Лабораторная работа №2. Методы поиска

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных. Для всех вариантов добавить реализацию добавления, поиска и удаления элементов. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

Задание 1

бинарный поиск

Фибоначчиев

интерполяционный

```
import random
import time
def random_arr(m = 50, min_limit = -250, max_limit = 1013):
    import random
    return [random.randint(min_limit, max_limit) for _ in range(m)]
def BinSearch(arr, x):
    i = 0
    j = len(arr)-1
    while i < j:
        m = int((i+j)/2)
        if x > arr[m]:
            i = m+1
        else:
            j = m
    return j
newMatrix = random_arr(15, -25, 100)
newMatrix.sort()
print (newMatrix)
k=5  #искомый элемент
g=BinSearch(newMatrix, k)
if newMatrix[g] == k:
    print ("Это элемент под номером ",g)
else: print ("Данного элемента в массиве нет")
print ("Добавить данный элемент в массив?")
s=(input())
if s == "да":
    newMatrix.insert(g, k)
    print (newMatrix)
if newMatrix[g] == k:
    print ("Удалить данный элемент из массива?")
    s=(input())
    if s == "да":
        newMatrix.pop(g)
        print (newMatrix)
[-15, -9, 0, 12, 12, 29, 35, 35, 40, 56, 65, 73, 89, 94, 94]
```

```
Данного элемента в массиве нет
Добавить данный элемент в массив?
[-15, -9, 0, 12, 12, 29, 35, 35, 40, 56, 65, 73, 89, 94, 94]
Данного элемента в массиве нет
Добавить данный элемент в массив?
lf
```

```
def FibonacciSearch(lys, val):
    fibM_minus_2 = 0
    fibM_minus_1 = 1
    fibM = fibM_minus_1 + fibM_minus_2
    while (fibM < len(lys)):
        fibM_minus_2 = fibM_minus_1
        fibM_minus_1 = fibM
        fibM = fibM_minus_1 + fibM_minus_2
    index = -1;
    while (fibM > 1):
        i = min(index + fibM_minus_2, (len(lys)-1))
        if (lys[i] < val):
            fibM = fibM_minus_1
            fibM_minus_1 = fibM_minus_2
            fibM_minus_2 = fibM - fibM_minus_1
            index = i
        elif (lys[i] > val):
            fibM = fibM_minus_2
            fibM_minus_1 = fibM_minus_1 - fibM_minus_2
            fibM_minus_2 = fibM - fibM_minus_1
        else :
            return i
    if(fibM_minus_1 and index<(len(lys)-1) and lys[index+1] == val):
        return index+1;
    return -1
from random import randint

newMatrix=[12,4,1,33,2,34]
newMatrix.sort()
print (newMatrix)
k=1
n=len(newMatrix)
g=FibonacciSearch(newMatrix, k)
if newMatrix[g] == k:
    print ("Это элемент под номером ",g)
else: print ("Данного элемента в массиве нет")
print ("Хотите добавить данный элемент в массив?")
s=(input())
if s == "да":
    newMatrix.insert(g, k)
    print (newMatrix)
if newMatrix[g] == k:
    print ("Хотите удалить данный элемент из массива?")
    s=(input())
```

```
    if s == "да":
        newMatrix.pop(g)
        print (newMatrix)
[1, 2, 4, 12, 33, 34]
Это элемент под номером  0
Хотите добавить данный элемент в массив?
[1, 2, 4, 12, 33, 34]
Это элемент под номером  0
Хотите добавить данный элемент в массив?
да
[1, 1, 2, 4, 12, 33, 34]
Хотите удалить данный элемент из массива?
[1, 1, 2, 4, 12, 33, 34]
Хотите удалить данный элемент из массива?
и
def interpolation(a, k) :
    left = 0
    right = len(a) - 1
    while a[left] < k and k < a[right]:
        mid = int(left + (k - a[left]) * (right - left) / (a[right] - a[left]))
        if a[mid] < k:
            left = mid + 1
        elif a[mid] > k:
            right = mid - 1
        else:
            return mid

    if a[left] == k:
        return left
    elif a[right] == k:
        return right
    else:
        return left
newMatrix = random_arr(15, -25, 100)
newMatrix.sort()
print (newMatrix)
k=12
g=interpolation(newMatrix, k)
if newMatrix[g] == k:
    print ("Это элемент под номером ",g)
else: print ("Данного элемента в массиве нет")
print ("Хотите добавить данный элемент в массив?")
s=(input())
if s == "да":
    newMatrix.insert(g, k)
    print (newMatrix)
if newMatrix[g] == k:
    print ("Хотите удалить данный элемент из массива?")
    s=(input())
    if s == "да":
```

```
    newMatrix.pop(g)
    print (newMatrix)
[-23, -21, -17, -2, 5, 23, 23, 27, 31, 31, 36, 38, 44, 86, 96]
Данного элемента в массиве нет
Хотите добавить данный элемент в массив?
[-23, -21, -17, -2, 5, 23, 23, 27, 31, 31, 36, 38, 44, 86, 96]
Данного элемента в массиве нет
Хотите добавить данный элемент в массив?
да
[-23, -21, -17, -2, 5, 12, 23, 23, 27, 31, 31, 36, 38, 44, 86, 96]
Хотите удалить данный элемент из массива?
[-23, -21, -17, -2, 5, 12, 23, 23, 27, 31, 31, 36, 38, 44, 86, 96]
Хотите удалить данный элемент из массива?
ь
```

