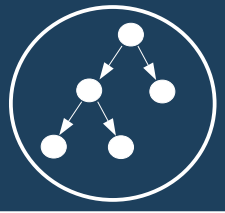


Data Structures and Algorithms

Алгоритмы.
Сортировка распределяющим подсчетом

Описание сути алгоритма

Сортировка распределяющим подсчетом является разновидностью сортировки подсчетом. Используется для сортировки массивов данных, ключи сортировки которых представимы целыми числами и их значения лежат в относительно узком диапазоне. Например, если нужно отсортировать массив котов по возрасту, то этот алгоритм покажет очень высокое быстродействие. Важным и положительным моментом является то, что это устойчивый алгоритм сортировки.

Сведение о алгоритме

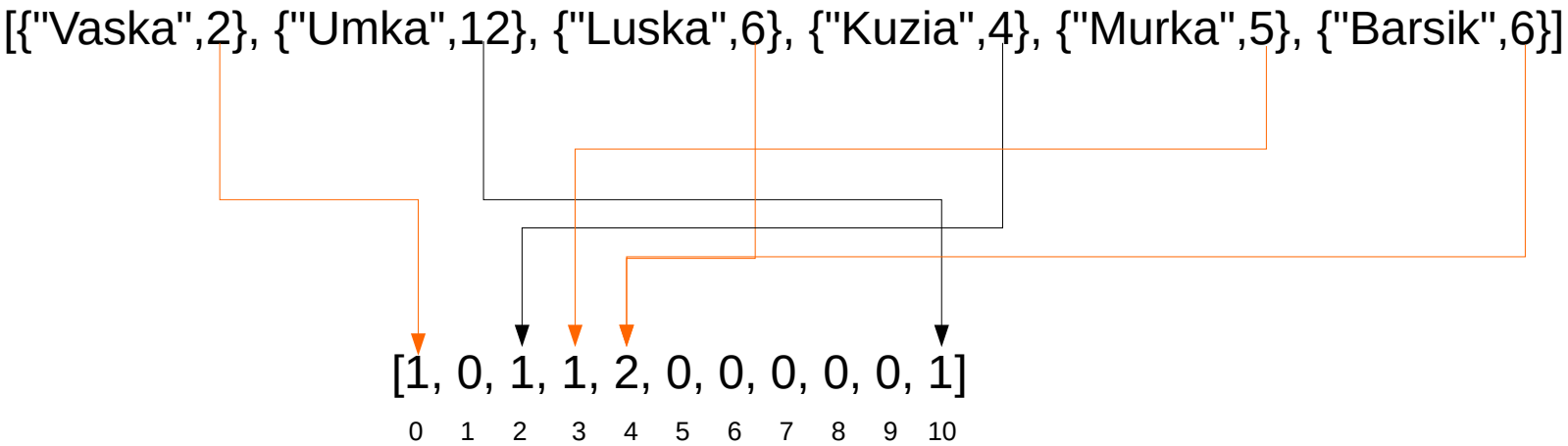
Сложность по времени в наихудшем случае O(n)

Требуется (n + диапазон ключей) дополнительной памяти

Описание алгоритма

1) Определяем минимальное и максимальное значение ключей сортировки в сортируемой последовательности(в дальнейшем sort) (обозначим их как min и max соответственно). Объявляем вспомогательную последовательность (в дальнейшем support) длинна которой вычисляется как max-min+1. Заполняем ее нулями.
2) Выполняем проход по sort, добавляем единицу к значению support[element-min] где element это значение ключа сортировки текущего элемента в sort
3) Объявляем дополнительную переменную size, значение которой равно длине сортируемой последовательности. Выполняем проход по вспомогательной последовательности в обратном порядке. Устанавливаем текущий элемент равный разности size и текущего элемента. Уменьшаем size на значение текущего элемента.
4) Создаем последовательность размер которой совпадает с размером сортируемой последовательности.
5) Выполняем проход по сортируемой последовательности размещаем текущий элемент на индекс вычисляемый как support[element-min]. Увеличиваем значение в support[element-min] на единицу.

