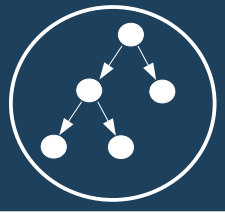


Data Structures and Algorithms

Алгоритмы.
Генерация сочетаний.

Сведение о алгоритме

Сложность по времени в наихудшем случае $O(\frac{n!}{k!(n-k)!} \cdot \frac{k}{n-k+1})$

Затраты памяти $O(k+2)$

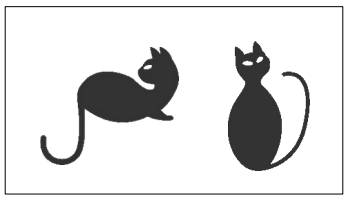
k — элементов выбранных из множества из n - элементов

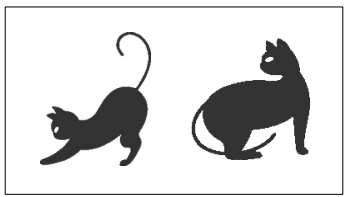
Сочетание

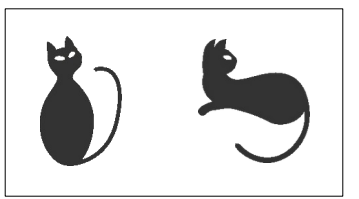
Сочетание из n по k - набор k элементов, выбранных из данного множества, содержащего n различных элементов. Наборы, отличающиеся только порядком следования элементов (но не составом), считаются одинаковыми. Т.е. порядок элементов не важен.

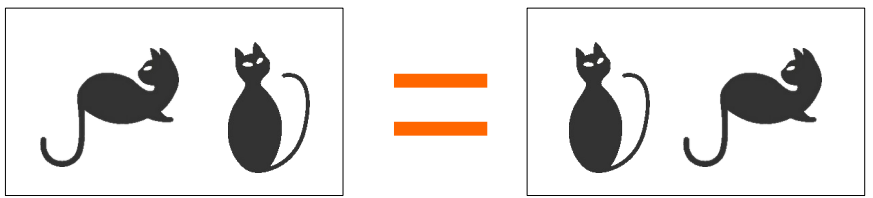
Множество из 5 элементов (n=5).

Сочетание из 5 по 2









В размещении порядок следования не учитывается.

Количество сочетаний

Число сочетаний из n по k равно биномиальному коэффициенту:

