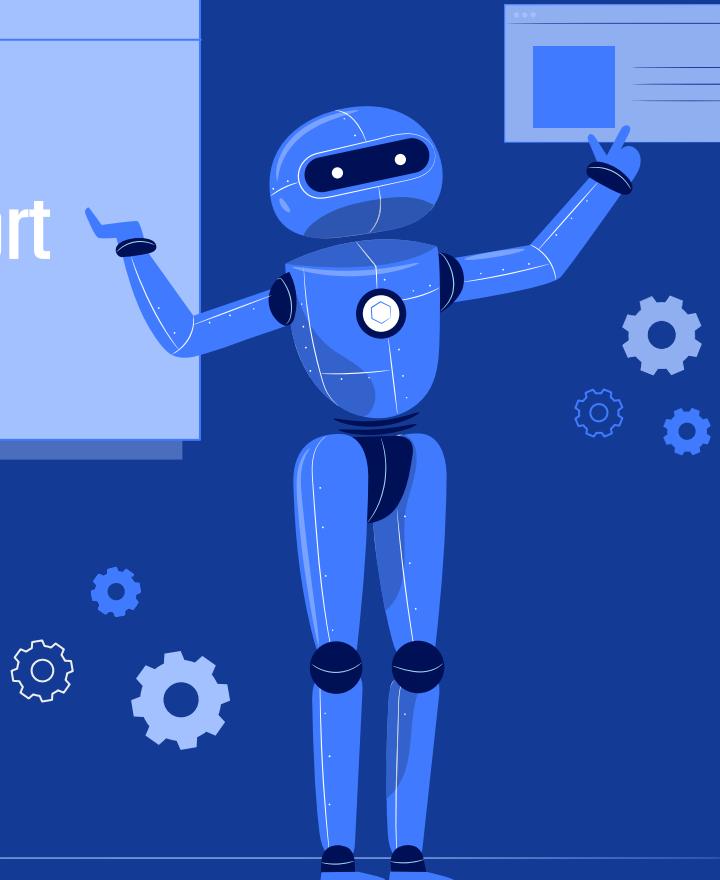


HPC project:

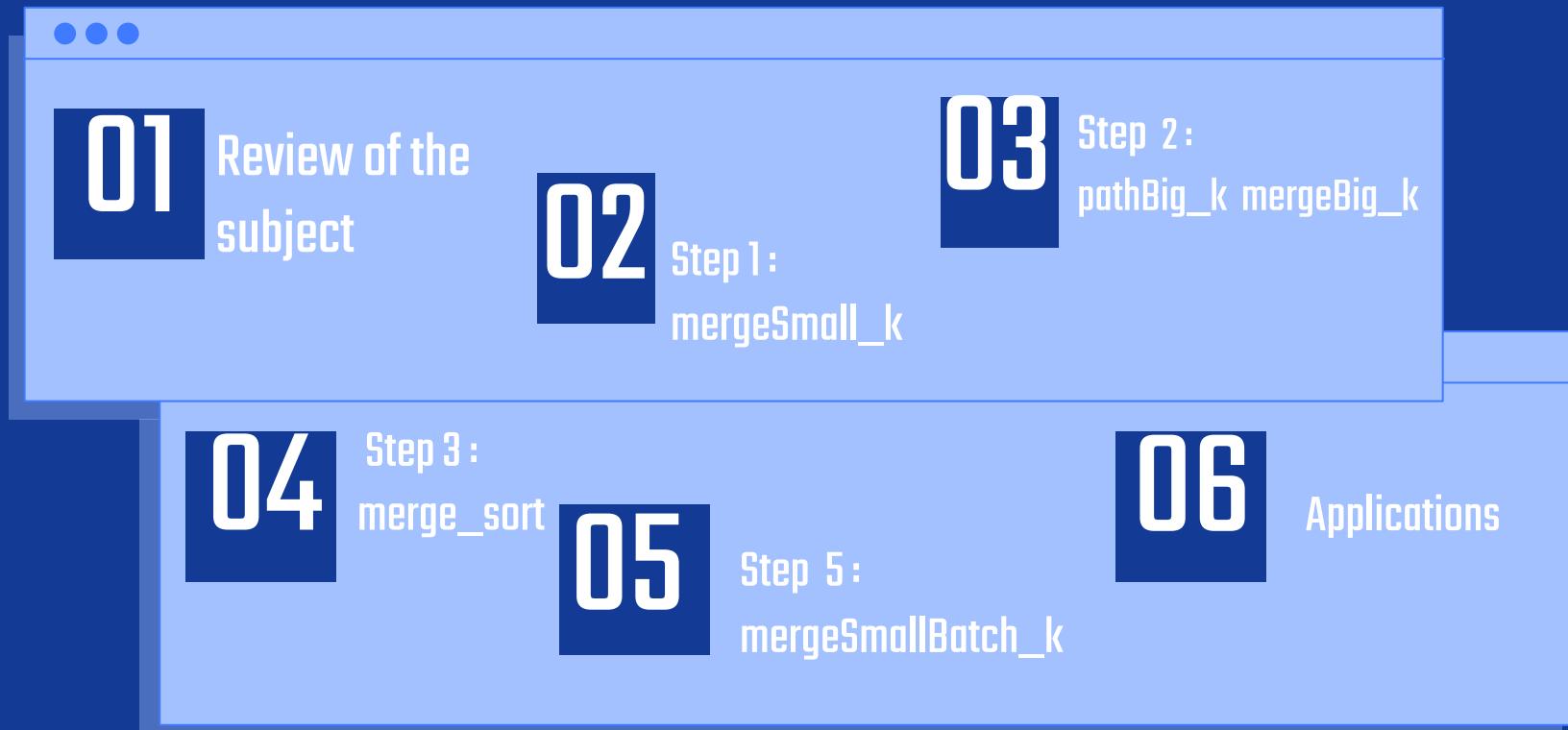
Batch merge and merge path sort



Astrid Legay – Marco
Naguib – MAIN5

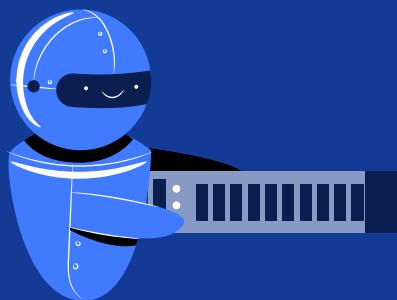


Summary



01

Review of the subject



Sorting tables



APPLICATIONS

Database
Image processing
Graph theory



ALGORITHMS

Bubble sort
Quick sort
Merge sort

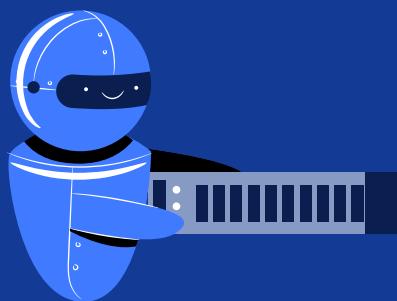


MERGESORT

Highly parallelizable
thanks to divide and
conquer

02

Step 1:
mergeSmall_k



Step 1



For $|A| + |B| \leq 1024$, write a kernel mergeSmall k that merges A and B using only one block of threads.

