Изучение различных алгоритмов поиска и выборки

Тема работы: Изучение различных алгоритмов поиска и выборки

1 Теоретические сведения

1.1. Алгоритмы поиска и выборки

Поиск необходимой информации в списке — одна из фундаментальных задач теоретического программирования. При обсуждении алгоритмов поиска предполагается, что информация содержится в записях, составляющих некоторый список, который представляет собой массив данных в программе. Записи, или элементы списка, идут в массиве последовательно и между ними нет промежутков. Номера записей в списке идут от 1 до N — полного числа записей. Списки могут быть неотсортированными или отсортированными по значению ключевого поля. В неотсортированном списке порядок записей случаен, а в отсортированном они идут в порядке возрастания ключа.

Поиск нужной записи в неотсортированном списке сводится к просмотру всего списка до того, как запись будет найдена. Это простейший из алгоритмов поиска. Мы увидим, что этот алгоритм не очень эффективен, однако он работает на произвольном списке.

В отсортированном списке возможен также двоичный поиск. Двоичный поиск использует преимущества, предоставляемые имеющимся упорядочиванием, для того, чтобы отбрасывать за одно сравнение больше одного элемента. В результате поиск становится более эффективным.

С поиском конкретного значения связана задача выборки, в которой требуется найти элемент, удовлетворяющий некоторым условиям. Например, необходимо найти пятый по величине элемент, седьмой с конца или элемент со средним значением.

1.2. Последовательный поиск

При последовательном поиске предполагается, что список не отсортирован, поскольку некоторые алгоритмы на отсортированных списках показывают лучшую производительность. Обычно поиск

производится не просто для проверки того, что нужный элемент в списке имеется, но и для того, чтобы получить данные, относящиеся к этому значению ключа. Например, ключевое значение может быть номером сотрудника или порядковым номером, или любым другим уникальным идентификатором. После того, как нужный ключ найден, программа может, частично изменить связанные с ним данные или просто вывести всю запись. Предполагается, что элементы списка имеют номера от 1 до *N*. Это позволит возвращать 0 в случае, если целевой элемент отсутствует в списке.

Алгоритм последовательного поиска последовательно просматривает по одному элементу списка, начиная с первого, до тех пор, пока не найдет целевой элемент. Очевидно, что чем дальше в списке находится конкретное значение ключа, тем больше времени уйдет на его поиск.

Алгоритм последовательного поиска для общего случая в псевдокоде:

```
SequentialSearch(list.target,N)
  list        список для просмотра
  target      целевое значение
  N           число элементов в списке

for i=l to N do
   if   (target=list[i])
      return i
   end if
end for
return 0
```

1.3. Двоичный поиск

При сравнении целевого значения со средним элементом отсортированного списка возможен один из трех результатов: значения равны, целевое значение меньше элемента списка, либо целевое значение

больше элемента списка. В первом, и наилучшем, случае поиск завершен. В остальных двух случаях можно отбросить половину списка.

Когда целевое значение меньше среднего элемента, известно, что если оно имеется в списке, то находится перед этим средним элементом. Когда же оно больше среднего элемента, понятно, что если оно имеется в списке, то находится после этого среднего элемента. Этого достаточно, чтобы одним сравнением отбросить половину списка. При повторении этой процедуры можно отбросить половину оставшейся части списка.

Алгоритм двоичного поиска для общего случая в псевдокоде:

```
BinarySearch(list.target,N)
list      список для просмотра
target    целевое значение
N         число элементов в списке
start=1
end=N
while start<=end do
  middle=(start+end)/2
    select(Compare(list[middle].target)) from
       case -1:   start=middle+l
       case  0:   return middle
       case  1:   end=middle-l
    end select
end while
return 0
```

1.4. Выборка

Иногда нужно найти элемент из списка, обладающий некоторыми специальными свойствами, а не имеющий некоторое конкретное значение. Другими словами, вместо записи с некоторым конкретным значением поля необходимо найти, запись с наибольшим, наименьшим или средним значением этого поля. В более общем случае - запись с К-ым по величине значением поля.

Один из способов найти такую запись состоит в том, чтобы отсортировать список в порядке убывания; тогда запись с К-ым по величине значением окажется на К-ом месте. Распространенным ре-

шением является следующий подход: Находим наибольшее значение в списке и помещаем его в конец списка. Затем находим наибольшее значение в оставшейся части списка, исключая уже найденное. В результате получаем второе по величине значение списка, которое можно поместить на второе с конца место в списке. Повторив эту процедуру К раз, найдем К-ое по величине значение.

