Εργαστήριο Παράλληλων Συστημάτων - Εργασία 2

Χρήστος Μαργιώλης

Ιανουάριος 2023

Περιεχόμενα

1 Προγράμματα

Οι κώδικες έχουν σχόλια μόνο στα σημεία που θεώρησα ότι μπορεί να προκύψει κάποιο «μπέρδεμα».

1.1 'Ασκηση 2A

1.1.1 Κώδικας

```
#include <err.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include <omp.h>

static void     pretty_print(int *, int, const char *);
static int      cmpfunc(const void *, const void *);
static void     merge(int *, int *, int *, int *, int *);
static void     multisort(int *, int *, int);

/*
 * Print the contents of an array like:
 *
 * array = [x, y, z]
 */
static void
pretty_print(int *arr, int n, const char *name)
{
        int i;

        printf("\n%s = [", name);
        for (i = 0; i < n; i++)
                printf("%d%s", arr[i], (i == n - 1) ? "" : ", ");
        printf("]\n");
}

/*
 * Passed to qsort(3).
 */
static int
cmpfunc(const void *a, const void *b)
{
        return (*(int *)a - *(int *)b);
}

/*
```

```
 * Merge sort
 */
static void
merge(int *a, int *enda, int *b, int *endb, int *res)
{
        while (a <= enda && b <= endb) {
                if (*a < *b)
                        *res++ = *a++;
                else
                        *res++ = *b++;
        }
        while (a <= enda)
                *res++ = *a++;
        while (b <= endb)
                *res++ = *b++;
}

static void
multisort(int *arr, int *space, int n)
{
        int quarter, *sta, *spa, *stb, *spb, *stc, *spc, *std, *spd;

        /*
         * Sort with qsort(3) directly if we can't split the array into 4
         * quarters.
         */
        if ((quarter = n / 4) < 4)
                qsort(arr, n, sizeof(int), cmpfunc);
        else {
                /* Split the array into 4 quarters. */
                sta = arr;
                spa = space;
                stb = sta + quarter;
                spb = spa + quarter;
                stc = stb + quarter;
                spc = spb + quarter;
                std = stc + quarter;
                spd = spc + quarter;
                /* Sort each quarter */
#pragma omp task
                multisort(sta, spa, quarter);
#pragma omp task
                multisort(stb, spb, quarter);
#pragma omp task
                multisort(stc, spc, quarter);
#pragma omp task
```

```
                multisort(std, spd, n - 3 * quarter);
                /* Wait for the tasks above to finish. */
#pragma omp taskwait
#pragma omp task
                /* Merge A and B into SpaceA */
                merge(sta, sta + quarter - 1, stb, stb + quarter - 1, spa);
#pragma omp task
                /* Merge C and D into SpaceC */
                merge(stc, stc + quarter - 1, std, arr + n - 1, spc);
#pragma omp taskwait
                /* Merge the two resulting couples (SpaceA and SpaceC). */
                merge(spa, spc - 1, spc, space + n - 1, arr);
        }
}

int
main(int argc, char *argv[])
{
        int *a, *space, i, n, ntd;
        double start, end;


        if (argc < 3) {
                fprintf(stderr, "usage: %s nthreads n\n", *argv);
                return (1);
        }
        if ((ntd = atoi(argv[1])) < 1)
                err(1, "can't use nthreads n < 1");
        if ((n = atoi(argv[2])) < 1)
                err(1, "can't use n < 1");

        srand(time(NULL));
        omp_set_num_threads(ntd);

        if ((a = malloc(n * sizeof(int))) == NULL)
                err(1, "malloc");
        if ((space = malloc(n * sizeof(int))) == NULL)
                err(1, "malloc");
        for (i = 0; i < n; i++)
                a[i] = rand() % 100;

        /* Calculate speed up */
        start = omp_get_wtime();

        pretty_print(a, n, "A_unsorted");
        multisort(a, space, n);
```

```
        pretty_print(a, n, "A_multisort");

        end = omp_get_wtime();
        printf("Total time: %f seconds\n", end - start);

        free(a);

        return (0);
}
```

1.1.2 Ενδεικτικά τρεξίματα

usage: ./a.out nthreads n

Για nthreads = 2 και n = 10:

```
christos@tpad$ ./ex2a 2 10

A_unsorted = [82, 43, 18, 24, 3, 17, 89, 13, 63, 44]

A_multisort = [3, 13, 17, 18, 24, 43, 44, 63, 82, 89]
Total time: 0.000030 seconds
```

Για nthreads = 8 και n = 100:

```
christos@tpad$ ./ex2a 8 100

A_unsorted = [39, 13, 48, 42, 60, 32, 38, 0, 13, 11, 82, 68, 25, 18, 52, 8, 45, 19, 17, 8, 47, 56, 77, 74,
58, 38, 34, 72, 63, 95, 23, 3, 9, 71, 97, 21, 55, 87, 73, 20, 51, 7, 88, 76, 25, 41, 36, 71, 60, 53, 31, 59
, 61, 60, 86, 71, 98, 20, 95, 14, 67, 18, 69, 28, 42, 66, 49, 97, 54, 74, 18, 57, 81, 58, 33, 59, 99, 21, 8
2, 12, 26, 13, 23, 87, 74, 9, 10, 24, 81, 58, 38, 1, 76, 7, 29, 70, 26, 31, 68, 80]

A_multisort = [0, 1, 3, 7, 7, 8, 8, 9, 9, 10, 11, 12, 13, 13, 13, 14, 17, 18, 18, 18, 19, 20, 20, 21, 21, 2
3, 23, 24, 25, 25, 26, 26, 28, 29, 31, 31, 32, 33, 34, 36, 38, 38, 38, 39, 41, 42, 42, 45, 47, 48, 49, 51,
52, 53, 54, 55, 56, 57, 58, 58, 58, 59, 59, 60, 60, 60, 61, 63, 66, 67, 68, 68, 69, 70, 71, 71, 71, 72, 73,
 74, 74, 74, 76, 76, 77, 80, 81, 81, 82, 82, 86, 87, 87, 88, 95, 95, 97, 97, 98, 99]
Total time: 0.000057 seconds
```

