



Computational Thinking - 2

Honor 4471
Spring 2021

A photograph of a logistics yard at dusk or dawn. In the foreground, a yellow container is being lifted by a crane onto a blue truck. To the right, there are large stacks of colorful shipping containers in shades of yellow, red, and blue. The sky is filled with soft, colorful clouds. The word "logistics" is written in white text across the center of the image.

logistics

Activity

Chat

Teams

Assignments

Calendar

Calls

Files

Walkie Talkie

Teams

Filter

Your teams

Paralab

Honor 4471

General

Concepts

How do I

Project ideas

Python

Search

HS

General

Posts

Files

Team

Meet

Let's get the conversation started

Try @mentioning a student or teacher to begin sharing ideas.

21

HS

Hari Sundar

Tuesday 9:40 AM

Like this message once you have logged on to teams

Reply

HS

Hari Sundar

Tuesday 10:11 AM Edited

Link for Lectures

<https://utah.zoom.us/j/98541169450>

Meeting ID: 985 4116 9450

See more

Join our Cloud HD Video Meeting

Zoom is the leader in modern enterprise video communications, with an easy, reliable cloud platform for video and audio conferencing, chat, and webinars across mobile, desktop, and room systems. Zo...

Poll (on Teams)



What is your major ?

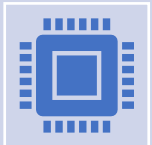
Social work, Linguistics, Psychology, Film, Marketing, Economics, International Studies, Special Education, English, Spanish, Environmental & Sustainability, Political Science, Business, Statistics, Biology, Finance, Criminology, Sociology, ...



What is your
academic
standing ?

Freshman,
sophomore,
...

Freshman – **40%**, Sophomore – **50%**, Junior – 10%



Have you programmed
before ?

45% - Yes, **55%** - No

Module 1: NYPD Stop & Frisk Data

<https://www.nyclu.org/en/stop-and-frisk-data>

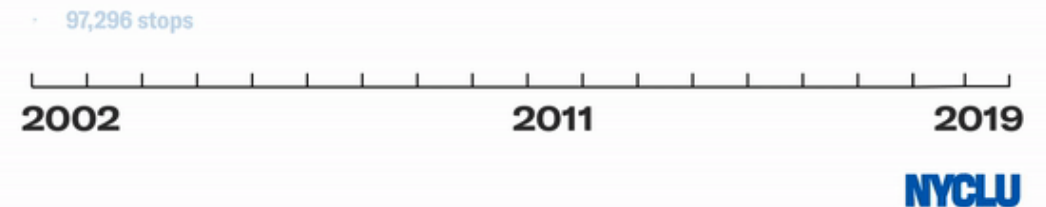
About the Data:

Every time a police officer stops a person in NYC, the officer *is supposed to* fill out a form recording the details of the stop. The forms were filled out by hand and manually entered into an NYPD database until 2017, when the forms became electronic. The NYPD reports stop-and-frisk data in two ways: a summary report released quarterly and a complete database released annually to the public.

