**Обработка разреженных матриц**

**Цель работы** – реализовать алгоритмы обработки разреженных матриц, сравнить эти алгоритмы со стандартными алгоритмами обработки матриц.

**Входные данные:**

N (натуральное) – количество ненулевых элементов вектора, N целых чисел-элементов вектора.

M( натуральное) –количество ненулевых элементов матрицы(M<=N^2), M- целых чисел-элементов вектора.

K(натуральное) – количество строчек в матрице(K<=N), К целых чисел – указателей на новую строчку матрицы.

**Выходные данные:**

Затраченная память для хранения в байтах, затраченное время на умножение( в мс) для умножения в разреженном виде и умножения в стандартном виде, результат умножения матриц в стандартном виде и в разреженном строчном формате.

**Внутренние структуры данных**

Разреженный строчный формат:

std::vector<int>ia; вектор для хранения номеров столбцов первой матрицы

std:: vector<вt> a; - вектор для хранения значений первой матрицы

std:: list<int> ja; - список для хранения указателей на новые строчки первой матрицы

std::vector<int>ia1; - вектор для хранения номеров столбцов вектора

std:: vector<int>a1- вектор для хранения значений вектора

std::vector<int> result – вектор для хранения значения результатов умножение

sdt::vector<int> result1 –вектор для хранения номеров столбцов значений результатов умножения

Стандартный вид:

vector<vector<int> > a\_normal; - матрица

vector <int> a1\_normal; -вектор

vector<int> out; - вектор для хранения результатов умножения

**Основные алгоритмы**

Умножение в разреженном виде

**int** y=ja.front();

**for**(**int** i=0;i<h;i++)

{

**if** (i==y)

{

ja.pop\_front();

y=ja.front();

j1++;

}

count1++;

result[ia[i]]+=(a[i]\*ip[j1]);

result1[ia[i]]=ia[i];

}

Умножение в стандартном виде

**for** (**int** ix = 0; ix < size\_n; ix++)

{

out[ix] = 0;

**for** (**int** jx = 0; jx < max1; jx++)

{

count2++;out[ix] += a\_normal[jx][ix] \* a1\_normal[jx];

;

}}

**Эффективность разреженных матриц**

Матрица 10000 элементов(умножение на вектор размера 100 с таким же % разреженности)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процент заполнения | Время умножения в разр. строчном формате | Время умножения в стандартном виде | Затраченная память(байт)  Ст. вид | Затраченная память(байт) в разреженном строчном формате | Выигрыш по времени % | Выигрыш по памяти% |
| 10 | 3.360690e-002 | 1.191379e-001 | 40800 | 9272 | 260 | 340 |
| 20 | 7.537241e-002 | 1.151021e-001 | 40800 | 17352 | 53 | 135 |
| 30 | 1.382710e-001 | 1.658600e-001 | 40800 | 25432 | 19 | 60 |
| 40 | 3.700300e-001 | 2.227945e-001 | 40800 | 33512 | -68 | 21 |
| 50 | 4.060252e-001 | 1.769372e-001 | 40800 | 41592 | -135 | -2 |

**Вывод:** Использование разреженного строчного формата имеет смысл при проценте заполнение матрицы менее 40, тогда мы получаем выигрыш по времени и по памяти. При заполнении матрицы 40-50% мы получаем только выигрыш по памяти. При заполнении матрицы выше 50% идет значительный проигрыш по времени и по памяти по сравнению со стандартным алгоритмом обработки матриц.

.

Тесты:

|  |  |  |
| --- | --- | --- |
|  | Ввод | Вывод |
| Пустой столбец | 5  0 1 2 3 5  1 2 3 4 5  9  0 1 2 3 4 1 2 3 4  1 1 2 3 5 8 3 7 4  5  0 5 6 7 8 | 1 17 11 31 5  0 1 2 3 4 |
| Отрицательное количество элементов вектора | Input number of non-zero elements of array:-2 | Repeat input |
| Отрицательный номер столбца | Input vector of number of colums(number must be positive)  1 -2 3 4 | Error  Repeat input |
| Пустые строки | 5  0 1 2 3 5  1 2 3 4 5  9  0 1 2 3 4 1 2 3 4  1 1 2 3 5 8 3 7 4  10 | There are full-zero lines, it is error |
| Некорректный указатель на новую строку | 5  0 1 2 3 5  1 2 3 4 5  9  0 1 2 3 4 1 2 3 4  1 1 2 3 5 8 3 7 4  5  0 5 6 7 10 | Error, incorrect number of line |