Реализации алгоритмов/Комбинаторика/Размещения

Содержание

Java

```java
import java.util.Arrays;

public class PermutationsWithRepetition {
    private Object[] source;
    private int variationLength;

    public PermutationsWithRepetition(Object[] source, int variationLength) {
        this.source = source;
        this.variationLength = variationLength;
    }

    public Object[][] getVariations() {
        int srcLength = source.length;
        int permutations = (int) Math.pow(srcLength, variationLength);

        Object[][] table = new Object[permutations][variationLength];

        for (int i = 0; i < variationLength; i++) {
            int t2 = (int) Math.pow(srcLength, i);
            for (int p1 = 0; p1 < permutations;) {
                for (int al = 0; al < srcLength; al++) {
                    for (int p2 = 0; p2 < t2; p2++) {
                        table[p1][i] = source[al];
                        p1++;
                    }
                }
            }
        }

        return table;
    }

    public static void main(String[] args) {
        PermutationsWithRepetition gen = new PermutationsWithRepetition(
                new Integer[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 0},
                5);

        Object[][] variations = gen.getVariations();

        for (Object[] s : variations) {
            System.out.println(Arrays.toString(s));
        }
    }
}
```

Python

```
from numpy import array

def permutations(n,length):
    numbers = range(n)
    permutations = n**length
    output = array([[0]*length]*permutations)

    for i in range(length):
        t2 = n**i
        p1 = 0
        while (p1 < permutations):
            for al in range(n):
                for p2 in range(t2):
                    output[p1,i] = numbers[al]
                    p1 += 1
    return output
```

Haskell

```
import Control.Monad
permutationsWithRepetition xs = iterate (liftM2 (:) xs) [[]]

Prelude> take 4 (permutationsWithRepetition "ab")
[[""],["a","b"],["aa","ab","ba","bb"],["aaa","aab","aba","abb","baa","bab","bba","bbb"]]
Prelude> permutationsWithRepetition [1,2,3] !! 2
[[1,1],[1,2],[1,3],[2,1],[2,2],[2,3],[3,1],[3,2],[3,3]]
```

C++

```
// pIn - входной массив
// N - размер входного массива
// K - количество элементов в размещении

void PermutationWithRepetition(const char* pIn, int N, int K)
{
  char* pOut = new char[K + 1]; // строка из K символов плюс 1 символ для терминального 0
  pOut[K] = 0;                  // помещаем 0 в конец строки
  K--;
  int *stack = new int[K * 2],  // стек псевдорекурсии, глубина рекурсии K - 1
      *pTop = stack,            // вершина стека
      k = 0,                    // переменные цикла
      n = 0,
      j = 0;
  for (;;)                      // цикл псевдорекурсии
  {
    while(n < N)
    {
      pOut[k] = pIn[n++];
      if (k == K)
        printf("%02d. %s\n", ++j, pOut);
      else
      {
        if (n < N)
        {
          *pTop++ = k;          // сохраняем k и n в стеке
          *pTop++ = n;
        }
        k++;                    // псевдорекурсивный вызов
        n = 0;
      }
    }
    if (pTop == stack)          // стек пуст, конец цикла
      break;

    n = *(--pTop);              // выталкиваем k и n из стека
    k = *(--pTop);
  }
  delete[] pOut;
  delete[] stack;
}
```

JavaScript

```
function PermutationsWithRepetition(src, len){

    var K = len - 1, k = 0,
        N = src.length, n = 0,
        out = [],
        stack = [];

    function next(){
        while (true) {
            while (n < src.length) {
                out[k] = src[n++];
                if (k == K) return out.slice(0);
                else {
                    if (n < src.length) {
                        stack.push(k);
                        stack.push(n);
                    }
                    k++;
                    n = 0;
                }
            }
            if (stack.length == 0) break;

            n = stack.pop();
            k = stack.pop();
        }
        return false;
    }

    function rewind(){ k = 0; n = 0; out = []; stack = []; }

    function each(cb) {
        rewind();
        var v;
        while (v = next()) if (cb(v) === false) return;
    }

    return {
        next: next,
        each: each,
        rewind: rewind
    };
}

/* пример использования */

var perms = PermutationsWithRepetition([1, 2, 3, 4, 5], 3);

perms.next(); // [1, 1, 1]
perms.next(); // [1, 1, 2]
//...
perms.next(); // [5, 5, 4]
perms.next(); // [5, 5, 5]
perms.next(); // false

perms.rewind();
perms.next(); // [1, 1, 1]
//...

perms.each(function(v){ console.log(v); }); // вывод всех размещений в консоль
```