Алгоритмы информационного поиска и сортировки

Задача поиска и ее разновидности

Задача поиска состоит в отыскании в последовательности элемента (или нескольких элементов) с заданными свойствами его значения.

A.Найти минимальное значение элементов последовательности (все элементы разные).

B. Найти номер минимального элемента последовательности (все элементы разные).

C. Найти минимальный элемент и его номер в последовательности с совпадающими элементами.

D. Найти номер элемента с заданным значением (все элементы разные).

Обозначим исследуемую последовательность x1, x2, x3, …, xn и составим алгоритмы решения каждой из предложенных задач.

А. Рассмотрим задачу А как задачу определения размера самого маленького яблока из лежащих в ящике, разделенном на ячейки.

Сделаем из мягкой проволоки рамку размером в любое произвольное яблоко. Таким образом, мы задались произвольным размером, который назовем эталоном.

Берем следующее яблоко и пытаемся протащить его через рамку.

Поиск будем проводить путем сравнения всех элементов последовательности с эталонной переменной, которой присвоим значение любого элемента последовательности.

Запишем пошаговую разработку алгоритма. Главное его свойство: все действия выполняются одинаково для всех элементов последовательности.

Значит, основой его будет следующий цикл.

1. Инициализация цикла.
2. Пока есть элементы делай.

Начало.

2.1.Сравнить эталон с очередным элементом последовательности.

2.2.Перейти к следующему элементу.

Конец.

Анализируем п.1. Нужно задать номер того элемента, с которого начнется сравнение, и определить эталон.
Если в качестве эталонной переменной MIN выберем x1, то можно начать цикл с элемента номер 2. Тогда
I.I:=2; MIN:=X[1];

Условие п.2. означает, что текущий номер элемента I не превысил заданную длину последовательности: I<N.

Шаг 2.1. – условный оператор. Его особенность – в отсутствии действий после иначе, поэтому можем записать укороченную форму оператора

2.1. IFX[I]<MIN

THEN MIN<X[I]; {Шаг 2.2. – в увеличении индекса I на 1}

2.2. I:=I+1;

Объединяя шаги детализации, получим алгоритм:

```
Ввод (последовательность X);
I:=2;MN:=X[1];
WHILE K=N DO
BEGIN IF X[I]<MIN
THEN MIN:=X[I]; I:=I+1;
 END;
Вывод (MN);
```

Алгоритм в задаче В имеет такую же структуру, что и в задаче А.

Процесс отыскания минимума и его номера можно совместить в одном цикле.

В алгоритме появится новая переменная NOM – номер минимального элемента. При этом на шаге инициализации цикла нужно не забыть присвоить NOM начальное значение 1 на случай, если выбранный в качестве эталона первый элемент и окажется минимальным.

Поэтому алгоритм будет иметь вид:

```
Ввод (последовательность X);
I:=2;MIN:=X[1];NOM:=1;
WHILE I <= N DO
BEGIN IF X[1]<MIN
   THEN
   BEGIN MIN:=X[I]; NOM=I; END;
   I:=I+1;
END;
Вывод (NOM).
```

С. Если стоит задача нахождения максимального элемента и его номера в последовательности с совпадающими значениями, то нужно оговорить особо, какой именно – первый или последний из одинаковых элементов считать искомым.

Алгоритм В решает задачу отыскания минимального (первого по порядку следования) элемента и его номера. Если мы хотим определить последний по порядку следования элемент и его номер, то в сравнении текущего элемента с эталоном должны изменить условие, а именно:

MIN>=X[I].

Тогда переприсваивание будет происходить и в случае равенства.

D. Пусть задана последовательность x1, x2, x3, ... ,xn и переменная P, которая называется ***поисковой***.
Известно, что все элементы последовательности имеют разные значения.
Требуется определить номер элемента, значение которого равно значению переменной P.

- Так как сравнение вещественных величин на равенство не имеет смысла, будем считать, что массив X и переменная P – целые (они могут быть и литерными).
- Одна из особенностей задачи состоит в том, что в последовательности может не оказаться элемента со значением P. Эту ситуацию нужно обязательно предусмотреть при конструировании алгоритма.

D1. *Неупорядоченная последовательность*

- Просматриваем последовательно все элементы (книги на полке) и сравниваем их с поисковым значением (автора и название книги – с требованием). Когда обнаружится искомый элемент (книга), запомним его номер (место книги на полке).
- Основой алгоритма является, таким образом, цикл.
- При инициализации цикла, как обычно, задаем исходное значение 1 индексу I элемента. Результат работы алгоритма – номер K найденного элемента.
- Вспомним, что наш алгоритм должен однозначно реагировать на ситуацию, когда искомого значения в последовательности нет. Резонно в этом случае считать K=0. Сделать это присваивание нужно до входа в цикл, иначе придется обременять алгоритм дополнительными проверками.

алгоритм

- Ввод (последовательность X);
- 1:=1; К:=0;
- WHILE I<=N DO
- BEGIN
- IF X[1]= P
- THEN
- BEGIN   K:= I; I:=N+1; END;
- ELSE I:=I+1
- END;
- Вывод (К).

D2. *Упорядоченная последовательность*

- Если последовательность упорядочена, то есть значения элементов возрастают (убывают) с увеличением номера элемента, то к ней можно применить другой алгоритм поиска. Договоримся рассматривать случай с возрастающей последовательностью.
- Идея алгоритма: выбираем элемент Xc из середины последовательности и сравниваем с поисковым значением P. Если он не равен P, то выясним, справа или слева от выбранного элемента находится искомый:
  - если Xc< P, то справа,
  - если Xc > P, то слева.
- Соответствующий промежуток снова делим пополам и так далее. То есть получим не что иное, как метод дихотомии.

алгоритм запишется следующим образом.

```
Ввод (последовательность X);
A:=1;B:=N;
WHILE A<B DO
    BEGIN
    C:=(A+B) DIV 2;
    IF X[C]<P
    THEN A:=C+1
    ELSE B:=C;
    END;
IFX[A]=P
THEN K:=A
ELSE K:=0;
Вывод (К).
```

Обобщение алгоритма на случай массива произвольной размерности

Так, алгоритм поиска минимального элемента и его номера в матрице:

```
Ввод (матрица A);
nom:= 1 ; nom2 := 1;
FOR I:=1 TO M DO
FOR J:=1TO N DO
     IF A[I,J]< min
     THEN
     BEGIN  min:= A[I,J];
       Nom1:= I;
       Nom2:= J;
     End;
Вывод (min, nom1, nom2).
```

Основные методы сортировки

- Под *сортировкой* понимают процесс перестановки элементов данного множества в определенном порядке. Цель сортировки – облегчить поиск элементов в упорядоченном множестве.
- Методы сортировки обычно разделяют на две группы: сортировка массивов и сортировка файлов (последовательных). Эти две группы часто называют внутренней и внешней сортировкой.
- Пусть даны элементы a_1 a_2, ...a_n. Сортировка означает перестановку этих элементов в таком порядке $a_{1к}$ $a_{2к}$, ...$a_{nк}$, что при заданной функции упорядочения f справедливо отношение f(aki)<f(ak2)<...<f(akN).

Основные методы сортировки

Обычно функция упорядочения не вычисляется по какому-то специальному правилу, а содержится в каждом элементе в виде явной компоненты (поля). Ее значение называется *ключом* элемента (key). Прочие компоненты – это все существенные данные об элементе.

С точки зрения же алгоритмов сортировки ключ – единственная существенная компонента, и нет необходимости определять остальные.

Простые методы сортировки массивов