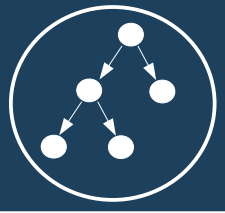

Data Structures and Algorithms

Алгоритмы.
Генерация перестановок. Метод обмена Эрлиха

Сведение о алгоритме

Сложность по времени в наихудшем случае O(n)

Внимание для одной перестановки !

Описание алгоритма

1) Создать две вспомогательные последовательности. Длинна первой равна длине базовой последовательности (в дальнейшем b). Длинна второй на один элемент больше (в дальнейшем c). Заполнить последовательность b числами от 0 и далее по возрастанию с шагом 1. Заполнить с нулями. Объявить переменную k=1, j=1. Перейти к пункту **2**.
2) Вернуть базовую последовательность как очередную перестановку. Перейти к пункту **3**.
3) Присвоить k=1. Выполняем проход от начала последовательности до тех пор пока c[k]=k. На каждом шаге устанавливаем c[k] = 0. По окончанию прохода проверяем если k = n **закончить алгоритм**. В противном случае c[k]=c[k]+1 и перейти к пункту **4**.
4) Выполнить обмен a[0] <-> a[b[k]]. Перейти к **5**.
5) Установить j=1, k = k -1. До тех пор пока j < k выполнять обмен b[j] <-> b[k] установить j = j + 1, k = k-1. Вернуться к **2**.

Графическое пояснение алгоритма

a	b	c	
[«a», «b», «c»]	[0, 1, 2]	[0, 0, 0,0]	Возврат самой последовательности
[«a», «b», «c»]	[0, 1, 2]	[0, 1, 0, 0]	Ищем такой k, что c[k]≠k. Увеличиваем c[k] на единицу. Меняем местами a[0] = a[b[k]]
[«b», «a», «c»]	[0, 1, 2]	[0, 1, 0, 0]	Уменьшаем k на единицу. Устанавливаем j =1. Ничего не делаем так как j>k

Графическое пояснение алгоритма

a	b	c	
[«b», «a», «c»]	[0, 1, 2] k	[0, 0, 1, 0] k	Ищем такой k, что c[k]≠k. По ходу поиска ставим c[k] = 0. Увеличиваем c[k] на единицу. Меняем местами a[0] = a[b[k]]
[«c», «a», «b»]	[0, 1, 2] k j	[0, 0, 1, 0]	Уменьшаем k на единицу. Устанавливаем j =1. Ничего не делаем так как j=k

Графическое пояснение алгоритма

a	b	c	
[«c», «a», «b»]	[0, 1, 2] k	[0, 1, 1, 0] k	Ищем такой k, что c[k]≠k. По ходу поиска ставим c[k] = 0. Увеличиваем c[k] на единицу. Меняем местами a[0] = a[b[k]]
[«a», «c», «b»]	[0, 1, 2] k j	[0, 1, 1, 0]	Уменьшаем k на единицу. Устанавливаем j =1. Ничего не делаем так как j>k

Python

Реализация алгоритма на Python

Функция генератор возвращающая перестановки

```
def get_permutation(sequince):
    n = len(sequince)
    a = sequince[::]
    b = [i for i in range(n)]
    c = [0 for i in range(n+1)]
    k = 1
    j = 1
    while True:
        yield a
        k = 1
        while c[k] == k:
            c[k] = 0
            k = k + 1
        if k == n:
            return
        c[k] += 1
        a[0], a[b[k]] = a[b[k]], a[0]
        j = 1
        k = k-1
        while j < k:
            b[j], b[k] = b[k], b[j]
            j = j+1
            k = k-1
```

Java

Реализация алгоритма на Java

Метод для обмена элементов массива местами

```java
public static <T> void swap(T[] array, int i, int j) {
    T temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}

public static void swap(int[] array, int i, int j) {
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
```

Метод для вывода на экран всех перестановок

```
public static <T> void printAllPermutation(T[] array) {
      T[] aSequince = Arrays.copyOf(array, array.length);
      int[] bTable = new int[aSequince.length];
      int[] cTable = new int[aSequince.length + 1];
      int k = 1;
      int j = 1;
      for (int i = 0; i < bTable.length; i++) {
            bTable[i] = i;
      }
      for (;;) {
            System.out.println(Arrays.toString(aSequince));
            k = 1;
            for (; cTable[k] == k;) {
                  cTable[k++] = 0;
            }
            if (k == aSequince.length) {
                  break;
            }
            cTable[k] += 1;
            swap(aSequince, 0, bTable[k]);
            j = 1;
            k = k - 1;
            for (; j < k;) {
                  swap(bTable, k--, j++);
            }
      }
}
```

Список литературы

1) Д. Кнут. Искусство программирования. Том 4. «Генерация всех кортежей и перестановок», 2-е изд. ISBN 5-8459-0082-4 стр. 72
2) Gideon Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. Journal of the ACM, 20(3):500-513, July 1973.