ФГБОУ ВО “Чувашский государственный университет им. И. Н. Ульянова”
Факультет: ИВТ
Кафедра: Вычислительной техники
Предмет: Структуры и алгоритмы обработки данных

Лабораторная работа № 4.
ИССЛЕДОВАНИЕ МЕТОДОВ ПОИСКА

Вариант 7

Выполнил: студент группы ИВТ-41-20
Галкин Дмитрий Сергеевич
Проверил:
доцент
Павлов Леонид Александрович

Чебоксары
2022

Цель работы: Ознакомление с методами быстрого поиска; получение навыков программирования задач быстрого поиска.

Задание:

1) Исследовать алгоритмы поиска (успешный и безуспешный поиск) в соответствии с вариантами, заданными табл. 1.

Таблица 1

№ варианта	1	2	3	4	5	6	7	8	9	10	11
1	+	+	+	+				+		+	+
2	+	+			+	+			+	+	+
3	+	+		+			+	+		+	+
4	+	+				+	+		+	+	+
5	+	+	+				+	+		+	+
6	+	+			+		+		+	+	+
7	+	+	+		+			+		+	+
8	+	+	+			+			+	+	+
9	+	+		+	+			+		+	+
10	+	+		+		+			+	+	+
11	+	+				+		+	+	+	+
12	+	+		+				+	+	+	+

Номера столбцов соответствуют следующим алгоритмам поиска:

1. Последовательный поиск в упорядоченной таблице.
2. Обычный бинарный поиск.
3. Однородный бинарный поиск с вычислением значений δ.
4. Однородный бинарный поиск с дополнительной таблицей с вычисленными заранее значениями δ.
5. Поиск Фибоначчи с проверкой условия $i \le 0$.
6. Поиск Фибоначчи без проверки условия $i \le 0$.
7. Интерполяционный поиск.
8. Поиск в АВЛ-деревьях.
9. Поиск в красно-черных деревьях.
10. Цифровой поиск.
11. Поиск с хешированием.

2) Исследовать алгоритмы поиска, включения и исключения для динамических таблиц в соответствии с вариантами, заданными в табл. 2.

Таблица 2

№ варианта	1	2	3	4	5
1	+	+			
2	+		+		
3	+			+	
4	+				+
5		+	+		
6				+	+
7		+		+	
8			+		+

Номера столбцов соответствуют следующим способам организации динамических таблиц:

1. Обычное бинарное дерево поиска.
2. АВЛ-дерево.
3. RB-дерево.
4. АВЛ-дерево, в котором в структуру каждого узла добавлено поле FATHER для указания на отца данной вершины.
5. RB-дерево, в котором в структуру каждого узла добавлено поле FATHER для указания на отца данной вершины.

1) Исследовать алгоритмы поиска (успешный и безуспешный поиск)

- Поиск Фибоначчи с проверкой условия $i \leq 0$

```
function Fib (arr, n, x) {
  j := 1
 t := 0
 while (F(j) < n+1) {
    Inc(j)
   m := F(j+1) – 1 – n
   i := F(j)
   p := F(j-1)
   q := F(j-2)
   i := i – m
     while (p >= 1 and q >=0) {
       if (i <= 0) then {
         if (p == 1) then break;
       else {
         i += q;
         p -= q;
         q -= p;
       }
     }
    else if (x == arr[i]) then
      return i   //поиск успешный, значение найдено
    else
    {
      if (x > arr[i]) then
      {
        if (p == 1) break;
        i := i + q
        p := p – q
        q := q – p
      }
      else
      {
        if (x < arr[i]) then
        {
          if (q == 0) break;
          i := i – q
          t := p – q
```

```
            p := q
            q := t
          }
        }
      }
    }
    return -1   //поиск безуспешный
  }
```

- Цифровой поиск

q – очередь, с нужной нам последовательностью

t – дерево, в котором будем искать

```
function Digital(q) {
        m := t.root
        q := {q(1), q(2), … , q(n)}
        i := 1
        m := m.left
        while (q != Ø ) do
        {
                case(q([i]== m)
                {
                        delete q[i]
                        m := m.left
                        Inc(i)
                }
                case(q[i] != m)
                {
                        while (q[i] != m)  do
                                m := m.right
                        if (m == 0) then
                                return -1   //безуспешный поиск
                }
        }
        return 1   //успешный поиск
}
```

- Поиск с хешированием

h – хеш-функция
n – количество ячеек в таблице
M[i] – ячейки таблицы

```
function hash(x) {
        i := h(x)
        while M[i] != Ø do
        {
                if x == M[i] then
                        return i
                else
                        i := (i+1) mod m
                return -1 //если мы дошли до этой строчки, то поиск безуспешен
        }
}
```

2) Исследовать алгоритмы поиска, включения и исключения для динамических таблиц

- Обычное бинарное дерево поиска.

Поиск:

```
function Bin_Search(x) {
        t := root
        while t != λ do
        {
                case (x == t.info)
                        return x // поиск удачный, значение найдено
                case (x < t.info)
                        t := t.left
                case (x > t.info)
                        t := t.right
        }
        return -1 //неудачный поиск
}
```

Включение:

```
function add_elem(x, Tree){
t := Tree
if (t := λ ) then
{
new (t)
t.info := x
t.right := λ
t.left := λ
}
 else
```

```
{
case (x == t.info):
        return x
        case (x < t.info):
t.left := add_elem(x, t.left)
case (x > t.info):
        t.right := add_elem(x, t.right)
}
        add_elem := t
        return 1
}
```

Исключение:

функция, заменяющее одно поддерево, являющееся дочерним, другим поддеревом с корнем в узле u заменяется поддеревом с корнем в узле v

```
function del (T, u, v) {
        if (u.p) == Ø
                T.root = v
        else if (u == u.p.left)
                u.p.left = v
            else (u.p.right = v)
        if (v != Ø)
                v.p = u.p
}

main function Delete (T, x) {
if (x.left == Ø) //если у узла x нет левого дочернего узла
        del(T, x, x.right)
else
{
if (x.right == Ø) // если есть левый узел, но нет правого
                then del(T, x, x.left)
else
                // находим узел z следующий за x
z = min_key(x.right) //находим элемент с минимальным значением
       //ключа

        if (z.p != x) then
        {
del(t, z, z.right)//если z - не правый дочерний узел
                z.right = x.right
                z.right.p = z
}
        del(T, x, z) // если z - правый дочерний узел
        z.left = x.left
```

```
        z.left.p = z
}
}
```

СКРИНЫ:

Поиск	1000		2000		10000		40000		90000	
Успешно?	Да	Нет	Да	Нет	Да	Нет	Да	Нет	Да	Нет
Послед.	620	1002	1307	2008	6362	10003	25256	40008	56042	90006
Бинар.	8	9	9	10	12	13	14	15	15	16
ОБВЧ.	9	10	10	11	13	14	16	16	17	17
ФиПров.	74	75	183	194	551	481	1023	1247	3120	3102
АВЛ	9	9	11	12	13	14	15	16	17	18
Цифр.	2	1	1	1	2	2	2	2	2	2
Хеш.	1	1	1	1	1	1	1	1	1	1

Поиск	1000	2000	10000	40000	90000
АВЛ.	2	4	2	7	6
RB.	1	1	1	1	1

Вывод: в ходе данной лабораторной работы я ознакомился с методами быстрого поиска, а также получил практические навыки.