ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук Департамент программной инженерии

КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ НАХОЖДЕНИЯ МАТРИЦЫ АЛГЕБРАИЧЕСКИХ ДОПОЛНЕНИЙ С ПРИМЕНЕНИЕМ OPENMP

Пояснительная записка

		Исполнитель:
		студент группы БПИ199
		К. Н. Борисов
,,	**	2020 =

Содержание

1	Текст задания	3
2	Применяемые расчетные методы 2.1 Алгоритм вычисления 2.2 Распределение по потокам 2.3 Вывод результата в консоль 2.4 Входные данные 2.3 Входные данны	4 4 4
3	Тестовые примеры	5
4	Список использованной литературы	7
5	Текст программы	8

1. Текст задания

Найти алгебраическое дополнение для каждого элемента матрицы. Входные данные: целое положительное число n, произвольная матрица A размерности $n \times n$. Количество потоков является входным параметром, при этом размерность матриц может быть не кратна количеству потоков. Использовать OpenMP. [1]

2. Применяемые расчетные методы

2.1. Алгоритм вычисления

Алгебраическое дополнение это $A_{ij}=(-1)^{i+j}M_{ij}$, где M_{ij} – дополнительный минор, определитель матрицы, получающейся из исходной матрицы A путём вычёркивания i-й строки и j-го столбца. [2] Для вычисления определителя использовался рекурсивный алгоритм $\Delta=\sum_{j=1}^{n}(-1)^{1+j}a_{1j}M_{j}^{1}$. [3] Именно поэтому программа не работает на матрицах больше чем 10×10 .

2.2. Распределение по потокам

В предыдущей задачке распределять по потокам надо было ручками, и там это была самая сложная часть. А тут за нас это делает OpenMP. И поэтому эта задачка получилась раза в два проще. Тут мы как будто пишем однопоточный код и он сам становится многопоточным. Нужно только написать #pragma omp parallel for, подключить omp.h и передать опцию -fopenmp компилятору, и наш код магически становится быстрее.

2.3. Вывод результата в консоль

В консоль матрицы выводятся в JSON-подобном формате, чтобы их можно было сразу вставить в питон и проверить правильность результата.

2.4. Входные данные

На вход в аргументах командной строки (argv) подаются количество потоков и имя входного файла с матрицей. Имя входного файла опционально: если его нет то матрица будет читаться из консоли (stdin). Файл с матрицей имеет следующий формат: на перовой строке два целых числа — размеры матрицы n и m, потом идут $n \times m$ действительных чисел, разделённых пробельными символами ('\n' или ' ').

3. Тестовые примеры

Программа корректно работает с большими матрицами. Она даже может выбирать оптимальное количество потоков, если ей дать 0. (см. рисунок 1)

orta	n122@ThinkPad	-T480: /mnt/c/l	Jsers/user/OneD	rive/dz2020/asn	n/task04					
		-T480:/mnt/	c/Users/user	/OneDrive/dz	:2 <mark>020/asm/tas</mark>	k04\$ time ./	matrix_openm	p 0 test100.	txt	
nputt	ed matrix:									
[0,	3,	1,	4,	1,	4,	4,	1,	0,	4,],
Ī	0,	0,	3,	1,	0,	4,	1,	0,	1,	1,],
Ī	0,	3,	1,	3,	2,	2,	3,	1,	0,	1,],
Ī	1,	0,	2,	0,	2,	1,	1,	1,	2,	0,]
Ī	1,	3,	0,	0,	1,	1,	0,	1,	1,	0,],
Ī	1,	2,	1,	0,	1,	1,	1,	4,	1,	1,],
Ļ	0,	3,	0,	1,	2,	4,	3,	0,	1,	0,],
Ļ	1,	1,	0,	1,	2,	0,	1,	1,	4,	1,],
Ļ	1,	0,	3,	1,	2,	1,	1,	1,	0,	4,],
L	1,	1,	1,	0,	1,	3,	1,	2,	2,	4,],
1+	ing matrix:									
Sult	ing matrix:									
Г	2184,	405,	468,	-624,	-1980,	-786,	2127,	-765,	132,	105,]
ī	-2212,	1190,	1540,	-1064,	-1036,	-798,	616,	-434,	910,	336,]
ī	1148,	-3544,	-3782,	5290,	5042,	3888,	-6092,	2572,	-2018,	-1386,
Ī	6650,	-3811,	-3398,	4036,	2390,	3186,	-2663,	1345,	-2138,	-1785,],
Ī	2030,	194,	-254,	586,	-124,	360,	-674,	-284,	-338,	-252,],
[-2842,	2336,	2518,	-3110,	-2434,	-2412,	3106,	-620,	1300,	798,],
[-3388,	2672,	2560,	-3908,	-2098,	-2454,	3568,	-1502,	1426,	1050,],
[-3206,	2242,	2228,	-2476,	-1892,	-2334,	2354,	-1102,	1808,	966,],
[-3374,	2908,	3146,	-3700,	-2180,	-2934,	3230,	-1618,	1382,	1386,],
[2968,	-3869,	-4240,	4664,	4240,	4134,	-5353,	2279,	-2014,	-1113,],
eal	0m0.654s									
ser	0m4.781s									
/S	0m0.031s	T400 - / /			2020 / //	Lord				
ortan	122@ThinkPad	-1480:/mnt/	c/Users/user	/OneDrive/dz	:2020/asm/tas	K04\$				