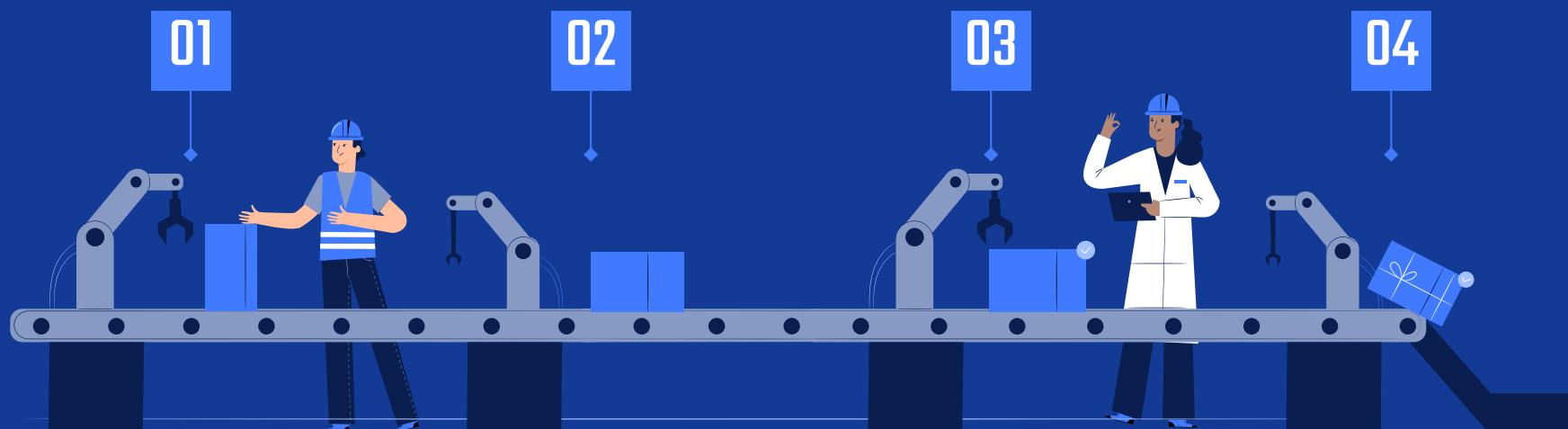
Process

Writing the algorithm A and B (sequentially)

Using the functions for testing and printing

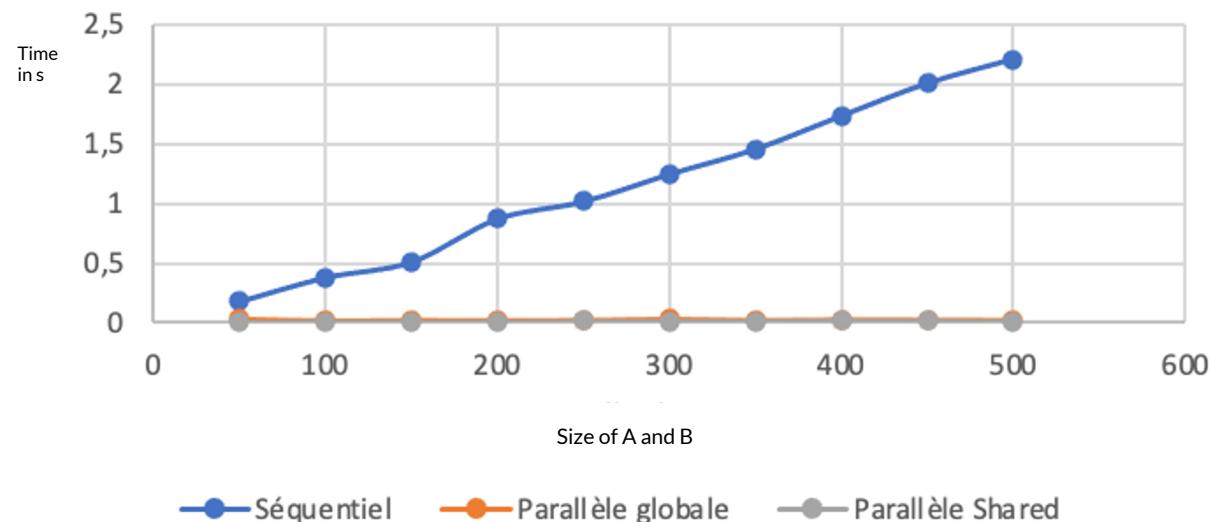
Writing MergeSmall_k on global and shared memory using Cuda

