.04.2023

Оглавление

Описание задачи.

Метод Монте-Карло

Пусть 2 игрока A и B играют в следующую игру: у игроков есть монетка, где 0 это орел, а 1 это решка, каждый игрок выбирает комбинацию из 3 цифр 0/1(например 001), затем подбрасывается монетка и результат записывается в длинную строку, побеждает тот чья комбинация будет на конце итоговой строки. Например: A - 001, B - 100, R - 01010101010100, победил B Необходимо смоделировать игру двух игроков и: Построить таблицу вероятности выигрыша одной из комбинаций, так чтобы столбцы соответствовали игроку А, а строки игроку Б, а на пересечении была бы вероятность побед игрока А над игроком Б, при выбранных ими комбинациях. Так же по результатом всех попыток определить суммарный средний шанс выигрыша игрока А и игрока Б вне зависимости от выбранных комбинаций Проанализировать полученные результаты

Описание метода/модели.

Метод Монте-Карло (методы Монте-Карло, ММК) — общее название группы численных методов, основанных на получении большого числа реализаций стохастического (случайного) процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи.

Многие системы слишком сложны для исследования влияния неопределенности с использованием аналитических методов. Однако такие системы можно исследовать, если рассматривать входные данные в виде случайных переменных, повторяя большое количество вычислений N (итераций), для получения результата с необходимой точностью.

Метод может быть применен в сложных ситуациях, которые трудны для понимания и решения с помощью аналитических методов. Модели систем могут быть разработаны с использованием таблиц и других традиционных методов. Однако существуют и более современные программные средства, удовлетворяющие высоким требованиям, многие из которых относительно недороги. Если модель разрабатывают и применяют впервые, то необходимое для метода Монте-Карло количество итераций может сделать получение результатов очень медленным и трудоемким. Однако современные достижения компьютерной техники и разработка процедур генерации данных по принципу латинского гиперкуба позволяют сделать продолжительность обработки незначительной во многих случаях.

Выполнение задачи.

Для начала необходимо реализовать алгоритм самой игры:

Генерируем комбинации для двух игроков. Проверяем не совпадают ли они. Если есть совпадение, то генерируем для второго игрока новую комбинацию до тех пор, пока совпадения не будет. Далее Генерируем последовательность подбрасываний до тех пор пока последовательность из последних трех бросков не будет совпадать с одной из комбинаций игроков. Проверяем какой игрок победил.

```
# реализация самой игры
for i in range(3):
  A=GenComb() # комбинация игрока A
  B=GenComb() # комбинация игрока B
# проверка различия комбинаций игроков
while A==B:
  B=""
  for i in range(3):
    B=GenComb()
print("A =", A)
print("B =", B)
R = ""
# пока кто-то не победит
while A!=R[-3:] and B!=R[-3:]:
  R+=str(random.randrange(2))
  print("R =", R)
if A==R[-3:]:
  print("Победил игрок A")
else:
  print("Победил игрок B")
```

```
# генерация комбинации
def GenComb():
  C=""
  for i in range(3):
    C+=str(random.randrange(2))
  return C
```

```
A = 011
B = 111
R = 1
R = 11
R = 110
R = 1100
R = 11001
R = 110011
Победил игрок A
```

Для реализации метода Монте-Карло нам необходимо проверить варианты для всех вариантов комбинаций у двух игроков. Для этого мы проводим несколько раундов игры для каждой пары комбинаций.

```
# варианты комбинаций для игроков
A = ['000', '001', '010', '011', '100', '101', '110', '111']
B = ['000', '001', '010', '011', '100', '101', '110', '111']
# количество побед при определенных комбинациях
countA= [[0] * len(A) for i in range(len(B))]
countB= [[0] * len(A) for i in range(len(B))]
# количестов раундов
rounds=1000
```

Проводим заданное количество раундов для каждой пары комбинаций.

```
for r in range(rounds):
  for i in range(len(A)):
    for j in range(len(B)):
      if i==j: # совпадение комбинаций
        countA[i][j]=rounds/2
        countB[i][j]=rounds/2
        continue
      R = "" # последовательность бросков
      while A[i]!=R[-3:] and B[j]!=R[-3:]: # пока кто-то не победит
        R+=str(random.randrange(2))
      if A[i]==R[-3:]:
        #print("Победил игрок A")
        countA[i][j]+=1
      else:
        #print("Победил игрок B")
        countB[i][j]+=1
```

Строим матрицу вероятностей для каждой комбинации. Аналогично, для игрока B.

```
# средняя вероятность победы игрока A при каждой комбинации
for i in range(len(A)):
  print(' %s '%B[i], end="| ")
  for j in range(len(B)):
    countA[i][j]/=rounds
    sumA+=countA[i][j]
    print('%.3f'%countA[i][j],end="| ")
  print()
```

```
# вероятность победы игрока A при определенной комбинации
sA=[0] * len(A)
print(" "*5 ,end="| ")
for i in range(len(A)):
  for j in range(len(B)):
    sA[i]+=countA[j][i]
  print('%.3f'%(sA[i]/len(A)),end="| ")
print()
```

```
# общая вероятность победы игрока A при любой комбинации
print("Вероятность выигрыша игрока A = ",round(sumA/64, 5))
print()
```

