Сортировка и множества

1

Что такое сортировка? (1)

Пусть даны элементы a_1, a_2,…, a_n и функция упорядочения $f(a_i)$, которая возвращает целое или вещественное число.
Сортировка означает перестановку этих элементов в таком порядке a_{k1}, a_{k2},..., a_{kn}, что $f(a_{k1}) \le f(a_{k2}) \le ... \le f(a_{kn})$ (сортировка по неубыванию) или $f(a_{k1}) \ge f(a_{k2}) \ge ... \ge f(a_{kn})$ (сортировка по невозрастанию).

Что можно сортировать?

- Числа: $f(a_i)=a_i$.
- Литеры: $f(a_i)=ord(a_i)$.
- Значения перечислимого типа: $f(a_i)=ord(a_i)$.
- Массивы и строки – использовать лексикографический порядок.
- Записи.

Мы будем сортировать массивы, в частности, массив a.

```
const n=…;
var a:array[1..n] of element;
```

2

Что такое сортировка? (2)

В качестве элементов массива рассматриваются любые данные, для которых можно ввести отношение порядка (естественное или какое-то особенное).

К алгоритмам сортировки предъявляется требование экономии памяти: т. е. *переупорядочивание элементов нужно выполнять на том же самом месте, не использовать вспомогательных массивов*.

3

Сортировка простыми включениями (1)

Этот метод обычно используют игроки в карты.

- Элементы (карты) условно разделяются на готовую последовательность a_1, a_2,..., a_{i-1} и входную последовательность a_i,..., a_n.
- На каждом шаге, начиная с i=2 и увеличивая i на 1, берут i-ый элемент входной последовательности и передают в главную последовательность, вставляя его на подходящее место.

Алгоритм

```
for i:=2 to n do
    begin
    x:=a[i];
    {вставить x на подходящее место в a[1],a[2],...,a[i-1]
     или оставить на месте}
    end
```

4

Сортировка простыми включениями (2)

В следующем примере каждой строке соответствует состояние массива после очередного шага цикла (выделен выбираемый элемент):

44 ***55*** 12 42 94 18 06 67
44 55 ***12*** 42 94 18 06 67
12 44 55 ***42*** 94 18 06 67
12 42 44 55 ***94*** 18 06 67
12 42 44 55 94 ***18*** 06 67
12 18 42 44 55 94 ***06*** 67
06 12 18 42 44 55 94 ***67***
06 12 18 42 44 55 67 94

Для этого алгоритма число сравнений в среднем равно $n^2/4$ (в худшем случае равно $n^2/2$). Число присваиваний (пересылок) – такое же.

5

Сортировка простым выбором (1)

Метод:

- Выбирается наименьший элемент.
- Он меняется с первым элементом *a*[1].
- Эти операции повторяются с оставшимися *n*–1 элементами, потом с *n*–2 элементами, пока не останется только один наибольший элемент.

В следующем примере каждой строке соответствует состояние массива после очередного шага цикла (выделен выбираемый элемент):

44 55 12 42 94 18 ***06*** 67
06 55 ***12*** 42 94 18 44 67
06 12 55 42 94 ***18*** 44 67
06 12 18 ***42*** 94 55 44 67
06 12 18 42 94 55 ***44*** 67
06 12 18 42 44 ***55*** 94 67
06 12 18 42 44 55 94 ***67***
06 12 18 42 44 55 67 94

6

Сортировка простым выбором (2)

Алгоритм:

```
for i:=1 to n-1 do
  begin
  {присвоить k индекс наименьшего элемента
   из a[i],a[i+1],...,a[n]}
  {поменять местами a[i] и a[k]}
  end
```

Для этого алгоритма число пересылок в среднем равно $n \ln n$, наихудшее – $n^2/4$.

Число сравнений равно $(n^2-n)/2$.

