PARALLEL AND SEQUENTIAL ALGORITHMS AND DATA STRUCTURES

LECTURE 1
INTRODUCTION

华中科技大学

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

TOPIC and Reference OF THIS COURSE

http://www.cs.cmu.edu/~15210/

https://www.diderot.one/courses/121/books

- This course covers
  - defining precisely the problem you want to solve,
  - learning the different algorithm-design techniques for solving problems,
  - designing abstract data types and the data structures that implement them
  - analyzing and comparing the cost of algorithms and data structures

SYNOPSIS

- **Parallelism**
- **Specification, Problem, and Implementation**

PARALLELSM

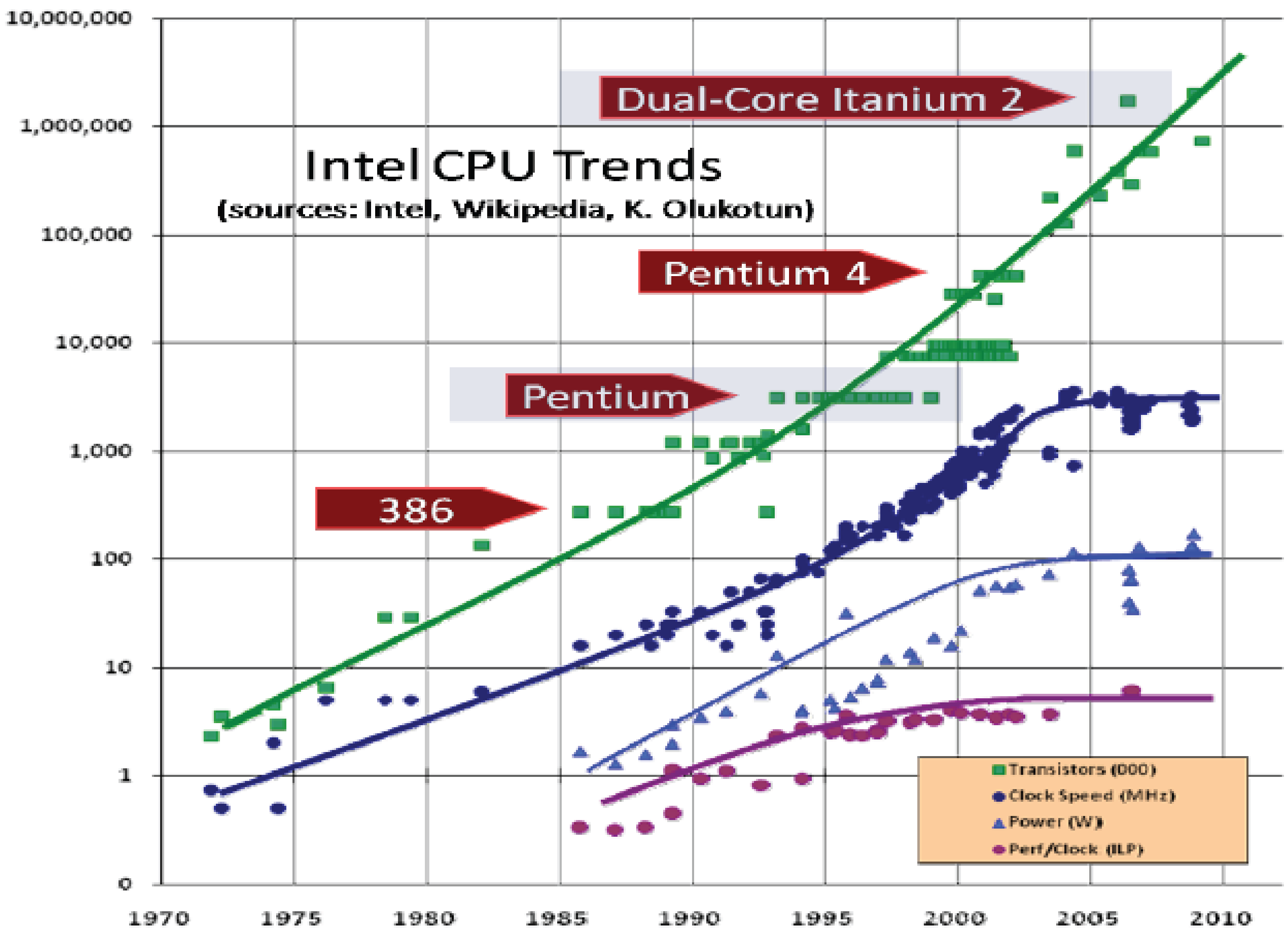
- Why should we care about parallelsm?
  - More powerful
  - Efficiency in terms of energy usage
- How much energy it takes to run a computation twice as fast using a sequential computer?
  - Energy consumption is a cubic function of clock frequency
  - How much ?

