Data Structures & Algorithms

Overview

- What does D.S. & A. cover?
- Computational Problems
  - Example Problem
  - Example Algorithms
  - Evaluation and analysis

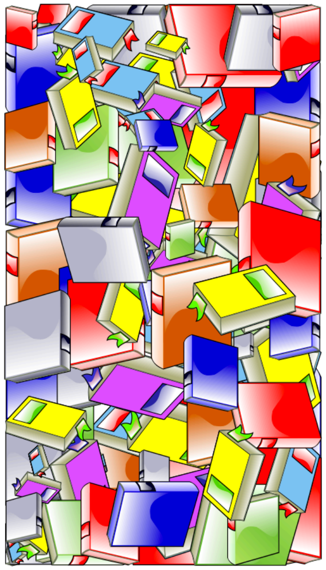
What does D.S. & A. cover?

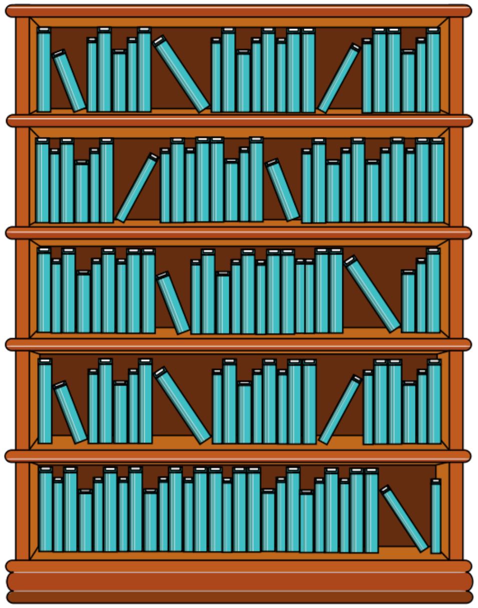
Largely focuses on

- Collections of data
- Organization & interaction with data
    - I.e., Computational Problems
    - Including comparison/contrast of different approaches

What does D.S. & A. cover?

A real-world example of importance





Computational Problems

A Defintion

Formal definition…other courses (347, 547)

Informally, for this class, a problem describes…

- A structured collection of input data
- Criteria for a correct solution

Computational Problems

An example: Sorting

Sorting

- Example: Sorting a pile of books into order by subject
- Example: Sorting integers in order

Computational Problems

An example: Sorting

Sorting?

A structured collection of input data

Input: A list of numbers

(List? Array? Linked List?)

Criteria for a correct solution

Um…Sorted?

Computational Problems

An example: Sorting

More precise

A structured collection of input data

An input *array* of data who's type is*"totally orderable"*

Total order?

Briefly: notion of

I.e., a way to clearly answer "does x come before y"

Computational Problems

An example: Sorting

More precise for this example

A structured collection of input data

An input ***array*** of ***integers***

Computational Problems

An example: Sorting an array of integers

Sorting?

Criteria for a correct solution

Should contain the same values

(Conservation of data: Data is neither created nor destroyed)

Sorted

For any two elements x & y,
if x<y x is before y in the array

Computational Problems

Other examples

- Shortest Path between two points
- Traveling salesperson (TSP)
  - Finds the shortest path that visits n sites and returns to home?
- Optimization problems
- Factoring
- Halting problem

Computational Problems

An Algorithm

- Algorithm
  - Named for 9th century mathematician
  - Definition
    - Effective procedure
    - for taking any instance of a computational problem
    - and finding a correct solution

Computational Problems

An Algorithm: Definition

Effective procedure

Can be turned into a program
(Halting problem can’t be)

for taking *any* instance of a computational problem

Not just some…ANY POSSIBLE

and finding a *correct solution*

100%, perfectly correct (not almost, or probably)

Computational Problems

An example: Sorting

Simple Problem:

Sort 89, 3, 1

Simple Algorithm: Try all permutations until we find one that’s sorted

Ugh.

