**Recursive Sorting algorithms**

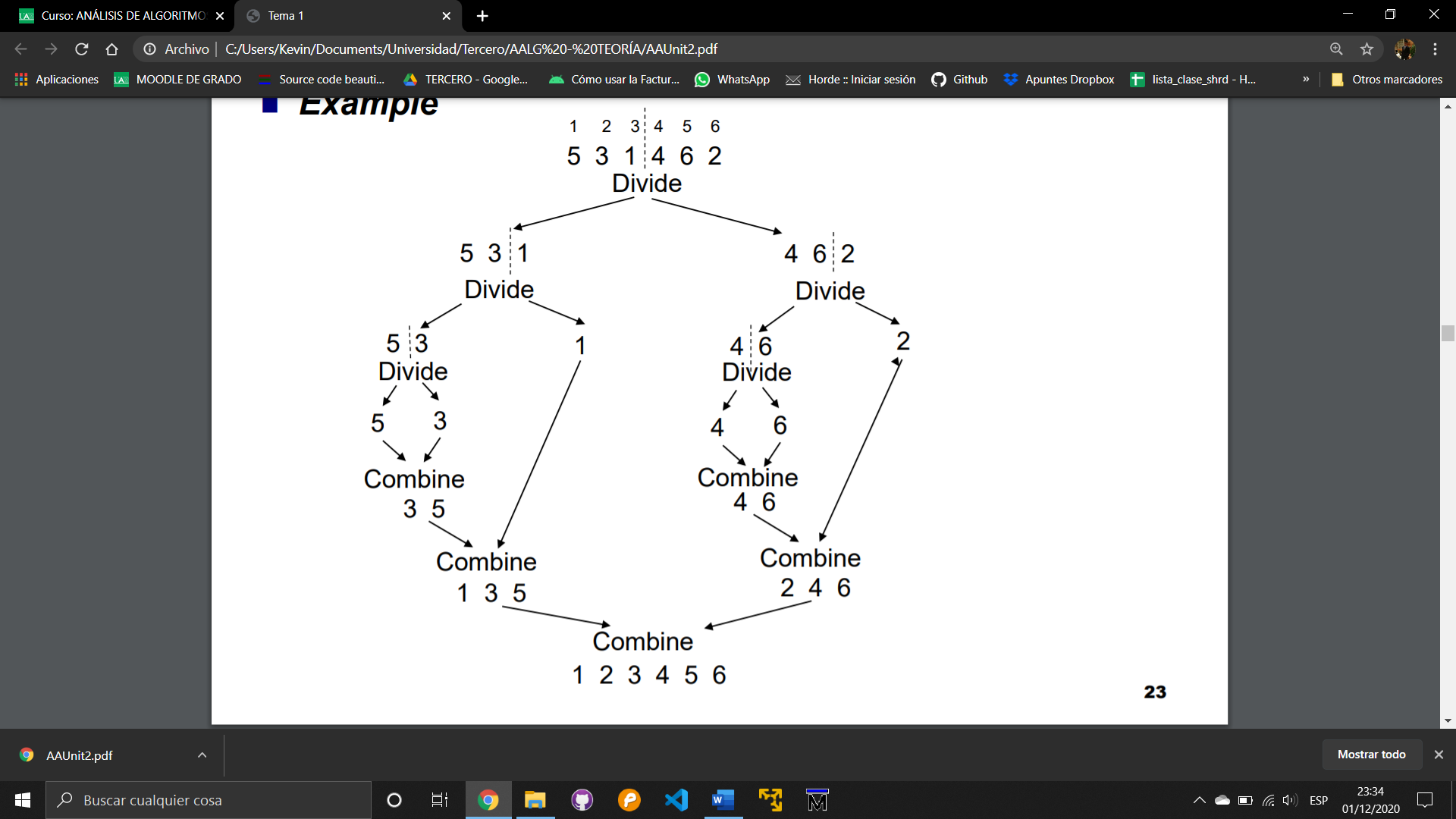
1. **Divide and conquer methods**

* **MergeSort**: Mergesort is a divide and conquer algorithm which is divided in two parts, **divide** and **combine**.

It uses recursive calls to itself, therefore the complexity is calculated with recursive equations.

