merge_sort.cu:

```
••• merge_sort.cu(117): warning #177-D: variable "left_im1" was declared but neve
          int left_im1 = (i > 0) ? in[L0 + i - 1] :
                ^

    Remark: The warnings can be suppressed with "-diag-suppress <warning-number>"

    merge_sort.cu(119): warning #177-D: variable "right_jm1" was declared but nev
          int right_jm1 = (j > 0) ? in[R0 + j - 1] :
                ^

    Enter N (e.g., 10000 / 100000 / 1000000): 10000
    CPU merge sort time: 2 ms
    GPU merge sort time: 0.182304 ms
    CPU sorted: true
    GPU sorted: true
    CPU == GPU: true
```

quick_sort.cu:

```
    !nvcc -arch=sm_75 quick_sort.cu -o app

    # Запуск
    !./app

••• quick_sort.cu(187): warning #177-D: variable "fillGrid" was declared but nev
                  int fillGrid = (fillN + fillBlock - 1) / fillBlock;
                        ^

    Remark: The warnings can be suppressed with "-diag-suppress <warning-number>

    Enter N (e.g., 10000 / 100000 / 1000000): 10000
    CPU quick sort time: 1 ms
    GPU quick sort time: 3.64138 ms
    CPU sorted: true
    GPU sorted: true
    CPU == GPU: true
```

heap_sort.cu:

```
# Компиляция
!nvcc -arch=sm_75 heap_sort.cu -o app

# Запуск
!./app
```

```
Enter N (e.g., 10000 / 100000 / 1000000): 10000
CPU heap sort time: 3 ms
GPU heap sort time: 397.216 ms
CPU sorted: true
GPU sorted: false
CPU == GPU: false
```

Контрольные вопросы

Различие между последовательной и параллельной merge sort

Последовательно рекурсивно делим и сливаем по одному. Параллельно можно сортировать куски одновременно и сливать пары сегментов параллельными потоками.

Как потоки и блоки влияют на производительность CUDA

Чем лучше загрузка SM (больше активных блоков/потоков) и чем меньше простаиваний/дивергенции, тем быстрее. Плохой размер блоков даёт мало occupancy и больше накладных расходов.

Сложности quick sort на GPU

Основная проблема — partition и рекурсия: много зависимостей, разные размеры подмассивов, нерегулярный доступ к памяти и сильная дивергенция ветвлений.

Когда GPU сортировка может быть хуже CPU

Если массив маленький, если много копирований CPU↔GPU, если алгоритм плохо параллелится или упирается в память/синхронизацию.

Почему важно выбирать размер блоков и потоков

Это влияет на occupancy, использование регистров/shared memory и на то, насколько эффективно скрывается задержка доступа к памяти.

Как shared memory влияет на скорость сортировки

Если данные часто переиспользуются внутри блока, shared memory резко уменьшает обращения к глобальной памяти и ускоряет сортировку (например, block-sort/bitonic).

Что значит “разделяй и властвуй”

Разбиваем задачу на подзадачи (части массива), решаем их отдельно и объединяем (merge) или рекурсивно сортируем части (quick/merge).