MOORE'S LAW IN TOP500

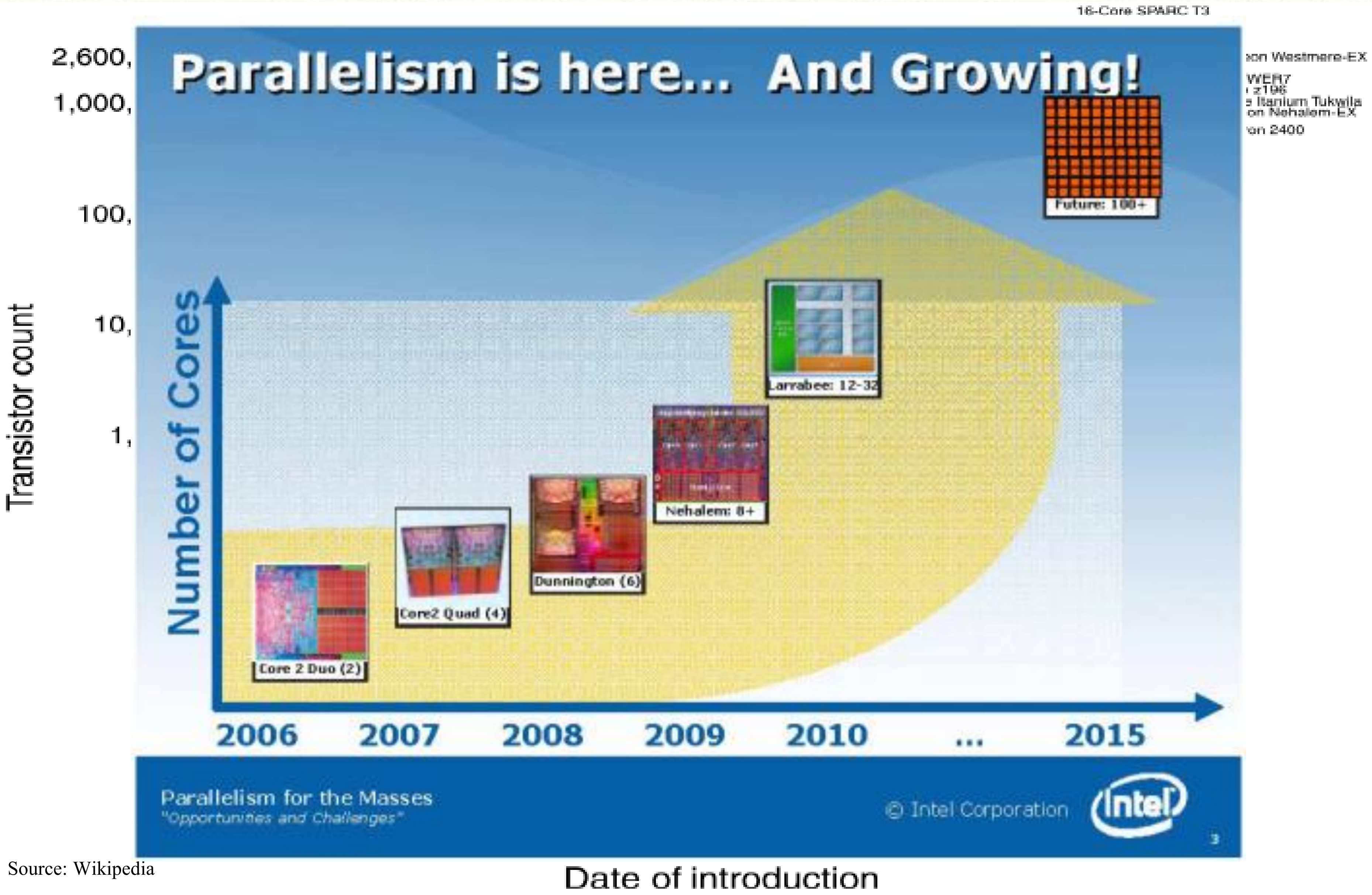
- Roughly 10x performance every 4 years

Source：
http://www.top500.org/statistics/perfdevel/

PROCESSOR TECHNOLOGY



TECHNOLOGY - MOORE'S LAW



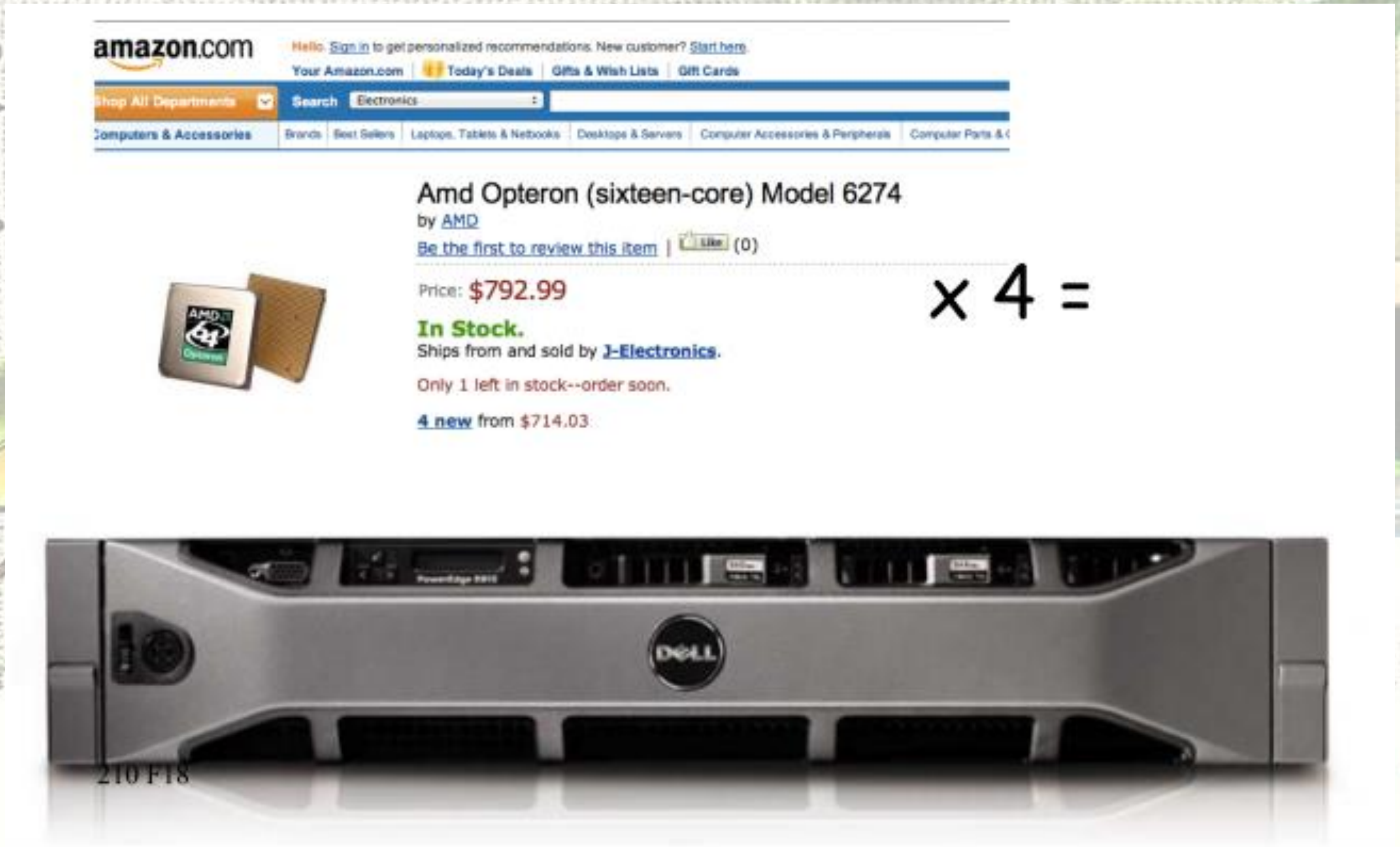
Source: Wikipedia

PARALLEL HARDWARE

- Nearly impossible to avoid parallelism
  - Can you give examples not have several processors?

PARALLEL HARDWARE

- 64 core blade servers ($6K) (shared memory)



PARALLEL HARDWARE

- **4992 "cuda" cores**

Roll over image to zoom in

Nvidia Tesla K80 24GB GPU Accelerator passive cooling 2x Kepler GK210 900-22080-0000-000

by NVIDIA

29 customer reviews | 11 answered questions

Price: $4,295.95 + $11.55 shipping

Note: Not eligible for Amazon Prime.

In Stock.

Ships from and sold by eServer PRO.

Estimated Delivery Date: Aug. 27 - Sept. 1 when you choose Expedited at checkout.

- Nvidia Tesla K80 GPU: 2x Kepler GK210
- Memory size (GDDR5) : 24GB (12GB per GPU)
- CUDA cores: 4992 (2496 per GPU)
- Memory bandwidth: 480 GB/sec (240 GB/sec per GPU)
- 2.91 Tflops double precision performance with NVIDIA GPU Boost - See more at: http://www.nvidia.com/object/tesla-servers.html#sthash.IF5LVWFq.dpuf

4 new from $4,135.00

Upgrading to a Solid-State Drive?