Рисунок 1: Большая матрица

Программа работает с большими матрицами быстрее, когда у неё несколько потоков. (см. рисунок 2)

		-T480:/ <mark>mnt</mark> /	c/Users/user	/OneDrive/dz	2 <mark>020/asm/ta</mark> s	k04\$ time ./	matrix_openn/	ıp 1 test100.	txt	
outt	ed matrix:									
	0,	3,	1,	4,	1,	4,	4,	1,	0,	4,
	0,	0,	3,	1,	0,	4,	1,	0,	1,	1,
	0,	3,	1,	3,	2,	2,	3,	1,	0,	1,
	1,	0,	2,	0,	2,	1,	1,	1,	2,	0,
	1,	3,	0,	0,	1,	1,	0,	1,	1,	0,
	1,	2,	1,	0,	1,	1,	1,	4,	1,	1,
	0,	3,	0,	1,	2,	4,	3,	0,	1,	0,
	1,	1,	0,	1,	2,	0,	1,	1,	4,	1,
	1,	0,	3,	1,	2,	1,	1,	1,	0,	4,
	1,	1,	1,	0,	1,	3,	1,	2,	2,	4,
1+	ing matrix:									
ult	Ing matrix.									
	2184,	405,	468,	-624,	-1980,	-786,	2127,	-765,	132,	105,
	-2212,	1190,	1540,	-1064,	-1036,	-798,	616,	-434,	910,	336
	1148,	-3544,	-3782,	5290,	5042,	3888,	-6092,	2572,	-2018,	-1386
	6650,	-3811,	-3398,	4036,	2390,	3186,	-2663,	1345,	-2138,	-1785
	2030,	194,	-254,	586,	-124,	360,	-674,	-284,	-338,	-252
	-2842,	2336,	2518,	-3110,	-2434,	-2412,	3106,	-620,	1300,	798
	-3388,	2672,	2560,	-3908,	-2098,	-2454,	3568,	-1502,	1426,	1050
	-3206,	2242,	2228,	-2476,	-1892,	-2334,	2354,	-1102,	1808,	966
	-3374,	2908,	3146,	-3700,	-2180,	-2934,	3230,	-1618,	1382,	1386
	2968,	-3869,	-4240,	4664,	4240,	4134,	-5353,	2279,	-2014,	-1113,
,	02 000-									
l r	0m2.008s									
	0m1.969s									

Рисунок 2: Большая матрица с одним потоком

4. Список использованной литературы

- [1] Практические приемы построения многопоточных приложений [Электронный ресурс]. //URL: http://softcraft.ru/edu/comparch/tasks/t04/ (Дата обращения: 25.11.2020, режим доступа: свободный)
- [2] Статья «Алгебраическое дополнение» Wikipedia.org //URL: https://ru.wikipedia.org/wiki/Алгебраическое дополнение (Дата обращения: 15.11.2020, режим доступа: свободный)
- [3] Статья «Определитель» Wikipedia.org //URL: https://ru.wikipedia.org/wiki/Определитель (Дата обращения: 15.11.2020, режим доступа: свободный)