Tests and measurements

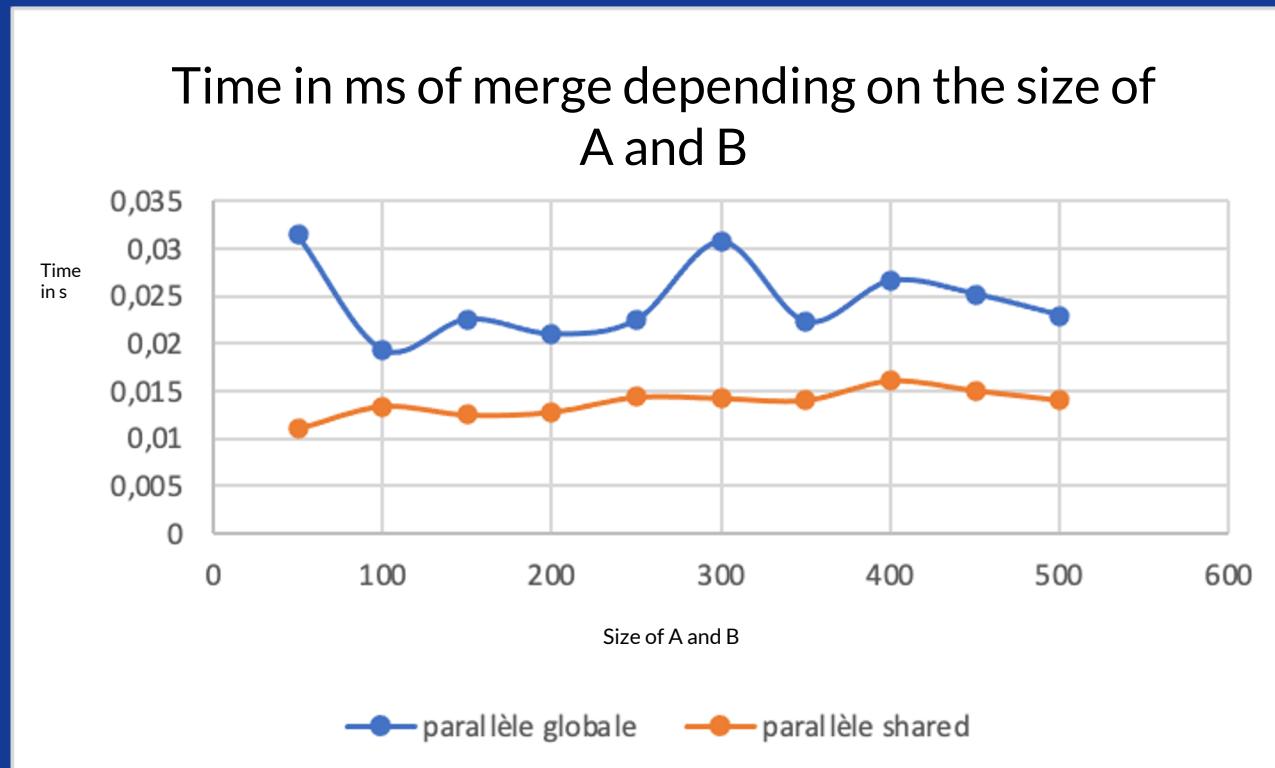


Results

Time in ms of merge depending on the size of A and B

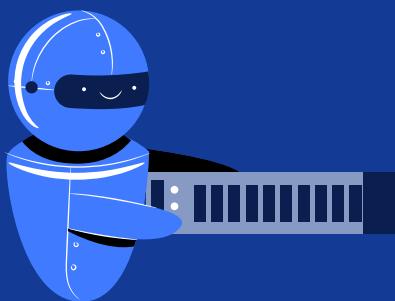


Global and shared memory comparison



03

Step 2 : pathBig_k et mergeBig_k

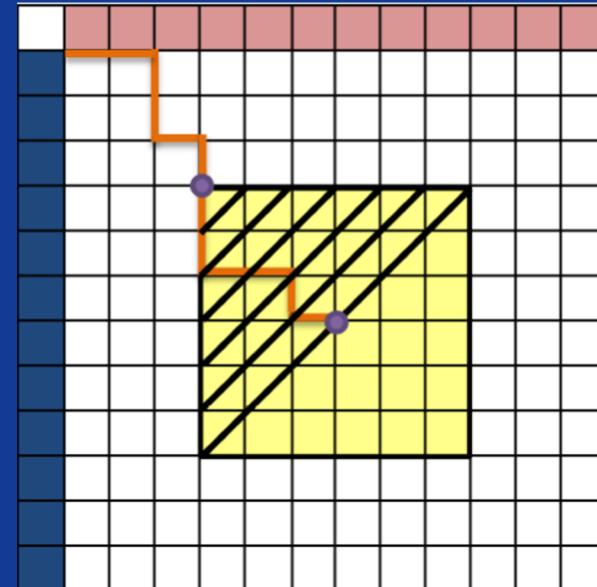
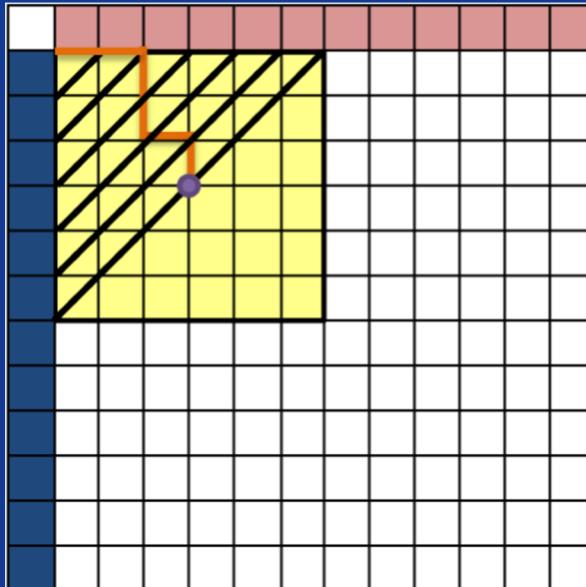


Step 2

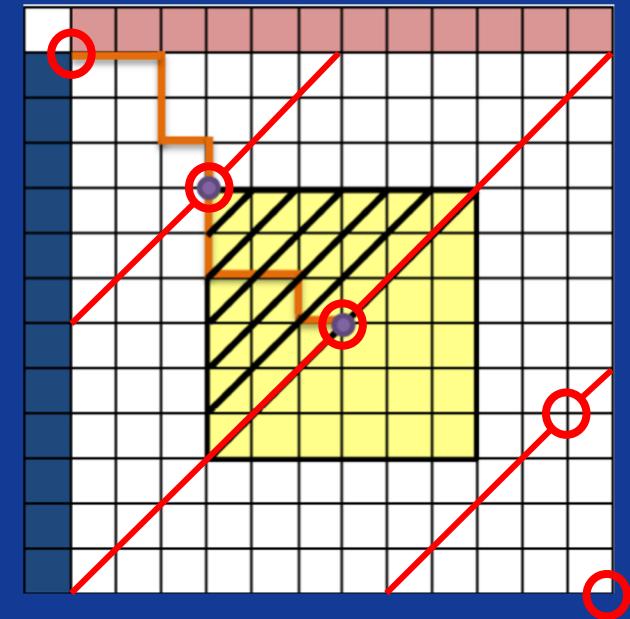
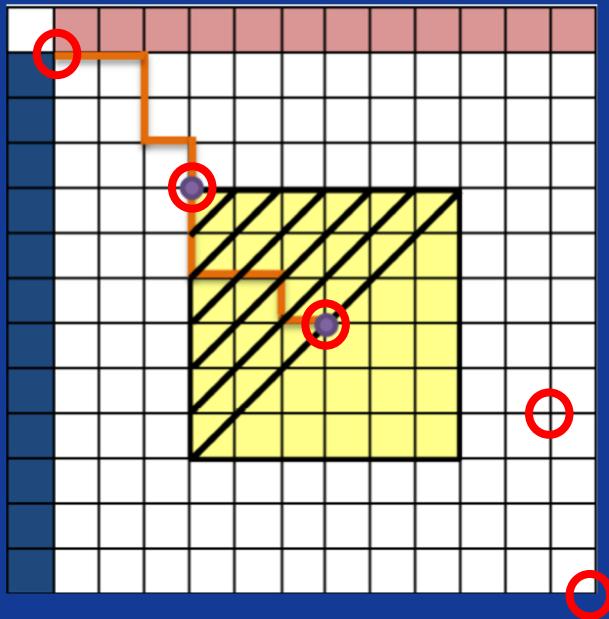


For any size $|A|+|B|=d$ sufficiently smaller than the global memory, write two kernels that merge A and B using various blocks: The first kernel pathBig k finds the merge path and the second one mergeBig k merges A and B.