Για nthreads = 16 και n = 1000000. Λόγω του αριθμού των στοιχείων, το στιγμιότυπο δείχνει μόνο τον χρόνο υπολογισμού:

```
Total time: 0.519716 seconds
```

1.2 'Ασκηση 2B-A

1.2.1 Κώδικας

```
#include <stdio.h>
#include <time.h>

#define N       (1 << 2)
#define DIM     (N * N)
/*
 * This formula for calculating the number of blocks is mentioned at "out of
 * the blocks" section in:
 *
 * https://developer.nvidia.com/blog/even-easier-introduction-cuda/
 */
#define BLKSIZE (1 << 8)
#define NBLK    ((DIM + BLKSIZE - 1) / BLKSIZE)

__global__ void
convolution(float *a, float *aconv)
{
        float c11, c12, c13, c21, c22, c23, c31, c32, c33;
        int i, j, x, stridex;

        /*
         * Each thread gets a slice of the rows to work with. Grid-stride idiom
         * mentioned at section "out of the blocks" in:
         *
         * https://developer.nvidia.com/blog/even-easier-introduction-cuda/
         */
        x = blockIdx.x * blockDim.x + threadIdx.x;
        stridex = blockDim.x * gridDim.x;

        /* Random weight values */
        c11 = +0.2;  c21 = +0.5;  c31 = -0.8;
        c12 = -0.3;  c22 = +0.6;  c32 = -0.9;
        c13 = +0.4;  c23 = +0.7;  c33 = +0.10;

        if (x < 1 || x > N - 1)
                return;
        for (i = x; i < N - 1; i += stridex) {
                for (j = 1; j < N - 1; j++) {
                        /* Taken from the lab's example code. */
                        aconv[i * N + j] =
                            c11 * a[(i - 1)     * N + (j - 1)] +
                            c12 * a[i           * N + (j - 1)] +
                            c13 * a[(i + 1)     * N + (j - 1)] +
```

```
                        c21 * a[(i - 1)     * N + j] +
                        c22 * a[i           * N + j] +
                        c23 * a[(i + 1)     * N + j] +
                        c31 * a[(i - 1)     * N + (j + 1)] +
                        c32 * a[i           * N + (j + 1)] +
                        c33 * a[(i + 1)     * N + (j + 1)];
                }
        }
}

__global__ void
min_diagonal(float *arr, float *min_arr)
{
        int x, stridex, i;

        x = blockIdx.x * blockDim.x + threadIdx.x;
        stridex = blockDim.x * gridDim.x;

        if (x >= N)
                return;
        /* Calculate local minimums */
        min_arr[x] = arr[x * N + x];
        for (i = x; i < N; i += stridex)
                if (arr[i * N + i] < min_arr[x])
                        min_arr[x] = arr[i * N + i];
}

static void
pretty_print(float *arr, const char *name)
{
        int i, j;

        printf("\n%s = [\n", name);
        for (i = 0; i < N; i++) {
                printf("\t[");
                for (j = 0; j < N; j++) {
                        printf("%.2f%s", arr[i * N + j],
                           (j == N - 1) ? "]\n" : ", ");
                }
        }
        printf("]\n");
}

int
main(int argc, char *argv[])
{
```

```
        float *a, *aconv, *min_arr, min;
        int i;

        srand(time(NULL));

        /*
         * Use unified memory to avoid having additional device arrays and
         * memcpying from host to device and vice versa.
         *
         * https://developer.nvidia.com/blog/unified-memory-cuda-beginners/
         */
        cudaMallocManaged(&a, DIM * sizeof(float));
        cudaMallocManaged(&aconv, DIM * sizeof(float));
        cudaMallocManaged(&min_arr, DIM * sizeof(float));

        /* Initialize array */
        for (i = 0; i < DIM; i++)
                a[i] = (float)(rand() % 100);

        convolution<<<NBLK, BLKSIZE>>>(a, aconv);
        /* Wait for all devices to finish */
        cudaDeviceSynchronize();

        min_diagonal<<<NBLK, BLKSIZE>>>(aconv, min_arr);
        cudaDeviceSynchronize();

        /*
         * Find global minimum using the local minimums calculated in
         * min_diagonal().
         */
        min = min_arr[0];
        for (i = 0; i < N; i++)
                if (min_arr[i] < min)
                        min = min_arr[i];

        pretty_print(a, "A");
        pretty_print(aconv, "A_conv");
        printf("Min_diagonal(A_conv): %.2f\n", min);

        cudaFree(a);
        cudaFree(aconv);
        cudaFree(min_arr);

        return (0);
}
```

1.2.2 Ενδεικτικά τρεξίματα

Για `NxN = 8x8` και `blocksize = 256`

```
cuda17@rncp-ubuntu:~$ nvcc ex2b_a.cu -o ex2b_a
cuda17@rncp-ubuntu:~$ ./ex2b_a

A = [
        [12.00, 20.00, 37.00, 60.00, 14.00, 43.00, 62.00, 48.00]
        [27.00, 90.00, 88.00, 73.00, 71.00, 4.00, 44.00, 84.00]
        [87.00, 89.00, 68.00, 0.00, 18.00, 89.00, 21.00, 26.00]
        [32.00, 16.00, 32.00, 10.00, 46.00, 7.00, 7.00, 58.00]
        [79.00, 44.00, 70.00, 93.00, 40.00, 84.00, 94.00, 67.00]
        [74.00, 82.00, 92.00, 98.00, 39.00, 37.00, 34.00, 78.00]
        [26.00, 2.00, 30.00, 97.00, 91.00, 51.00, 23.00, 76.00]
        [67.00, 8.00, 86.00, 13.00, 15.00, 93.00, 23.00, 94.00]
]

A_conv = [
        [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
        [0.00, 53.40, 17.80, 8.70, 23.20, -12.20, 3.70, 0.00]
        [0.00, -26.70, 47.50, -14.90, 14.50, 34.10, -68.40, 0.00]
        [0.00, 48.10, 133.10, 51.30, 29.70, 99.60, 63.50, 0.00]
        [0.00, 24.70, 63.30, 82.70, 10.10, 5.80, -24.20, 0.00]
        [0.00, -59.20, -56.70, 113.60, 39.70, 29.10, -6.60, 0.00]
        [0.00, -10.40, -21.20, 48.50, 14.10, 55.10, -45.20, 0.00]
        [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
]
Min_diagonal(A_conv): -45.20
```

