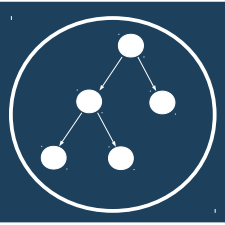
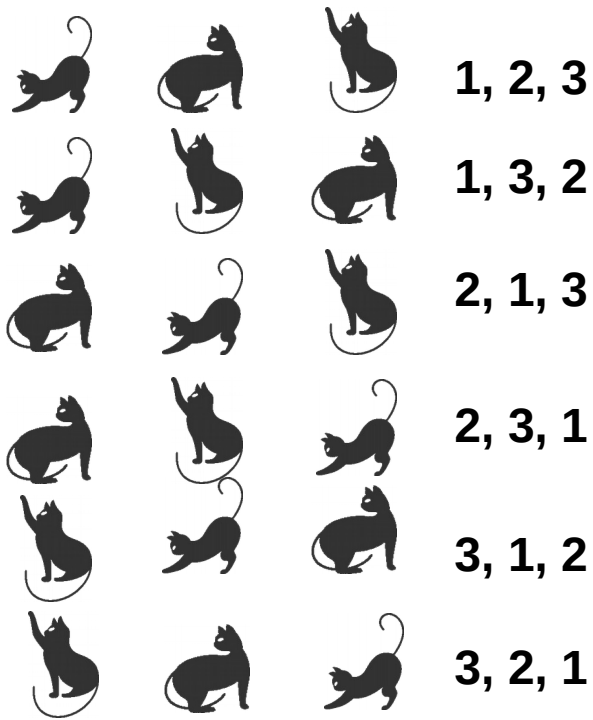


Data Structures and Algorithms

Алгоритмы.
Генерация перестановок.
Алгоритм Джонсона - Троттера.

Перестановки

В комбинаторике перестановка — это упорядоченный набор без повторений чисел 1 , 2 , … , n , обычно трактуемый как биекция на множестве { 1 , 2 , … , n } которая числу i ставит в соответствие i-й элемент из набора. Число n при этом называется длиной перестановки.



1, 2, 3

1, 3, 2

2, 1, 3

2, 3, 1

3, 1, 2

3, 2, 1

1, 2, 3 →

Генерация всех перестановок

Ряд задач предполагает генерацию всех перестановок длинной n. Именно для этого используется алгоритм Джонсона — Троттера.

Преимущества данного алгоритма:

- Относительная легкость не рекурсивной реализации
- Малый расход памяти

Для генерации перестановок элементов любых типов данных, можно рассматривать все равно перестановки целых чисел. В таком случае перестановки целых чисел можно рассматривать в качестве индексов последовательностей. В свою очередь в последовательностях можно хранить любые типы данных.

Сведение о алгоритме

Алгоритм Джонсона — Троттера.

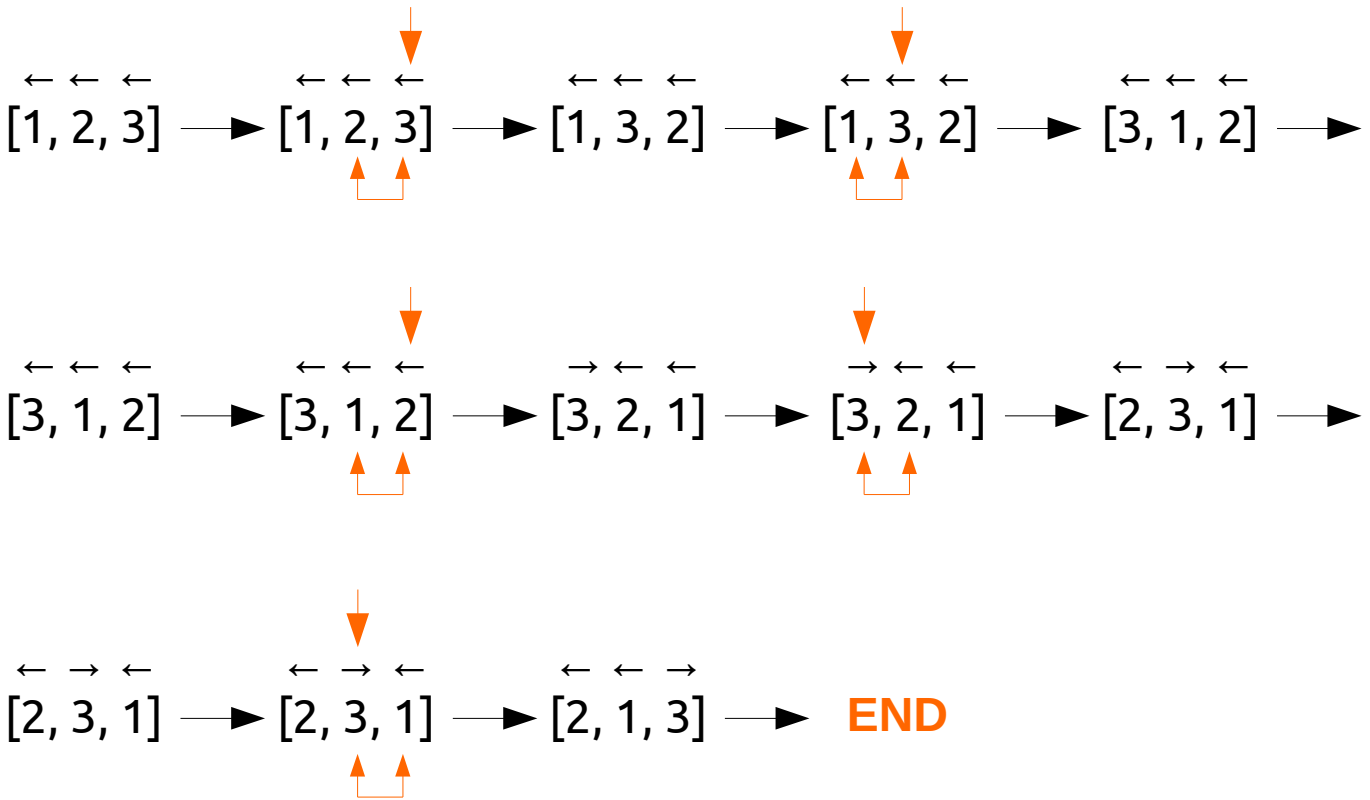
Сложность по времени в наихудшем случае	O(n!)
Затраты памяти	O(n)