More complex problem: More data

Computational Problems

An example: Sorting

Let’s find the “first value”

- Let’s select it. Search through all values until we find the minimum
- Move it to the front

Repeat with remaining data

- Find min of everything after 1st spot (2nd smallest overall) and swap to 2nd position
- Find min of everything after 2nd spot (3rd smallest overall) and swap to 3rd position

Computational Problems

An example: Sorting Code (In Place Selection Sort)

```
    /**
     *
     * @param array
     * @param start Index to start with
     * @return the index of the minimum item in
array[start..length-1]
     */
    public static int minLocation(int[] array, int start)
{
        int loc = start;
        for(int i=start;i<array.length; i++) {
            if(array[i]<array[loc]) {
                loc = i;
            }
        }
        return loc;
    }
```

```
/**
 * Swap the locations of items at indices i and j
 * @param array
 * @param i
 * @param j
 */
public static void swap(int[] array, int i, int j) {
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
```

```
public static void main(String[] args) {
    int[] input = {2,34,1,3,89,13,55};

    // Iterate through each position and swap in
    // the minimum of the remaining items.
    for(int i=0;i<input.length;i++) {
        int loc = minLocation(input, i);
        swap(input, i, loc);
    }

    System.out.println(Arrays.toString(input));

}
```

Computational Problems

An example: Sorting Code (ArrayList Based Selection Sort)

```
public static void selectionSortList(ArrayList<Integer> input, ArrayList<Integer> sorted) {
    // While there's still data in the input list

    while(input.size()>0) {
        // Get the minimum value from the input list
        int locationOfMin = 0;
        for(int i=0;i<input.size();i++) {
            if(input.get(i) < input.get(locationOfMin)) {
                locationOfMin = i;
            }
        }
        // Now we've found the min: add it to the solution
        sorted.add(input.get(locationOfMin));
        // Now remove it
        input.remove(locationOfMin);
    }
}
```

Selection Sort

An Algorithm

Now we have algorithms for sorting!

Two questions

- Is it *correct* (will solve the computational problem on all inputs)?
- Is it “*good*”?

Selection Sort

Is it correct?

Seems so…

  - Not a compelling answer in most contexts
    - Medical equipment
    - Aerospace control
    - Financial transactions

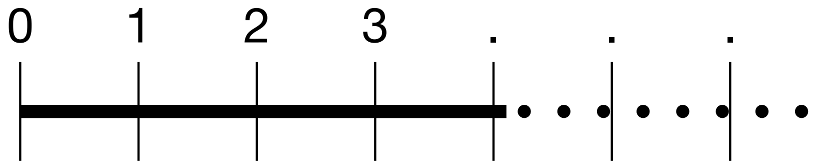
Selection Sort

Is it correct?

- “Seems so…” doesn’t cut it
  - Your intuition or informal reasoning will lead you astray
  - We’ll need to prove it (i.e., do a proof)
    - This is training…practice to prepare for more challenging problems
    - And review
  - And I’m going to *cheat*…
    - I already know a technique that works/fits…Proof by induction

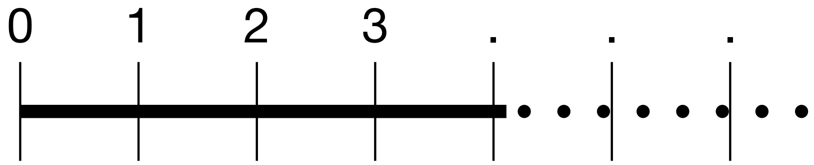
Proof by Induction

Review: The number line



Proof by Induction

Review: The number line

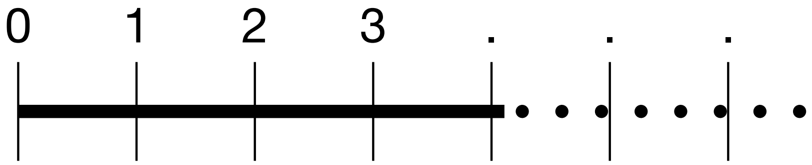


Associate integers with problem instances,
so we can talk about all the instances with numbers