Полученные результаты

Вывод таблиц вероятностей для каждого игрока при количестве раундов равному 1000:

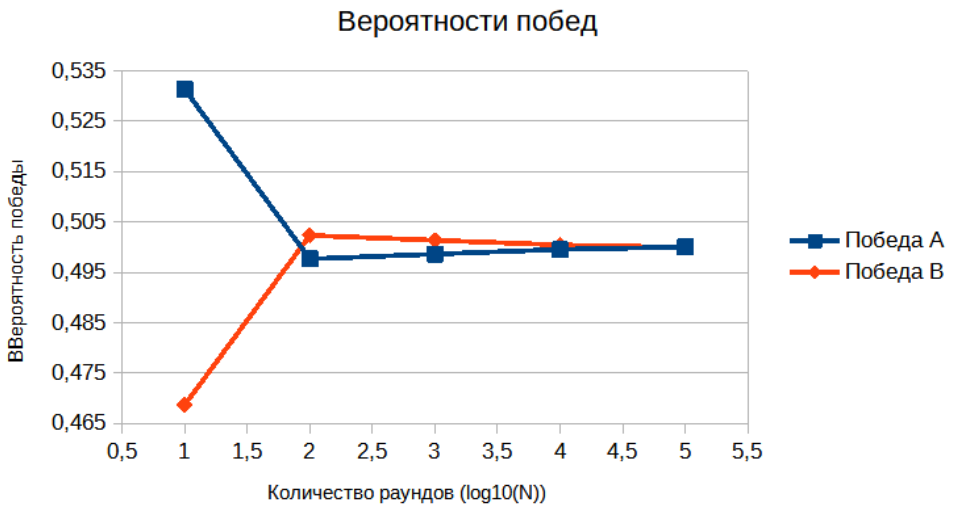
```
     |  000 |  001 |  010 |  011 |  100 |  101 |  110 |  111 |
 000 | 0.500| 0.526| 0.393| 0.379| 0.139| 0.431| 0.297| 0.495|
 001 | 0.479| 0.500| 0.662| 0.653| 0.259| 0.628| 0.502| 0.705|
 010 | 0.613| 0.313| 0.500| 0.480| 0.486| 0.477| 0.370| 0.577|
 011 | 0.613| 0.331| 0.499| 0.500| 0.507| 0.489| 0.731| 0.866|
 100 | 0.874| 0.750| 0.514| 0.496| 0.500| 0.528| 0.349| 0.615|
 101 | 0.592| 0.377| 0.517| 0.494| 0.500| 0.500| 0.341| 0.612|
 110 | 0.694| 0.471| 0.638| 0.264| 0.655| 0.664| 0.500| 0.495|
 111 | 0.480| 0.314| 0.431| 0.133| 0.384| 0.391| 0.486| 0.500|
     | 0.606| 0.448| 0.519| 0.425| 0.429| 0.514| 0.447| 0.608|
Вероятность выигрыша игрока A =  0.49936

     |  000 |  001 |  010 |  011 |  100 |  101 |  110 |  111 |
 000 | 0.500| 0.474| 0.607| 0.621| 0.861| 0.569| 0.703| 0.505|
 001 | 0.521| 0.500| 0.338| 0.347| 0.741| 0.372| 0.498| 0.295|
 010 | 0.387| 0.687| 0.500| 0.520| 0.514| 0.523| 0.630| 0.423|
 011 | 0.387| 0.669| 0.501| 0.500| 0.493| 0.511| 0.269| 0.134|
 100 | 0.126| 0.250| 0.486| 0.504| 0.500| 0.472| 0.651| 0.385|
 101 | 0.408| 0.623| 0.483| 0.506| 0.500| 0.500| 0.659| 0.388|
 110 | 0.306| 0.529| 0.362| 0.736| 0.345| 0.336| 0.500| 0.505|
 111 | 0.520| 0.686| 0.569| 0.867| 0.616| 0.609| 0.514| 0.500|
     | 0.394| 0.552| 0.481| 0.575| 0.571| 0.486| 0.553| 0.392|
Вероятность выигрыша игрока B =  0.50064

Сумма вероятностей выиграша обоих игроков (должно быть ~1) =  1.0
```

Зависимость вероятности победы игрока от количество раундов:

Количество раундов	log10(N)	Победа A	Победа B
10	1	0,53125	0,46875
100	2	0,49766	0,50234
1000	3	0,49864	0,50136
10000	4	0,49956	0,50044
100000	5	0,50004	0,49996



Заключение.

Из полученных результатов можно увидеть, что при увеличении количества раундов вероятность победы какого-либо игрока сходится к вероятности 50%. Это означает, что вероятность победы любого игрока абсолютно случайна.

Аналогичная ситуация происходит и с вероятностью победы конкретной комбинации. При увеличении количества раундов, вероятность победы определенной комбинации сводится к 50% шансу.

Полученные данные говорят о том, что победа в данной игре равна 50% шансу, что в свою очередь говорит об абсолютной случайности выигрыша.