Описание алгоритма

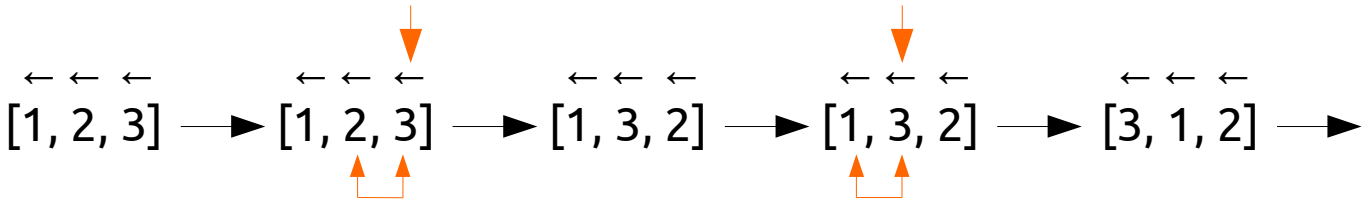
С каждым элементом перестановки связываем направление. Направление — указатель на соседний элемент (может указывать на элемент справа или слева). Элемент перестановки называется мобильным, если его направление указывает на меньший соседний элемент.

1) Создается первая перестановка. Ряд чисел по возрастанию 1, 2, 3, … n. Направление каждого элемента указывает влево.
2) Ищем наибольший мобильный элемент. Если не находим, то алгоритм закончен.
3) Производим обмен, найденного мобильного элемента с элементом на который указывает направление найденного мобильного элемента.
4) Меняем направление у всех элементов, которые больше чем найденный на шаге 2 элемент.
5) Переходим к шагу 2.

Графическое пояснение



Описание алгоритма

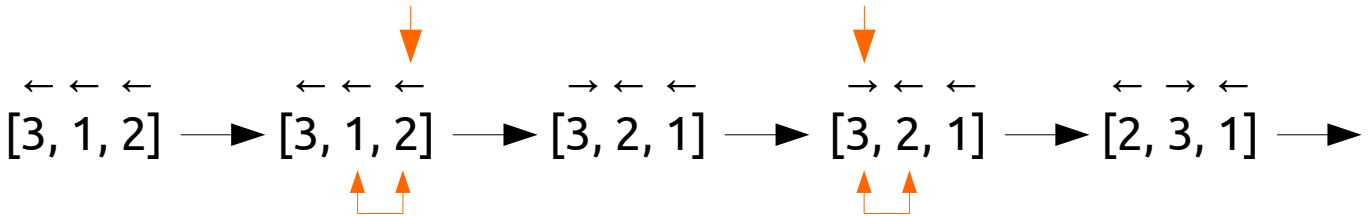


Начальный шаг. Последовательность 1,2,3 все направления влево.

Максимальным мобильным элементом является 3 (т. к. число на которое указывает его направление меньше его 2 < 3). Производится их обмен, так как больше 3 элементов нет, направление у элементов не меняется.

Потом находится следующий максимальный мобильный элемент. Это опять 3. Происходит обмен с элементом 1.