Για `NxN = 32x32` και `blocksize = 1024`

```
A = [
        [61.00, 53.00, 72.00, 84.00, 44.00, 94.00, 2.00, 66.00, 23.00, 27.00, 64.00, 2.00, 69.00, 86.00, 34.00, 58.00, 9.00, 58.00, 0.00, 45.00, 44.00, 7.00, 31.00, 15.00, 6.00, 60.00, 39.00, 33.00, 75.00, 24.00, 10.00, 36.00]
        [29.00, 34.00, 72.00, 73.00, 81.00, 75.00, 39.00, 4.00, 54.00, 56.00, 58.00, 76.00, 94.00, 45.00, 86.00, 55.00, 55.00, 86.00, 53.00, 51.00, 94.00, 36.00, 66.00, 0.00, 56.00, 5.00, 33.00, 32.00, 82.00, 95.00, 20.00, 11.00]
        [30.00, 93.00, 37.00, 63.00, 60.00, 76.00, 67.00, 74.00, 32.00, 70.00, 50.00, 79.00, 23.00, 37.00, 34.00, 30.00, 23.00, 87.00, 81.00, 17.00, 23.00, 47.00, 69.00, 80.00, 5.00, 54.00, 64.00, 87.00, 50.00, 84.00, 50.00, 32.00]
        [77.00, 87.00, 95.00, 97.00, 16.00, 14.00, 24.00, 0.00, 44.00, 74.00, 79.00, 19.00, 11.00, 66.00, 49.00, 87.00, 53.00, 30.00, 4.00, 29.00, 30.00, 26.00, 9.00, 35.00, 80.00, 73.00, 74.00, 82.00, 9.00, 76.00, 14.00, 39.00]
        [16.00, 61.00, 36.00, 32.00, 20.00, 60.00, 84.00, 72.00, 87.00, 16.00, 92.00, 90.00, 82.00, 93.00, 85.00, 35.00, 24.00, 42.00, 16.00, 6.00, 20.00, 77.00, 41.00, 0.00, 50.00, 67.00, 35.00, 12.00, 43.00, 49.00, 51.00, 11.00]
        [63.00, 39.00, 95.00, 91.00, 52.00, 80.00, 15.00, 39.00, 96.00, 7.00, 89.00, 70.00, 1.00, 27.00, 65.00, 77.00, 21.00, 34.00, 83.00, 41.00, 11.00, 76.00, 93.00, 14.00, 43.00, 28.00, 26.00, 38.00, 30.00, 77.00, 2.00, 93.00]
        [68.00, 97.00, 36.00, 20.00, 77.00, 51.00, 59.00, 25.00, 11.00, 1.00, 55.00, 12.00, 20.00, 73.00, 41.00, 49.00, 7.00, 24.00, 90.00, 10.00, 52.00, 35.00, 32.00, 95.00, 64.00, 10.00, 85.00, 94.00, 87.00, 87.00, 39.00, 56.00]
        [85.00, 75.00, 20.00, 14.00, 70.00, 40.00, 40.00, 89.00, 41.00, 47.00, 53.00, 69.00, 20.00, 94.00, 79.00, 27.00, 79.00, 64.00, 98.00, 22.00, 47.00, 82.00, 69.00, 11.00, 93.00, 7.00, 57.00, 32.00, 46.00, 48.00, 40.00, 83.00]
        [75.00, 69.00, 98.00, 54.00, 9.00, 90.00, 95.00, 50.00, 37.00, 49.00, 71.00, 10.00, 95.00, 41.00, 89.00, 66.00, 53.00, 87.00, 40.00, 0.00, 22.00, 62.00, 12.00, 15.00, 21.00, 21.00, 99.00, 67.00, 70.00, 40.00, 51.00, 97.00]
        [9.00, 1.00, 3.00, 18.00, 91.00, 99.00, 20.00, 80.00, 48.00, 91.00, 90.00, 95.00, 32.00, 32.00, 13.00, 37.00, 71.00, 54.00, 37.00, 93.00, 60.00, 1.00, 60.00, 89.00, 23.00, 60.00, 0.00, 45.00, 0.00, 59.00, 94.00, 9.00]
        [60.00, 98.00, 79.00, 3.00, 97.00, 51.00, 84.00, 97.00, 42.00, 26.00, 92.00, 26.00, 10.00, 6.00, 15.00, 82.00, 12.00, 52.00, 75.00, 32.00, 54.00, 88.00, 21.00, 29.00, 48.00, 81.00, 26.00, 48.00, 41.00, 20.00, 9.00, 53.00]
        [70.00, 40.00, 57.00, 67.00, 91.00, 93.00, 16.00, 85.00, 71.00, 9.00, 63.00, 82.00, 67.00, 70.00, 64.00, 79.00, 82.00, 91.00, 63.00, 88.00, 79.00, 36.00, 17.00, 79.00, 17.00, 95.00, 27.00, 58.00, 68.00, 88.00, 12.00, 38.00]
        [88.00, 21.00, 58.00, 23.00, 14.00, 74.00, 60.00, 85.00, 83.00, 75.00, 67.00, 58.00, 5.00, 83.00, 81.00, 88.00, 75.00, 44.00, 76.00, 6.00, 32.00, 46.00, 86.00, 58.00, 93.00, 65.00, 8.00, 61.00, 6.00, 72.00, 52.00, 86.00]
        [45.00, 10.00, 62.00, 59.00, 84.00, 22.00, 97.00, 20.00, 98.00, 16.00, 22.00, 55.00, 0.00, 56.00, 43.00, 27.00, 52.00, 72.00, 85.00, 85.00, 70.00, 23.00, 35.00, 63.00, 89.00, 95.00, 77.00, 47.00, 20.00, 29.00, 85.00, 65.00]
        [91.00, 47.00, 77.00, 75.00, 22.00, 26.00, 47.00, 72.00, 42.00, 70.00, 27.00, 94.00, 26.00, 23.00, 73.00, 78.00, 95.00, 59.00, 63.00, 17.00, 82.00, 58.00, 88.00, 23.00, 46.00, 9.00, 70.00, 66.00, 38.00, 8.00, 83.00, 81.00]
        [55.00, 60.00, 57.00, 29.00, 86.00, 56.00, 1.00, 81.00, 26.00, 29.00, 75.00, 4.00, 52.00, 49.00, 35.00, 99.00, 60.00, 58.00, 16.00, 42.00, 1.00, 40.00, 18.00, 47.00, 58.00, 40.00, 65.00, 40.00, 48.00, 48.00, 30.00, 56.00]