Goal: Prove that some statement is true for all instances

Proof by Induction

Review: The number line



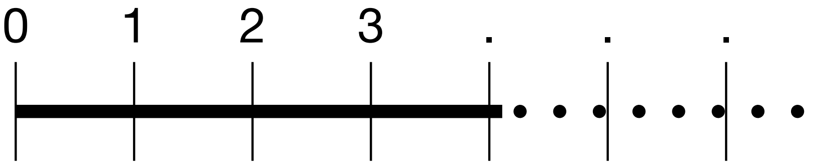
Associate integers with problem instances,
so we can talk about all the instances with numbers

Goal: Prove that some statement is true for all instances

Selection Sort Goal: Prove that list-based selection sort will “sort” any list

Proof by Induction

Review: The number line



Pick some “base case” or “basis”

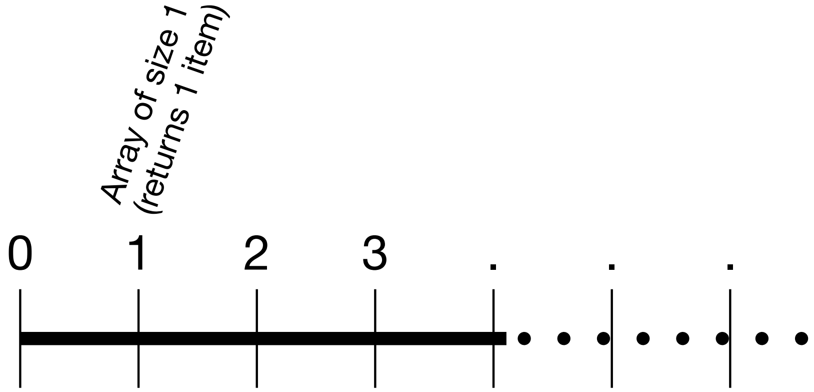
Pick something that’s easy to prove is correct