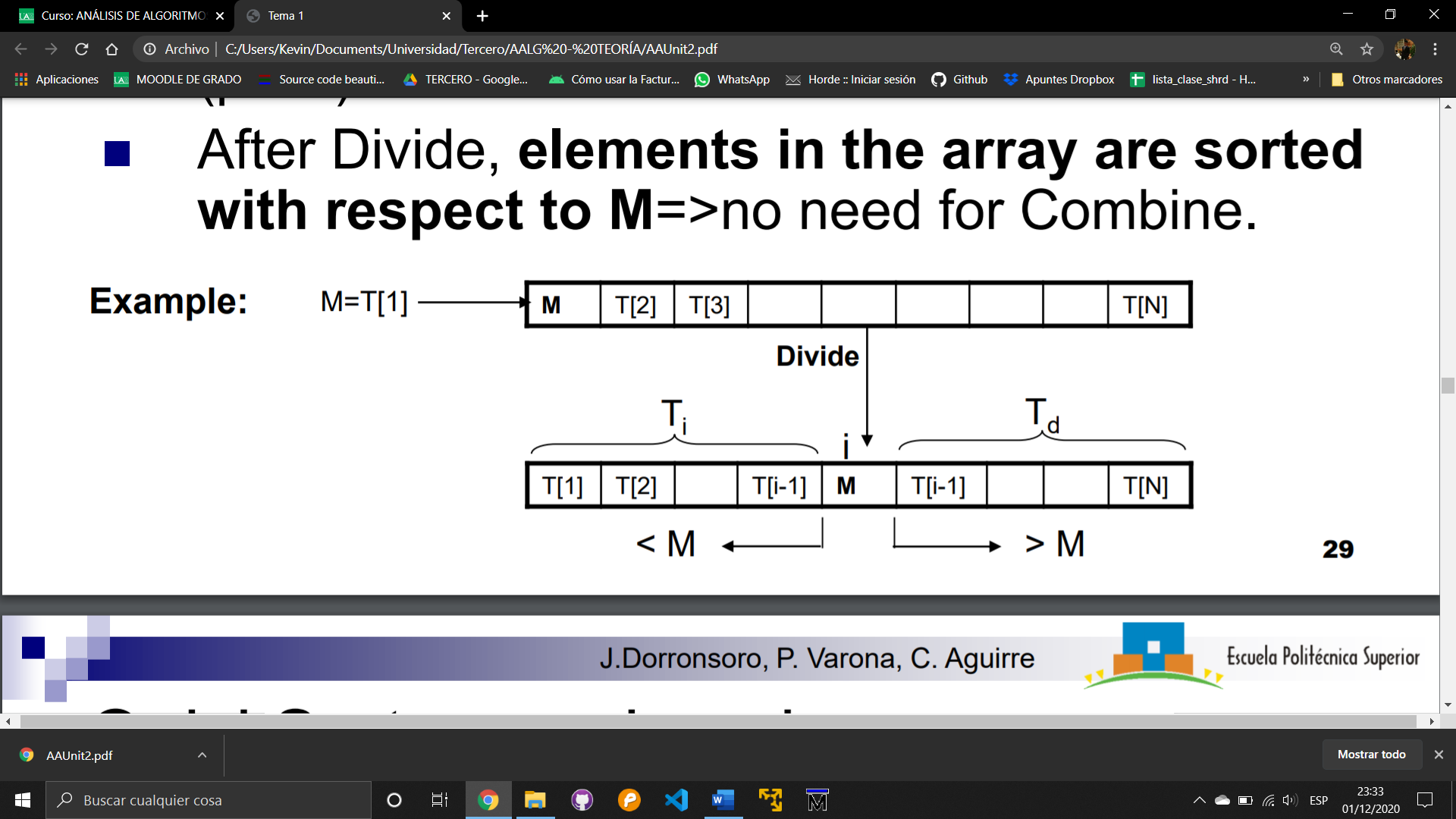
The comparisons made by combine in mergesort are:

The runtime of the algorithm is good, but it uses dynamic memory and being recursive makes it less efficient regarding the stack.



* **Quicksort**: Quicksort is also a divide and conquer algorithm which is divided in two parts, **divide** and **combine**.

