1 Генерирование разбиений заданного множества

Задание 1. Сгенерировать все разбиения заданного множества.

Пусть $X=\{x_1,x_2,\ldots,x_n\}$ – произвольное множество. Если каждому элементу x_i сопоставить его номер i, то, не нарушая общности, можем считать, что множество X имеет вид: $X=\{1,2,\ldots,n\}$.

Под ***разбиением*** n-элементного множества X на k блоков будем понимать произвольное семейство $\pi=\{B_1,\ldots,B_k\}$, такое что $B_1\cup\ldots\cup B_k=X$, $B_i\cap B_j=\varnothing$ для $1\le i<j\le k$, и $B_i\neq\varnothing$ для $1\le i\le k$. Подмножества $B_1,\ldots,B_k$ будем называть *блоками* семейства π. Множество всех разбиений множества X на k блоков будем обозначать $\Pi_k(X)$, а множество всех разбиений через $\Pi(X)$. Очевидно, что $\Pi(X)=\Pi_1(X)\cup\ldots\cup\Pi_n(X)$ (более того, $\{\Pi_1(X),\ldots,\Pi_n(X)\}$ является разбиением множества $\Pi(X)$).

Опишем алгоритм генерирования всех разбиений множества. Идею этого алгоритма легче всего объяснить, сформулировав его в рекуррентной форме. Отметим сначала, что каждое разбиение π множества $\{1,\ldots,n\}$ однозначно определяет разбиение π_{n-1} множества $\{1,\ldots,n-1\}$, возникшее из π после удаления элемента n из соответствующего блока (и удаления образовавшегося пустого блока, если элемент n образовывал одноэлементный блок). Напротив, если дано разбиение $\sigma=\{B_1,\ldots,B_k\}$ множества $\{1,\ldots,n-1\}$, легко найти все разбиения π множества $\{1,\ldots,n\}$, такие что $\pi_{n-1}=\sigma$, т.е. следующие разбиения:

$$\begin{gathered} B_1\cup\{n\},B_2,\ldots,B_k,\\ B_1,B_2\cup\{n\},\ldots,B_k,\\ \ldots\\ B_1,B_2,\ldots,B_k\cup\{n\},\\ B_1,B_2,\ldots,B_k,\{n\}. \end{gathered}\qquad (*)$$

Это дает следующий рекуррентный метод генерирования всех разбиений: если дан список L_{n-1} всех разбиений множества $\{1,\ldots,n-1\}$, то список L_n всех разбиений множества $\{1,\ldots,n\}$ можно создавать, заменяя каждое разбиение σ в списке L_{n-1} на соответствующую ему последовательность (*). При этом мы можем гарантировать, что каждое следующее разбиение в списке образуется из предыдущего посредством удаления некоторого элемента из некоторого блока (это может повлечь за собой удаление одноэлементного блока) и добавления его в другой блок либо создания из него одноэлементного блока.

Если обратить порядок последовательности (*) для каждого второго разбиения списка L_{n-1}, то элемент n будет двигаться попеременно вперед и назад, и разбиения «на стыке» последовательностей, образованных из соседних разбиений списка L_{n-1}, будут мало отличаться друг от друга (при условии, что соседние разбиения списка L_{n-1} мало отличаются друг от друга). На рис.1.9 показано построение списка L_n для n= 1,2,3.

Приведем нерекуррентную реализацию этого алгоритма. Разбиение множества {1,…,n} будем представлять с помощью последовательности блоков, упорядоченной по возрастанию самого маленького элемента в блоке. Этот наименьший элемент блока будем называть номером блока. Отметим, что номера соседних блоков, вообще говоря, не являются соседними натуральными числами. В этом алгоритме будем использовать переменные ПРЕД [i], СЛЕД [i], $1 \le i \le n$, содержащие соответственно номер предыдущего и номер следующего блока с номером i (СЛЕД [i] = 0, если блок с номером i является последним блоком разбиения). Для каждого элемента i, $1 \le i \le n$, номер блока, содержащего элемент i, будет храниться в переменной БЛОК[i], направление, в котором «движется» элемент i, будет закодировано в булевской переменной ВПЕР [i] (ВПЕР [i] = true, если i движется вперед).

А л г о р и т м
генерирования всех разбиений множества {1,…,*n*}

Данные: n.

Результат: последовательность всех разбиений множества {1,…,n}, в котором каждое следующее разбиение образуется из предыдущего путем перенесения единственного элемента в другой блок.

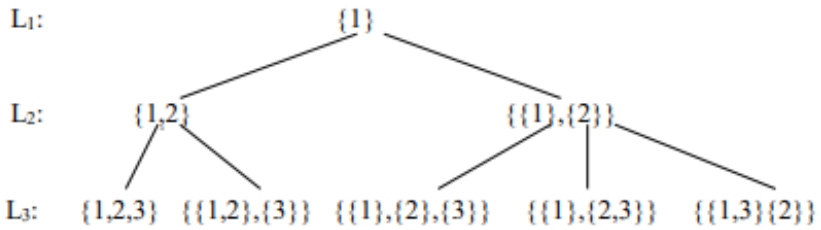

Рис.1. Генерирование разбиений множества {1,2,3}

Алгоритм

1) Поместить все элементы множества в первый блок, т.е. БЛОК[i]=1, i=1,… n. Положить ВПЕР[i] = true, i = 1,…,n.
2) Положить СЛЕД[1] = 0.

