Лекция по джаве для кт

1 Предисловие

Целью данной лекции является понимание особенностей поиска максимума/минимума в последовательностях и в массивах.

Для упрощения терминологии мы, в основном, будем говорить о поиске в "массивах". Хотя, в действительности, это также относится и к последовательностям.

Также, в большинстве случаев, будем использовать термин "максимум хотя, в действительности, все сказанное будет использоваться также и для поиска минимума (возможно, с изменением знака).

Под максимумом имеется ввиду "наибольшее значение элемента".

Под минимумом имеется ввиду "наименьшее значение элемента".

Под номером максимума имеется ввиду "номер (индекс/положение в массиве) элемента, имеющего наибольшее значение". Будем считать, что нумерация элементов массива начинается с единицы.

Под номером минимума имеется ввиду "номер (индекс/положение в массиве) элемента, имеющего наименьшее значение".

Под первым элементом массива имеется ввиду элемент с самым маленьким номером (то есть, с номером 0).

Под вторым элементом массива имеется ввиду элемент, расположенный после первого (то есть, с номером 1).

2 Изучаемая задача. Простейший случай

В данной лекции рассматривается задача поиска максимума в массиве из n элементов.

В самом простом случае нужно найти значение наибольшего элемента массива.

Это и будет называться поиск максимума.

"Самый простой случай это когда не задано никаких условий/ограничений на поиск нужного элемента.

В этом случае для поиска максимума достаточно одной дополнительной переменной. Например, max.

В качестве начального значения этой переменной нужно взять первый элемент массива.

После этого нужно перебрать все остальные элементы (то есть, со второго до последнего). Каждый элемент нужно сравнивать с max. Если он окажется лучше (больше) max, то нужно поменять max.

```
max = a[0];
for (int i = 1 ; i < n ; i++)
  if (a[i] > max)
    max = a[i];
```

3 Поиск номера максимума. Первый/последний

При поиске номера максимума возникает понятие первого/последнего максимума.

Эта особенность возникает в том случае, если в массиве/последовательности имеется несколько максимумов. То есть, несколько элементов, которые имеют такое же значение, как и наибольший элемент.

Этой особенности не возникает, если нужно найти просто значение максимума. Потому что для всех этих элементов это значение одинаково.

Однако, когда нужно найти номер максимума, возникает дополнительный вопрос – номер какого из этих одинаковых элементов нам требуется найти?

Как правило, требуется найти номер первого максимума. То есть, того максимума, который имеет наименьший индекс.

Поиск первого максимума – самый простой. О нем не нужно заботиться как-то особенно. Первый максимум получится "сам собой". Потому что при сравнении очередного элемента с текущим максимумом используется знак "строго больше". (при поиске минимума – "строго меньше"). То есть, при нахождении очередного элемента, который имеет такое же значение, как max, условие "строго больше"не выполнится и поэтому max и номер максимума не изменится.

Если же нужно найти номер последнего максимума, нужно как раз при нахождении каждого очередного элемента, равного max, запоминать номер этого элемента. Это значит, что условие изменения max должно быть не "строго больше a "больше или равно".

4 Что хранить для номера в массиве

При поиске номера максимума в массиве (не в последовательности), если не задано ограничение (условие) на искомые элементы, можно использовать только одну дополнительную переменную – номер максимума (например, imax).

Это возможно потому, что зная номер элемента массива, можно быстро получить значение этого элемента (просто написав имя массива и, в квадратных скобках, номер элемента).

То есть, при поиске номера максимума достаточно задать начальное значение номера, равное нулю (номер первого элемента).

При сравнении очередного элемента массива с максимумом, нужно сравнивать его с элементом номер imax.

А при изменении максимума – в переменную imax класть номер текущего элемента (счетчик цикла).

Заметим, что при поиске номера максимума в последовательности или с условием, такое решение не работает.

Пример реализации

```
imax = 0;
for (int i = 1 ; i < n ; i++)
    if (a[i] > a[imax])
        imax = i;
```

5 Поиск номера в последовательности

При поиске номера максимума в последовательности (не в массиве), нужно хранить как значение максимума, так и значение номера максимума.

