Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчёт по лабораторной работе № 2

«Методы поиска»

по дисциплине «Сиаод»

Выполнил: студент группы

БВТ1902

Сорокин Никита Андреевич

Москва

2021

Цель работы

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных. Для всех вариантов добавить реализацию добавления, поиска и удаления элементов.

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.

Код программы

```
from random import randint
import time
import copy

x = int(input("Введите x: "))

min = int(input("Введите min: "))
max = int(input("Введите max: "))

print(" ")
print("Результат: ")

mass2 = []

for j in range(x):
    mass2.append(randint(min, max))
print(mass2)

print("Отсортированный результат: ")
mass2.sort()

print(mass2)

print("")

NN = len(mass2)


def bin_poisk(mass2):
    global bin_mass
    bin_mass = copy.copy(mass2)
    print("")
    print("БИНАРНЫЙ ПОИСК ЭЛЛЕМЕНТА")
    print()
    ell = input("Введите эллемент для поиска: ")
    ell = int(ell)

    mid = len(bin_mass) // 2
    low = 0
    high = len(bin_mass) - 1

    while bin_mass[mid] != ell and low <= high:
        if ell > bin_mass[mid]:
            low = mid + 1
        else:
            high = mid - 1
        mid = (low + high) // 2
```

```
    if low > high:
        print("Нет такого числа")
    else:
        print("Есть такое число")


def bin_poisk_del(mass2,bin_mass):
    ell = input("Введите эллемент для удаления: ")
    ell = int(ell)

    mid = len(bin_mass) // 2
    low = 0
    high = len(bin_mass) - 1

    while bin_mass[mid] != ell and low <= high:
        if ell > bin_mass[mid]:
            low = mid + 1
        else:
            high = mid - 1
        mid = (low + high) // 2

    if low > high:
        print("Нет такого числа")
    else:
        print("Результат: ")
        del bin_mass[mid]
        print(bin_mass)

def bin_poisk_ins(mass2,bin_mass):
    ell_ins = input("Введите эллемент для добавления: ")
    ell_ins = int(ell_ins)
    dd_mass = [ell_ins]

    if dd_mass[0] > bin_mass[len(bin_mass)-1]:
        bin_mass = bin_mass + dd_mass
        print("Результат: ")
        print(bin_mass)
        return

    for i in range(len(mass2)):
        if ell_ins >= mass2[i] and ell_ins <= mass2[i+1]:
            dd = mass2[i+1]
            dd_indx = i+1
            break

    mid = len(bin_mass) // 2
    low = 0
    high = len(bin_mass) - 1

    while bin_mass[mid] != dd and low <= high:
```

```
        if dd > bin_mass[mid]:
            low = mid + 1
        else:
            high = mid - 1
        mid = (low + high) // 2

    else:
        print("Результат: ")
        bin_mass2 = bin_mass[dd_indx:]
        bin_mass1 = bin_mass[:dd_indx]
        bin_mass = bin_mass1 + dd_mass + bin_mass2
        print(sorted(bin_mass))



bin_poisk(mass2)
print("")
bin_poisk_del(mass2,bin_mass)
print("")
bin_poisk_ins(mass2,bin_mass)


def Interpolation_del(mass2,interpolation_mass):

    ell = input("Введите эллемент для удаления: ")
    ell = int(ell)

    low = 0
    high = (len(interpolation_mass) - 1)
    per_k = 0
    while low <= high and ell >= interpolation_mass[low] and ell <= interpolation_mass[high]:
        index = low + int(((float(high - low) / (interpolation_mass[high] - interpolation_mass[low]))
*(ell - interpolation_mass[low])))
        if interpolation_mass[index] == ell and per_k != 1:
            print("Результат: ")
            del interpolation_mass[index]
            print(interpolation_mass)
            per_k = 1
            break

        if interpolation_mass[index] < ell:
            low = index + 1;
        else:
            high = index - 1;

    if per_k == 0:
        print("Нет эллемента")


def Interpolation(mass2):
```

```
    global interpolation_mass
    interpolation_mass = copy.copy(mass2)
    print("ИНТУРПОЛЯЦИОННЫЙ ПОИСК ЭЛЛЕМЕНТА")
    print()
    ell = input("Введите эллемент для поиска: ")
    ell = int(ell)

    low = 0
    high = (len(interpolation_mass) - 1)
    per_k = 0
    while low <= high and ell >= interpolation_mass[low] and ell <= interpolation_mass[high]:
        index = low + int(((float(high - low) / (interpolation_mass[high] - interpolation_mass[low])) * (
                ell - interpolation_mass[low])))
        if interpolation_mass[index] == ell:
            print("Результат: число входит")
            per_k = 1
            break
        if interpolation_mass[index] < ell:
            low = index + 1;
        else:
            high = index - 1;