Вычисление счетчика повторений



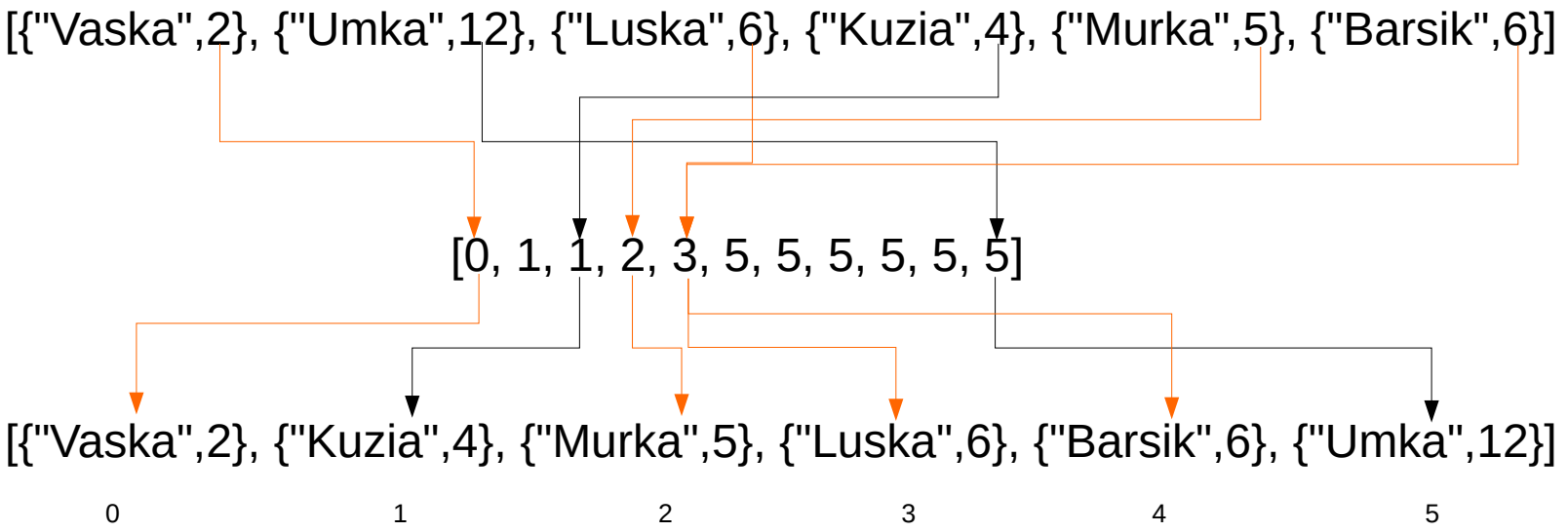
min = 2
max = 12
Длинна вспомогательной последовательности max-min+1 = 12-2+1 = 11

Для элемента основной последовательности вычисляется соответствующий ему индекс во вспомогательной последовательности. Для вычисления используется зависимость вида index = ключ-min. Так, например для 5 получим 5-2=3. После чего увеличиваем значение элемента во вспомогательной последовательности на этом индексе на единицу.

Преобразование счетчиков в индексы

Длинна сортируемой последовательности равна 6 устанавливаем n = 6. Выполняем обратный проход устанавливая значение текущего элемента как разность n и текущего элемента. При этом устанавливаем n равный текущему элементу.

Размещение элементов



При размещении элементов выполняется проход по сортируемой последовательности текущий элемент устанавливается в позицию равную значению во вспомогательной последовательности по индексу [key-min]. При этом значение во вспомогательном индексе увеличивается на единицу.

Python

Реализация алгоритма на Python

Тестовый класс для демонстрации

```
class Cat:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return "Cat[name = {}, age = {}]".format(self.name, self.age)
```

Функция для сортировки распределяющим подсчетом

```
def get_min_max_key(sequince):
    min_key = min(sequince, key=lambda c: c.age).age
    max_key = max(sequince, key=lambda c: c.age).age
    return(min_key, max_key)


def counting_sort(sequince):
    min_max = get_min_max_key(sequince)
    min_key = min_max[0]
    max_key = min_max[1]
    n = max_key - min_key + 1
    support = [0 for i in range(n)]
    for element in sequince:
        support[element.age-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[element.age-min_key]] = element
        support[element.age-min_key] += 1
    return result
```

Java

Реализация алгоритма на Java

Тестовый класс для демонстрации

```
public class Cat {
	private String name;
	private int age;

	public Cat(String name, int age) {
		super();
		this.name = name;
		this.age = age;
	}

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}
	@Override
	public String toString() {
		return "Cat [name=" + name + ", age=" + age + "]";
	}
}
```

Методы для сортировки распределяющим подсчетом

```
public static int[] findMinMaxKey(Cat[] cats) {
        int minKey = cats[0].getAge();
        int maxKey = cats[0].getAge();
        for (Cat cat : cats) {
                if (cat.getAge() < minKey) {
                        minKey = cat.getAge();
                }
                if (cat.getAge() > maxKey) {
                        maxKey = cat.getAge();
                }
        }
        return new int[] { minKey, maxKey };
}

public static Cat[] countingSort(Cat[] cats) {
        int[] minMaxKey = findMinMaxKey(cats);
        int minKey = minMaxKey[0];
        int maxKey = minMaxKey[1];
        int n = maxKey - minKey + 1;
        int[] support = new int[n];
        for (Cat element : cats) {
                support[element.getAge() - minKey] += 1;
        }
        int size = cats.length;
        for (int i = support.length - 1; i >= 0; i--) {
                size -= support[i];
                support[i] = size;
        }
        Cat[] result = new Cat[cats.length];
        for (Cat cat : cats) {
                result[support[cat.getAge() - minKey]] = cat;
                support[cat.getAge() - minKey] += 1;
        }
        return result;
}
```