При изменении одной из этих переменных, нужно тут же изменять и вторую переменную.

То есть, перед началом поиска нужно задать начальное значение максимума, равное первому элементу, а значение номера максимума – равное единице.

Далее в цикле, во второго до последнего элемента, нужно сравнивать очередной элемент со значением максимума, и, если очередной элемент больше (лучше) максимума, запоминать его номер в переменной-максимуме, а его номер (текущее значение счетчика цикла) – в переменной-номере максимума.

Пример реализации:

```
max = in.nextInt();
imax = 0;
for (int i = 1 ; i < n ; i++) {
    a = in.nextInt();
    if (a > imax) {
        max = a;
        imax = i;
    }
}
```

6 Максимум с условием

Иногда при поиске максимума на значение искомого элемента накладывается дополнительное условие. Например, "найти наибольший четный элемент".

В этом случае существенно меняется принцип поиска.

Первое. Начальное значение. Потому что в этом случае нельзя в качестве начального значения максимума брать первый элемент. (Ведь он может не удовлетворять условию, и при этом быть больше, чем любой элемент, который удовлетворяет условию. В этом случае результатом поиска будет этот первый элемент. Что неверно).

Таким образом, возникает проблема: что же взять в качестве начального значения максимума!?

Как правило, в такой задаче нам известно ограничение на значения элементов. Например, "не превышают по модулю число 1000".

В этом случае в качестве начального значения максимума нужно взять значение, выходящее за диапазон допустимых значений элементов.

То есть, для заданного примера при поиске максимума нужно взять число -1001. А при поиске минимума – число 1001.

Второе. Границы цикла. Так как теперь значение первого элемента не попадает в максимум, нужно начинать перебор элементов в цикле не со второго элемента, а с первого!

Третье. Наличие максимума. После окончания поиска, возможно, что требуемого элемента вообще нет. (Например, требуется найти наибольший четный, а все элементы – нечетные).

Это значит, что после окончания цикла необходимо проверить, нашли ли мы хотя бы один нужный элемент.

Для этого достаточно сравнить максимум с тем значением, которое мы положили в него перед циклом. Если это значение по-прежнему лежит в максимуме, то нужного элемента нет.

Пример реализации:

```
 max = -1001;
for (int i = 0 ; i < n ; i++)
    if (a[i] > max)
        max = a[i];
if (max == -1001)
    out.println("          ");
else
    out.println(max);
```

7 Поиск с условием в произвольном массиве

В редком случае, нужно будет найти максимум, удовлетворяющий какому-нибудь условию, когда не известно ограничение на диапазон значений элементов.

В этом случае проблема начального значения максимума оказывается более сложной.

Решение – сначала найти первый элемент, который удовлетворяет условию поиска.

Если такой элемент есть – взять его значение за начальное значение максимума, и, начиная со следующего элемента, сравнивать элементы, подходящие под условие поиска, с максимумом, как и в "обычном"поиске максимума с условием (см. предыдущий раздел).

А если такого элемента нет, то его и максимума такого нет.

Самая короткая запись такого решения – использовать переменную-номер максимума и как признак того, что мы еще не нашли ни одного нужного элемента, и для хранения номера нужного максимума.

Для этого зададим начальное значение этой переменной, равное -1. Это будет признаком того, что мы еще не нашли ни одного нужного элемента. Переберем в цикле все элементы, от первого до последнего. Если текущий элемент удовлетворяет условию, то, если номер максимума равен -1, или

если текущий элемент больше максимального элемента, поменяем номер максимума на номер текущего элемента.

После цикла сравним номер максимума с -1, чтобы понять, нашелся ли нужный максимум.

Пример реализации

```
imax = 0;
for (int i = 0 ; i < n ; i++)
    if ( a[i] "search condition" )
        if ( imax == -1 || a[i] > a[imax] )
            imax = i;
if ( imax == -1 )
    out.println("NO");
else
    out.println(imax);
```