Use of sliding windows



Use of sliding windows



Process

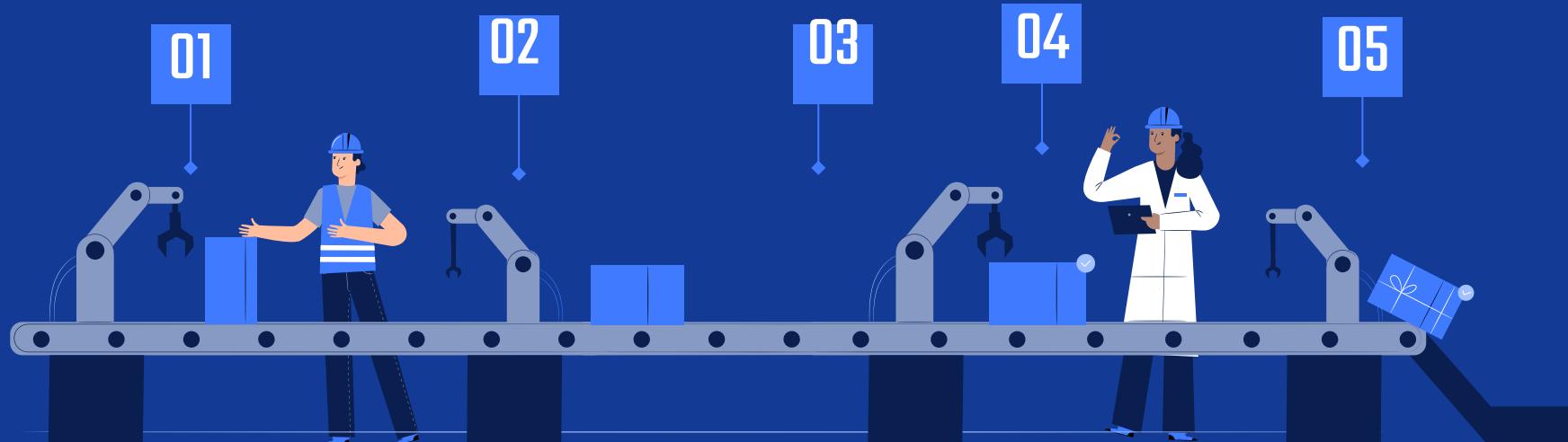
Using testing functions of question 1

Pathbig : finds the intersection between the way and the diagonal

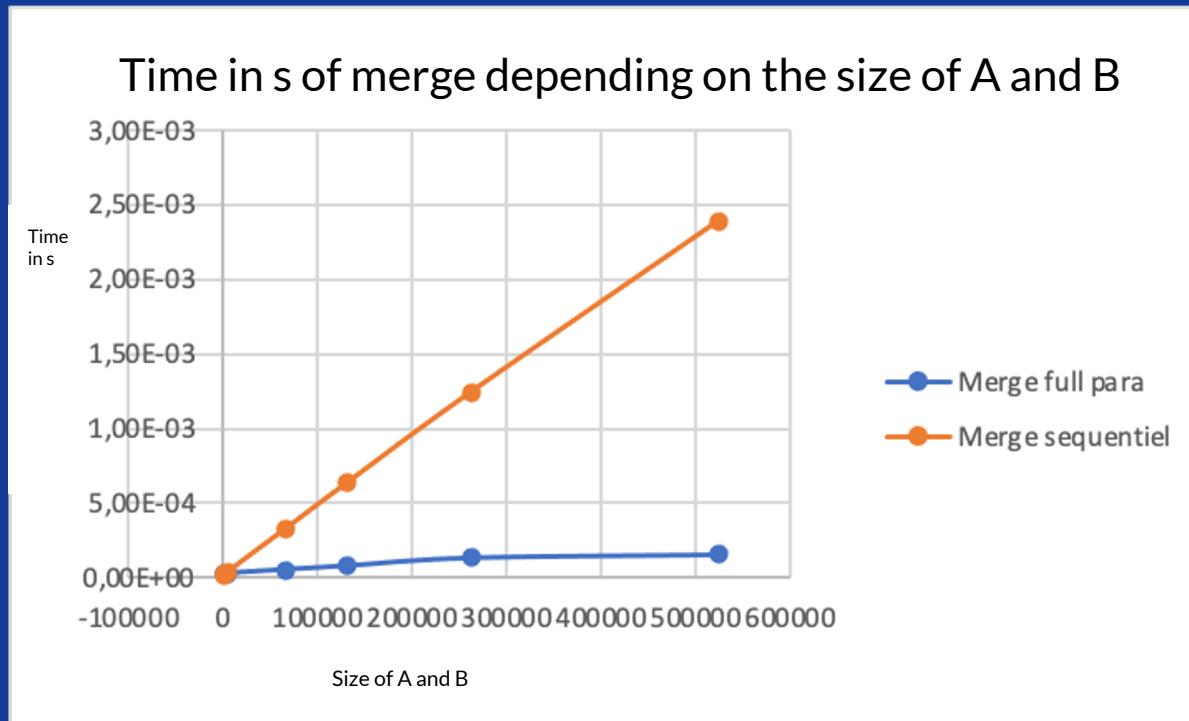
Mergebig : merge the sliding windows 1 by 1