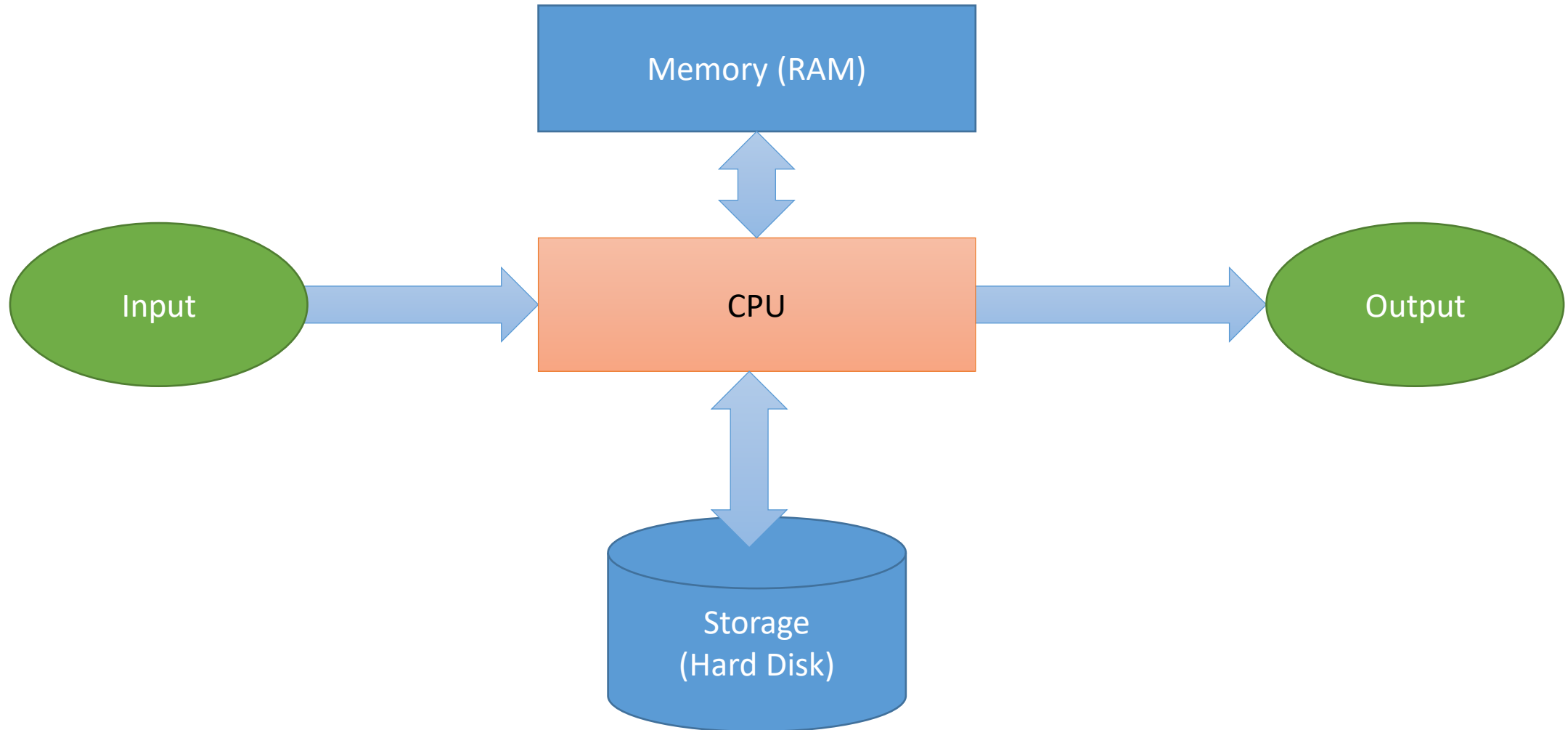
The quarterly reports are released by the NYCLU every three months (available [here](#)) include data on stops, arrests, and summonses. The data are broken down by precinct of the stop and race and gender of the person stopped.

The annual database includes nearly all of the data recorded by the police officer after a stop such as the age of the person stopped, if a person was frisked, if there was a weapon or firearm recovered, if physical force was used, and the exact location of the stop within the precinct. The NYPD uploads this database to their [website](#) annually. The most recent annual dataset and codebook is located below. It contains over 100 variables and 12,404 observations, each of which represents a stop conducted by an NYPD officer.

Number of Reported Stops by Year



Computer Hardware



Sort some
cards





Can you write precise
instructions to sort cards ?



Algorithms

Sort some
cards





The model

What is important ?

- Paper or plastic ?
- The design ?
- Images ?
- Suit ?
- 2, 3, 4, 5, 6, 7, 8, 9, 10
- A, J, Q, K
- 1, 11, 12, 13

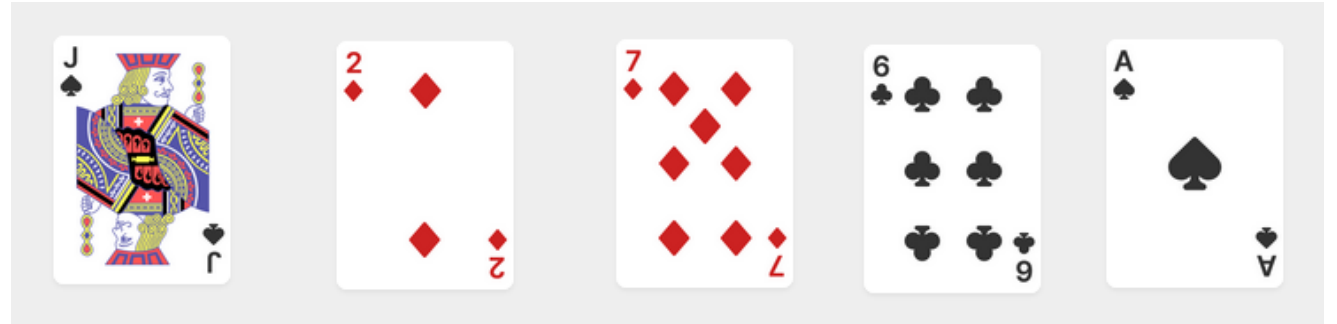


Representation

- We can represent each card by a number from 1-13
- A set of cards will be stored on the hard-disk or in memory (RAM) as a **list** of numbers