Fortran

Реализация алгоритма на Fortran

Структура для описания кошки

```
type::Cat
    character(len=10)::c_name
    integer(4)::age
end type Cat
```

Функции для поиска минимального и максимального ключа

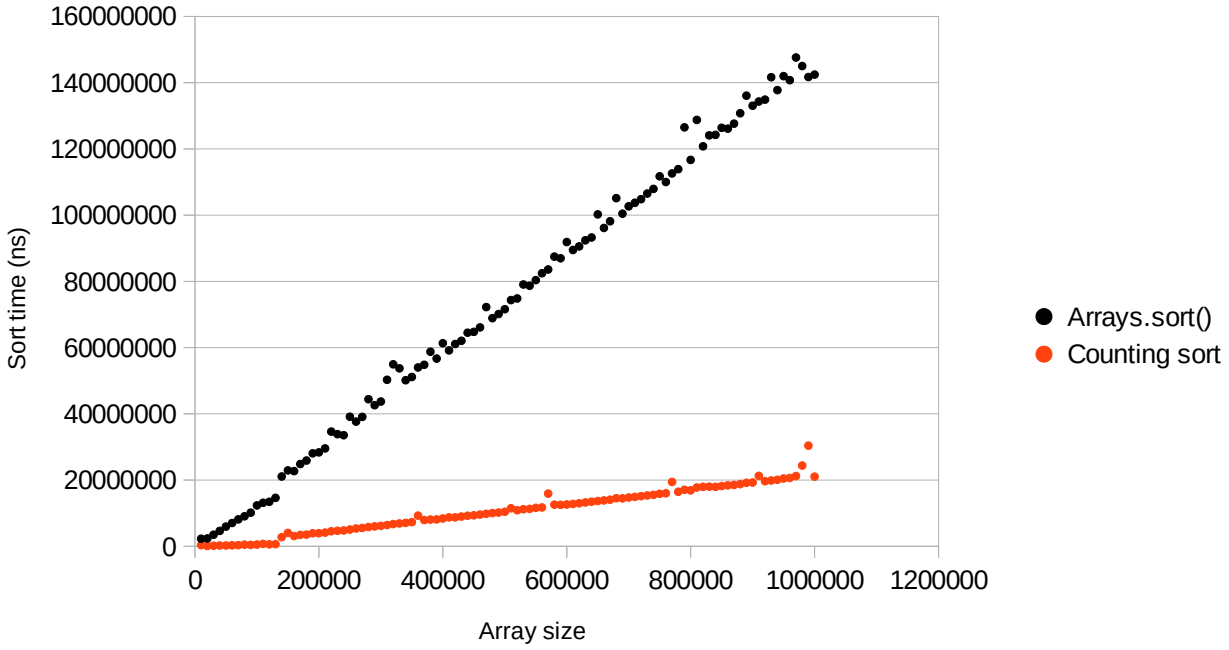
```
function find_min_max_keys(cats) result(min_max)
    type(Cat),dimension(:),intent(in)::cats
    integer(4),dimension(2)::min_max
    integer(4)::i, min_key, max_key
    min_key = cats(1)%age
    max_key = cats(1)%age
    do i = 2,size(cats)
        if (cats(i)%age < min_key) then
            min_key = cats(i)%age
        end if
        if (cats(i)%age > max_key) then
            max_key = cats(i)%age
        end if
    end do
    min_max = [min_key, max_key]
end function find_min_max_keys
```

Функция для сортировки распределяющим подсчетом

```
function counting_sort(cats) result(cat_res)
    type(Cat),dimension(:),intent(in)::cats
    type(Cat),dimension(size(cats))::cat_res
    integer(4)::i, min_key, max_key,n,cats_size
    integer(4), dimension(2)::min_max
    integer(4),dimension(:),allocatable::support
    min_max = find_min_max_keys(cats)
    min_key = min_max(1)
    max_key = min_max(2)
    n = max_key - min_key + 1
    allocate(support(n))
    do i = 1,size(cats)
        support(cats(i)%age - min_key + 1) = support(cats(i)%age - min_key + 1) + 1
    end do
    cats_size = size(cats) + 1
    do i = n, 1,-1
        cats_size = cats_size - support(i)
        support(i) = cats_size
    end do
    do i = 1, size(cats)
        cat_res(support(cats(i)%age - min_key + 1)) = cats(i)
        support(cats(i)%age - min_key + 1) = support(cats(i)%age - min_key + 1) + 1
    end do
    deallocate(support)
end function counting_sort
```

Вычислительный эксперимент

Для проверки эффективности алгоритма по сравнению с алгоритмом сортировки общего назначения проведен вычислительный эксперимент. Проводится сортировка массивов кошек (ключом сортировки выступает возраст) разного размера. При этом использовался алгоритм сортировки общего назначения (реализован в стандартной библиотеке) и текущая версия сортировки распределяющим подсчетом. Результат приведен на графике.



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.