Санкт-Петербургский Политехнический Университет Петра Великого
Институт прикладной математики и механики
Кафедра прикладной математики

Дискретная математика

Лабораторная работа 1. Точные вычисления комбинаторных чисел

Преподаватель: Стахов С. В.

Студент: Колесник В. Н.

23631/1

Оглавление

I. Постановка задачи

Пакет должен обеспечивать вычисления следующих комбинаторных чисел:

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

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

II. Решение задачи и обоснование

1. Проверки переполнения

Для контроля переполнения производятся следующие проверки:

1) Проверка при сложении:
```
if (MaxUINT32 - a < *this)
  throw ADD_OVERFLOW;
```
2) Проверка при умножении:
```
if ((double)MaxUINT32.value_ / a.value_ < value_)
  throw MUL_OVERFLOW;
```
3) Проверка при вычитании:
```
if (*this < a)
  throw SUB_OVERFLOW;
```
4) Проверка при делении:
```
if (a == UINT32(0u))
  throw DIV_BY_ZERO;
```

2. Размещения

Определение

Число всех функций $f: 1..n \to 1..m$ (при отсутствии ограничений), или число всех возможных способов разместить n объектов по m ящикам, называется числом размещений:

$$U(m,n)$$

Теорема

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

Доказательство

Так как ограничений нет, объекты размещаются независимо друг от друга, то есть каждый из n объектов можно разместить m способами. Таким образом, по правилу произведения получаем m^n возможных размещений.

Реализация

Использовалось бинарное возведение в степень. В обычном цикле количество операций равно m. При бинарном возведении в степень $\log(m)$ операций

```
UINT32 Placing(byte m, byte n) {
  UINT32 U(1);
  UINT32 M(m);
  while (n) {
    if (n & 1) {
      U *= M;
      --n;
    }
    else {
      M *= M;
      n >>= 1;
    }
  }
  return U;
}
```

Реализация вытекает из определения бинарного возведения в степень.

3. Размещения без повторений

Определение

Число инъективных функций $f: 1..n \to 1..m$, или число способов разместить n объектов по m ящикам, не более чем по одному в ящик, называется числом размещений без повторений:

$$A(m,n)$$

Теорема

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

Доказательство

Ящик для первого объекта можно выбрать m способами, для второго – m-1 способами и т.д. Таким образом $A(m,n) = m * (m-1) * \ldots * (m-n+1) = \frac{m!}{(m-n)!}$

Граничные случаи

$A(m,n) = 0$ при $n > m$

$A(m,0) = 1$

Реализация

```
UINT32 PlacingWithoutRepeat(byte m, byte n) {
  if (n > m)
    return UINT32(0);
  if (n == 0)
    return UINT32(1);

  UINT32 A(m);
  for (byte i = m - n + 1; i < m; ++i)
    A *= i;
  return A;
}
```

В начале обработаны граничные случаи. Реализация вытекает из определения $A(m,n) = m * (m-1) * \ldots * (m-n+1)$

4. Перестановки

Определение

Число взаимно-однозначных функций $f: 1..n \rightarrow 1..n$, или число перестановок n предметов обозначается:

$$P(n)$$

Теорема

$$P(n) = n!$$

Доказательство

$$P(n) = A(n,n) = n * (n-1) * \ldots * (n-n+1) = n * (n-1) * \ldots * 1 = n!$$

Реализация

```
UINT32 Permutation(byte n) {
  UINT32 P(1);
  for (byte i = 2; i <= n; ++i)
    P *= UINT32(i);
  return P;
}
```

Реализация вытекает из определения

5. Сочетания

Определение

Число строго монотонно возрастающих функций $f \quad 1..n \rightarrow 1..m$, или число размещений n неразличимых предметов по m ящикам не более чем по одному в ящик, то есть число способов выбрать из m ящиков n ящиков с предметами, называется числом сочетаний:

$$C(m,n)$$

Граничные случаи

$$C(m,n) = 0 \text{ при } n > m$$

Реализация

```
UINT32 Combination(byte m, byte n) {
  if (n > m)
    return UINT32(0);

  if (n > m - n)
    n = m - n;

  UINT32 *A = new UINT32[n + 1];
  for (byte i = 0; i != n + 1; ++i)
    A[i] = UINT32(1);

  for (byte i = 0; i != m - n; ++i)
    for (byte j = 1; j != n + 1; ++j)
      A[j] = A[j] + A[j - 1];

  UINT32 Abuf = A[n];
  delete[] A;
  return Abuf;
}
```

В начале обрабатывается граничный случай и, если возможно, уменьшается n, что можно сделать, исходя из следующего тождества:

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

Для реализации использован треугольник Паскаля. Число сочетаний C(m,n) находится в (m+1)-м ряду на (n+1)-м месте. Весь треугольник хранить в памяти нет нужды, достаточно хранить только одну строку. Первая строка содержит все единицы, а каждая следующая получается перевычислением слева направо по формуле:

$$A[y] \coloneqq A[j] + A[j-1]$$

При этом в правой части оператора A[j] = это «старое» значение, то есть элемент из «верхней» строки, а A[j-1] – только что вычисленное значение слева. Перевычислив таким образом массив m-n раз получаем C(m,n)=A[n].