| | | | | |
|----|---|---|---|---|
| 11 | 2 | 7 | 6 | 1 |
|----|---|---|---|---|

| | | | | |
|---|---|---|---|----|
| 1 | 2 | 6 | 7 | 11 |
|---|---|---|---|----|

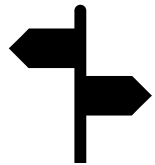


| | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|
| 10 | 2 | 1 | 7 | 6 | - | - | - | - | - |
|----|---|---|---|---|---|---|---|---|---|



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|----|

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|



Privet drive

4 Privet Dr

`privet[4]`

Write instructions to sort
cards



Algorithm - Selection Sort

- Find the smallest number in the list (1st or 0th number)
- Place it in the 0th position
 - Where to put the number in the 0th position ?
 - Swap
- Repeat by **selecting** the 1st, 2nd ... numbers and placing in the appropriate location
- Effectively divides the list into 2 parts, a sorted and an unsorted part
- We operate on the unsorted part, and always find the smallest number in the unsorted part

| | |
|--|---|
| | 8 |
| | 5 |
| | 2 |
| | 6 |
| | 9 |
| | 3 |
| | 1 |
| | 4 |
| | 0 |
| | 7 |

Algorithm – Find smallest number in a list

| | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|
| 10 | 2 | 1 | 7 | 6 | - | - | - | - | - |
|----|---|---|---|---|---|---|---|---|---|

- Need a variable to store the smallest number (thus far)
 - `min`
- Initialize `min` to the first number
 - `min = cards[0]` or `min = cards[i]`
- compare min with other cards, updating min if needed
 - `if cards[i] < min, min = cards[i]`

Algorithm – Find smallest number in a list

| | | | | | | | | | |
|----|---|---|---|---|---|---|---|---|---|
| 10 | 2 | 1 | 7 | 6 | - | - | - | - | - |
|----|---|---|---|---|---|---|---|---|---|

- Need a variable to store the index of the smallest number (thus far)
 - `min_idx`
- Initialize `min_idx` to the first index
 - `min = 0` or `min = i`
- compare min with other cards, updating min if needed
 - `if cards[i] < cards[min_idx], min_idx = i`