7

Сортировка простым выбором (3)

Задача. Дана вещественнозначная матрица *a* с *n* строками и *m* столбцами. Определим для каждой строки *ai* функцию $f(a_i) = a_{i1} - a_{i2} + a_{i3} - \ldots + (-1)^{m+1} a_{im}$. Переставить строки матрицы *a* по неубыванию значений функции *f*.

```
const n=3; m=4;
type
   element = array[1..m] of real;
   mas = array[1..n] of element;

function f(s:element):real;
var
   i:integer;
   y:real;
begin
   y:=0;
   for i:=1 to m do
       if i mod 2 = 0 then y:=y-s[i]
                      else y:=y+s[i];
   f:=y
end;
```

8

Сортировка простым выбором (4)

```
procedure sort(var a:mas);
var i,j,k:integer;  x:element;
begin
   for i:=1 to n-1 do
       begin
       k:=i;
       x:=a[i];
       for j:=i+1 to n do
           if f(a[j])< f(x) then begin k:=j; x:=a[j] end;
       a[k]:=a[i];
       a[i]:=x
       end;
end;

var   a:mas;
…
begin
{ввод матрицы a}
sort(a);
{печать матрицы a}
end.
```

9

Сортировка простым обменом («пузырек») (1)

Метод:

Будем представлять массив в виде столбца элементов – первый элемент массива является самым верхним.

Сравниваем и обмениваем два соседних элемента, до тех пор, пока не будут рассортированы все элементы. На каждом шаге, двигаясь снизу вверх, меняем все подходящие пары. «Легкие» элементы «всплывают» вверх после каждого прохода.

44	***06***	06	06	06	06
55	44	***12***	12	12	12
12	55	44	***18***	18	18
42	12	55	44	***42***	42
94	42	18	55	44	44
18	94	42	42	55	55
06	18	94	67	67	67
67	67	67	94	94	94

10

Сортировка простым обменом («пузырек») (2)

Алгоритм:

```
for i:=2 to n do
for j:=n downto i do
 {Если a[j] и a[j-1] не в нужном порядке, то их меняем местами}
```

Число перемещений в среднем равно $3n^2/4$, в худшем случае – $3n^2/2$. Число сравнений равно $(n^2–n)/2$.

11

Определение множественного типа (1)

Чтобы ввести в язык Паскаль вычислительную структуру множеств, используют **множественный** тип.

Значения множественного типа, так же, как и массивы, строятся из нескольких значений одного (базового) типа.

Однако в отличие от массивов, значение множественного типа может содержать *любое* количество *различных* элементов базового типа – от нуля элементов (пустое множество) до всех возможных значений базового типа.

Иными словами, *возможными значениями переменных множественного типа являются все подмножества значений базового типа*.

12

Определение множественного типа (2)

Множественный тип задается с помощью двух служебных слов – set и of – и следующего за ним базового типа.

Например:

```
type
digits = set of 1..5;
var
s:digits;
```

Переменная s в качестве значений может принимать следующие множества целых чисел:

пустое множество, {1}, …, {5}, {1,2}, …, {4,5}, {1,3,5}, …, {4,5,3,2}, …, {1,2,3,4,5} (всего 32 различных множества).

Обратите внимание:

- *Все значения базового типа в множестве должны быть различны.*
- *Порядок «расположения» элементов в множестве никак не фиксируется.*

13

Определение множественного типа (3)

Каким может быть базовый тип множества?

- символьный,
- перечислимый,
- ограниченный.

Базовый тип должен содержать **не более 256 значений.** Если базовый тип – ограниченный целый, то значения элементов множеств должны быть в диапазоне от 0 до 255.

Пример:

```
var s:set of char;
```

Переменная v принимает в качестве значения множество символов. Различных таких множеств всего 2^{256} (приблизительно 10^{77}).

14

Определение множественного типа (4)

В Паскале допускаются явные изображения значений множественного типа:

- пустое множество изображается [],
- в общем случае изображение строится из списка элементов множества, разделенных запятыми, и весь список заключается в квадратные скобки.