        [61.00, 39.00, 85.00, 99.00, 95.00, 87.00, 88.00, 74.00, 68.00, 8.00, 30.00, 72.00, 57.00, 65.00, 23.00, 17.00, 16.00, 91.00, 11.00, 17.00, 39.00, 81.00, 16.00, 49.00, 22.00, 33.00, 58.00, 22.00, 33.00, 32.00, 78.00, 94.00]
        [23.00, 64.00, 46.00, 78.00, 3.00, 26.00, 96.00, 23.00, 86.00, 27.00, 95.00, 43.00, 92.00, 78.00, 12.00, 8.00, 61.00, 76.00, 77.00, 52.00, 57.00, 45.00, 54.00, 31.00, 78.00, 56.00, 54.00, 12.00, 88.00, 84.00, 58.00, 63.00]
        [0.00, 56.00, 85.00, 3.00, 83.00, 82.00, 78.00, 69.00, 9.00, 25.00, 65.00, 53.00, 95.00, 29.00, 62.00, 8.00, 57.00, 91.00, 61.00, 67.00, 37.00, 15.00, 98.00, 67.00, 71.00, 4.00, 79.00, 11.00, 89.00, 98.00, 26.00, 89.00]
        [46.00, 11.00, 45.00, 81.00, 93.00, 23.00, 3.00, 54.00, 1.00, 28.00, 8.00, 96.00, 49.00, 22.00, 5.00, 7.00, 65.00, 18.00, 74.00, 2.00, 33.00, 24.00, 78.00, 56.00, 29.00, 1.00, 19.00, 78.00, 43.00, 45.00, 59.00, 98.00]
        [8.00, 56.00, 23.00, 54.00, 88.00, 26.00, 8.00, 81.00, 46.00, 68.00, 29.00, 48.00, 42.00, 86.00, 7.00, 8.00, 1.00, 33.00, 62.00, 89.00, 57.00, 84.00, 57.00, 38.00, 86.00, 16.00, 8.00, 29.00, 13.00, 68.00, 71.00, 22.00]
        [24.00, 95.00, 76.00, 56.00, 73.00, 36.00, 89.00, 72.00, 57.00, 71.00, 28.00, 99.00, 57.00, 79.00, 7.00, 14.00, 12.00, 78.00, 3.00, 69.00, 54.00, 53.00, 8.00, 92.00, 21.00, 16.00, 74.00, 35.00, 36.00, 97.00, 9.00, 13.00]
        [44.00, 37.00, 21.00, 18.00, 73.00, 63.00, 42.00, 82.00, 34.00, 62.00, 82.00, 43.00, 93.00, 89.00, 57.00, 5.00, 11.00, 13.00, 26.00, 18.00, 18.00, 34.00, 18.00, 39.00, 3.00, 36.00, 74.00, 91.00, 34.00, 35.00, 56.00, 78.00]
        [72.00, 78.00, 48.00, 98.00, 41.00, 98.00, 88.00, 27.00, 4.00, 62.00, 22.00, 97.00, 4.00, 32.00, 54.00, 67.00, 97.00, 81.00, 85.00, 15.00, 15.00, 48.00, 54.00, 78.00, 36.00, 81.00, 62.00, 22.00, 16.00, 18.00, 53.00, 89.00]
        [48.00, 1.00, 39.00, 41.00, 44.00, 19.00, 68.00, 48.00, 34.00, 43.00, 98.00, 98.00, 75.00, 52.00, 9.00, 72.00, 33.00, 95.00, 87.00, 1.00, 95.00, 93.00, 23.00, 83.00, 74.00, 85.00, 6.00, 91.00, 56.00, 59.00, 32.00, 56.00]
        [12.00, 71.00, 98.00, 56.00, 42.00, 18.00, 57.00, 76.00, 61.00, 7.00, 18.00, 88.00, 59.00, 28.00, 68.00, 45.00, 75.00, 99.00, 98.00, 22.00, 93.00, 21.00, 5.00, 67.00, 59.00, 63.00, 58.00, 15.00, 22.00, 98.00, 71.00, 35.00]
        [13.00, 21.00, 43.00, 56.00, 48.00, 0.00, 84.00, 1.00, 7.00, 3.00, 98.00, 19.00, 83.00, 2.00, 16.00, 58.00, 2.00, 14.00, 32.00, 47.00, 35.00, 37.00, 14.00, 94.00, 1.00, 25.00, 61.00, 75.00, 67.00, 85.00, 18.00, 81.00]
        [6.00, 6.00, 89.00, 46.00, 6.00, 73.00, 0.00, 66.00, 28.00, 42.00, 85.00, 11.00, 96.00, 53.00, 21.00, 58.00, 67.00, 53.00, 97.00, 54.00, 43.00, 12.00, 1.00, 96.00, 89.00, 62.00, 71.00, 56.00, 99.00, 34.00, 89.00, 6.00]
        [48.00, 78.00, 4.00, 46.00, 4.00, 56.00, 64.00, 32.00, 58.00, 49.00, 44.00, 99.00, 2.00, 17.00, 49.00, 21.00, 71.00, 47.00, 76.00, 66.00, 11.00, 77.00, 62.00, 0.00, 91.00, 33.00, 8.00, 43.00, 67.00, 58.00, 1.00, 7.00]
        [28.00, 57.00, 6.00, 32.00, 66.00, 78.00, 17.00, 16.00, 72.00, 13.00, 15.00, 26.00, 38.00, 65.00, 0.00, 53.00, 64.00, 28.00, 19.00, 75.00, 57.00, 81.00, 27.00, 48.00, 67.00, 35.00, 43.00, 34.00, 37.00, 44.00, 94.00, 66.00]
        [2.00, 52.00, 58.00, 68.00, 74.00, 19.00, 36.00, 98.00, 32.00, 52.00, 77.00, 63.00, 69.00, 77.00, 68.00, 85.00, 57.00, 48.00, 68.00, 14.00, 21.00, 39.00, 14.00, 48.00, 26.00, 58.00, 75.00, 64.00, 2.00, 21.00, 82.00, 56.00]
        [25.00, 32.00, 24.00, 51.00, 4.00, 13.00, 58.00, 36.00, 17.00, 79.00, 51.00, 86.00, 8.00, 28.00, 23.00, 65.00, 68.00, 83.00, 79.00, 33.00, 22.00, 93.00, 74.00, 8.00, 3.00, 1.00, 64.00, 58.00, 74.00, 46.00, 14.00, 99.00]