Задание 2

```
class HashMap:
  def Fhash (a, id):
    n=64
    i=0
    hash = (ord(id[0])+ord(id[len(id)//2])+ord(id[len(id)-1])+i)//n
    if a[hash] == None:
      a[hash]=id
    else:
      k=hash
      i=i+1
      while (a[hash]!=None) and (hash !=255):
        hash = (ord(id[0])+ord(id[len(id)//2])+ord(id[len(id)-1])+i)//n
        i=i+1
      if a[hash] == None:
        a[hash]=id
      else:
        hash = i//n
        while (a[hash]!=None) or (hash !=k):
          hash = i//n
          i=i+1
        if a[hash] == None:
          a[hash]=id
        else:
          print ("There are no places in the table")
    return a
  def Fhashsearch (a, id):
    n=256
    i=0
    hash = (ord(id[0])+ord(id[len(id)//2])+ord(id[len(id)-1])+i)//n
    if a[hash] == id:
      return hash
    else:
      k=hash
      i=i+1
```

```
            while (a[hash]!=id) and (hash !=255):
                hash = (ord(id[0])+ord(id[len(id)//2])+ord(id[len(id)-1])+i)//n
                i=i+1
        if a[hash] == id:
            return hash
        else:
            hash = i//n
            while (a[hash]!=id) and (hash <250):
                hash = i//n
                i=i+1
            if a[hash] == id:
                return hash
            else:
                print ("Идентификатор не найден")
    def Fhashdel (a, id, n):
        a[n]=None
        return (a)


   a=[None]*64
print ("Хотите добавить новый идентификатор в таблицу?")
s=(input())
if s == "да":
    while s =="да":
        id=(input())
        HashMap.Fhash (a, id)
        print ("Хотите добавить новый идентификатор в таблицу?")
        s=(input())
print (a)
print ("Хотите найти идентификатор в таблице?")
k=(input())
if k == "да":
    while k =="да":
        id=(input())
        hash =HashMap.Fhashsearch (a, id)
        print (hash)
        print ("Хотите найти идентификатор в таблице?")
        k=(input())
    if hash!= None:
        print ("Хотите удалить идентификатор из таблицы?")
        k=(input())
        if k == "да":
            HashMap.Fhashdel(a, id, hash)
            print (a)
Хотите добавить новый идентификатор в таблицу?
нет
[None, None, None, None, None, None, None, None, None, None, None, None, None
, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, No
```

```
ne, None, None, None, None, None, None, None, None, None, None, None, None, N
one, None, None, None, None, None, None, None, None, None, None, None, None]
Хотите найти идентификатор в таблице?
нет
```

```
class RandomHashMap:
    def Fhash2 (a, id, i):
        import random
        i=i+1
        if i<63:
            n=64
            hash = (ord(id[0])+ord(id[len(id)//2])+ord(id[len(id)-1])+random.randint(0, 1500))//n
            if a[hash] == None:
                a[hash]=id
            else:
                k=hash
                while (a[hash]!=None):
                    hash = (ord(id[0])+ord(id[len(id)//2])+ord(id[len(id)-1])+random.randint(0, 1500))//n

        else:
            print ("There are no places in the table")
        return a
    def Fhashsearch2(a, id):
        i=0
        while i<63 and a[i]!=id:
            i=i+1
        if a[i]==id:
            return i
        else:
             print ("Идентификатор не найден")
    def FhashDel2(a, id):
        i=0
        while i<63 and a[i]!=id:
            i=i+1
        if a[i]==id:
            a[i]=None
            return a
        else:
             print ("Идентификатор не найден")
a=[None]*64
i=0
print ("Хотите добавить новый идентификатор в таблицу?")
s=(input())
if s == "да":
    while s =="да":
        id=(input())
        RandomHashMap.Fhash2 (a, id, i)
        print ("Хотите добавить новый идентификатор в таблицу?")
        s=(input())
print (a)
print ("Хотите найти идентификатор в таблице?")
```

```
k=(input())
if k == "да":
   while k =="да":
      id=(input())
      hash =RandomHashMap.Fhashsearch2 (a, id)
      print (hash)
      print ("Хотите найти идентификатор в таблице?")
      k=(input())
   if hash!= None:
      print ("Хотите удалить идентификатор из таблицы?")
      k=(input())
      if k == "да":
         RandomHashMap.Fhashdel2 (a, id)
         print (a)
Хотите добавить новый идентификатор в таблицу?
да
5
Хотите добавить новый идентификатор в таблицу?
нет
[None, None, None, None, None, None, None, None, None, None, None, None, None
, None, None, None, None, None, None, None, None, None, '5', None, None
, None, None, None, None, None, None, None, None, None, None, None, Non
e, None, None, None, None, None, None, None, None, None, None, None, None, No
ne, None, None, None, None, None, None, None, None, None, None, None, None]
Хотите найти идентификатор в таблице?
да
5
23
Хотите найти идентификатор в таблице?
нет
Хотите удалить идентификатор из таблицы?
нет
class ChainMap:
   def Chainmethod (arr, id):
      h=0
      for i in range (0, len(id)):
         h=ord(id[i])+h
      key=h%10
      arr[key].append(id)
      return key
   def ChainmethodSearch (a, id):
      h=0
      for i in range (0, len(id)):
         h=ord(id[i])+h
      key=h%10
      k=0
      for i in range (len(a[key])):
         if a[key][i]==id:
            print("Строка ", key, "Элемент ", i)
            k=1
```

```
        if k==0:
            print("Идентификатор не найден")
            return (1)

    def ChainmethodDeletions (a, id):
        h=0
        for i in range (0, len(id)):
            h=ord(id[i])+h
        key=h%10
        for i in range (len(a[key])):
            if a[key][i]==id:
                del(a[key][i])
        return (key)
key=[[]*1 for i in range (10)]
print ("Хотите добавить новый идентификатор в таблицу?")
s=(input())
if s == "да":
    while s =="да":
        id=(input())
        x=ChainMap.Chainmethod (key, id)
        print ("Хотите добавить новый идентификатор в таблицу?")
        s=(input())
print (key)

print ("Хотите найти идентификатор в таблице?")
k=(input())
if k == "да":
    while k =="да":
        id=(input())
        hash =ChainMap.ChainmethodSearch (key, id)
        print ("Хотите найти идентификатор в таблице?")
        k=(input())
    if hash!= 1:
        print ("Хотите удалить идентификатор из таблицы?")
        k=(input())
        if k == "да":
            hash =ChainMap.ChainmethodDeletions (key, id)
            print (key)
Хотите добавить новый идентификатор в таблицу?
нет
[[], [], [], [], [], [], [], [], [], []]
Хотите найти идентификатор в таблице?
нет
```

Расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого». Подразумевается, что ферзь бьёт все клетки, расположенные по вертикалям, горизонталям и обеим диагоналям
Написать программу, которая находит хотя бы один способ решения задач.

```
board = [[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
         [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
         [1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

         [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
         [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]]

def clearBoard(board):
    for i in range(8):
        for j in range(8):
            board[i][j] = 0

def printBoard(board):
    for i in range(8):
        for j in range(8):
            if board[i][j] == 1:
                print("1", end="")
            else:
                print("0", end="")
        print("")
    print("")
def checkColsOK(board):
    for i in range(8):
        sum = 0
        for j in range(8):
            sum += board[j][i]
        if sum > 1:
            return 0
def checkRowsOK(board):
    for i in range(8):
        sum = 0
        for j in range (8):
            sum += board[i][j]
        if sum > 1:
            return 0
def checkDiagsOK(board):

# left to right, bottom up
    counter = 8
    sum = 0
```

```
    for i in range(8):
        x = i
        y = 0
        for j in range(counter):
            #print(board[y][x], end="")
            sum += board[y][x]
            x += 1
            y +=1
        counter -= 1

        if sum > 1:
            return 0
        sum = 0


# right to left, top down
    counter = 8
    sum = 0

    for i in range(8):
        x = i
        y = 0
        for j in range(counter):
            #print(board[x][y], end="")
            sum += board[x][y]
            x += 1
            y +=1
        counter -= 1

        if sum > 1:
            return 0
        sum = 0


# right to left, bottom up
    counter = 8
    sum = 0

    for i in reversed(range(8)):
        x = i
        y = 0
        for j in range(counter):
            #print(board[x][y], end="")
            sum += board[x][y]
            x -= 1
            y += 1
        counter -= 1
        if sum > 1:
            return 0
        sum = 0
```

```
# left to right, top down
    counter = 8
    sum = 0

    for i in range(8):
        x = 7
        y = i
        for j in range(counter):
            #print(board[x][y], end="")
            sum += board[x][y]
            x -= 1
            y += 1
        counter -= 1
        if sum > 1:
            return 0
        sum = 0

def addQueen(board, col):

    row = 0

    for row in range(8):
        board[row][col] = 1
        if (checkRowsOK(board) != 0 and checkDiagsOK(board) != 0):
            if col == 7:
                printBoard(board)
            else:
                addQueen(board, col + 1)
        board[row][col] = 0


clearBoard(board)
addQueen(board, 0)
```

```
10000000
00000010
00001000
00000001
01000000
00010000
00000100
00100000

10000000
00000010
00010000
00000100
00000001
01000000
00001000
00100000
```

Вывод: в ходе лабораторной работы мы освоили методы поиска (бинарный поиск, Фибоначчиев, интерполяционный, простое рехэширование,рэхэширование с помощью случ.чисел и метод цепочек) и реализовали задачу про расстановку ферзей.