Лабораторная работа № 1
Генерация подмножеств. Коды Грея (4 часа)

Краткий теоретический материал

1. Общие сведения о комбинаторной генерации

Пусть есть некоторое множеств комбинаторных объектов (например, подмножеств, перестановок, кортежей и т.п.). Требуется сгенерировать все элементы этого множества.

Мы будем рассматривать алгоритмы, генерирующие комбинаторные объекты в *лексикографическом порядке*, и алгоритмы, генерирующие объекты в так называемом *порядке минимального изменения*. Порядок минимального изменения подразумевает, что каждый следующий элемент генерируется из предыдущего при помощи очень небольшого изменения. Эти способы *последовательной генерации* часто реализуются с помощью *алгоритма генерации следующего элемента*.

Помимо последовательной генерации, нас интересуют *алгоритмы ранжирования* и *алгоритмы генерации объектов по номеру*. Алгоритм ранжирования определяет положение (или ранг) комбинаторного объекта среди всех объектов (относительно заданного порядка); алгоритм генерации объекта по номеру находит объект, имеющий заданный ранг. Таким образом, операции ранжирования и генерации объектов по номеру можно рассматривать как обратные операции.

Приведём более формальные математические описания этих понятий. Пусть S – конечное множество и $N = |S|$. Функция ранжирования – это биекция

$$rank : S \to \{0, \dots, N-1\}.$$

Функция ранжирования задаёт отношение полного порядка на множестве S по очевидному правилу

$$s < t \Leftrightarrow rank(s) < rank(t).$$

Обратно: существует единственная функция ранжирования, связанная с любым отношением полного порядка на S.

Если `rank` – это функция ранжирования, заданная на S, то существует единственная функция получения объекта по его номеру, соответствующая функции `rank`. Эта функция `unrank` также является биекцией:

$$unrank : \{0, \dots, N-1\} \to S.$$

Функция `unrank` является обратной функцией для функции `rank`, то есть

$$rank(s) = i \Leftrightarrow unrank(i) = s,$$

для всех $s \in S$ и всех $i \in \{0, \dots, N-1\}$.

Эффективные алгоритмы ранжирования и генерации объекта по номеру имеют ряд возможных применений. Одним из приложений является генерация случайных объектов из заданного множества S. Другое применение алгоритмов ранжирования и получения объекта по номеру заключается в хранении комбинаторных объектов в памяти компьютера. Вместо хранения комбинаторного объекта, который (возможно) занимает достаточно много места в памяти, можно просто сохранить его ранг, являющийся просто целым числом. При необходимости объект может быть восстановлен при помощи алгоритма получения объекта по номеру.

Функция генерации следующего элемента `successor` может быть легко построена с помощью функций `rank` и `unrank` по следующей формуле:

$$successor(s) = \begin{cases} unrank(rank(s) + 1), & \text{если } rank(s) < N - 1, \\ \text{не определено}, & \text{если } rank(s) = N - 1 \end{cases}$$

Однако, важно понимать, что, в зависимости от множества S, могут существовать более эффективные способы генерации следующего элемента.

Если функция конструирования следующего элемента известна, то сгенерировать все элементы множества S достаточно просто. Необходимо начать с первого элемента и применить функцию `successor` $N - 1$ раз.

2. Генерация подмножеств в лексикографическом порядке

Пусть n – положительное целое число, $A = \{1, \ldots, n\}$. Рассмотрим множество S, состоящее из всех 2^n подмножеств A. Опишем способ генерации элементов S в лексикографическом порядке.

Для каждого подмножества $T \subseteq A$ определим характеристический вектор $\chi(T)$. Элемент характеристического вектора показывает, входит ли соответствующий элемент множества A в T. Например, если $n = 3$ и множество $A = \{1, 2, 3\}$, то характеристические векторы для подмножеств A будут иметь следующий вид:

T	$\chi(T) = [x_2, x_1, x_0]$	$\mathsf{rank}(T)$
$\emptyset$	$[0,0,0]$	0
$\{3\}$	$[0,0,1]$	1
$\{2\}$	$[0,1,0]$	2
$\{2,3\}$	$[0,1,1]$	3
$\{1\}$	$[1,0,0]$	4
$\{1,3\}$	$[1,0,1]$	5
$\{1,2\}$	$[1,1,0]$	6
$\{1,2,3\}$	$[1,1,1]$	7

Лексикографический порядок на множестве характеристических векторов определяет порядок генерации подмножеств в лексикографическом порядке.

Ниже приведены алгоритмы `rank` и `unrank` для рассматриваемого порядка.