Selection Sort: Let the integer represent the list size

Basis is a size of 1, which moves the 1 item to the sorted list

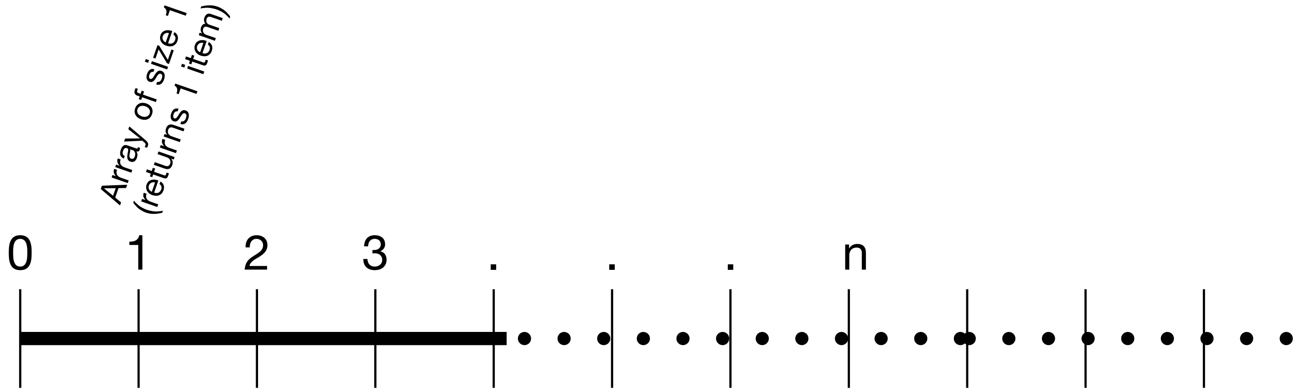
Proof by Induction

Review: The number line



Proof by Induction

Review



Next we pick an arbitrary other instance, n

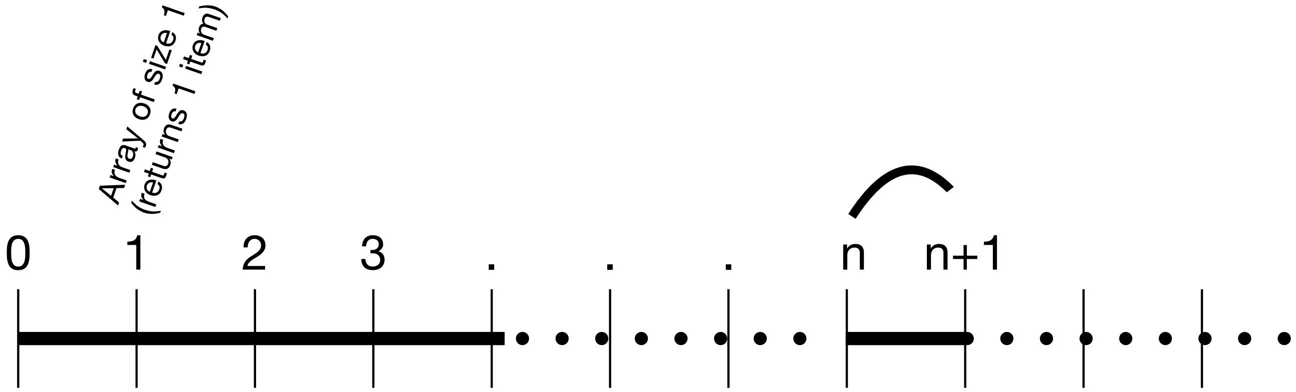
Form a hypothesis (the “Induction Hypothesis”):

They hypothesis is always: Our statement is also true at n

Selection Sort: assume sorting n things will work!

Proof by Induction

Review



Now we use logic/math to show that if whatever we're trying to prove is true for n, it'll be true at n+1 too. This is called the inductive step.

That is, showing that "being true on n" induces "being true on n+1".

Proof by Induction

List-Based Selection Sort

Basis: List based selection sort correctly sorts one item

Induction Hypothesis: A list of n items will be correctly sorted

Consider a list of n+1 elements

Goal: Show it’ll be sorted

Algorithm removes smallest element and we have two lists

smallest
element

n remaining items

Proof by Induction

List-Based Selection Sort

Due to the induction hypothesis, we assume that the “n remaining items” will be appended in-order

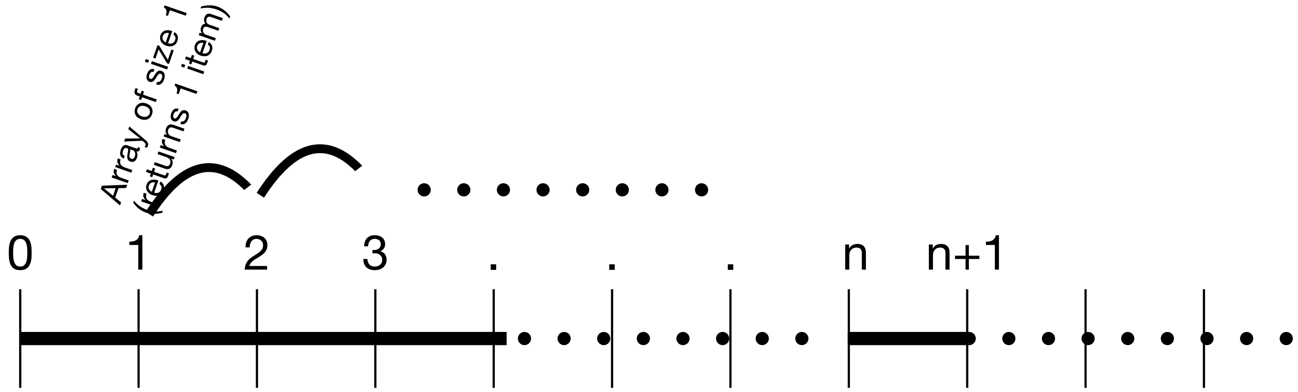
smallest element

n remaining items in order (smallest was removed)

Hence the entire list is sorted (QED)

Proof by Induction

Review



Aside: Formal Proofs

First decide *what* you want to argue

You want your argument to be

- not wrong
- convincing (to you and others)

Formal proof techniques help you achieve these

- Proof by induction is a known, accepted formal technique that works on certain types of problems

Selection Sort

Is it “good”?