Learn how to install an SSD with Amazon Tech Shorts. Learn more

PARALLEL HARDWARE

- **Nearly impossible to avoid parallelism**
  - **Do you know how many computers are engaged in answering a typical web search?**

Parallel Software

	Serial	Parallel		
		1-core	8-core	32h-core
Sorting 10M strings	2.90	2.90	0.40	.095 (30.5)
Remove dupl. 10M strings	0.66	1.00	0.14	.038 (17.4)
Min. span. tree 10M edges	1.60	2.50	0.42	.140 (11.4)
BFS 10M edges	0.82	1.20	0.20	.046 (17.8)

Running times in seconds

Which one is better?
Why serial != 1-core?

- **Do we have to do anything differently to take advantage of parallelism?**

Parallel Software

- Challenges of parallel software
  - The fundamental difference is that two computations can be performed in parallel only if they do not depend on each other
- Two Challenges
  - Algorithm design: max number finding
  - Practical
    - The many forms of parallelism, ranging from large to small scale, and from general to special purpose, currently requires many different programming languages, libraries, and implementation techniques

PARALLEL SOFTWARE

- This course focus on: Parallel Thinking
  - think about parallelism at a high-level
  - learning general techniques for designing parallel algorithms and data structures
  - learning how to approximately analyze their costs

WORK AND SPAN

- Work
  - The work of an algorithm corresponds to the total number of primitive operations performed by an algorithm
- perfect speedup
  - If we had W work and P processors to work on it in parallel, then even sharing would imply each processor does W/P work, and hence the total time is W/P

Is easy to get?

华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

WORK AND SPAN

What is the longest sequence of dependences?

- **Span**
  - The span of an algorithm basically corresponds to the longest sequence of dependences in the computation
  - It can be thought of the time an algorithm would take if we had an unlimited number of processors on an ideal machine

華中科技大學
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

WORK AND SPAN

- greedy scheduler
  - Consider a scheduler that runs on P processors and that has the following greedy property
    - no processor will sit idle (not work on some part of the computation) if there is work ready to do
- It can be shown that when using a greedy schedule on a computation with W work and S span, the runtime will be at most:

$$T \leq \frac{W}{P} + S .$$

Why?