Mergebig : merge the sliding windows in parallel

Tests and measurements

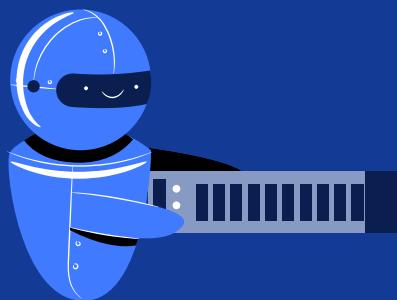


Results



04

Step 3 :
mergesort

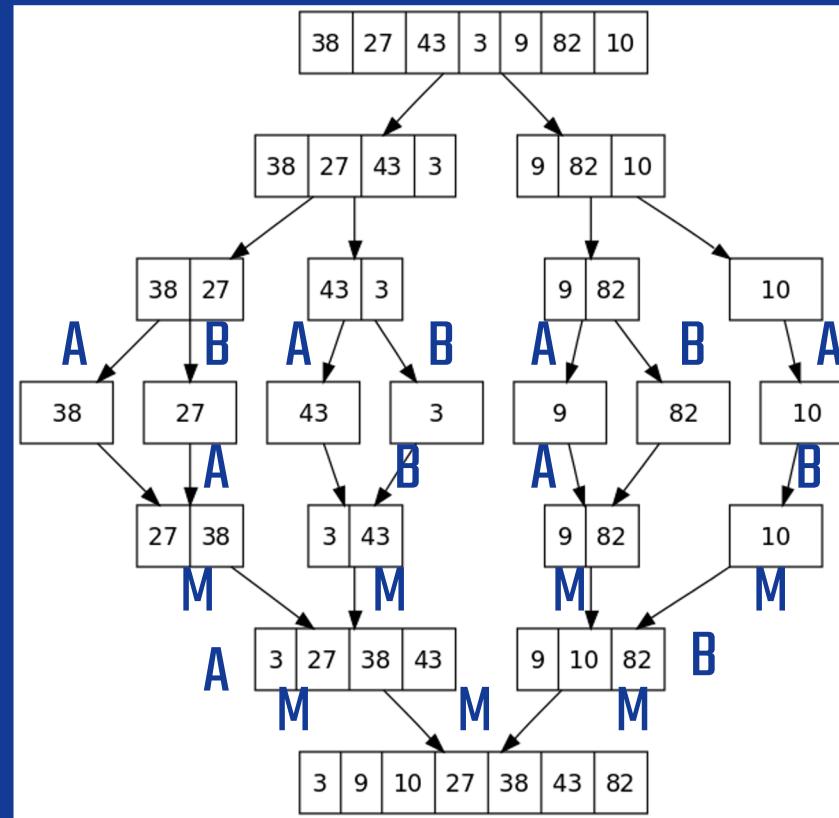


Step 3



Looping on appropriate calls of pathBig k and of mergeBig k, write a function that sorts any array M of size d sufficiently smaller than the global memory. Give the execution time with respect to d.

Merge Sort



Process

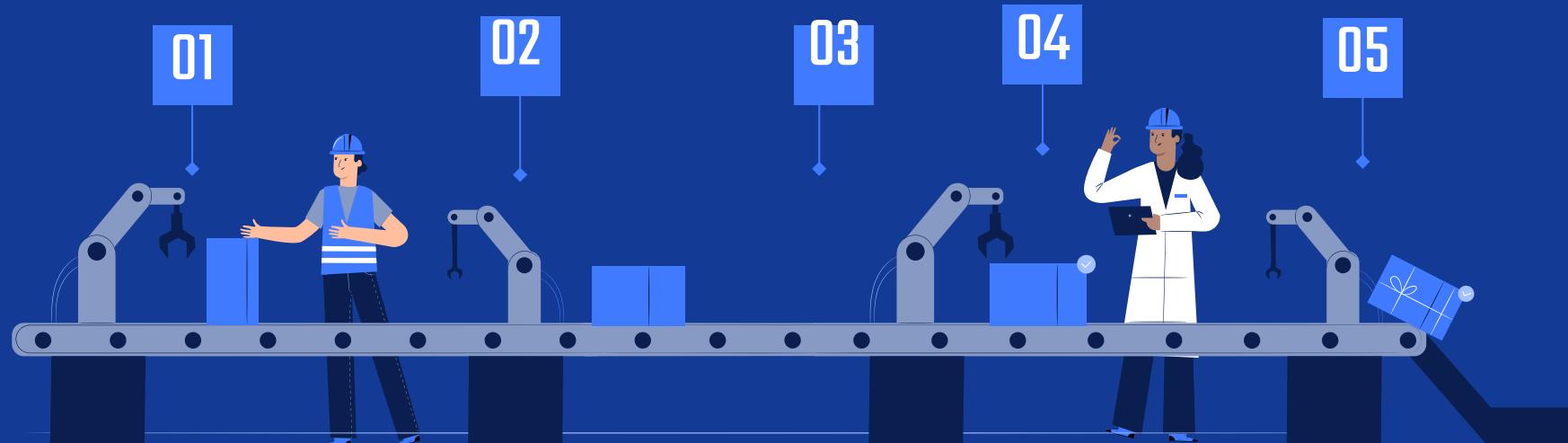
Sequential code
in C

Using the functions
for testing and
printing

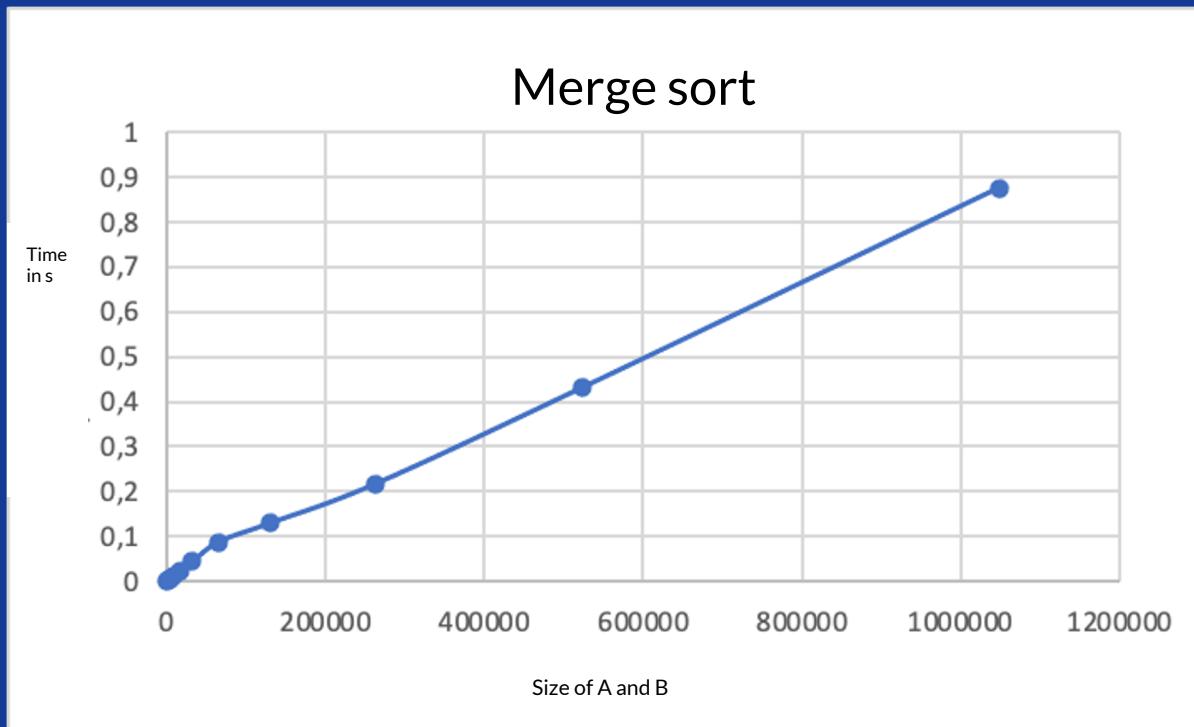
Parallelizing the
sequential code
to CUDA