What does “good” mean?

“Good” is relative!

In this class it’s often a comparison between alternatives.

The “try all sort” vs. the “selection sort”
(and the selection sort example with ArrayLists vs. LinkedLists)

In terms of what?

Time taken and/or memory used

Selection Sort

Is it “good”?

Time: What’s a fundamental operation

Often these are comparable to assembly language instructions
(Covered in CSE132 and CSE260M)

Things like comparisons (x<y), assignment to variables (x=y),
array access (x=a[y]), array assignment (a[x]=y), math (x=x+y), etc.

In some simple processors nearly all these do take the same time

Generally all the same order of magnitude on modern machines

Selection Sort

Is it “good”?

For now we’ll focus on

Operations

In terms of input size
(Selection Sort: the number of items to sort)

Selection Sort

Is it “good”?

- Concrete problem
  - Sorting 5 items
  - Fundamental operation?
    - “Looking at” each number

Selection Sort

Is it “good”?

Sorting 5 items

Find the min of 5 items (look at 5 items => 5 fundamental operations)
(Then other stuff: swap, remove, etc.)

Find the min of 4 items (4 fundamental operations)

Find the min of 3 items (3 fundamental operations)

Find the min of 2 items (2 fundamental operations)

Find the min of 1 item (1 fundamental operation)

Total “Fundamental operations” = 5+4+3+2+1 = 15

Selection Sort

Is it “good”?

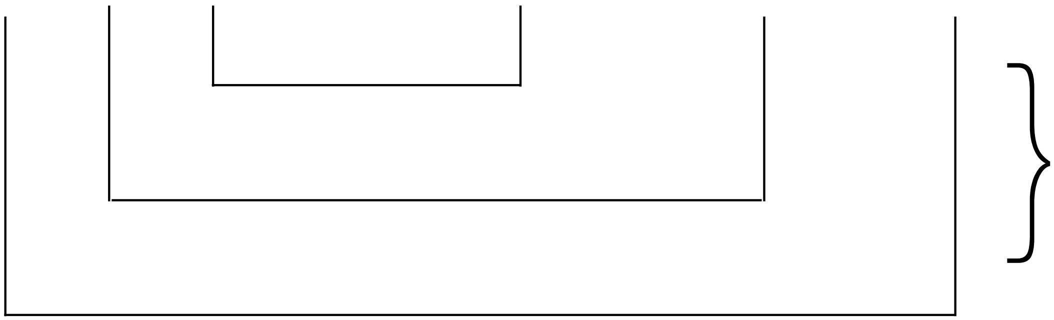
Sorting n items

Total “Fundamental operations”

Selection Sort

Is it "good"?

The sum: the "Gauss story"



Selection Sort

Is it “good”?

Summary for selection sort: fundamental operations (“look at”)

5	15	25
100	5,050	10,000
10,000	50,005,000	100,000,000
100,000	5,000,050,000	10,000,000,000

Selection Sort

Is it “good”?

Comparisons

We often focus on estimating the “worst case number of operations” based on the problem size

Special notation (more later):

There are other sorts. Some are

Selection Sort

Is it “good”?

Comparisons

We often focus on estimating the “worst case number of operations” based on the problem size

Special notation (more later):

