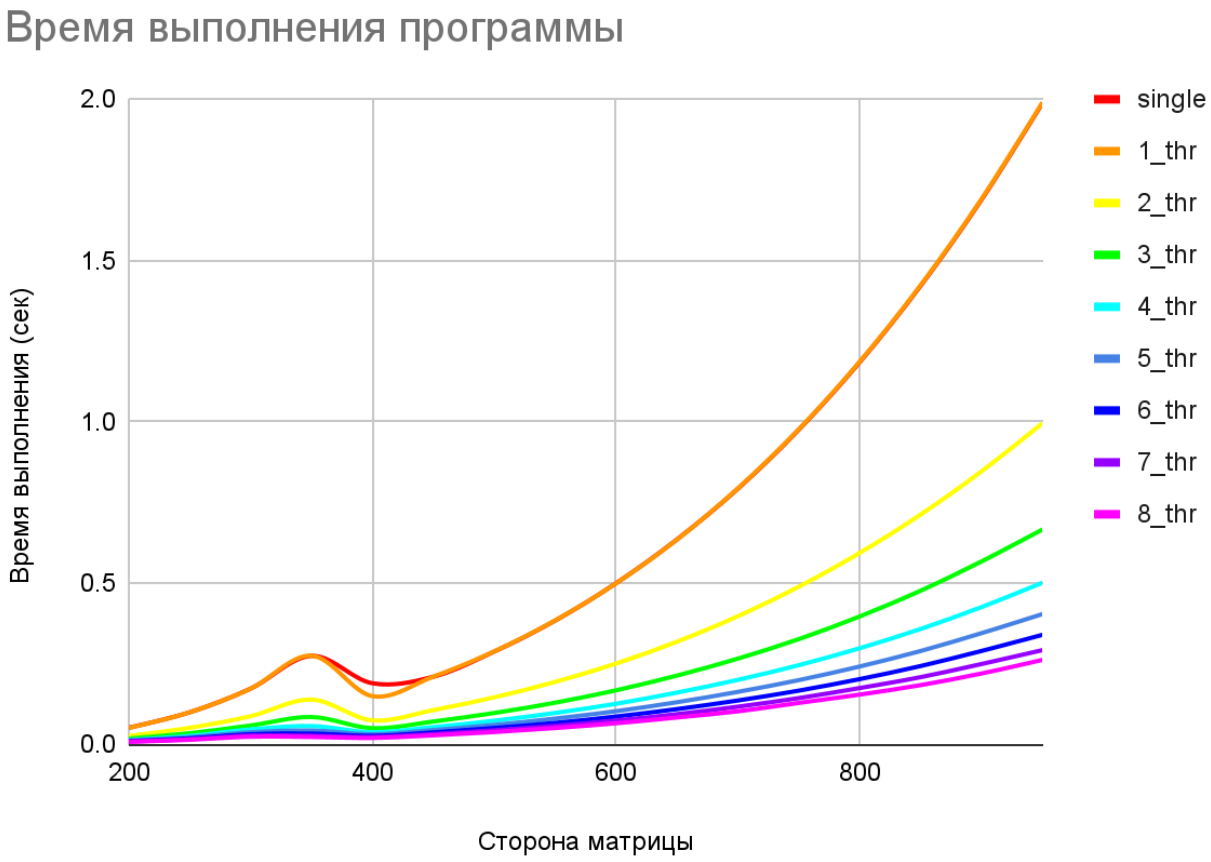
«Параллельные и распределённые вычисления» Лабораторная работа №3 Работа с OpenMP. Распараллеливание циклов.	Выполнил:	Зенин М.А.
	Преподаватель:	Катаев А.В.
	Дата выполнения:	
	Подпись преподавателя	

Цель: изучение основ работы с библиотекой openmp при распараллеливании циклов for.

9. Brute-Force matcher: Есть 2 матрицы из N(строк)*M(столбцов) float элементов, необходимо реализовать алгоритм, для каждой строки первой матрицы подбирающий номер строки во второй матрице, так, что бы сумма квадратов разностей элементов выбранных строк была минимальна. Можно использовать каждую строку второй матрицы несколько раз.



Код:

```
int* match(matrix* A, matrix* B) {
  assert(A->w == B->w && A->h == B->h);
  int* output = malloc(sizeof(float) * A->h);

  for (int y = 0; y < A->h; y++) {
    float* A_row = A->v[y];
    float min = FLT_MAX;
    int min_index = -1;

    for (int other_y = 0; other_y < A->h; other_y++) {
      float sum = 0;
      for (int x = 0; x < A->w; x++)
        sum += pow((A_row[x] - B->v[other_y][x]), 2);

      if (sum < min) {
        min = sum;
        min_index = other_y;
      }
    }

    output[y] = min_index;
  }

  return output;
}
```

```
int* match_parallel(matrix* A, matrix* B) {
  assert(A->w == B->w && A->h == B->h);
  int* output = malloc(sizeof(float) * A->h);

  #pragma omp parallel for
  for (int y = 0; y < A->h; y++) {
    float* A_row = A->v[y];
    float min = FLT_MAX;
    int min_index = -1;

    // #pragma omp parallel for
    for (int other_y = 0; other_y < A->h; other_y++) {
      float sum = 0;
      // #pragma omp parallel for reduction(+:sum)
      for (int x = 0; x < A->w; x++)
        sum += pow((A_row[x] - B->v[other_y][x]), 2);

      #pragma omp critical
      {
      if (sum < min) {
        min = sum;
        min_index = other_y;
      }
      }
    }

    output[y] = min_index;
  }
```