Tests and
measurements

Optimization



Results



• • •

05

Step 5:
`mergeSmallBatch_k`



Step 4

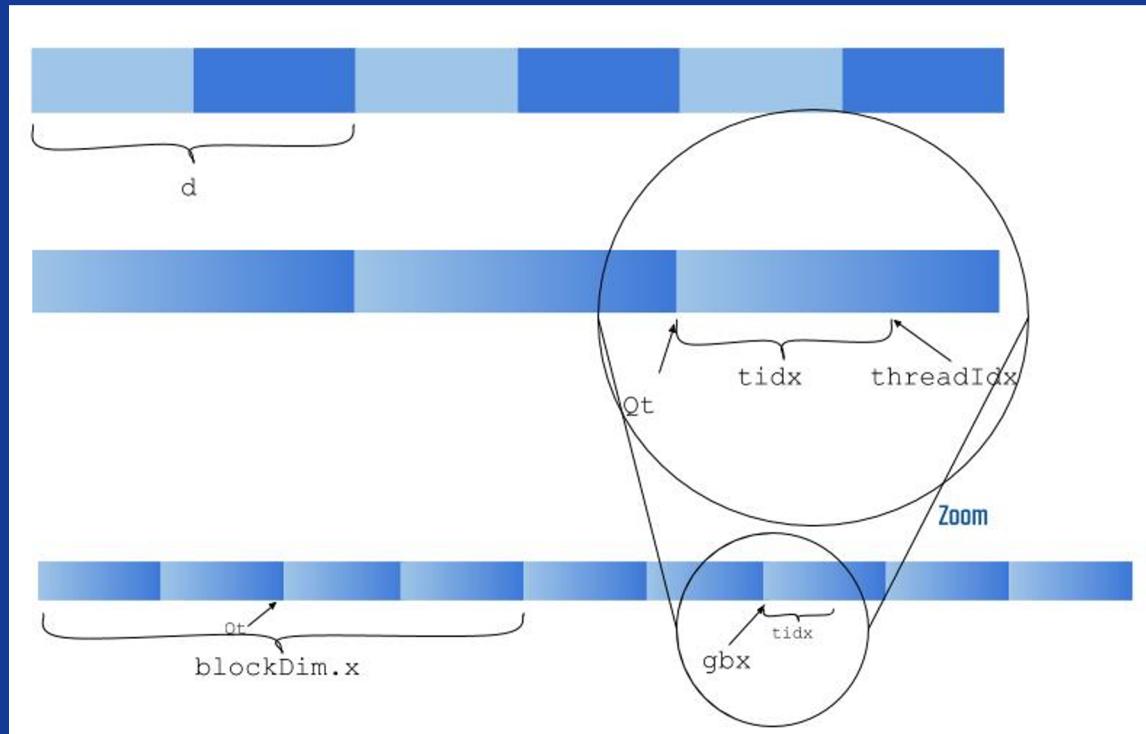


Explain why the indices

- `int tidx = threadIdx.x%d;`
- `int Qt = (threadIdx.x-tidx)/d;`
- `int gbx = Qt +
blockIdx.x*(blockDim.x/d);`

are important in the definition of
`mergeSmallBatch k`.

Question 4



Step 5



Write the kernel `mergeSmallBatch k` that batch merges two by two $\{A_i\}_{1 \leq i \leq N}$ and $\{B_i\}_{1 \leq i \leq N}$. Give the execution time with respect to $d = 4, 8, \dots, 1024$.

