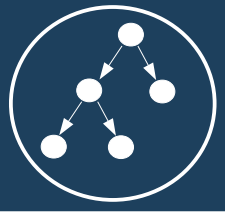

Data Structures and Algorithms

Алгоритмы.
Поразрядная сортировка

Описание сути алгоритма

Поразрядная сортировка — алгоритм сортировки не использующий сравнение элементов между собой. Предназначен для сортировки данных ключи которых можно представить в виде последовательности «разрядов» каждому из которых можно сопоставить целое число. Например это могут быть целые числа (разряды записаны явно), строки (каждый символ это разряд). Алгоритм сводится к повторению алгоритма сортировки распределяющим подсчетом для каждого разряда. Важным моментом является способ «выравнивания» ключей сортировки, т. е. как сравнивать ключи которые имеют разное количество разрядов.

В зависимости от того как выполнять выравнивание ключей сортировки поразрядная сортировка делится на:

- LSD (least significant digit) — выравнивание по младшему разряду.
- MSD (most significant digit) — выравнивание со старшему разряду.

123	Hello
1234	Hello world
LSD	**MSD**

Сведение о алгоритме

Сложность по времени в наихудшем случае O(n)

Требуется (n + диапазон ключей) дополнительной памяти

Описание алгоритма

1) Определяем максимальное количество разрядов в ключах сортировки (в дальнейшем k). Определяем способ выравнивания ключей сортировки (LSD или MSD).
2) Выполняем последовательно k раз сортировку используя алгоритм распределяющего подсчета, где в качестве ключа сортировки используется значение соответствующего разряда.

Графическое пояснение алгоритма на примере LSD выравнивания для целых положительных чисел

[121, 5, 24, 9, 32] → LSD выравнивание → [121, 005, 024, 009, 032]

[121, 005, 024, 009, 032] → [121, 032, 024, 005, 009]

[121, 032, 024, 005, 009] → [005, 009, 121, 024, 032]

[005, 009, 121, 024, 032] → [005, 009, 024, 032, 121]

Python

Реализация алгоритма на Python

Вспомогательные функции для работы с разрядами целых чисел

```
def get_number_of_digits(number):
    i = 1
    while(number >= 10**i):
        i = i+1
    return i


def get_digit(number, i):
    return number % (10**(i+1))//(10**i)


def max_number_of_digits(numbers):
    number_of_digits = 1
    for number in numbers:
        current_digits = get_number_of_digits(number)
        if current_digits > number_of_digits:
            number_of_digits = current_digits
    return number_of_digits
```

Python

Функция для сортировки распределяющим подсчетом ключ соответствующий разряд в числе

```
def counting_sort(sequince, position):
    min_key = min([get_digit(x, position) for x in sequince])
    max_key = max([get_digit(x, position) for x in sequince])
    n = max_key - min_key + 1
    support = [0 for i in range(n)]
    for element in sequince:
        support[get_digit(element, position)-min_key] += 1
    size = len(sequince)
    for i in range(n-1, -1, -1):
        size -= support[i]
        support[i] = size
    result = [None for i in range(len(sequince))]
    for element in sequince:
        result[support[get_digit(element, position)-min_key]] = element
        support[get_digit(element, position)-min_key] += 1
    return result
```

Функция для поразрядной сортировки

```
def radix_sort(sequince):
    number_of_digits = max_number_of_digits(sequince)
    for i in range(number_of_digits):
        sequince = counting_sort(sequince, i)
    return sequince
```

Java

Реализация алгоритма на Java

Методы для работы с разрядами чисел

```
public static int numberOfDigits(int number) {
     int i = 1;
     long n = 10;
     for (; number >= n;) {
          i++;
          n *= 10;
     }
     return i;
}

public static int findMaxNumberOfDigits(int[] numbers) {
     int result = 1;
     for (int i = 0; i < numbers.length; i++) {
          int digits = numberOfDigits(numbers[i]);
          if (digits > result) {
               result = digits;
          }
     }
     return result;
}

public static int getDigit(int number, int devider) {
     return number % (devider * 10) / (devider);
}
```

Методы для сортировки распределяющим подсчетом

```java
public static int[] findMinMaxKey(int[] numbers, int devider) {
        int minKey = getDigit(numbers[0], devider);
        int maxKey = minKey;
        for (int number : numbers) {
                int digit = getDigit(number, devider);
                if (digit < minKey) {
                        minKey = digit;
                }
                if (digit > maxKey) {
                        maxKey = digit;
                }
        }
        return new int[] { minKey, maxKey };
}

public static int[] countSort(int[] numbers, int devider) {
        int[] minMaxKey = findMinMaxKey(numbers, devider);
        int minKey = minMaxKey[0];
        int maxKey = minMaxKey[1];
        int n = maxKey - minKey + 1;
        int[] support = new int[n];
        for (int number : numbers) {
                support[getDigit(number, devider) - minKey] += 1;
        }
        int size = numbers.length;
        for (int i = support.length - 1; i >= 0; i--) {
                size -= support[i];
                support[i] = size;
        }
        int[] result = new int[numbers.length];
        for (int number : numbers) {
                result[support[getDigit(number, devider) - minKey]] = number;
                support[getDigit(number, devider) - minKey] += 1;
        }
        return result;
}
```