    if per_k == 0:
        print("Нет эллемента")


Interpolation(mass2)
print("")
Interpolation_del(mass2,interpolation_mass)
print("")
Interpolation_ins(mass2,interpolation_mass)


print("ПОИСК ФИБОНАЧЧИ")

def fib(n):
    if n == 0 or n ==1:
        return n
    k, l = 0, 1
    while n > 1:
        sum = k + l
        k = l
        l = sum
        n = n-1
    return sum

def fibSearch(arr, x):
    if x < arr[0]:
        return -1
    if x > arr[len(arr) - 1]:
```

```
        return -1
    if len(arr) == 1 and arr[0] != x:
        return -1
    n = 0
    index = -1
    while (fib(n)<len(arr)-1):
        n = n + 1
    for i in range (0, n+1):
        k = fib(i)
        if k >= len(arr):
            k = len(arr) - 1
        if x == arr[k]:
            return k
        if x < arr[k]:
            arr2 = []
            for j in range (fib(i-1), k):
                arr2.append(arr[j])
            m = fibSearch(arr2, x)
            if m == -1:
                index = -1
            else:
                index = fib(i - 1) + m
            break
    if index == -1:
        print("Нет такого числа")
    if index != -1:
        print("Есть такое число")

ell2 = input("Введите эллемент для поиска: ")
ell2 = int(ell2)

print(fibSearch(mass2,ell2))

def rehashfunction(key, mass, data, indx, data_mass):
    z = 0
    ell = key % len(mass)
    for i in range(len(data_mass)):
        if data_mass[i] == ell:
            z += 1

    if z != 0:
        key = (key + z)
        rehashfunction(key, mass, data, indx, data_mass)

    if z == 0:
        data[indx] = ell
        data_mass.append(ell)

print("Результат рехэширования: ")
data = dict.fromkeys(mass2)
```

```
data_mass = [None] * len(mass2)


for i in range(len(mass2)):
    indx = mass2[i]
    key = mass2[i]
    rehashfunction(key, mass2, data, indx, data_mass)

for key, value in data.items():
        print(value, key)


print("")

import random

print("Результат метода цепочек: ")

def hashCode(len, el):
    return el % len

def sReHash(len, el, j):
    return (hashCode(len,el)+j) % len

class LinkedList:
    def __init__(self, data):
        self.next = None
        self.data = data

    def append(self, val):
        end = LinkedList(val)
        n = self
        while n.next:
            n = n.next
        n.next = end

    def list(self):
        l = []
        n = self
        while n.next:
            l.append(n.data)
            n = n.next
        l.append(n.data)
        return l

n = len(mass2)
arr = mass2
d={}

for i in range (n):
```

```
    k = hashCode(len(arr), arr[i])
    if list(d.keys()).count(k) > 0:
        if type(d[k]) == LinkedList:
            d[k].append(arr[i])
        else:
            ll = LinkedList(d[k])
            ll.append(arr[i])
            d[k] = ll
    else:
        d[k] = arr[i]

for i in d.keys():
    c = str(i) + " "
    if type(d[i]) == LinkedList:
        c += str(d[i].list())
    else:
        c += str(d[i])
    print(c)


from random import randrange
from random import randint
import time


def genmass():
    global mass
    x = 8
    y = 8
    mass = []
    for i in range(y):
        mass2 = []
        for j in range(x):
            mass2.append(randint(0, 0))
        mass.append(mass2)

def randommass(mass):
    print("")
    for i in range(8):
        random_index = randrange(0, 8)
        mass[i][random_index] = 1
    print("Позиции ферзей: ")
    for i in range(8):
        print(mass[i])

def proverka(mass):
    global k
    n = 8
    x = [1, 2, 3, 4, 5, 6, 7, 8]
    y = []
```

```
    for i in range(8):
        for j in range(8):
            if mass[i][j] > 0:
                y.append(j)

    correct = True
    for i in range(n):
        for j in range(i + 1, n):
            if x[i] == x[j] or y[i] == y[j] or abs(x[i] - x[j]) == abs(y[i] - y[j]):
                correct = False

    if correct:
        k = 1
        print("Результат: Не бьют")
        print("Координаты ферзей: ")
        for i in range(8):
            print(x[i], y[i])
        mass = []
    else:
        k = 0
        print("Результат: Бьют")
        mass = []


x = 8
y = 8
mass = []
print("Исходный массив: ")
for i in range(y):
    mass2 = []
    for j in range(x):
        mass2.append(randint(0, 0))
    mass.append(mass2)

k = 0
start_time = time.time()
while k == 0:
    genmass()
    randommass(mass)
    proverka(mass)

print("Потраченное время(с): ")
sec1 = float('{:.6f}'.format(time.time() - start_time))
print(sec1)
```