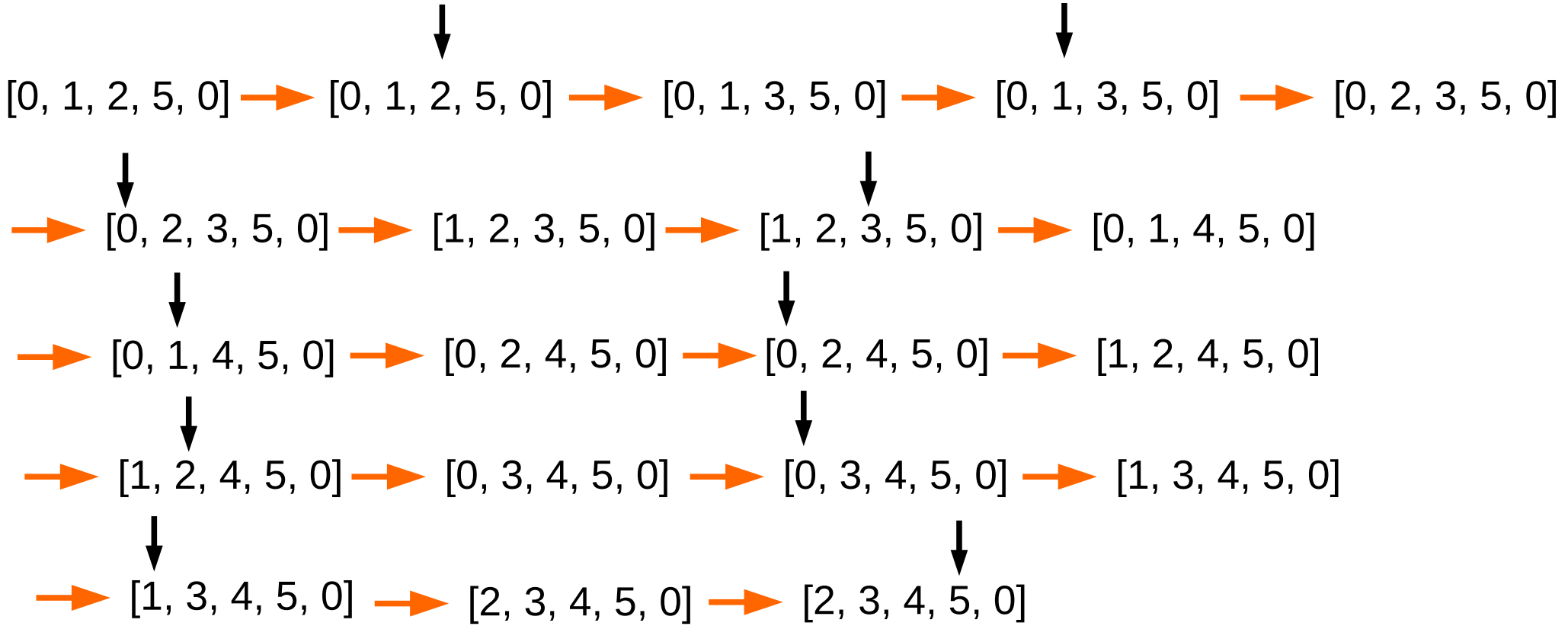
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Принцип работы алгоритма

Генерируем все сочетания из n целых чисел [0,1,2...n-1] по k.

1) Создаем последовательность размером k+2 элемента. Первые k — элементов устанавливаем равный индексу элемента. Элемент k+1 устанавливаем равным n, k+2 — равный 0.
2) Начиная с начала последовательности проверяем условие $k_i+1 = k_{i+1}$. Если это так, то устанавливаем элемент равный его индексу. Как только это условие нарушено, то переходим к пункту 3.
3) Если индекс элемента где нарушено условие больше k заканчиваем алгоритм, если нет то увеличиваем элемент, на который указывает индекс, на единицу. Возвращаемся к пункту 2.

Графическая иллюстрация работы алгоритма (сочетание 3 из 5)



Python

Реализация алгоритма на Python

Реализация алгоритма на Python

```
def print_combination(k, n):
    comb_ls = []
    for i in range(k):
        comb_ls.append(i)
    comb_ls.append(n)
    comb_ls.append(0)
    while True:
        print(comb_ls[0:k])
        for j in range(len(comb_ls)-1):
            if comb_ls[j]+1 == comb_ls[j+1]:
                comb_ls[j] = j
            else:
                break
        if j < k:
            comb_ls[j] += 1
        else:
            break
```

Java

Реализация алгоритма на Java

Реализация алгоритма на Java

```
public static void printAllCombination(int n, int k) {
     int[] comb = new int[k + 2];
     for (int i = 0; i < k; i++) {
          comb[i] = i;
     }
     comb[k] = n;
     comb[k + 1] = 0;
     for (;;) {
          printArrayPart(comb, 0, k);
          int j = 0;
          for (; comb[j] + 1 == comb[j + 1];) {
               comb[j] = j;
               j = j + 1;
          }
          if (j < k) {
               comb[j]++;
          } else {
               break;
          }
     }
}
```

Вспомогательный метод для вывода сочетания

```
public static void printArrayPart(int[] array, int s, int e) {
    System.out.print("[");
    for (int i = s; i < e - 1; i++) {
        System.out.print(array[i] + ",");
    }
    System.out.println(array[e - 1] + "]");
}
```

Список литературы

1) Д. Кнут. Искусство программирования. Том 4. Генерация всех сочетаний и разбиений, 3-е изд. Стр. [10-14]