]

A_conv = [
        [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
        [0.00, 0.00, 16.10, 36.20, 3.20, 126.90, 49.00, 30.90, 43.70, 4.50, 59.50, -20.20, 10.60, -10.50, 18.00, 23.20, -54.40, 96.40, 28.90, -36.00, 61.70, -29.50, 108.00, 22.00, 21.60, 6.70, 71.30, -0.70, 57.20, 117.20, 26.40, 0.00]
        [0.00, 79.90, 33.00, 59.10, 0.00, 6.00, 4.50, -23.90, -52.00, 61.90, 1.70, 32.10, 4.90, 12.70, 59.70, 73.60, -28.90, 26.60, 27.00, -49.00, 30.00, -25.70, 15.70, 27.30, 31.70, 51.10, 25.00, 32.90, -53.60, 96.50, 40.30, 0.00]
        [0.00, 19.20, -16.90, 39.40, -7.90, 41.60, 89.70, 70.70, 19.50, 20.70, 66.10, 122.50, 45.10, 94.50, 31.90, 54.10, -30.40, 22.40, 40.20, 0.10, -12.70, 37.40, 5.50, 17.50, -11.20, 0.40, -25.60, 61.90, -72.60, 90.90, 23.10, 0.00]
        [0.00, 44.30, 53.00, 144.80, 50.20, 21.30, 28.80, -47.00, 62.50, -64.50, 74.20, 64.20, -72.00, -23.50, 23.70, 66.80, 34.90, 66.50, 52.20, 35.10, -27.40, 82.00, 88.10, -50.70, -3.20, 26.40, 9.30, 46.60, -17.90, 65.30, 30.50, 0.00]
        [0.00, 22.60, 34.00, 16.20, -28.70, 59.70, 10.50, -42.10, 102.50, -142.30, -0.00, 54.20, -29.20, 18.70, 57.50, 74.50, -28.00, -19.00, 88.70, 33.80, -77.40, 16.80, 102.20, -4.30, 51.60, 37.10, 67.20, 75.10, 29.20, 127.30, -11.40, 0.00]
        [0.00, 50.80, 8.00, -16.40, 38.70, 56.00, 42.80, 15.50, 117.40, -46.10, 68.20, 93.00, -7.00, 41.00, 24.20, 99.40, 32.40, -34.10, 140.90, 24.60, -2.90, 25.50, 55.90, 25.60, 73.00, -40.10, -2.50, 2.60, -43.50, 89.90, -55.60, 0.00]
        [0.00, 115.70, 104.70, -36.70, 45.20, 34.40, 59.10, 96.70, 16.10, -16.90, 44.00, 47.00, -60.70, 72.30, 70.70, 46.40, 46.20, -45.70, 119.20, -54.90, -29.00, 28.70, -11.40, -77.70, 112.00, -92.20, 57.00, 48.60, 47.30, 69.00, -11.60, 0.00]
        [0.00, -32.60, 11.60, -32.00, -1.20, 77.10, 3.40, 56.70, 38.50, 35.60, 100.50, 34.00, 29.30, -52.50, 39.50, -25.10, -4.00, 35.00, 98.10, 47.90, 4.60, 48.90, 68.50, 20.00, 98.00, -54.30, 31.20, -27.20, 0.80, 9.00, -15.20, 0.00]
        [0.00, 66.80, 99.70, 10.00, -22.40, 67.00, 24.70, 118.90, -17.00, -21.50, 55.90, 0.40, -3.40, -24.10, -14.10, 27.40, -0.60, 48.60, 36.20, 31.70, 32.00, 37.30, 14.80, 41.30, -16.90, 35.90, 1.00, 70.00, -0.30, -24.70, 6.10, 0.00]
        [0.00, 31.30, 65.20, -93.60, 81.10, 88.70, -29.50, 73.00, 24.50, -55.70, 67.10, 108.20, 89.70, 86.90, 0.60, 87.30, 33.60, 74.50, 44.00, 42.30, 89.50, 42.30, -46.70, 69.00, -10.50, 115.90, -9.20, 66.40, 33.60, 35.90, 58.70, 0.00]
        [0.00, 2.00, 69.90, -70.40, -14.50, 55.20, -40.10, 105.00, 128.20, -32.40, 73.60, 55.70, -17.40, 30.30, -14.30, 90.30, 16.50, 10.10, 25.00, -11.60, -19.00, 74.20, 16.00, 86.00, -42.40, 140.40, -36.00, 8.40, -5.90, 98.60, -13.50, 0.00]
        [0.00, -44.00, 44.00, 30.40, 4.00, 96.00, -25.40, 9.00, 88.30, -19.30, -32.30, 52.70, -71.60, 20.10, -15.90, -11.00, 17.40, 26.70, 103.10, 45.90, 89.60, 8.40, -14.70, 16.00, 27.60, 143.00, 13.10, 74.50, -60.60, 79.40, 3.00, 0.00]
        [0.00, -6.20, 76.10, 38.60, 13.30, -75.80, 60.90, -43.80, 102.10, 29.40, 23.70, 140.00, -62.70, 6.40, 39.50, 50.00, 81.80, 12.90, 64.30, 0.40, 45.40, -2.30, 47.90, -32.30, -4.20, 37.40, 2.80, 92.40, -16.80, -40.40, 24.80, 0.00]
        [0.00, -34.30, 17.20, 28.50, 88.90, -9.40, 23.70, 2.00, 34.30, 53.30, -54.70, 94.80, -24.20, -7.50, 31.80, 9.20, 40.90, -10.80, 29.10, -43.10, 55.00, -33.30, 28.50, -38.40, 57.10, -10.00, 66.10, 71.70, 49.60, -75.00, 16.60, 0.00]
        [0.00, 8.50, 63.00, 71.00, 112.50, 93.60, -19.80, 127.30, 22.60, -0.30, 16.40, 32.80, 74.50, 11.20, -60.30, -4.80, 4.30, 66.90, 32.30, -4.80, -0.60, 63.60, 34.20, 4.20, 43.40, -57.30, 18.20, 21.40, 32.10, -24.00, 8.30, 0.00]
