Отчёт по дисциплине “Дискретная математика”

ЛАБОРАТОРНАЯ №1

23631/2, ИЛЬЯ КОЗЛОВ

Contents

Постановка задачи

Реализовать пакет программ, предназначенный для точного вычисления основных комбинаторных чисел, рассматриваемых в курсе основы дискретной математики для программистов.

Пакет должен обеспечивать вычисления следующих комбинаторных чисел:

- Число размещений без повторений A(m,n)
- Число размещений U(m,n)
- Число перестановок P(n)
- Число сочетаний C(m,n)
- Число Стирлинга второго рода S(m,n)
- Число Белла B(n)

Пакет имеет простейший интерфейс типа "командная строка".

Предусмотрены следующие команды:

- H - получение справки
- Q - завершение работы
- U, A, P, C, S, B - вычисление соответствующего комбинаторного числа.

Входными данными (параметрами) могут быть произвольные целые числа (возможно, со знаком), записанные в обычной позиционной десятичной системе счисления. Пакет обеспечивает вычисления для всех значений параметров, указанных в учебнике (Ф.А.Н. «Дискретная математика для программистов»), в том числе для тех, для которых значения соответствующего комбинаторного числа приписаны определением или соглашением, а не формулой. В случае нарушения любого из указанных условий пакет выдает сообщение об ошибке, диагностирующее, что именно было введено неправильно. При этом работоспособность пакета сохраняется. Результатом вычислений является целое число, записанное в обычной позиционной десятичной системе счисления.

Пакет должен обеспечивть точное (в математическом смысле) вычисление значения комбинаторного числа во всех возможных случаях, когда параметры и само значение представимы 32-битными целыми числами (0..4294967295).

Решение задачи

Решение проблемы переполнения

Умножение

В условиях наложенных ограничениях для отработки переполения и «безопасного» умножения комбинаторных чисел была реализована следующая функция:

```
uint calc::comb::mul(uint a, uint b, error * er)
{
  if (b != 0 && UINT32_MAX / b < a)
  {
    *er = ERR_MUL_OVERFLOW;
    return -1;
  }

  return a * b;
}
```

Пояснение: если UINT32_MAX (= 4294967295) / b < a, то a * b > 4294967295, что приводит к переполнению.

Сложение

Точно так же необходима реализация «безопасного» сложения:

```
uint calc::comb::add(uint a, uint b, error * er)
{
  uint max = a > b ? a : b, res = a + b;

  if (res < max)
    *er = ERR_SUM_OVERFLOW;

  return res;
}
```

Пояснение: так как программа работает с 32 битными числами (uint), то сумма a + b представима все теми же 32 битами, однако если a + b > 4294967295, то эта сумма уже не может быть правильно представлена 32 битами, поэтому a + b представляет число [(a + b) – 4294967295], что заведомо меньше max(a, b), а значит, если мы сравним результат суммы с max(a, b), то сможем избежать ошибочного сложения и вывести ошибку переполнения.

Число размещений без повторений A(m, n)

$$A(m,n) = \frac{m!}{(m-n)!} = m * (m-1) * \ldots * (m-n+1)$$

$$A(m,n) \stackrel{\text{def}}{=} 0, n > m \;\&\; A(m,0) \stackrel{\text{def}}{=} 1$$

В данном случае функция представляет из себя просто последовательное «безопасное» умножение от m до (m-n+1).

Пояснение: максимум A(m, n) наблюдается при n = m - 1, в этом случае перемножение представляет из себя обычный факториал => переполнение наблюдается уже при A(13, 12).

Число размещений U(m, n)

$$U(m,n) = m^n$$

$$U(m,0) \stackrel{\text{def}}{=} 1$$

Функция представляет из себя бинарную реализацию степени.

Пояснение: бинарная реализация позволяет сократить время исполнения с O(n) до O(logn).

Число перестановок P(n)

$$P(n) = n!$$

$$P(0) \stackrel{\text{def}}{=} 1$$

Функция представляет из себя последовательное «безопасное »перемножение от 2 до n.