Например:

[1,2,5]
['r','y','+']

В качестве элементов в изображении множества допускаются выражения, тип которых должен совпадать с базовым типом. Кроме того, можно указывать диапазоны значений.

Так, например, следующие два множества равны: [1,3..5] и [1,3,4,5].

Следующие изображения представляют одно и то же множество:

[1..3],[1,2,3],[1,2,3,1],[3,3,1,1,2,2,2,3].

15

Определение множественного типа (5)

Примеры:

```
type
Setofchar = set of char;
digits = set of 0..100;
var
  mychars: setofchar;
  mydig1,mydig2:digits;
  x,y:0..20;
.....
mychars:=['a'.. 'z','"'..'9','_'];
mydig1:=[];
mydig2:=mydig1;
mydig1:=[x..x+10,0,y-1,y+1];
```

16

Операции с множествами (1)

Множества языка Паскаль обладают свойствами математических множеств. В частности, над ними можно выполнять те же операции.

Объединение множеств - бинарная, коммутативная и ассоциативная операция.
C:=A+B;
Пересечение множеств - бинарная, коммутативная и ассоциативная операция.
C:=A*B;
Разность множеств - бинарная операция.
C:=A-B;

Примеры:
[1,3]+[2,4] = [1..4]
[1..10]*[5..15] = [5..10]
[1,2]*[3,4] = []
[1..10] - [5..15] = [1..4]

17

Операции с множествами (2)

Как добавить элемент во множество?
Для этого можно использовать объединение множеств. Пример:
mydig1:=mydig1+[5];

Проверка принадлежности множеству
x in A (x принадлежит A?) - бинарная булевская операция:

2 in [1..10,21] = true,
5 in [1,2,x,10] = true тогда и только тогда, когда x=5.

Использование постоянных множеств для проверок:
(ch='a') or (ch='b') or (ch='x') or (ch='y')
эквивалентно
ch in ['a','b','x','y']

('0'<c) and (c<'9') эквивалентно c in ['0'..'9'].

18

Операции с множествами (3)

Равенство множеств - бинарная булевская операция **A=B**.
Неравенство множеств - бинарная булевская операция **A<>B**.
Множество A является частью (подмножеством) множества B - бинарная булевская операция **A<=B** (**B>=A**).
Операции < и > для множеств не используются.
Примеры:
Значение выражения **[1,2,3]=[1,2]** равно **false**.
Значение выражения **[1,2,3]>=[1,2]** равно **true**.
Значение выражения **[s]<=[1..10]** равно **true**, т. и т. т., когда 1<=s<=10.

19

Операции с множествами (4)

Задача 1. *Состоят ли строки s1 и s2 из одних и тех же символов?*

```
a1:=[];a2:=[];
for j:=1 to length(s1) do
    a1:=a1+[s1[i]];
for j:=1 to length(s2) do
    a2:=a2+[s2[i]]
writeln(a[1]=a[2]);
```

Задача 2. Используя решето Эратосфена, найти простые числа, меньшие 255.

20

Операции с множествами (5)

Программа для решета Эратосфена

```
const n=255; {количество перебираемых натуральных  чисел}
Var    S,  {исходное множество}
      Primes: set of 2..n; {результирующее множество}
      next,j:integer;
begin
   S:=[2..n]; {все числа в заданном диапазоне}
   Primes:=[];  next:=2;  {начинаем с минимального простого числа}
   repeat
     {поиск очередного простого числа}
     while not (next in S) do
                next:=next+1;  {ищем в S наименьшее число}
     Primes:=Primes+[next]; {помещаем его в Primes}
     j:=next;
     while j<=n do {удаляем из S все числа, кратные next}
                 begin S:=S-[j]; j:=j+next  end;
   until S=[]; {повторяем цикл до исчезновения S}
   writeln('Простые числа < 256:');
   for j:=2 to n do
               if j in Primes then write(j:5)
end.
```

21