        [0.00, -17.40, 32.30, -26.80, -14.30, 25.10, -17.90, 42.20, 89.50, -20.10, 72.50, 35.60, 30.30, 95.10, -35.70, 13.90, -14.10, 149.40, 37.90, 68.10, -18.90, 92.00, -18.90, 36.60, 47.80, 23.60, 63.80, 5.90, 54.40, 48.50, 12.90, 0.00]
        [0.00, 1.50, -1.30, 60.40, 22.60, 25.20, 121.60, -13.40, 99.20, -73.90, 30.00, -16.50, 60.80, 93.60, 45.70, -17.30, -43.40, 90.50, 72.50, 11.30, -9.60, 31.90, 57.30, 35.50, 58.40, -18.40, 76.80, -45.40, 44.60, 33.20, -13.80, 0.00]
        [0.00, -12.50, 55.30, 27.40, 69.60, -55.50, 11.60, 10.90, 12.50, -76.30, 25.50, -31.00, 88.50, 12.40, 47.90, -94.10, -28.70, 17.20, 20.30, 18.70, 17.80, -62.60, 69.40, -19.10, 62.10, -75.40, 85.60, -89.90, 10.90, 71.70, -27.80, 0.00]
        [0.00, -43.00, 46.00, -41.50, 67.50, 29.40, -29.40, 138.00, 36.10, 35.70, -40.20, 24.00, 72.90, 55.60, 60.60, -76.20, -12.70, -36.30, 97.80, 58.20, 101.40, -30.70, 88.10, 56.80, 122.40, -25.80, -5.50, -27.50, -33.50, 62.90, -20.20, 0.00]
        [0.00, 72.90, 5.10, 5.50, 129.80, 75.40, -29.10, 123.30, 13.20, 79.20, -62.70, 75.70, 46.90, 128.30, 18.40, -34.60, 18.80, -20.30, 23.20, 27.80, 0.30, -31.90, 29.80, 4.80, 103.80, -5.30, -18.70, 65.40, -13.70, 40.10, 15.30, 0.00]
        [0.00, 38.20, -22.50, -59.80, 89.20, 19.70, -15.00, 48.10, -3.70, 92.70, -22.20, 78.50, -13.60, 175.00, 58.20, 25.10, -71.80, 19.50, -89.70, 28.00, -33.70, 16.20, -34.50, 35.10, 29.70, -18.70, 67.40, 68.80, -54.60, 62.80, 57.10, 0.00]
        [0.00, 69.80, 72.10, 15.50, 77.50, 26.90, 21.30, 65.50, -54.10, 31.30, -13.20, 2.50, -7.30, 48.90, 68.40, 53.00, 49.00, 128.10, 49.20, 24.70, -5.50, 81.90, -34.30, 122.70, 35.10, -15.60, 41.90, 47.10, -69.20, 33.20, 19.60, 0.00]
        [0.00, 16.30, -47.50, 11.00, -47.00, 19.90, 24.70, 75.00, -30.40, 41.90, 21.10, 121.20, 22.10, 54.20, 39.10, -0.90, 22.70, 19.30, 113.40, 14.00, 22.90, 93.70, -2.00, 87.20, 1.70, 31.10, 11.80, 85.00, 75.50, 4.70, -29.50, 0.00]
        [0.00, 30.40, 50.00, 81.70, 18.70, -30.70, 83.00, 86.20, -1.10, -27.30, -65.10, 81.50, 46.30, 23.40, -37.50, 40.50, -9.00, 69.00, 179.90, -34.60, 21.10, 42.80, -89.60, 48.10, -15.40, 95.00, -6.90, 40.90, -18.40, 38.50, -0.00, 0.00]
        [0.00, -46.10, 18.10, 19.90, 57.60, -64.80, 18.70, 19.00, 5.30, -66.50, -18.80, 56.50, 63.60, 15.10, -37.00, 0.10, -76.70, -53.00, 103.70, -108.80, 50.60, 68.60, -65.90, 44.00, -15.00, 44.40, 9.80, 57.30, 12.10, 85.90, 16.20, 0.00]
        [0.00, -55.00, 56.80, 67.10, 54.90, -62.30, 56.60, 18.70, 91.50, -30.90, 53.40, -38.60, 143.90, 17.30, -2.70, 52.80, 10.70, 38.70, 110.40, 24.20, 60.60, 49.90, -119.10, 115.10, 49.30, 42.80, 71.30, 45.40, -21.70, 86.80, 9.60, 0.00]
        [0.00, -28.60, 29.70, 34.30, -17.90, 30.00, 30.30, 79.10, 13.20, -70.10, 119.20, -57.10, 93.00, 6.50, -43.10, 35.40, 41.70, -24.90, 53.50, 44.20, 37.90, 72.20, -76.00, 60.10, 35.10, -29.20, -27.80, -40.70, 61.60, 20.40, 16.40, 0.00]
        [0.00, 15.90, -23.30, 90.20, -42.00, 89.60, -3.90, -9.20, 28.00, -27.20, -17.00, 17.30, 1.90, 51.90, 17.00, -46.90, 63.90, -52.10, 20.00, 84.70, 18.20, 83.10, -3.70, -72.40, 100.60, 32.90, 6.00, -29.20, 59.70, 29.60, 115.80, 0.00]
        [0.00, 106.40, 1.10, 53.60, 14.30, 31.00, 35.00, 14.70, 85.70, 28.90, 15.30, 122.60, 40.10, 88.00, 30.00, 30.10, 77.30, 14.20, -10.50, 63.30, -79.90, 35.40, 24.50, -63.00, 53.40, 49.00, 43.00, 18.70, 3.60, -0.00, 60.10, 0.00]
        [0.00, 49.70, -23.30, -30.70, 14.80, 7.10, -18.00, 19.90, 6.00, 28.40, 39.10, 26.00, -36.60, 22.30, -57.60, 15.30, 69.20, 54.50, 58.30, 24.10, -10.10, 116.10, 41.00, 2.10, -31.30, -35.70, 22.30, 83.70, 32.00, -44.40, 33.60, 0.00]
        [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
]
Min_diagonal(A_conv): -71.60
```