Selection sort - pseudocode

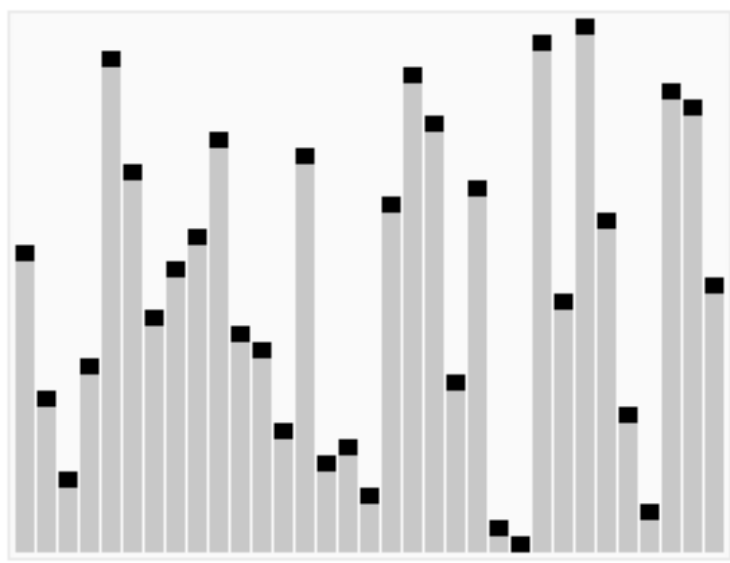
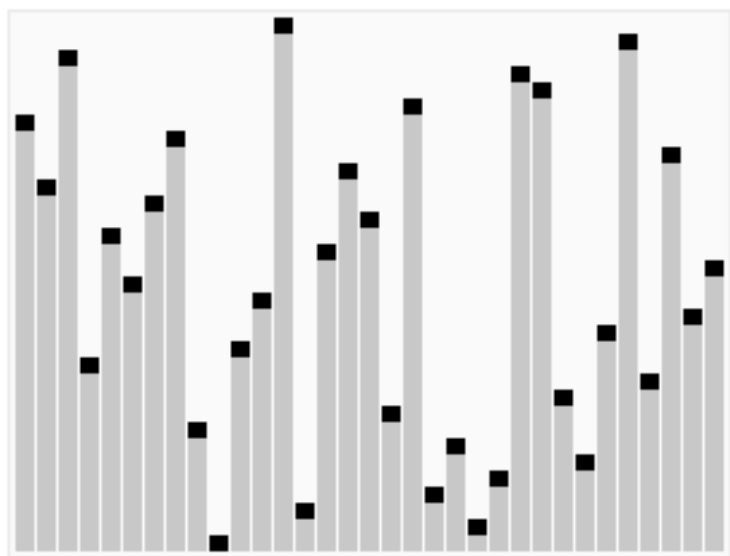
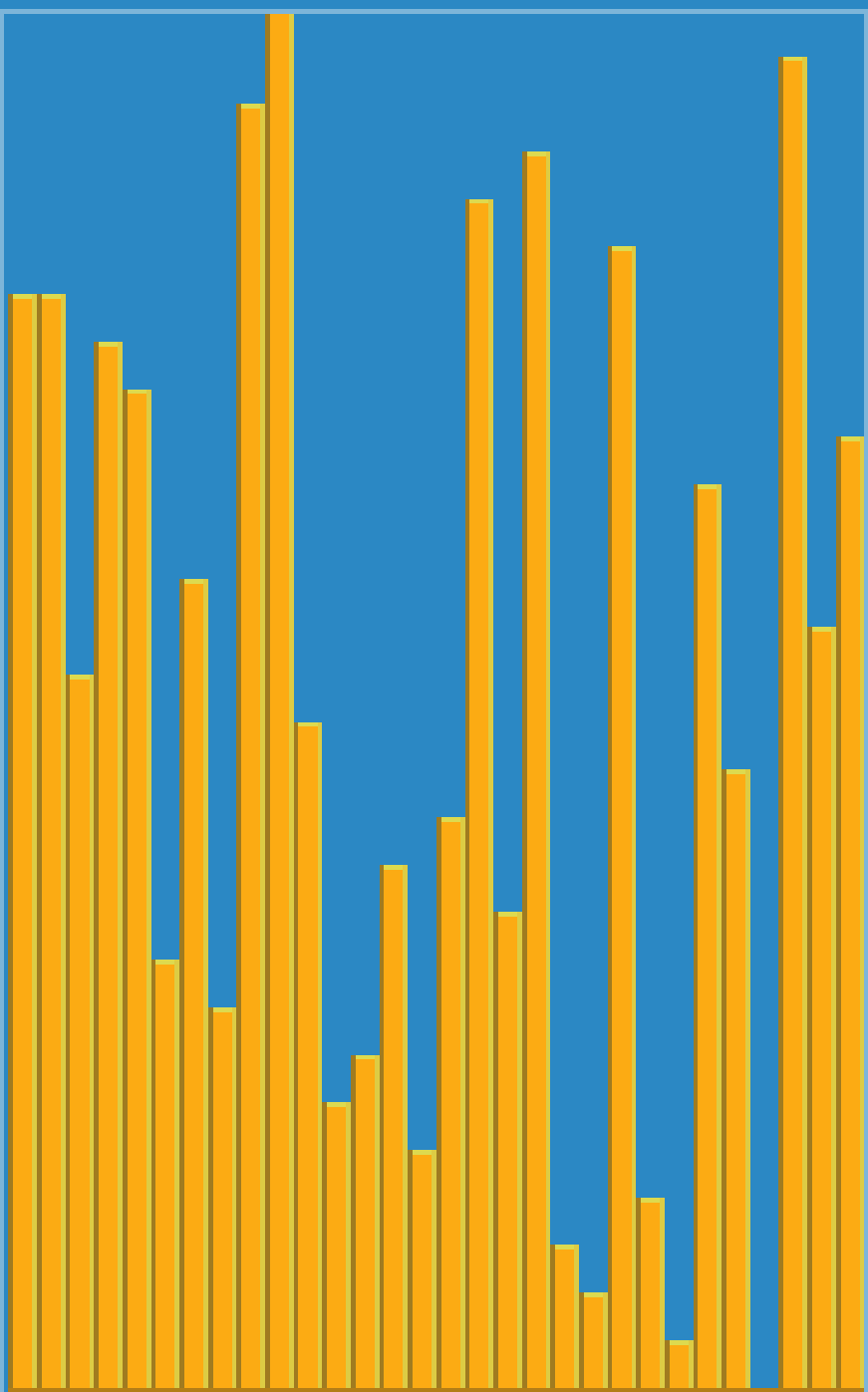
- Data n , `card[0]`, ... `card[n-1]` (input)
- Index i which indicates the current selection
 - Goes from 0 to $n-1$ (or 1 to n)
 - Find location $j \geq i$ of the smallest number in `card`
 - Swap `card[j]` and `card[i]`

