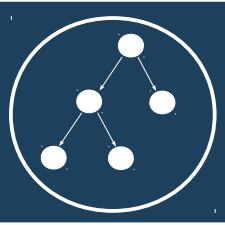

Data Structures and Algorithms

Алгоритмы.
Адаптация алгоритмов сортировки. Сортировка пользовательских типов данных.

Постановка задачи

Большинство примеров алгоритмов сортировки демонстрируются для последовательностей целых чисел. Однако очень часто возникают вопросы:

- Как адаптировать эти алгоритмы сортировки для сортировки пользовательских типов данных? Т.е. есть например класс Cat, как отсортировать последовательность объектов этого класса?
- Если в последовательности есть пустые ссылки (null, None и т. д.), как это учесть?

Описание способа адаптации

В большинстве алгоритмов сортировки используется сравнение между собой двух элементов последовательности. Для целых чисел для этого используют операторы сравнения (такие как <, >, == и т.д). Для пользовательских типов данных чаще всего такие операторы не применимы.

Одним из возможных решений является написание функции (метода), которая в качестве параметров принимает два объекта пользовательского класса, а в качестве результата своей работы возвращает целое число. В таком случае в алгоритме сортировки можно использовать результат работы этой функции.

Правила работы такой функции сравнения можно сформулировать следующим образом:

- Если первый объект больше второго, то вернуть любое положительное число.
- Если первый объект меньше второго, то вернуть любое отрицательное число.
- Если объекты равны, то вернуть ноль.

Внимание!! Критерий сортировки вы определяете сами именно в этой функции. Если нужна поддержка несколько критериев сортировки, то следует написать несколько таких функций.

Как учесть наличие «пустых» объектов в последовательности

В случае, если нужно учитывать наличие «пустых» объектов в последовательности, то перед сравнением полей объекта стоит сравнить между собой именно ссылки на объекты которые хранятся в последовательности.

Одним (но не единственным) способом сравнения может стать например такой:

1) Если 1-й объект не «пустой», а второй «пустой» вернуть положительное число.
2) Если 1-й объект «пустой», а второй не «пустой» вернуть отрицательное число.
3) Если 1-й объект «пустой», и второй «пустой» вернуть 0.

После этих проверок выполнить сравнение по значениям нужных полей.

Внимание! Это стоит делать, только при вероятности нахождения «пустых» объектов в последовательностях.

Постановка задачи

1) Создать пользовательский тип данных Cat (чаще всего это класс). Наделить его полями name, age.
2) Создать последовательность хранящую несколько объектов этого типа и также несколько «пустых» объектов (ссылки типа null, None в зависимости от используемого языка программирования).
3) Написать функцию сравнения для объектов этого типа. В качестве критерия сравнения выберем возраст.
4) Используя любой алгоритм сортировки (например сортировку выбором рассмотренную недавно) и эту функцию провести сортировку последовательности.

Python

Реализация на Python

Создание пользовательского типа Cat

```
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)
```

Создание объектов типа Cat и добавление в последовательность

```
cat_1 = Cat("Vaska", 6)
cat_2 = Cat("Barsik", 2)
cat_3 = Cat("Umka", 12)
cat_4 = Cat("Kuzia", 4)


cats = [cat_1, cat_2, None, cat_3, cat_4]
```

Функция сравнения двух объектов

```
def compare_cat(cat_a, cat_b):
    if cat_a is not None and cat_b is None:
        return 1
    if cat_a is None and cat_b is not None:
        return -1
    if cat_a is None and cat_b is None:
        return 0
    if cat_a.age > cat_b.age:
        return 1
    if cat_a.age < cat_b.age:
        return -1
    return 0
```

Проверка на наличие None

Сравнение по значению поля age

Использование функции сравнения при сортировке

```
for i in range(0, len(cats)-1):
    min_index = i
    for j in range(i+1, len(cats)):
        if compare_cat(cats[min_index], cats[j]) > 0:
            min_index = j
    if min_index != i:
        temp = cats[i]
        cats[i] = cats[min_index]
        cats[min_index] = temp
```

Использование функции сравнения

Реализация на Java

Создание пользовательского типа Cat

```
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 + "]";
     }
}
```

Создание объектов типа Cat и добавление в последовательность

```
Cat cat1 = new Cat("Vaska", 6);
Cat cat2 = new Cat("Barsik", 2);
Cat cat3 = new Cat("Umka", 12);
Cat cat4 = new Cat("Kuzia", 4);

Cat[] cats = new Cat[] { cat1, cat2, null, cat3, cat4 };
```

Метод сравнения двух объектов

```
public static int compareCat(Cat a, Cat b) {
    if (a != null && b == null) {
        return 1;
    }
    if (a == null && b != null) {
        return -1;
    }
    if (a == null && b == null) {
        return 0;
    }
    if (a.getAge() > b.getAge()) {
        return 1;
    }
    if (a.getAge() < b.getAge()) {
        return -1;
    }
    return 0;
}
```

Проверка на наличие null

Сравнение по значению поля age

Использование метода сравнения при сортировке

```
for (int i = 0; i < cats.length - 1; i++) {
    int minIndex = i;
    for (int j = i + 1; j < cats.length; j++) {
        if (compareCat(cats[minIndex], cats[j]) > 0) {
            minIndex = j;
        }
    }
    if (minIndex != i) {
        Cat temp = cats[i];
        cats[i] = cats[minIndex];
        cats[minIndex] = temp;
    }
}
```

Использование метода сравнения

Список литературы

1) Лафоре Р. Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд. —СПб.: Питер, 2013. — 704 с.:ISBN 978-5-496-00740-5. Стр.[112-115]