Για `NxN = 4x4` και `blocksize = 256`

```
cuda17@rncp-ubuntu:~$ ./ex2b_a

A = [
        [83.00, 6.00, 87.00, 47.00]
        [7.00, 30.00, 38.00, 78.00]
        [27.00, 31.00, 9.00, 12.00]
        [90.00, 89.00, 53.00, 13.00]
]

A_conv = [
        [0.00, 0.00, 0.00, 0.00]
        [0.00, -34.90, -29.40, 0.00]
        [0.00, 92.00, 21.90, 0.00]
        [0.00, 0.00, 0.00, 0.00]
]
Min_diagonal(A_conv): -34.90
```

1.3 'Ασκηση 2B-B

1.3.1 Κώδικας

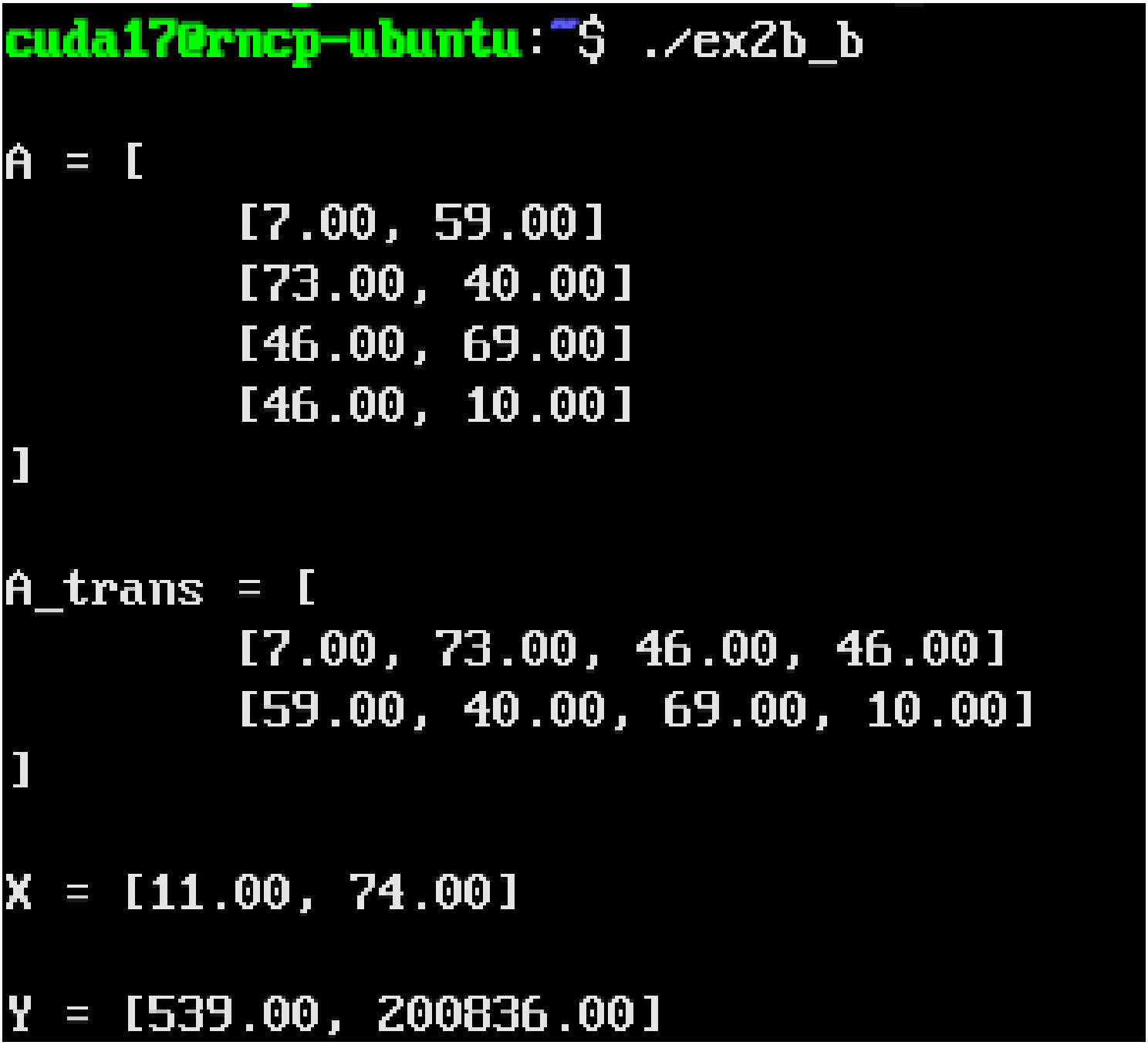
```
#include <stdio.h>
#include <time.h>

#define N       (1 << 3)
#define M       (1 << 3)
#define DIM     (N * M)
#define BLKSIZE (1 << 10)
#define NBLK    ((DIM + BLKSIZE - 1) / BLKSIZE)

/*
 * Calculations taken from lab's example code.
 */
__global__ void
transnorm(float *a, float *atrans, float *x, float *y)
{
        int i, j, idx, stridex;

        /* Each thread gets a slice of the rows to work with */
        idx = blockIdx.x * blockDim.x + threadIdx.x;
        stridex = blockDim.x * gridDim.x;

        if (idx >= N)
                return;
        /* First thread initializes y */
        if (threadIdx.x == 0) {
                for (i = 0; i < M; i++)
                        y[i] = 0;
        }
        for (i = idx; i < N; i += stridex) {
                for (j = 0; j < M; j++) {
                        /* Transpose A */
                        atrans[j * N + i] = a[i * M + j];
                        y[j] = atrans[j * M + i] * a[i * M + j] * x[j];
                }
        }
}

static void
pretty_print_1d(float *arr, const char *name, int n)
{
        int i;

        printf("\n%s = [", name);
```

```
        for (i = 0; i < n; i++) {
                printf("%.2f%s", arr[i],
                   (i == n - 1) ? "" : ", ");
        }
        printf("]\n");
}

static void
pretty_print_2d(float *arr, const char *name, int w, int h)
{
        int i, j;

        printf("\n%s = [\n", name);
        for (i = 0; i < w; i++) {
                printf("\t[");
                for (j = 0; j < h; j++) {
                        printf("%.2f%s", arr[i * h + j],
                           (j == h - 1) ? "]\n" : ", ");
                }
        }
        printf("]\n");
}

int
main(int argc, char *argv[])
{
        float *a, *atrans, *x, *y;
        int i, j;

        srand(time(NULL));

        /*
         * Use unified memory to avoid having additional device arrays and
         * memcpying from host to device and vice versa.
         */
        cudaMallocManaged(&a, DIM * sizeof(float));
        cudaMallocManaged(&atrans, DIM * sizeof(float));
        cudaMallocManaged(&x, M * sizeof(float));
        cudaMallocManaged(&y, M * sizeof(float));

        /* Initialize arrays */
        for (i = 0; i < N; i++) {
                x[i] = (float)(rand() % 100);
                for (j = 0; j < M; j++)
                        a[i * M + j] = (float)(rand() % 100);
        }
```

```
        transnorm<<<NBLK, BLKSIZE>>>(a, atrans, x, y);
        /* Wait for all devices to finish */
        cudaDeviceSynchronize();

        pretty_print_2d(a, "A", N, M);
        pretty_print_2d(atrans, "A_trans", M, N);
        pretty_print_1d(x, "X", M);
        pretty_print_1d(y, "Y", M);

        cudaFree(a);
        cudaFree(atrans);
        cudaFree(x);
        cudaFree(y);

        return (0);
}
```