There are other sorts. Some are

Selection Sort

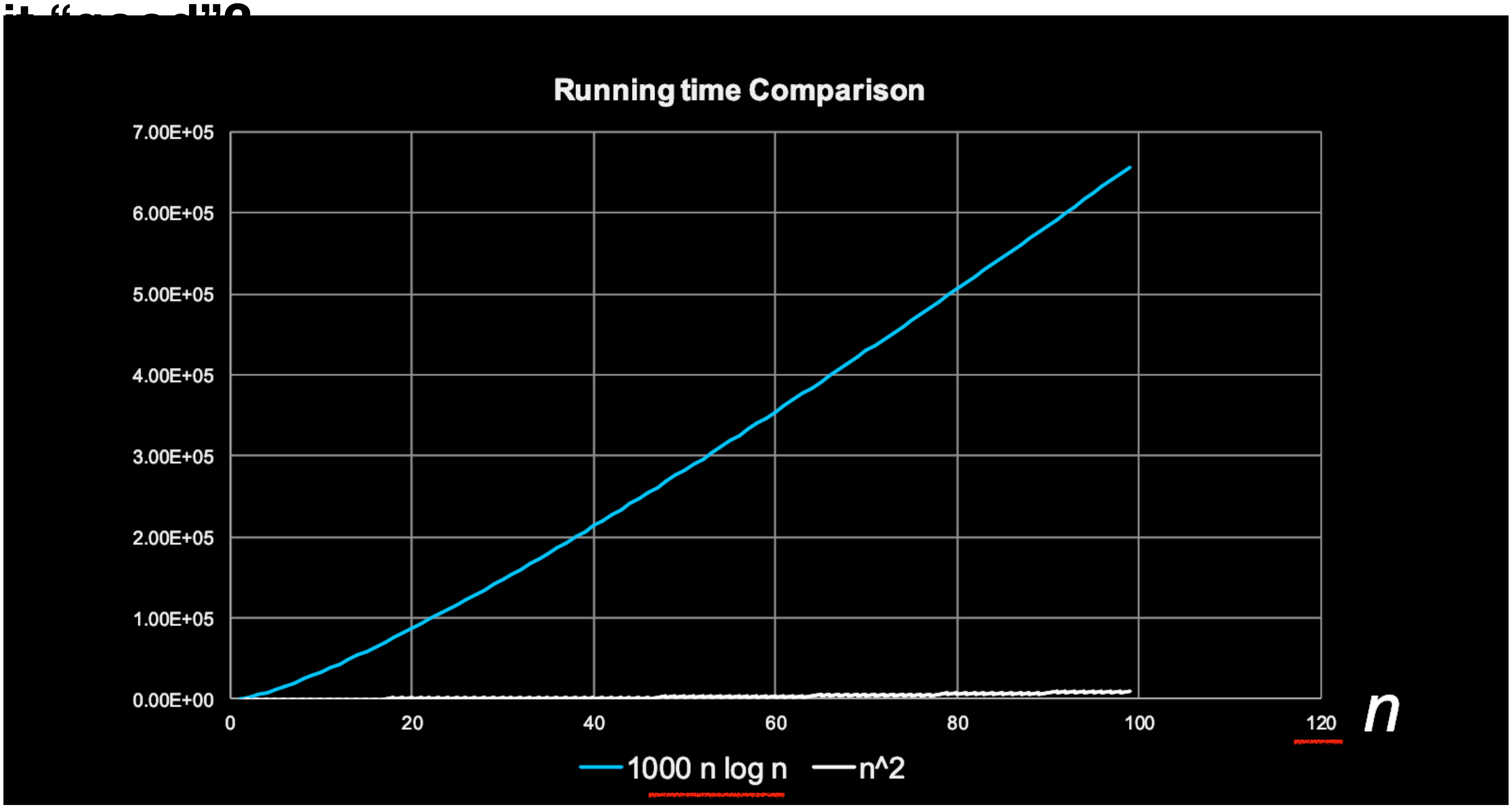
Is it “good”?

Comparisons

- We often focus on estimating the “worst case number of operations” based on the problem size
  - Special notation (more later):
  - There are other sorts. Some are
- Comparisons on small problems depend on details
  - Things that we dropped.