Число сочетаний C(m, n)

$$C(m,n) = \frac{m!}{n!\,(m-n)!}$$

$C(m,n) \stackrel{\text{def}}{=} 0, n > m$; $C(m,n) \stackrel{\text{def}}{=} m, n = 1$ или $m = n + 1$; $C(m,n) \stackrel{\text{def}}{=} 1, m = n$

Но формула эта крайне невыгодная, на деле используется рекуррентная:

$$C(m,n) = C(m-1,n) + C(m-1,n-1)$$

Совместно с «повернутым» треугольником Паскаля:

1	1	1	1	1	1
1	2	3	4	5	
1	3	6	10		
1	4	10			
1	5				
1					

Можно воспользоваться симметричностью и при n > m – n, задавать n = m – n

В таком случае функция реализуется как «пробег» по m – n + 1 строке представленного треугольника с заполнением лишь одного массива размера n+1 с помощью рекурентной формулы, причём ответ будет находится в n+1-ой ячейке массива.

Пояснение: максимум C(m, n) наблюдается при n = m / 2. То есть, в виду переполнения факториала уже при n = 13, для формулы через факториалы максимальное значение – это C(12, 6) = 924, а для приведенного алгоритма максимальное m = 34, C(34, 17), что существенно расширяет вычислительные способности пакета.

Число Стрилинга второго рода S(m, n)

$$S(m,n) = S(m-1,n-1) + nS(m-1,n)$$

$$S(m,n) \stackrel{\text{def}}{=} 1, m-n\,; S(m,0) \stackrel{\text{def}}{=} 0, m>0\,; S(m,n) \stackrel{\text{def}}{=} 0, n>m\,; S(m,1) \stackrel{\text{def}}{=} 1$$

Функция реализуется путём построения таблицы чисел Стирлинга, используя рекуррентную формулу

$m \setminus n$	0	1	2	3	4	5	6	7
0	1							
1	$\boxed{0}$	$\boxed{1}$						
2	$\boxed{0}$	$\boxed{1}$	$\boxed{1}$					
3	$\boxed{0}$	$\boxed{1}$	$\boxed{3}$	$\boxed{1}$				
4	$\boxed{0}$	$\boxed{1}$	$\boxed{7}$	$\boxed{6}$	1			
5	0	$\boxed{1}$	$\boxed{15}$	$\boxed{25}$	10	1		
6	0	1	$\boxed{31}$	$\boxed{90}$	65	15	1	
7	0	1	63	$\boxed{301}$	350	140	21	1

Пояснение: хранится только одна диагональ длины n, которая вначале инициализируется единичным значением. Далее на каждом шаге цикла по i вычисляется диагональ, начинающаяся с i-ой строки. Таким образом вычисляются только необходимые промежуточные элементы, причём по одному разу.

Число Белла B(n)

$$B(m) = \sum_{n=0}^{m} S(m,n)$$

$$B(0) \stackrel{\text{def}}{=} 1$$

В прямом виде число Белла не считается, для этого используется следующая рекуррентная формула:

$$B(m+1) = \sum_{i=0}^{m} C(m,i)B(i)$$

совместно с соответсвующим треугольником Белла, хранящимся в виде нижней треугольной матрицы:

1					
1	2				
2	3	5			
5	7	10	15		
15	20	27	37	52	
.	.	.	.	.	.

Тогда функция представляет из себя «пробег» по m строчкам матрицы с заполнением одного массива размера m в соотвествии с рекуррентной формулой, причём итоговый ответ будет в m-той ячейке массива.

Вычислительный эксперимент

В первую очередь были проведены проверки на значения по определению.

Далее были проверки частных краевых случаев, по типу P(12) и P(13), C(12, 6) и C(222, 5) и C(13, 6), A(13, 12), U(2, 31) и U(2, 32) и т.д.

Так же были проведены проверки частичных «обычных» случаев, то есть для значений из области О.Д.З.

Замечания

- Меня попросили убрать излишние проверки (допустим в U я проверял дополнительно n на 0, однако в теле функции реализован цикл while (n), в виду чего эта проверка лишняя)
- В виду того, что я реализовал «систему ошибок», меня попросили её дополнить, чтобы она могла выдавать более точный результат ошибки, то есть не только, что произошла ошибка умножения, но и в какой функции конкретно.

Источники

- Ф.А.Н. «Дискретная математика для програмиистов», 3-е издание
- http://en.cppreference.com/w/