6. Число Стирлинга второго рода

Определение

Число разбиений m-элементного множества на n блоков, или число способов разложить m объектов по n неразличимым ящикам, называется числом Стирлинга первого рода:

$$S(m,n)$$

Граничные случаи

$$S(m,0) = 0 \text{ при } m > 0$$

$$S(m,n) = 0 \text{ при } n > m$$

Теорема

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

Реализация

```
UINT32 StirlingNumberSecondKind(byte m, byte n) {
  if (n == 0)
    return UINT32(0);
  if (n > m)
    return UINT32(0);

  byte d = (m - n + 1 < n) ? (m - n + 1) : n;
  byte l = (m - n + 1 > n) ? (m - n + 1) : n;
  UINT32 *A = new UINT32[d];
  for (byte i = 0; i != d; ++i)
    A[i] = UINT32(1);
  if (d == n)
    for (byte i = 1; i != l; ++i)
      for (byte j = 1; j != d; ++j)
        A[j] = A[j - 1] + A[j] * UINT32(j + 1);
  else
    for (byte i = 1; i != l; ++i)
      for (byte j = 1; j != d; ++j)
        A[j] = A[j] + A[j - 1] * UINT32(i + 1);
  UINT32 Abuf = A[d - 1];
  delete[] A;
  return Abuf;
}
```

В начале обрабатываются граничные случаи.

Чтобы вычислить число Стирлинга, необходимо вычислить все значения в «параллелограмме». При этом хранить достаточно лишь рабочий массив, равный длине более короткой стороны «параллелограмма». Количество повторений соответствует

более длинной стороне «параллелограмма». Таким образом рабочий массив соответствует либо диагонали, которая на каждой итерации опускается вниз, либо столбцу, который на каждой итерации сдвигается вправо вниз. В любом случае вначале рабочий массив инициализируется значениями 1. Далее на каждом шаге по I перевычисляется рабочий массив. Таким образом, вычисляются только необходимые промежуточные элементы, причем по одному разу.

7. Число Белла

Определение

Число всех разбиений m-элементного множества называется числом Белла:

$B(n)$

Граничные случаи

$B(0) = 1$

Теорема

$B(m) = A_{m+1,1} = A_{m,m}$ ($A_{i,j}$ – j-ый элемент в i-ой строке треугольника Белла)

Реализация

```
UINT32 BellNumber(byte n) {
  if (n == 0)
    return UINT32(1);
  UINT32 *A = new UINT32[n];
  A[0] = UINT32(1);
  for (byte k = 1; k != n; ++k) {
    UINT32 buf = A[0];
    A[0] = A[k - 1];
    for (byte i = 1; i != k + 1; ++i) {
      UINT32 buf2 = A[i];
      A[i] = A[i - 1] + buf;
      buf = buf2;
    }
  }
  UINT32 Abuf = A[n - 1];
  delete[] A;
  return Abuf;
}
```

В реализации используется треугольник Белла – бесконечная треугольная матрица, где каждое число (кроме чисел на левой боковой стороне) является суммой 2 чисел: стоящего слева от него в этой же строке и слева над ним. Первое число в каждой строке (кроме первой) является последним числом из предыдущей строки. Число в первой строке – единица.

Алгоритм вытекает из приведенной теоремы и определения треугольника Белла.

III. Выводы

Программа вычисляет основные комбинаторные числа. Для этого были использованы наиболее эффективные алгоритмы и проведен контроль переполнения. Оказалось, что для числа размещений без повторений более простая формула без деления оказалась эффективней более «математической» и «красивой» в определении. Для вычисления числа сочетаний, числа Стирлинга первого рода и числа Белла алгоритмы с использованием дополнительной памяти оказались лучше рекуррентных формул.

Вычисление числа размещений было улучшено бинарным возведением в степень. А для вычисления числа перестановок не нашлось алгоритма лучше, чем алгоритм, основанный на определении.

IV. Список литературы

- Ф. А. Новиков «Дискретная математика», 2-ое издание