Selection Sort - Python

```
def selectionSort(card, n):  
    for step in range(n):  
        min_idx = step  
  
        for i in range(step + 1, n):  
            # select the minimum element  
            if card[i] < card[min_idx]:  
                min_idx = i  
  
            # put min at the correct position - swap  
            (card[step], card[min_idx]) = (card[min_idx], card[step])  
  
card = [11, 2, 7, 6, 1]  
n = len(card)  
selectionSort(card, n)  
print('Sorted Array in Ascending Order:')  
print(card)
```

A close-up, shallow depth-of-field photograph of a mechanical assembly. The image shows several metal rods or wires extending from a dark, possibly black, base plate. These rods are secured with various metal fasteners, including hex nuts and bolts. Some of the fasteners have a distinctive corrugated or knurled texture. The background is a warm, out-of-focus brown, suggesting a wooden surface. The overall lighting is soft and directional, highlighting the metallic textures and the precision of the assembly.

demo



6 5 3 1 8 7 2 4

INEFFECTIVE SORTS

```
DEFINE HALFHEARTEDMERGESORT(LIST):  
  IF LENGTH(LIST) < 2:  
    RETURN LIST  
  PIVOT = INT(LENGTH(LIST) / 2)  
  A = HALFHEARTEDMERGESORT(LIST[:PIVOT])  
  B = HALFHEARTEDMERGESORT(LIST[PIVOT:])  
  // UMMMMM  
  RETURN[A, B] // HERE. SORRY.
```

```
DEFINE FASTBOGOSORT(LIST):  
  // AN OPTIMIZED BOGOSORT  
  // RUNS IN O(N LOG N)  
  FOR N FROM 1 TO LOG(LENGTH(LIST)):  
    SHUFFLE(LIST):  
    IF ISSORTED(LIST):  
      RETURN LIST  
  RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"
```

```
DEFINE JOBIINTERVIEWQUICKSORT(LIST):  
  OK SO YOU CHOOSE A PIVOT  
  THEN DIVIDE THE LIST IN HALF  
  FOR EACH HALF:  
    CHECK TO SEE IF IT'S SORTED  
    NO, WAIT, IT DOESN'T MATTER  
    COMPARE EACH ELEMENT TO THE PIVOT  
    THE BIGGER ONES GO IN A NEW LIST  
    THE EQUAL ONES GO INTO, UH  
    THE SECOND LIST FROM BEFORE  
  HANG ON, LET ME NAME THE LISTS  
  THIS IS LIST A  
  THE NEW ONE IS LIST B  
  PUT THE BIG ONES INTO LIST B  
  NOW TAKE THE SECOND LIST  
  CALL IT LIST, UH, A2  
  WHICH ONE WAS THE PIVOT IN?  
  SCRATCH ALL THAT  
  IT JUST RECURSIVELY CALLS ITSELF  
  UNTIL BOTH LISTS ARE EMPTY  
  RIGHT?  
  NOT EMPTY, BUT YOU KNOW WHAT I MEAN  
  AM I ALLOWED TO USE THE STANDARD LIBRARIES?
```

```
DEFINE PANICSORT(LIST):  
  IF ISSORTED(LIST):  
    RETURN LIST  
  FOR N FROM 1 TO 10000:  
    PIVOT = RANDOM(0, LENGTH(LIST))  
    LIST = LIST[PIVOT:] + LIST[:PIVOT]  
    IF ISSORTED(LIST):  
      RETURN LIST  
  IF ISSORTED(LIST):  
    RETURN LIST  
  IF ISSORTED(LIST): // THIS CAN'T BE HAPPENING!  
    RETURN LIST  
  IF ISSORTED(LIST): // COME ON COME ON  
    RETURN LIST  
  // OH JEEZ  
  // I'M GONNA BE IN SO MUCH TROUBLE  
  LIST = [ ]  
  SYSTEM("SHUTDOWN -H +5")  
  SYSTEM("RM -RF ./")  
  SYSTEM("RM -RF ~/*")  
  SYSTEM("RM -RF /")  
  SYSTEM("RD /S /Q C:\*") // PORTABILITY  
  RETURN [1, 2, 3, 4, 5]
```

Can you design
your own Sorting
algorithm ?