Метод поразрядной сортировки

```
public static int[] radixSort(int[] numbers) {
     int maxNumberOfDigits = findMaxNumberOfDigits(numbers);
     int devider = 1;
     for (int i = 0; i < maxNumberOfDigits; i++) {
          numbers = countSort(numbers, devider);
          devider *= 10;
     }
     return numbers;
}
```

Fortran

Реализация алгоритма на Fortran

Fortran

Функции для работы с разрядами числа

```
function number_of_digits(c_number)
    implicit none
    integer(4),intent(in)::c_number
    integer(4)::number_of_digits
    integer(8)::pow
    pow = 10
    number_of_digits = 1
    do
        if(pow>c_number) then
            exit
        end if
        number_of_digits = number_of_digits + 1
        pow = pow * 10
    end do
end function number_of_digits

function get_digit(c_number,position)
    implicit none
    integer(4),intent(in)::c_number,position
    integer(4)::get_digit
    get_digit = mod(c_number,10**(position))/(10**(position-1))
end function get_digit

function get_max_number_of_digit(numbers) result(max_numbers)
    implicit none
    integer(4), dimension(:),intent(in)::numbers
    integer(4)::max_numbers,i
    max_numbers = 1
    do i = 1, size(numbers)
        if(number_of_digits(numbers(i))>max_numbers) then
            max_numbers = number_of_digits(numbers(i))
        end if
    end do
end function get_max_number_of_digit
```

Функции для поиска минимального и максимального ключа

```
 function find_min_max_keys(numbers,position) result(min_max)
    implicit none
    integer(4),dimension(:),intent(in)::numbers
    integer(4),intent(in)::position
    integer(4),dimension(2)::min_max
    integer(4)::i, min_key, max_key,temp_key
    min_key = get_digit(numbers(1),position)
    max_key = min_key
    do i = 2,size(numbers)
        temp_key = get_digit(numbers(i),position)
        if (temp_key < min_key) then
            min_key = temp_key
        end if
        if (temp_key > max_key) then
            max_key = temp_key
        end if
    end do
    min_max = [min_key, max_key]
end function find_min_max_keys
```

Fortran

Функция для сортировки распределяющим подсчетом на основании разряда числа

```
 subroutine counting_sort(numbers,position)
    implicit none
    integer(4),dimension(:),intent(inout)::numbers
    integer(4),intent(in)::position
    integer(4),dimension(size(numbers))::temp_array
    integer(4)::i, min_key, max_key,n,c_size,c_index
    integer(4), dimension(2)::min_max
    integer(4),dimension(:),allocatable::support
    min_max = find_min_max_keys(numbers,position)
    min_key = min_max(1)
    max_key = min_max(2)
    n = max_key - min_key + 1
    allocate(support(n))
    do i = 1,size(numbers)
        c_index = get_digit(numbers(i),position) - min_key + 1
        support(c_index) = support(c_index) + 1
    end do
    c_size = size(numbers) + 1
    do i = n, 1,-1
        c_size = c_size - support(i)
        support(i) = c_size
    end do
    do i = 1, size(numbers)
        c_index = get_digit(numbers(i),position) - min_key + 1
        temp_array(support(c_index)) = numbers(i)
        support(c_index) = support(c_index) + 1
    end do
    deallocate(support)
    do i = 1,size(numbers)
        numbers(i)=temp_array(i)
    end do
end subroutine counting_sort
```

Процедура для поразрядной сортировки

```
    subroutine radix_sort(numbers)
        implicit none
        integer(4),dimension(:),intent(inout)::numbers
        integer(4)::m_number_of_digits,i
        m_number_of_digits = get_max_number_of_digit(numbers)
        do i = 1,m_number_of_digits
            call counting_sort(numbers,i)
        end do
    end subroutine radix_sort
```

Список литературы

1) Роберт Седжвик, Кевин Уэйн «Алгоритмы на java 4-е издание» Пер. с англ. - М. : ООО "И.Д. Вильямс", 2013. ISBN 978-5-8459-1781-2.
2) Дональд Кнут. Искусство программирования, том 3. Сортировка и поиск — 2-е изд. — М.: «Вильямс», 2007. — ISBN 5-8459-0082-4.