3) Выписать разбиение.
4) Положить $j=n$ (j – активный элемент).
5) Положить k=БЛОК[j].
6) Если ВПЕР[j]=true, т.е. j движется вперед, то перейти к шагу 7, иначе к шагу 10.
7) Если СЛЕД[k]=0, т.е. k – последний блок, то положить СЛЕД[k]=j, ПРЕД[j]=k и СЛЕД[j]=0.
8) Если СЛЕД[k] > j (j образует новый блок), то ПРЕД[j]=k, СЛЕД[j] = СЛЕД[k], ПРЕД[СЛЕД[j]]=j, СЛЕД[k]=j.
9) БЛОК[j]=СЛЕД[k] и перейти к шагу 15.
10) Если ВПЕР[j]=false, т.е. j движется назад, то положить БЛОК[j]= =ПРЕД[k].
11) Если $k = j$, т.е. j образует одноэлементный блок, то перейти к шагу 12, иначе к шагу 15.
12) Если СЛЕД[k] = 0, то перейти к шагу 13, иначе к шагу 14.
13) Положить СЛЕД[ПРЕД[k]]=0 и перейти к шагу 15.
14) Положить СЛЕД[ПРЕД[k]]=СЛЕД[k], ПРЕД[СЛЕД[k]]= ПРЕД[k].
15) Выписать полученное разбиение.
16) Положить $j = n$.
17) Если $j = 1$, то алгоритм заканчивает работу. Если $j > 1$ и ((ВПЕР[j] and (БЛОК[j]=j))=true или (not ВПЕР[j] and (БЛОК[j]=1))=true), то перейти к шагу 18, иначе к шагу 5.
18) Положить ВПЕР[j] = not ВПЕР[j].
19) Положить $j = j - 1$ и перейти к шагу 17.

Этот алгоритм строит сначала разбиение $\{\{1,\ldots,n\}\}$ (шаг 1). Заметим, что это первое разбиение в списке L_n , созданном при помощи описанного выше рекуррентного метода. Задача основного цикла (шаг 17) – перемещение «активного» элемента j в соседний блок – предыдущий или последующий (в последнем случае может возникнуть необходимость создания нового блока вида $\{j\}$), а затем определение активного элемента во вновь образованном разбиении. Из описанного рекуррентного построения следует, что данный элемент перемещается только тогда, когда все элементы, которые больше его, достигают своего крайнего левого или правого положения; точнее, активный элемент j^* является таким наименьшим элементом, что для каждого большего элемента j выполняется одно из двух следующих условий:

1. ВПЕР[j] and (БЛОК[j]=j), т.е. элемент движется вперед и достигает своего крайнего правого положения (очевидно, j не может быть элементом блока с наименьшим элементом, большим j).
2. not ВПЕР[j] and (БЛОК[j] = 1), т.е. элемент j движется назад и достигает своего крайнего левого положения (в первом блоке).

Этот принцип реализуется в шагах 16 – 19. Заодно меняется направление движения элементов $j > j^*$. Дополнительным условием данного цикла является $j > 1$, так как из самого представления разбиения следует, что $j = 1$ не может быть активным элементом (очевидно, что элемент 1 всегда является элементом блока с номером 1). Если каждый из элементов $j>1$ отвечает условию 1) или 2), то легко убедиться, что уже порождены все разбиения. В таком случае на выходе цикла имеем $j = 1$ и следует выход из основного цикла, т.е. имеем окончание работы алгоритма. Из рекуррентности алгоритма вытекает также, что активным элементом для первого разбиения списка L_n, т.е. для $\{\{1,\ldots,n\}\}$, является элемент n. Такое же значение приписывается переменной j перед входом в цикл (шаг 16).

Проанализируем теперь процесс переноса активного элемента (шаги 5 – 14). Сначала находим номер блока, содержащего активный элемент. Пусть это будет k. Если этот элемент движется вперед, то достаточно перенести его в блок с номером СЛЕД[k] (шаг 9), а в двух остальных случаях переменную СЛЕД[k] нужно сначала модифицировать. Первый случай имеет место, когда СЛЕД[k]=0, т.е. когда k есть номер последнего блока разбиения. Тогда j образует одноэлементный блок; при этом достаточно принять СЛЕД[k]=j и соответственно изменить значения переменных СЛЕД[j] и ПРЕД[j] (шаг 7). Второй случай имеет место, когда СЛЕД[k]>j, он рассматривается аналогично. Условие СЛЕД[k]> j означает, что все блоки справа от блока с номером k содержат элементы, большие j (все эти элементы занимают свои крайние правые позиции, в противном случае j не был бы активным элементом). Из рекуррентности алгоритма легко вытекает, что в этом случае нужно создать одноэлементный блок, содержащий j. Это выполняется в п.8 (единственная разница с первым случаем состоит в том, что в данном случае вновь созданный блок не является последним блоком разбиения).

В случае, когда элемент j движется назад (шаг 10) достаточно поместить его в предыдущий блок и выполнить соответствующее изменение значений переменных СЛЕД и ПРЕД, если j создавал одноэлементный блок – это имеет место тогда, когда БЛОК[j]=k= j, т.к. каждый элемент $m>j$ блока с номером j был бы выбран активным элементом в цикле 17.

Построим все разбиения для n = 4.

{1,2,3,4}
{1,2,3} {4}
{1,2} {3} {4}
{1,2} {3,4}
{1,2,4} {3}
{1,4} {2} {3}
{1} {2,4} {3}
{1} {2} {3,4}
{1} {2} {3} {4}
{1} {2,3} {4}
{1} {2,3,4}
{1,4} {2,3}
{1,3,4} {2}
{1,3} {2,4}
{1,3} {2} {4}