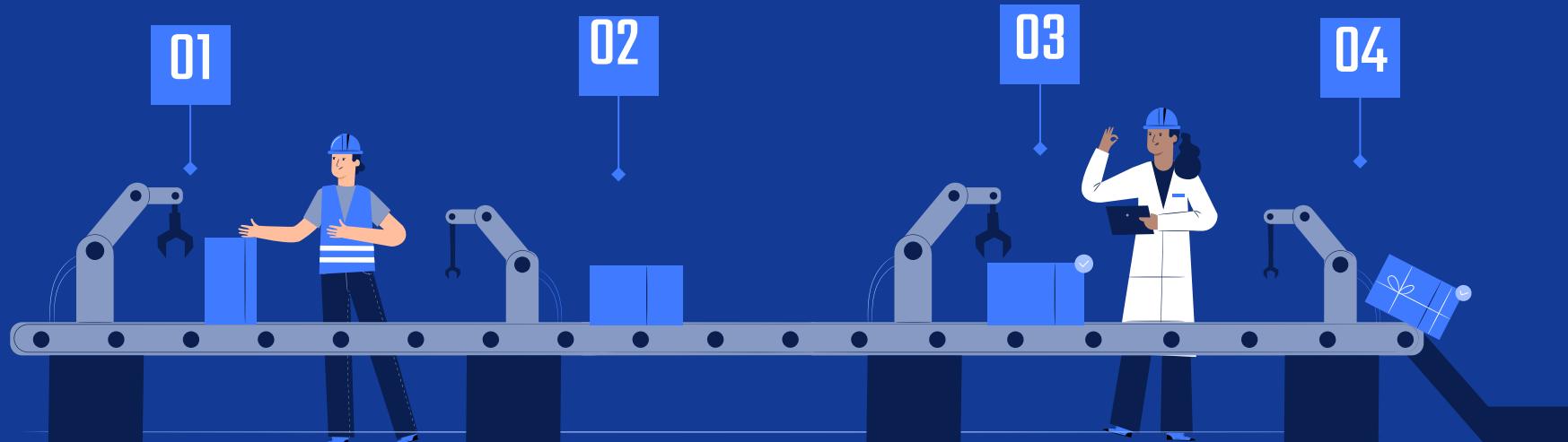
Process

Reusing MergeSmall

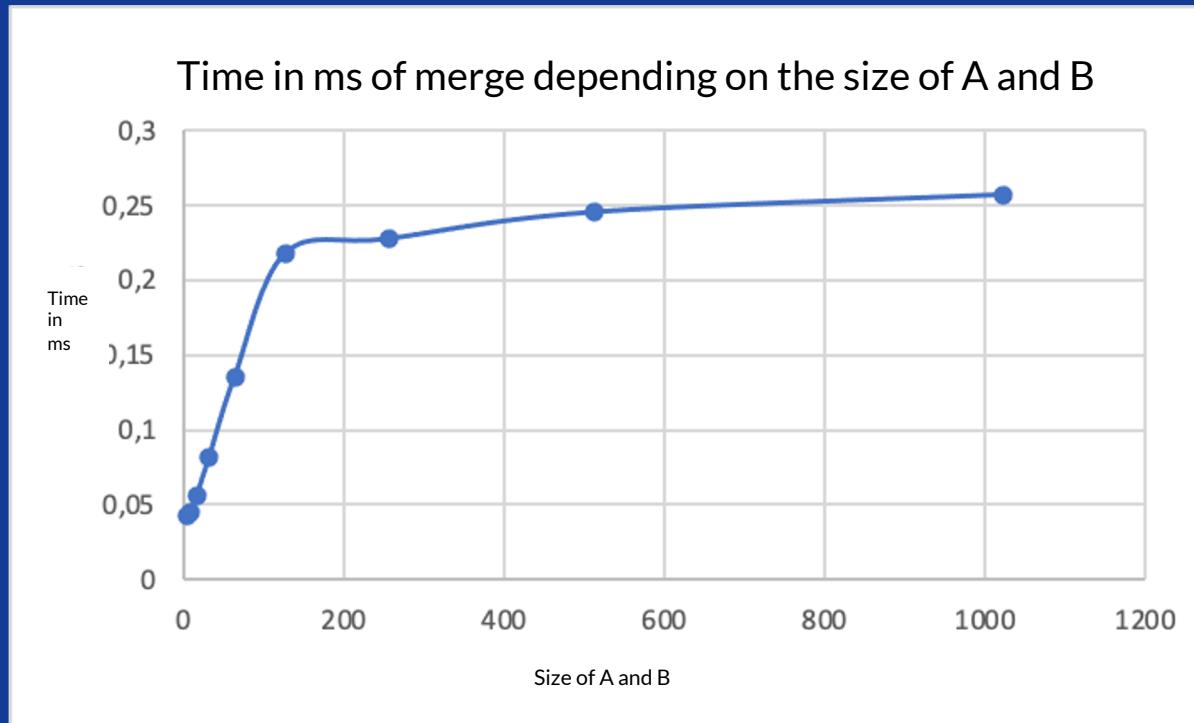
Using the indices in
question 4

Tests and
measurements

Optimization

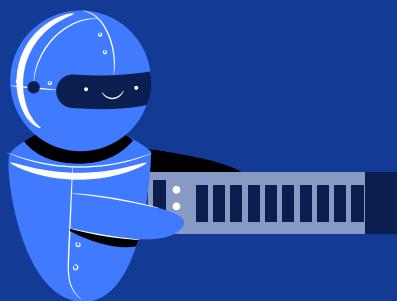


Results



05

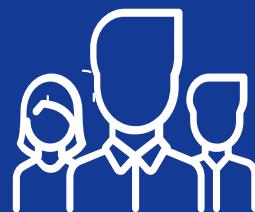
Applications



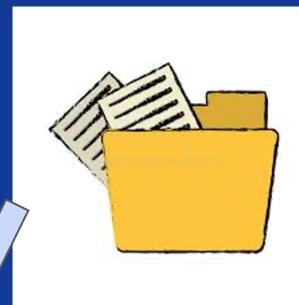
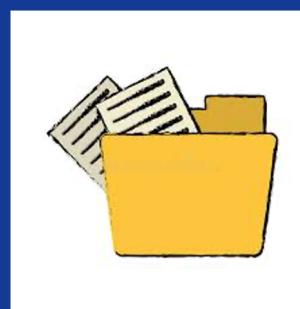


From a "practical" point of view

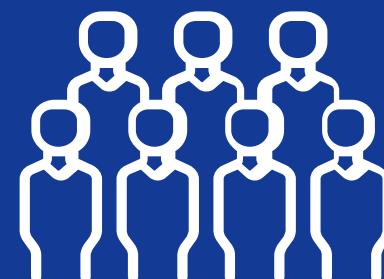
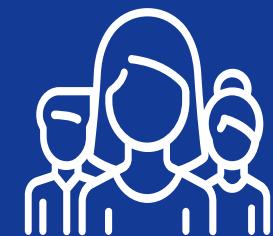
Company Merger



Compagny A

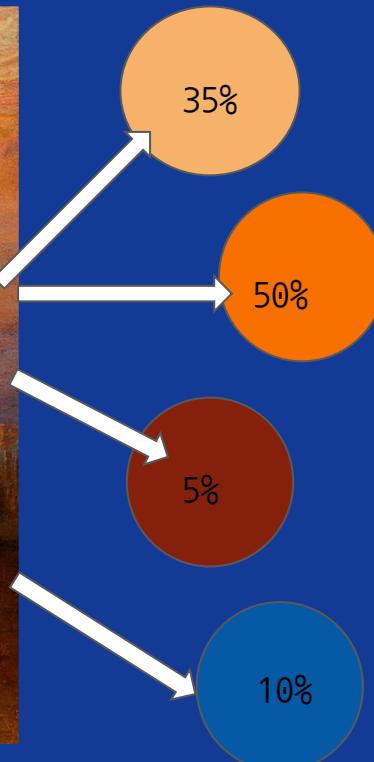
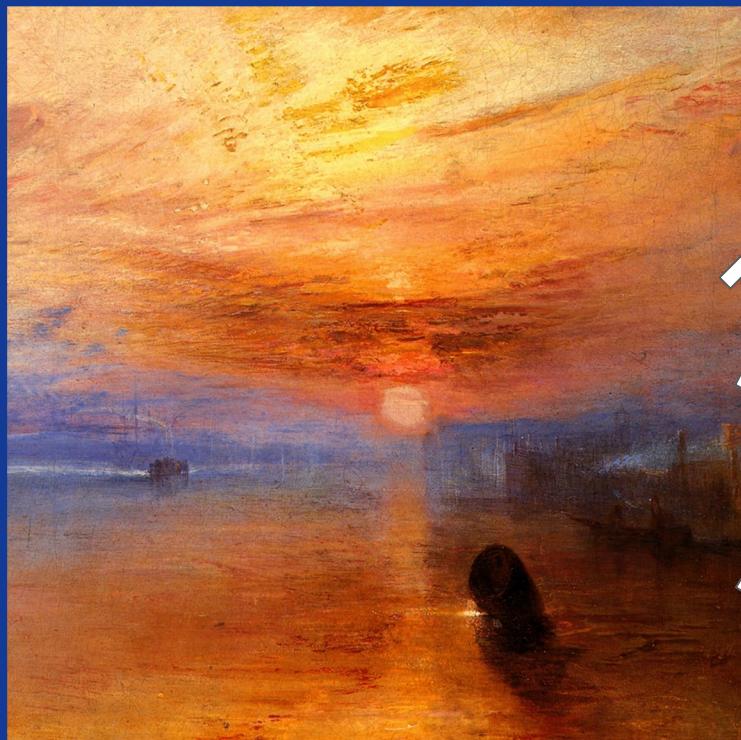