WORK AND SPAN

- **parallelism** $\mathcal{P}$ $$\mathcal{P} = \frac{W}{S}.$$
  - **For $P = \mathcal{P}$ we are getting half of perfect speedup**
  - **For any $P < \mathcal{P}$ the speedup is better**
  - **For any $P > \mathcal{P}$ it is worse**
- **Therefore $\mathcal{P}$ gives us a rough upper bound on the number of processors we can effectively use**

Why?

WORK AND SPAN

- **How to calculate the work and span of an algorithm?**

	work (W)	span (S)
Sequential composition	$1 + W_1 + W_2$	$1 + S_1 + S_2$
Parallel composition	$1 + W_1 + W_2$	$1 + \max(S_1, S_2)$

華中科技大学

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Example: Parallel Merge Sort

- As an example, consider the parallel mergeSort algorithm for sorting a sequence of length n
  - The work is the same as the sequential time, which you might know is
    - $W(n)=O(n\lg n)$
  - the span for mergeSort is
    - $S(n)=O(\lg^2 n)$
  - when sorting a million keys and ignoring constant factors
  - work is $10^6 lg(10^6)>10^7$, and span is $lg^2(10^6)<500$!

Work Efficiency

- If algorithm A has less work than algorithm B, but has greater span then which algorithm is better?
  - In general the work is more important than the span
    - first reduce the work and then reduce the span by designing asymptotically work-efficient algorithms that perform no more work than the best sequential algorithm for the same problem
    - sometimes it is worth giving up a little in work to gain a large improvement in span
- a parallel algorithm is *(asymptotically) work efficient* , if the work is asymptotically the same as the time for an optimal sequential algorithm that solves the same problem

SYNOPSIS

- Parallelism
- Specification, Problem, and Implementation

Algorithm Specification

- The **problem specification** describes ***what*** is expected of the algorithm, the algorithm should do
- The **algorithm specification** describes ***how*** it achieves what is asked
  - **Problem specification (Comparison Sort)**
    - Input: Given a sequence A of n elements taken from a totally ordered set with comparison operator <,
    - Output: return a sequence B containing the same elements but such that $B[i] \le B[j]$ for $0 \le i < j < n$
  - **Quicksort, Mergesort, Insertion Sort , ..., are algorithm specifications for sorting**

Cost Specification

- **Assuming the comparison function < does constant work, the cost for parallel comparison sorting a sequence of length n is $O(n\log n)$ work and $O(\log^2 n)$ span**
  - Comparison Sort: Efficient & Parallel
- **Assuming the comparison function < does constant work, the cost for parallel comparison sorting a sequence of length n is $O(n^2)$ work and $O(\log n)$ span**
  - Comparison Sort: Inefficient but Parallel

Data Structure Specification

- **Abstract Data Type (ADT)** specification describes *what* is expected of the data structure
  - Data Type (Priority Queue)
    - A **priority queue** consists of a priority queue type and supports three operations on values of this type
      - *The operation* ***empty*** *returns an empty queue*
      - *The operation* ***insert*** *inserts a given value with a priority into the queue and returns the queue*
      - *The operation* ***removeMin*** *removes the value with the smallest priority from the queue and returns it*

Data Structure Specification & Cost Specification

- **Cost Specification (Priority Queue: Basic)**
  - As with algorithms, we usually give cost specifications to data structures
    - The work and span of a priority queue operations are as follows
    - *empty*: $O(1)$, $O(1)$
    - *insert*: $O(\log n)$, $O(\log n)$
    - *removeMin*: $O(\log n)$, $O(\log n)$

Problem

- A **problem** requires meeting an algorithm or an ADT specification and a corresponding cost specification
  - Definition (Algorithmic Problem)
    - *An algorithmic problem or an algorithms problem requires designing an algorithm that satisfies the given algorithm specification and cost specification if any*
  - Definition (Data-Structures Problem)
    - *A data-structures problem requires meeting an ADT specification by designing a data structure that can support the desired operations with the required efficiency specified by the cost specification*