1.3.2 Ενδεικτικά τρεξίματα

Για `NxM = 4x2` και `blocksize = 256`



Για NxM = 4x8 και blocksize = 256

```
cuda17@rncp-ubuntu:~$ ./ex2b_b

A = [
        [67.00, 59.00, 97.00, 12.00, 98.00, 5.00, 56.00, 45.00]
        [50.00, 17.00, 87.00, 98.00, 94.00, 28.00, 54.00, 20.00]
        [73.00, 38.00, 3.00, 67.00, 71.00, 38.00, 25.00, 57.00]
        [66.00, 87.00, 85.00, 74.00, 7.00, 97.00, 23.00, 19.00]
]

A_trans = [
        [67.00, 50.00, 73.00, 66.00]
        [59.00, 17.00, 38.00, 87.00]
        [97.00, 87.00, 3.00, 85.00]
        [12.00, 98.00, 67.00, 74.00]
        [98.00, 94.00, 71.00, 7.00]
        [5.00, 28.00, 38.00, 97.00]
        [56.00, 54.00, 25.00, 23.00]
        [45.00, 20.00, 57.00, 19.00]
]

X = [8.00, 47.00, 98.00, 70.00, 0.00, 0.00, 0.00, 0.00]

Y = [35912.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
```

Για NxM = 8x8 και blocksize = 1024

```
cuda17@rncp-ubuntu:~$ ./ex2b_b

A = [
        [17.00, 88.00, 47.00, 98.00, 69.00, 77.00, 61.00, 29.00]
        [42.00, 59.00, 79.00, 29.00, 67.00, 31.00, 37.00, 83.00]
        [91.00, 42.00, 75.00, 67.00, 42.00, 91.00, 91.00, 74.00]
        [32.00, 67.00, 69.00, 49.00, 37.00, 57.00, 48.00, 35.00]
        [77.00, 48.00, 8.00, 4.00, 91.00, 67.00, 84.00, 72.00]
        [67.00, 61.00, 69.00, 96.00, 52.00, 63.00, 71.00, 72.00]
        [15.00, 15.00, 32.00, 36.00, 47.00, 99.00, 5.00, 48.00]
        [62.00, 96.00, 23.00, 41.00, 26.00, 72.00, 49.00, 30.00]
]

A_trans = [
        [17.00, 42.00, 91.00, 32.00, 77.00, 67.00, 15.00, 62.00]
        [88.00, 59.00, 42.00, 67.00, 48.00, 61.00, 15.00, 96.00]
        [47.00, 79.00, 75.00, 69.00, 8.00, 69.00, 32.00, 23.00]
        [98.00, 29.00, 67.00, 49.00, 4.00, 96.00, 36.00, 41.00]
        [69.00, 67.00, 42.00, 37.00, 91.00, 52.00, 47.00, 26.00]
        [77.00, 31.00, 91.00, 57.00, 67.00, 63.00, 99.00, 72.00]
        [61.00, 37.00, 91.00, 48.00, 84.00, 71.00, 5.00, 49.00]
        [29.00, 83.00, 74.00, 35.00, 72.00, 72.00, 48.00, 30.00]
]

X = [16.00, 75.00, 28.00, 69.00, 26.00, 86.00, 5.00, 88.00]

Y = [4624.00, 580800.00, 61852.00, 662676.00, 123786.00, 509894.00, 18605.00, 74008.00]
```

2 Προβλήματα

Δεν υλοποίησα την άσκηση 2Β-Γ (συνδιακύμανση).