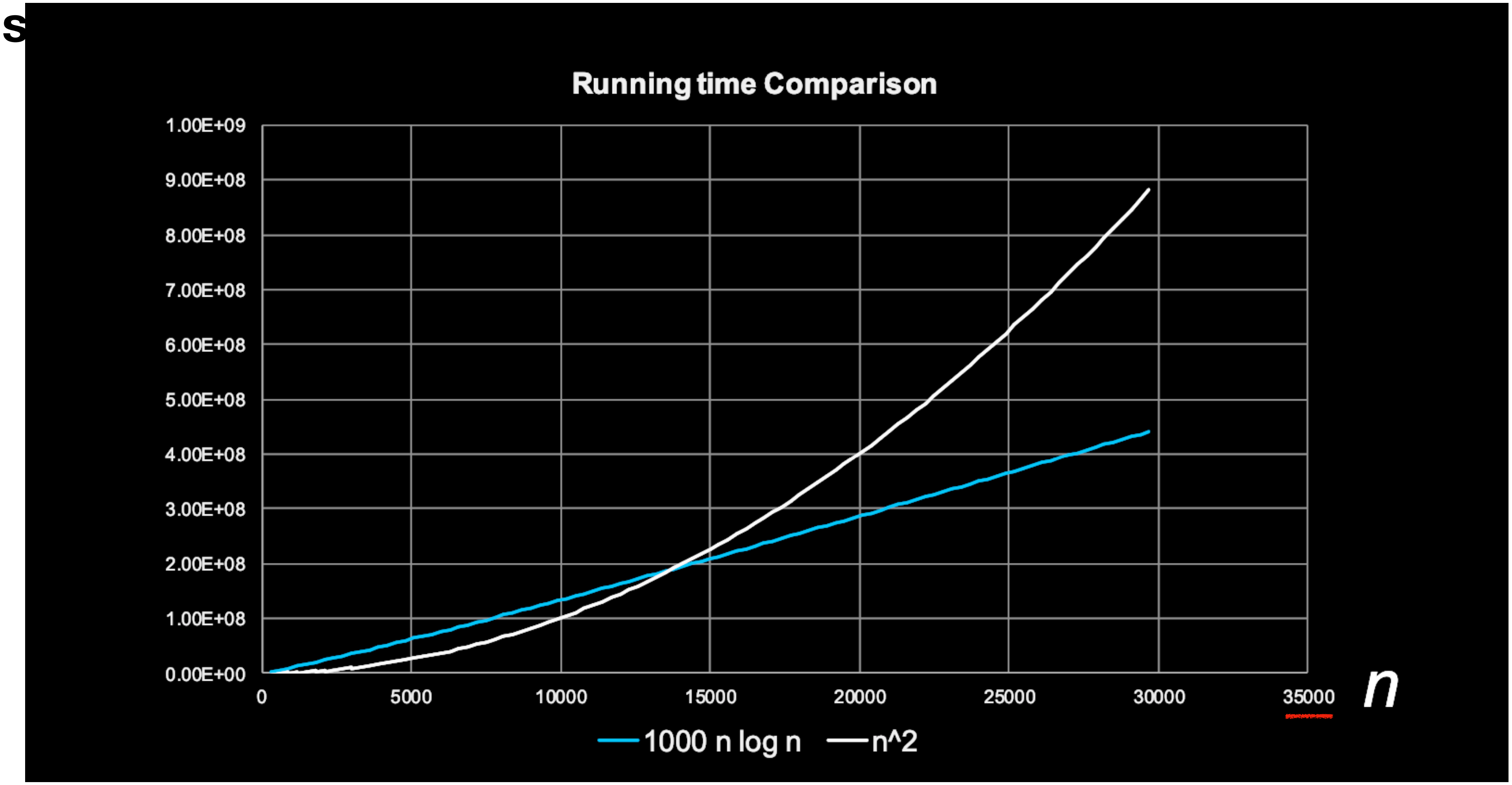
Examples

Is it “good”?