华中科技大学

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

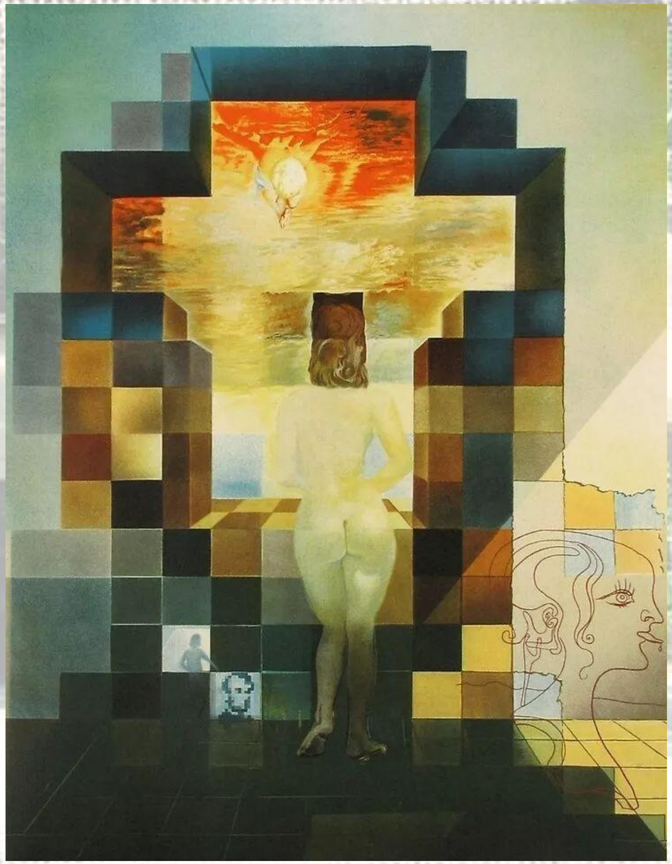
Implementation

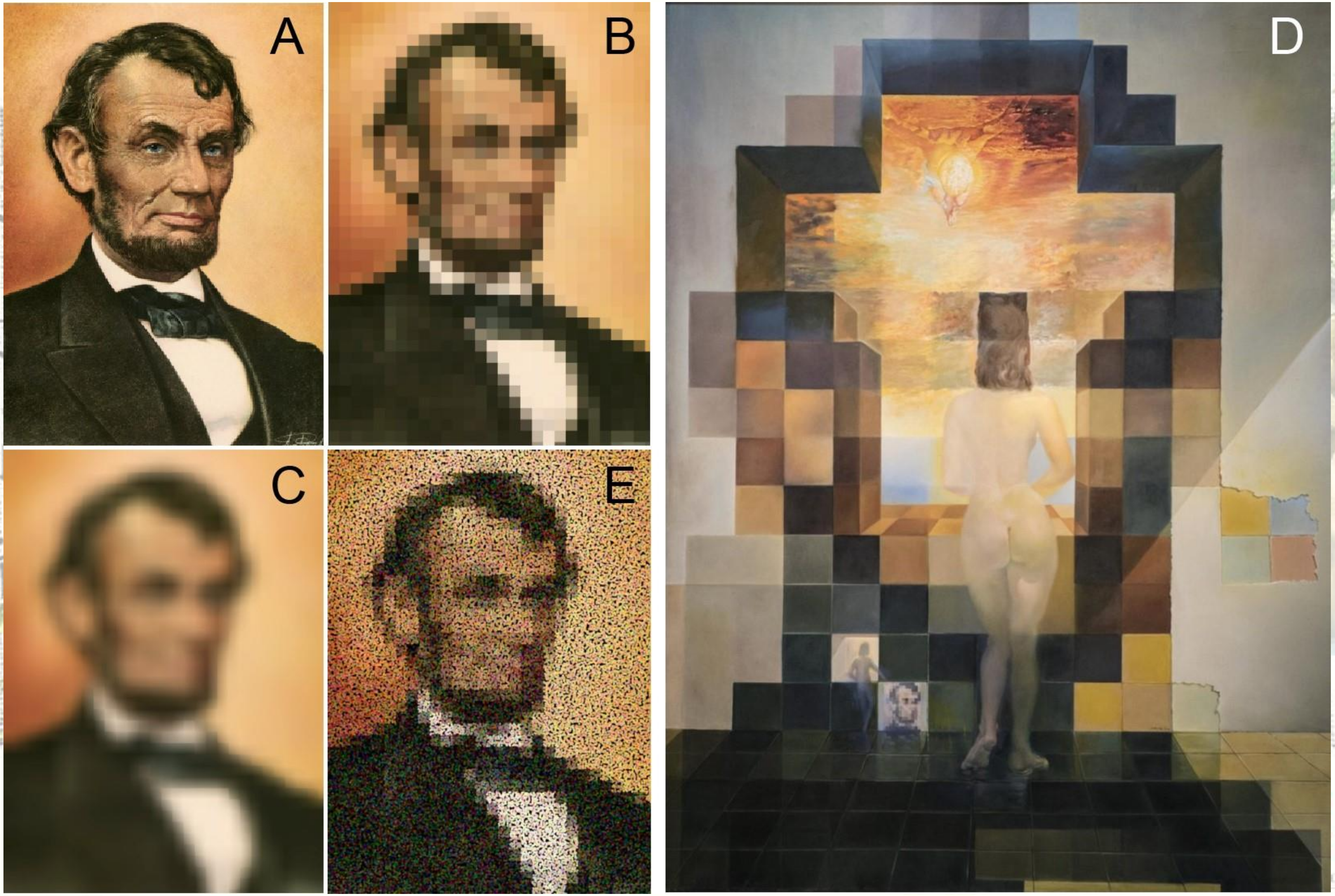
- The term **algorithm** <u>refers to an implementation that solves an algorithms problem</u> and the term **data structure** to refer to <u>an implementation that solves a data-structures problem</u>