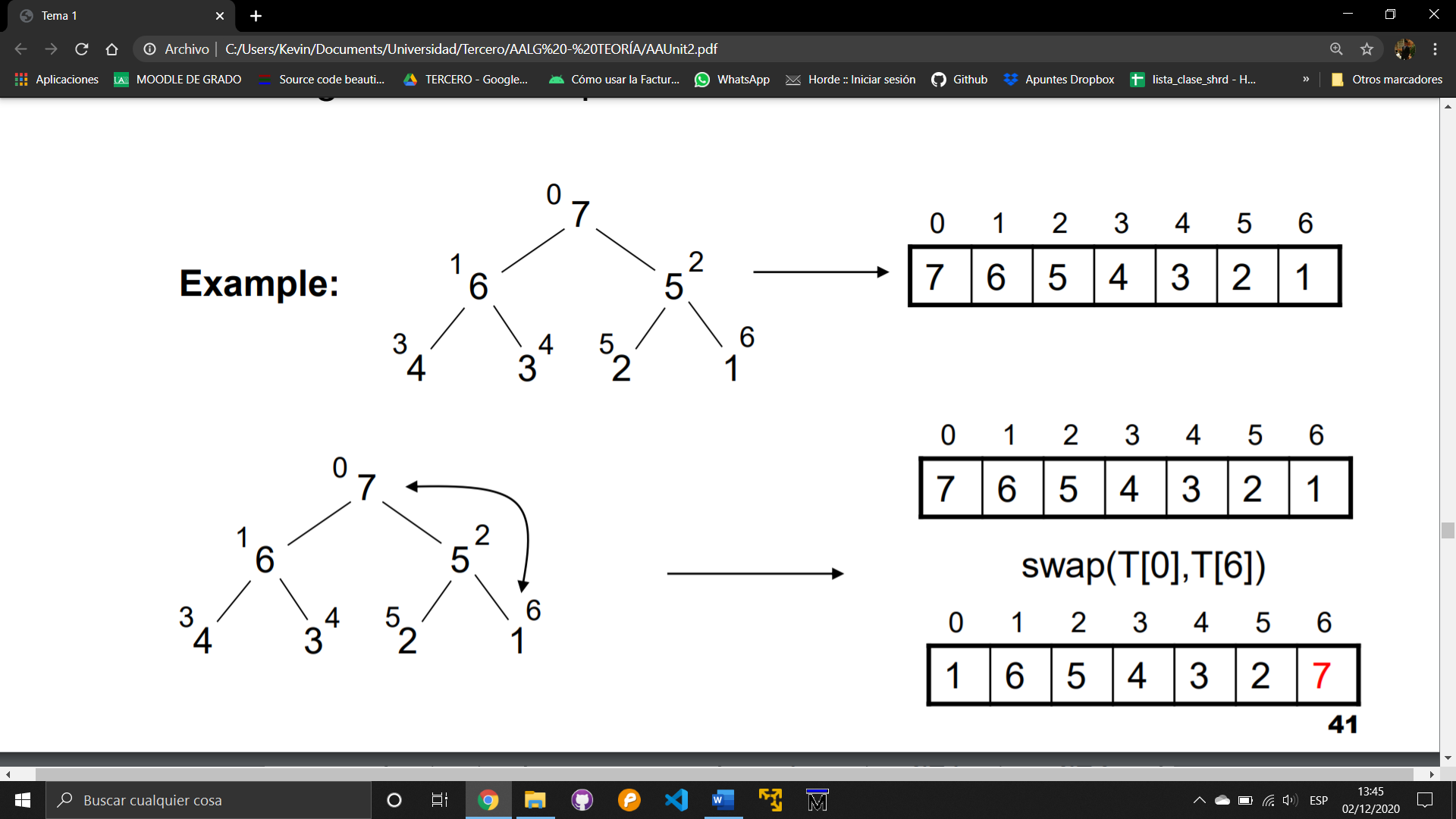
*Divide* takes an element of the array and then the array is sorted based in that element.



The **pivot** (M) can be selected in a lot of ways, not just the first element.

Mergesort uses another table and quicksort has a worse worst case.

* **HeapSort**: A heap is an almost complete binary tree. A **Max-heap** is a heap such that for all subtrees, the children are smaller than the parent.



In order to get the index of a child given the index of a parent we just apply this formula:

Parent: J Left child: 2·J+1 Right child: 2·J+2

If we have the child index we apply this:

Child: J Parent:

Heapsort is divided in 2 parts, first we **create the Max-heap** with the function **heapify** and finally you **sort the Max-heap**.

The height of the **Max-heap** depends on the number of nodes. **Height(T)=**

Abstract runtime of heapsort:

1. **Decision trees for sorting algorithms**

There is not any sorting algorithm which is better than heapsort( ), in order to prove this, we have decision trees.

A decision tree is composed by a **comparison-based sorting algorithm A** and a **size N**.

The **height of a node** in a tree is the number of edges on the longest path from a node to a leaf.

The **depth of a node** in a tree is the number of edges from the node to the root node.

The **height of a tree** = **Depth of a tree** is the height of the root node.

The **number of leaves** in is **N!**, and the **height of a decision tree** with N! leaves is

The **nA(σ) = number of key comparisons** is

The worst case of an algorithm is

The **minimum height/worst case of an algorithm**, regarding comparisons, with N! leaves is .

The **minimum average height/average case of an algorithm**, regarding comparisons,with N! leaves is

The **external path length (EPL)** are the sum of lengths of all paths from the root to a leaf