Algorithm 2.1: SUBSETLEXRANK (n, T)

$r \leftarrow 0$
for $i \leftarrow 1$ **to** n
  do { **if** $i \in T$
    then $r \leftarrow r + 2^{n-i}$
return (r)

Algorithm 2.2: SUBSETLEXUNRANK (n, r)

$T \leftarrow \emptyset$
for $i \leftarrow n$ **downto** 1
  do { **if** $r \bmod 2 = 1$
    then $T \leftarrow T \cup \{i\}$
    $r \leftarrow \lfloor \frac{r}{2} \rfloor$
return (T)

3. Генерация подмножеств в порядке минимального изменения

Характеристические векторы подмножеств при использовании порядка минимального изменения образуют структуру, которая известна как *код Грея*. То есть код Грея – это упорядочивание 2^n двоичных векторов длины n таким образом, что любые два последовательных вектора отличаются ровно в одном разряде.

Например, ниже приведён пример кода Грея при $n = 3$:

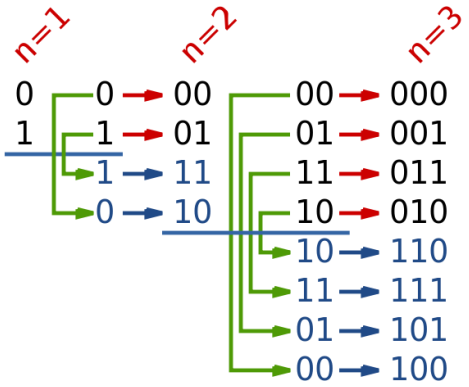
000, 001, 011, 010, 110, 111, 101, 100.

Порядок генерации подмножеств, соответствующий этому коду, выглядит так:

$\emptyset$, {3}, {2, 3}, {2}, {1, 2}, {1, 2, 3}, {1, 3}, {1}.

Мы будем рассматривать класс кодов Грея, называемый *зеркальными двоичными кодами Грея*. Строится он так. Для получения кода длины n производится n шагов. На первом шаге код имеет длину 1 и состоит из двух векторов 0 и 1. На каждом следующем шаге в конец списка заносятся все уже имеющиеся вектора в обратном порядке, и затем к первой половине получившихся векторов дописывается 0, а ко второй 1. С каждым шагом длина векторов увеличивается на 1, а их количество — вдвое. Таким образом, количество векторов длины n равно 2^n.

Проиллюстрируем процесс построения зеркального кода Грея:



Корректность этого алгоритма можно доказать, используя метод математической индукции.

С другой стороны, существует нерекурсивный алгоритм генерации следующего подмножества в порядке минимального изменения. Псевдокод этого алгоритма приведён ниже. Его корректность доказывается также с использованием метода математической индукции.

Algorithm 2.3: GRAYCODESUCCESSOR (n, T)

if $|T|$ is even
  then $U \leftarrow T \Delta \{n\}$
  else
    $j \leftarrow n$
    while $j \notin T$ **and** $j > 0$
      do $j \leftarrow j - 1$
    if $j = 1$
      then return ("undefined")
    $U \leftarrow T \Delta \{j - 1\}$
    return (U)

Здесь символом Δ обозначена симметрическая разность двух множеств, то есть

$$T_1 \Delta T_2 = (T_1 \backslash T_2) \cup (T_2 \backslash T_1).$$

Например, $\{1, 2, 3, 4\} \Delta \{4\} = \{1, 2, 3\}$.

Задание

Задание 1. На языке C/C++ написать программу генерации подмножеств в лексикографическом порядке. Программа должна обеспечивать следующую функциональность.

1. Вычисление номера заданного подмножества.
2. Генерация подмножества по заданному номеру.
3. Генерация всех подмножеств множества $A = \{1, \dots, n\}$ при заданном n.
4. Генерация по заданному подмножеству следующего подмножества.

Задание 2. На языке C/C++ написать программу генерации всех подмножеств множества $A = \{1, \dots, n\}$ при заданном n в порядке минимального изменения. *Использовать рекурсивный алгоритм*.

Задание 3. На языке C/C++ написать программу генерации всех подмножеств множества $A = \{1, \dots, n\}$ при заданном n в порядке минимального изменения. *Использовать нерекурсивный алгоритм*.

Требования к отчету

Отчет по лабораторной работе должен включать:

1. Титульный лист; задание; исходный код.
2. Примеры работы программы (скриншоты).
3. Выводы.

Литература

Kreher D. L., Stinson D. R. Combinatorial algorithms. Generation, enumeration, and search. Berlin: Springer, 1999. 344 p. (DMA Discrete Mathematics and its Applications).