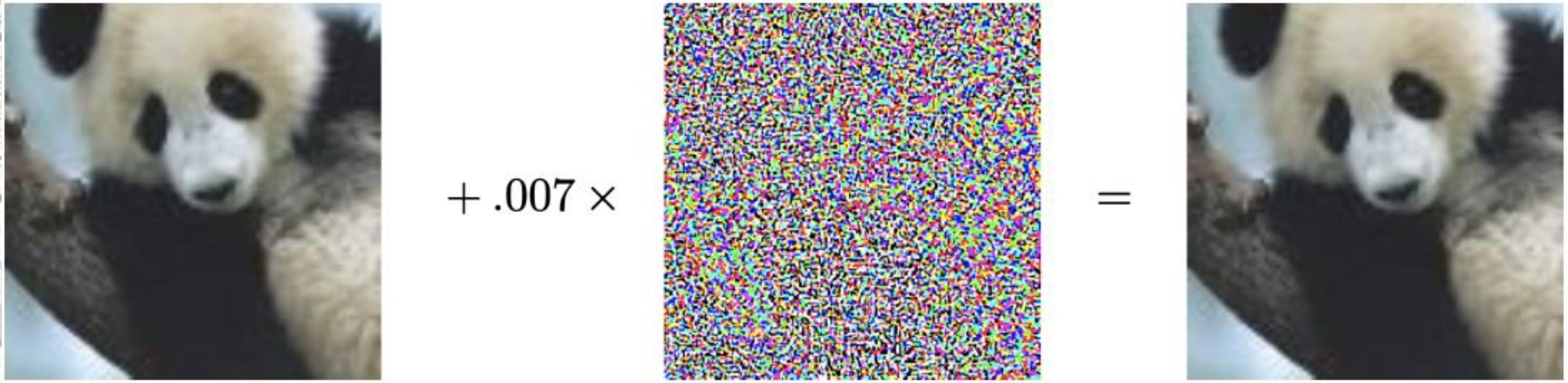
	Abstraction	Implementation
Functions	Problem	Algorithm
Data	Abstract Data Type	Data Structure

About abstraction

- Paul Cezanne
  - I can paint in different ways, in ways that don't necessarily mimic vision, and the viewer can still create a reality.
- Picasso and Braque




A
B
C
E
D



SUMMARY

- **Parallelism**
- **Specification, Problem, and Implementation**

Exercise

- **Prove： when using a greedy schedule on a computation with W work and S span, the runtime will be at most:**

$$T \le \frac{W}{P} + S .$$