Алгоритм выборки для общего случая в псевдокоде:

```
FindKthLargest(list,K,N)
list      список для просмотра
N         число элементов в списке
K         порядковый номер по величине требуемого элемента
for i=I to K do
    largest=list[1]
    largestLocation=1
       for j=2 to N-(i-I)  do
           if list [j]>largest then
                 largest=list[j]
                largestLocation=j
           end if
       end for
    Swap(list[N-(i-I)],list[largestLocation])
end for
return largest
```

2 Практическая часть

2.1. Выполнение лабораторной работы

1. Согласно индивидуальному заданию составить два алгоритма поиска целевого значения в выборке. Первый алгоритм последовательного поиска, второй алгоритм двоичного.
2. Реализовать оба алгоритма на любом выбранном языке программирования. Реализация двоичного алгоритма с использованием рекурсии приветствуется.
3. Составить отчет в электронном виде.
4. Ответить на дополнительные вопросы преподавателя.

2.2. Варианты индивидуальных заданий

1. Сгенерировать список из n целых чисел, удалить из него все делящиеся на 3 и на 5, упорядочить оставшиеся элементы. Найти индекс числа, введенного пользователем или сообщить об его отсутствии.

2. Сгенерировать n строк длины от 1 до m (n и m вводятся с клавиатуры), выбрать все строки, длина которых превышает среднюю длину строк списка, упорядочить строки по длине, в полученной выборке найти строку, содержащую количество символов равных числу введенному пользователем (вывести строку, либо сообщение об отсутствии).

3. Сгенерировать случайным образом список из n (вводится с клавиатуры) элементов, сделать выборку с 2-го от начала списка по предпоследний, упорядочить значения, найти среди выбранных элементов введенное с клавиатуры число (вывести индекс, либо сообщение об отсутствии). Выборка должна выводиться предварительно на экран перед введением ключа.

4. Сгенерировать список из n целых чисел, выбрать из него все нечетные, не превышающее среднее значение элементов списка. В полученной последовательности найти индекс числа, введенного пользователем, или сообщить об отсутствии такого числа.

5. Сгенерировать случайным образом список из 20 элементов, сделать выборку с 3-го по величине по 10-ый по величине, упорядочить значения, найти среди выбранных элементов введенное с клавиатуры число (вывести индекс, либо сообщение об отсутствии). Выборка должна выводиться предварительно на экран, перед введением ключа.

6. Считать последовательность чисел, выбрать элемент, который меньше наибольшего на 5, если такой есть, либо выбрать наибольший. Выбрать второй элемент, который больше наименьшего на 5, если такой есть, либо выбрать наименьший элемент. Принять оба элемента за границы интервала поиска, упорядочить на этом интервале все элементы и найти введенное с клавиатуры число (вывести индекс, либо сообщение об отсутствии). Интервал должен выводиться предварительно на экран,

перед введением ключа.

7. Сгенерировать список из n простых чисел, выбрать из него все числа, не превышающие среднее значение элементов списка. Упорядочить выбранные элементы. Найти среди них индекс числа, введенного пользователем или сообщить об его отсутствии.

8. Сгенерировать случайным образом список из 100 элементов, выбрать из первой половины наибольшее, а из второй третье по величине, принять их за границы интервала для поиска, упорядочить на этом интервале все элементы и найти введенное с клавиатуры число (вывести индекс, либо сообщение об отсутствии). Интервал должен выводиться предварительно на экран, перед введением ключа.

9. Сгенерировать строку из n (вводится с клавиатуры) символов, сделать выборку с 2-го от конца списка по предпоследний, упорядочить символы по возрастанию согласно их значениям в ASCII, найти среди выбранных элементов символ, значение кода которого вводится с клавиатуры (вывести сам символ, либо сообщение об отсутствии). Выборка должна выводиться предварительно на экран, перед введением ключа.

10. Сгенерировать список из n целых положительных и отрицательных чисел, удалить из него все простые числа. Упорядочить оставшиеся элементы по возрастанию и вывести индекс числа, равного по модулю введенному пользователем.

11. Сгенерировать список из n целых чисел, не превышающих m. Отобрать из списка n/2 чисел, меньше среднего значения. В полученном списке найти индекс числа, введенного пользователем, или сообщить об отсутствии такого числа.

12. Сгенерировать последовательность из n натуральных чисел, выбрать из них все четные, упорядочить выбранные элементы по убыванию. В полученной последовательности найти введенное с клавиатуры число (вывести индекс, либо сообщение об отсутствии). Последовательность должна выводиться предварительно на экран, перед введением ключа.