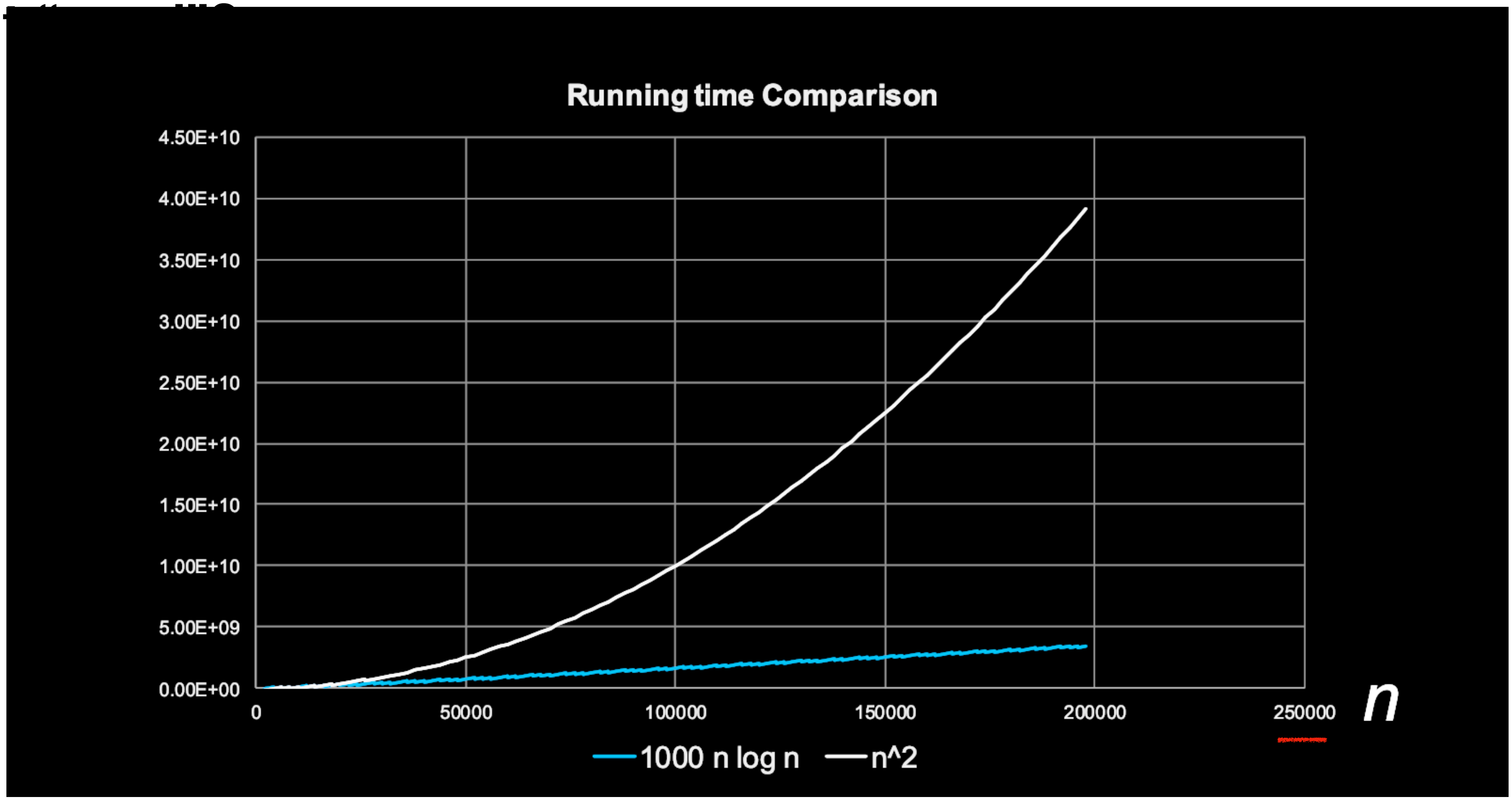
Examples

Is


Running time Comparison
1.00E+09
9.00E+08
8.00E+08
7.00E+08
6.00E+08
5.00E+08
4.00E+08
3.00E+08
2.00E+08
1.00E+08
0.00E+00
0
5000
10000
15000
20000
25000
30000
35000
n
1000 n log n
n^2

Examples

Is it


Running time Comparison
4.50E+10
4.00E+10
3.50E+10
3.00E+10
2.50E+10
2.00E+10
1.50E+10
1.00E+10
5.00E+09
0.00E+00
n
1000 n log n
n^2