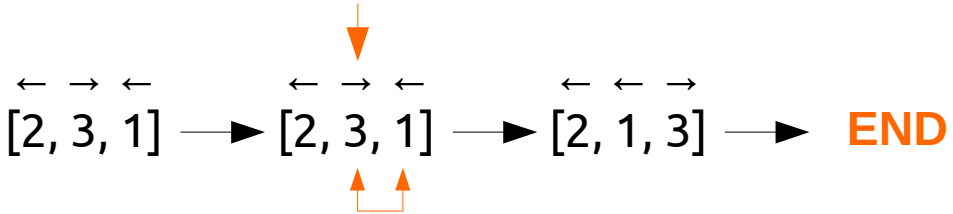
Описание алгоритма



Производится поиск максимального мобильного элемента. Это элемент 2 (он больше чем элемент по его направлению) и производится обмен. Так как элемент 3 больше чем 2, то у него меняется направление (теперь направление элемента 3 — вправо).

Теперь максимальным мобильным элементом стало 3, обмен с элементом 2. Направление нигде не меняется.

Описание алгоритма



Производится поиск максимального мобильного элемента. Это элемент 3 (он больше чем элемент по его направлению) и производится обмен. Направление нигде не меняется.

Поиск мобильного элемента неудачен. Алгоритм закончен.

Python

Реализация алгоритма на Python

Python

Реализация алгоритма на Python

```
def find_max_mobile_element(permutation, direction):
    index = -1
    for i in range(len(permutation)):
        next_element_index = i + direction[i]
        if next_element_index >= 0 and next_element_index < len(permutation):
            if permutation[i] > permutation[next_element_index]:
                if index == -1:
                    index = i
                else:
                    if permutation[i] > permutation[index]:
                        index = i
    return index
```

Функция поиска индекса максимального мобильного элемента

Реализация алгоритма на Python

```
def change_direction(permutation, direction, mobile_element):
    for i in range(len(permutation)):
        if permutation[i] > mobile_element:
            direction[i] = direction[i]*(-1)
```

Функция смены направления у элементов значение которых больше значения максимального мобильного элемента

Реализация алгоритма на Python

```
def swap_element(permutation, direction, i, j):
    permutation[i], permutation[j] = permutation[j], permutation[i]
    direction[i], direction[j] = direction[j], direction[i]
```

Функция обмена элементов перестановки и направления

Python

Реализация алгоритма на Python

```
def permutation_generator(n):
    permutation = list(range(1, n+1))
    direction = [-1]*n
    print(permutation)
    mobile_element_index = find_max_mobile_element(permutation, direction)
    while mobile_element_index != -1:
        mobile_element = permutation[mobile_element_index]
        next_index = mobile_element_index + direction[mobile_element_index]
        swap_element(permutation, direction, mobile_element_index, next_index)
        change_direction(permutation, direction, mobile_element)
        print(permutation)
        mobile_element_index = find_max_mobile_element(permutation, direction)
```

Функция генерации перестановок длинной n

Java

Реализация алгоритма на Java

Реализация алгоритма на Java

```
public static int findMaxMobileElement(int[] permutation, int[] direction) {
    int index = -1;
    for (int i = 0; i < permutation.length; i++) {
        int nextIndex = i + direction[i];
        if (nextIndex >= 0 && nextIndex < permutation.length) {
            if (permutation[i] > permutation[nextIndex]) {
                if (index == -1) {
                    index = i;
                } else {
                    if (permutation[i] > permutation[index]) {
                        index = i;
                    }
                }
            }
        }
    }
    return index;
}
```

Функция поиска индекса максимального мобильного элемента

Java

Реализация алгоритма на Java

```
public static void changeDirection(int[] permutation, int[] direction, int mobileElement) {
    for (int i = 0; i < permutation.length; i++) {
        if (permutation[i] > mobileElement) {
            direction[i] = direction[i] * (-1);
        }
    }
}
```

Функция смены направления у элементов значение которых больше значения максимального мобильного элемента

Реализация алгоритма на Java

```
public static void swap(int[] permutation, int[] direction, int i, int j) {
    int permutationTemp = permutation[i];
    permutation[i] = permutation[j];
    permutation[j] = permutationTemp;

    int directionTemp = direction[i];
    direction[i] = direction[j];
    direction[j] = directionTemp;
}
```

Функция обмена элементов перестановки и направления

Реализация алгоритма на Java

```
public static void permutationGenerator(int n) {
    int[] permutation = new int[n];
    int[] direction = new int[n];
    for (int i = 0; i < permutation.length; i++) {
        permutation[i] = i + 1;
        direction[i] = -1;
    }
    System.out.println(Arrays.toString(permutation));
    int mobileElementIndex = findMaxMobileElement(permutation, direction);
    for (; mobileElementIndex != -1;) {
        int mobileElement = permutation[mobileElementIndex];
        int nextIndex = mobileElementIndex + direction[mobileElementIndex];
        swap(permutation, direction, mobileElementIndex, nextIndex);
        changeDirection(permutation, direction, mobileElement);
        System.out.println(Arrays.toString(permutation));
        mobileElementIndex = findMaxMobileElement(permutation, direction);
    }
}
```

Функция генерации перестановок длинной n

Список литературы

1) Ананий Левитин. Алгоритмы: введение в разработку и анализ. : Пер. с англ. — М. : Издательский дом "Вильямс", 2006. — 576 с. — ISBN 5-8459-0987-2. Стр. [227-229]