Скриншоты работы программы

```
Введите x: 10
Введите min: -100
Введите max: 100

Результат:
[-45, 88, 30, -86, 68, -45, 31, -88, 36, -7]
Отсортированный результат:
[-88, -86, -45, -45, -7, 30, 31, 36, 68, 88]
```

Рис. 1 – Генератор матрицы

```
БИНАРНЫЙ ПОИСК ЭЛЛЕМЕНТА

Введите эллемент для поиска: 68
Есть такое число

Введите эллемент для удаления: 68
Результат:
[-88, -86, -45, -45, -7, 30, 31, 36, 88]

Введите эллемент для добавления: 68
Результат:
[-88, -86, -45, -45, -7, 30, 31, 36, 68, 88]
```

Рис. 2 – Результат бинарного поиска

```
Интерполяционный ПОИСК ЭЛЛЕМЕНТА
Введите эллемент для поиска: 68
Результат: число входит

Введите эллемент для удаления: 68
Результат:
[-88, -86, -45, -45, -7, 30, 31, 36, 88]

Введите эллемент для добавления: 68
[-88, -86, -45, -45, -7, 30, 31, 36, 68, 88]
```

Рис. 3 – Результат интерполяционного поиска

```
ПОИСК ФИБОНАЧЧИ
Введите эллемент для поиска: 68
Есть такое число
```

Рис. 4 – Результат поиска фибоначчи

```
Результат простого рехэширования:
2 -88
4 -86
6 -45
3 -7
0 30
1 31
7 36
8 68
9 88
```

Рис. 5 – Результат простого рехэширования

```
Результат метода цепочек:
2 -88
4 -86
5 [-45, -45]
3 -7
0 30
1 31
6 36
8 [68, 88]
```

Рис. 6 – Результат простого рехэширования методов цепочек

```
Координаты ферзей:
1 5
2 3
3 6
4 0
5 2
6 4
7 1
8 7
```

Рис. 7 – Результат задачи о 8 ферзях

Вывод

В ходе выполнения лабораторной работы, я научился реализовывать различные алгоритмы поиска. В ходе тестов было понятно, что скорость зависит от количества исходный элементов, но самыми быстрым является интерполяционный поиск.