Compagny B



Compagny M

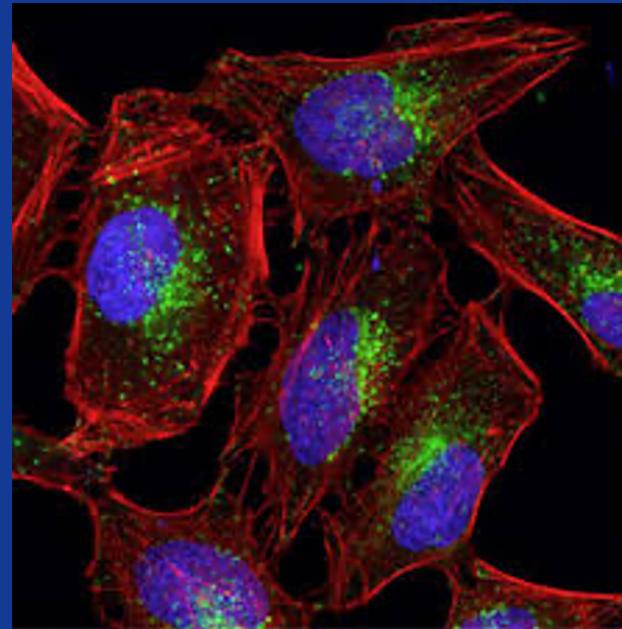
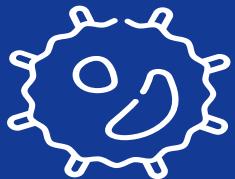
Image processing: in art



- Recognize the color palette
- Recognize the painter
- Sort by color
- Recognizing the artistic movement
- Recognize the artist's period

Image processing: in medicine

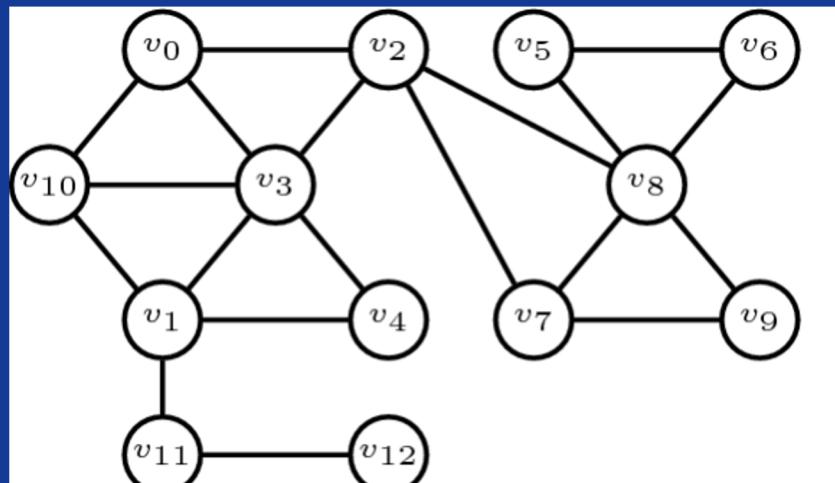
- Counting of cancer cells
- Identify how many there are



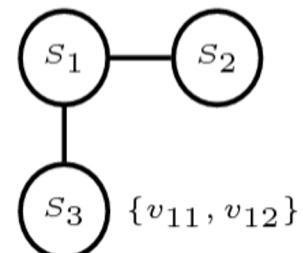


From a "technical" point of view

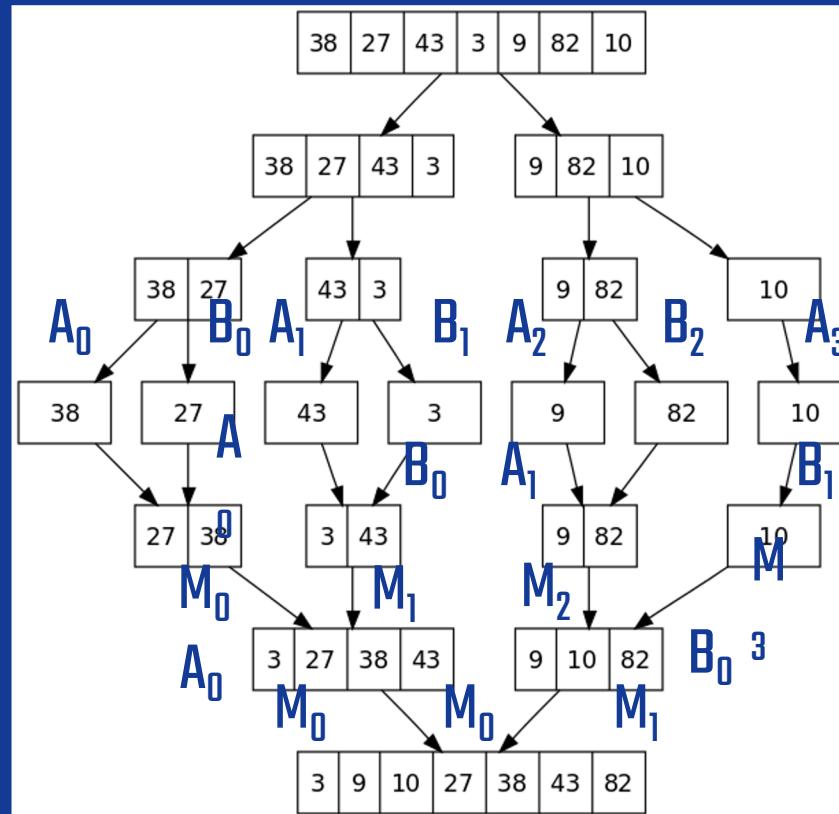
Web graph compression



$\{v_3, v_0, v_1, v_2, v_4, v_{10}\}$ $\{v_8, v_5, v_6, v_7, v_9\}$



Merge Sort with MergeSmallBatch



Comparaisons

01.

01.

Comparison between
the execution
times of question
3 and 5

02.

Comparison
between
MergeSmallBatch
and Numpy.sort()



Results

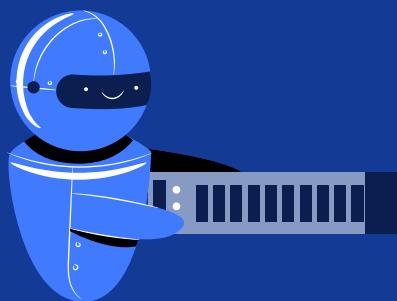
Comparaison between MergeSmallBatch and numpy





End

Conclusion



Conclusion



GPU

Discover the world of the GPU in practice

Cuda - Parallelism

Discover another way to set up parallelism

Problems

- Middle
- Infinite loop
- Managing the leftovers

Think about different possible applications

Thank you

Do you have any
question ?

CREDITS: This presentation template was created by Slidesgo,
including icons by Flaticon, infographics & images by Freepik and
illustrations by Stories