5. Текст программы

```
#include <stdlib.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <time.h>
#include <assert.h>
#include <omp.h>
typedef struct Matrix {
  int n;
  int m;
  double* mat;
} Matrix;
Matrix makeMatrix(int n, int m){
  Matrix r;
  r.n = n;
  r.m = m;
  r.mat = calloc(n*m*sizeof(double), 1);
  return r;
}
// печатает матрицу в JSON-подобном формате
void printMatrix(Matrix* m){
  printf("[\n");
  for(int i = 0; i < m->n; i++){
    printf(" [");
    for(int j = 0; j < m->m; j++){
      printf("%10.6g, ", m->mat[i*m->m + j]);
    }
    printf("],\n");
  }
  printf("]\n");
}
// находит минормую матрицу (делает копию)
Matrix minorMatrix(Matrix* m, int i_, int j_){
  Matrix matrix = makeMatrix(m->n - 1, m->n - 1);
  int i1 = 0;
  for(int i = 0; i < m->n; i++){
    if(i == i_)continue;
    int j1 = 0;
    for(int j = 0; j < m->m; j++){
      if(j == j_)continue;
      matrix.mat[i1*matrix.m + j1] = m->mat[i*m->m + j];
      j1++;
    }
    i1++;
  return matrix;
}
// тут надо объявлять хедер для рекурсии
double det(Matrix* m);
// находит дополнительный минор
double minorDet(Matrix* m, int i, int j){
  Matrix t = minorMatrix(m, i, j);
  double r = det(&t);
  free(t.mat);
```

```
return r;
}
// рекурсивно считаем определитель
double det(Matrix* m){
 assert(m->m == m->n);
 assert(m->n > 1);
 if(m->n == 2){
    return m->mat[0]*m->mat[3] - m->mat[1]*m->mat[2];
 double r = 0;
 int delta = 1;
 for(int i = 0; i < m->m; i++){
    r += m->mat[i] * minorDet(m, 0, i) * delta;
    delta *= -1;
 }
 return r;
}
// находит алгебраическое дополнение
double adjunct(Matrix* m, int i, int j){
 return minorDet(m, i, j) * ((i+j) % 2? -1 : 1);
}
// считывает матрицу из файла
Matrix readMatrix(char* filename){
 FILE* file = fopen(filename, "r");
 if(!file){
    perror(filename);
    exit(1);
  }
 int n,m;
 fscanf(file, "%d %d", &n, &m);
 Matrix matrix = makeMatrix(n,m);
 for(int i = 0; i < n*m; i++){
    fscanf(file, "%lf", matrix.mat + i);
 }
 fclose(file);
 return matrix;
}
int main(int argc, char** argv){
  if(argc == 1){
    // формат командный строки для тех кому лень читать ПЗ
    fprintf(stderr, "./matrix numberOfThreads [infile]\n");
    return 1;
  }
 int threadCount = atoi(argv[1]);
 char* infile = "/dev/fd/0";
 if(argc > 2){
    infile = argv[2];
  }else{
    fprintf(stderr, "readring stdin...\n");
 Matrix matrix = readMatrix(infile);
  if(matrix.n != matrix.m){
    fprintf(stderr, "we need a square matrix\n");
    free(matrix.mat);
```

```
return 1;
  }else if(matrix.n > 10){
    fprintf(stderr, "this matrix is SO big.... and i think i cant handle it\n");
    free(matrix.mat);
    return 1;
  }
 Matrix outmatrix = makeMatrix(matrix.n, matrix.m);
 printf("Inputted matrix:\n");
 printMatrix(&matrix);
 printf("\n");
 if(threadCount > 0){
    omp_set_dynamic(0);
    omp_set_num_threads(threadCount);
 }
 #pragma omp parallel for
 for(int i = 0; i < matrix.m*matrix.m; i++){</pre>
   outmatrix.mat[i] = adjunct(&matrix, i / matrix.m, i % matrix.m);
 }
 printf("Resulting matrix:\n");
 printMatrix(&outmatrix);
 free(matrix.mat);
 free(outmatrix.mat);
 return 0;
}
```