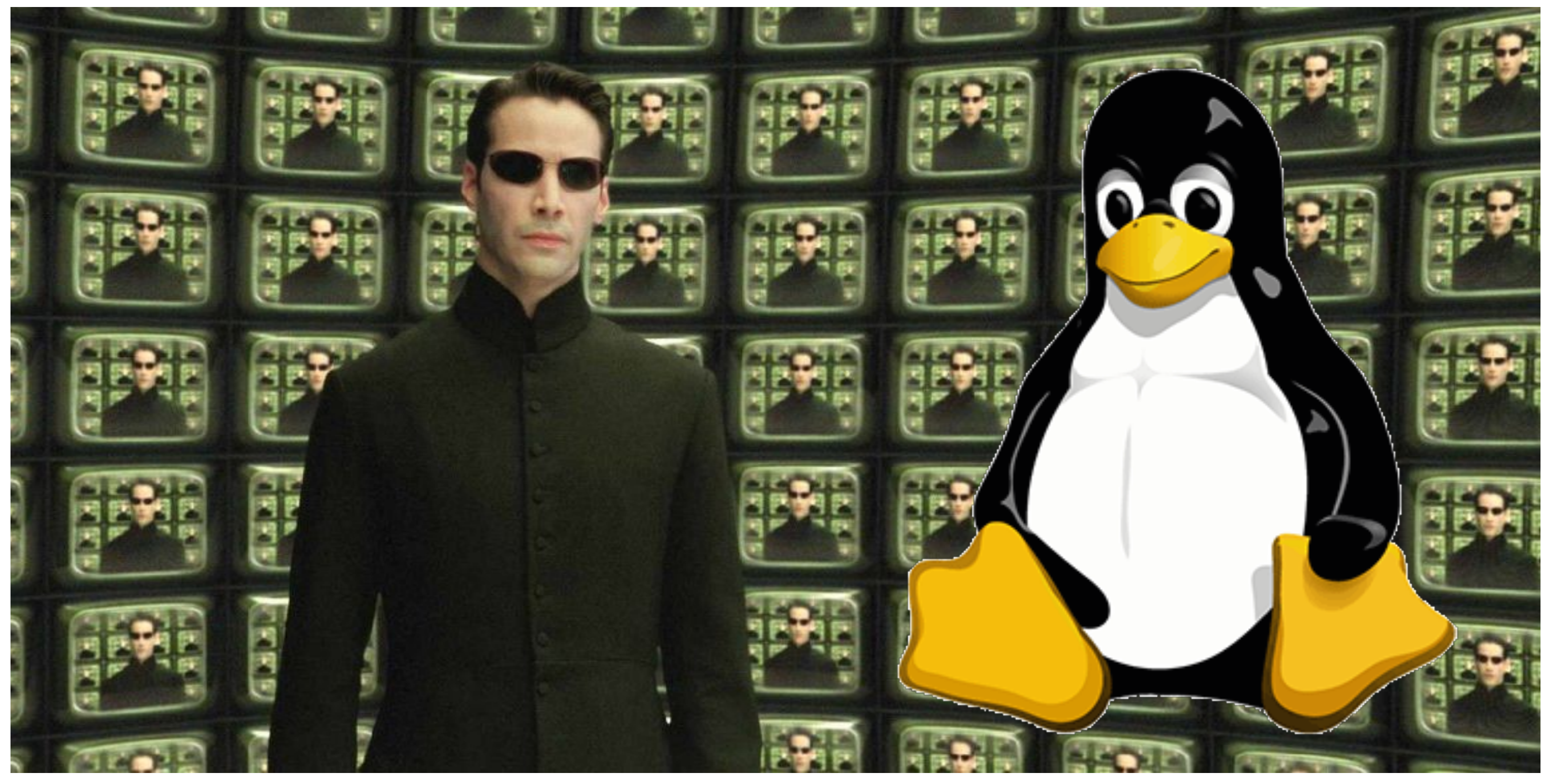
CSCI 125
Introduction to Computer Science and Programming II
Lecture 2: Array



Jetic Gū
2020 Summer Semester (S2)

Overview

- Focus: Basic C/C++ Syntax
- Architecture: Linux/Unix OS
- Core Ideas:
    1. What are arrays?
    2. C/C++ array declaration and manipulation
    3. A simple algorithm: selection sort

P0
Review

Data Types

- Primary √
  - Integers, Characters, Boolean √
  - Floating point √
  - Void
- Derived
  - Function, Array, Pointer, Reference
- User Defined
  - Struct, Class, Enumerate, Typedef

Review

Array

C/C++ Array

- Array: a collection of a fixed number of items with the **same data type** stored in **contiguous** memory locations

```
int a=0;

int b=1;

int c=2;

int d[5];
```

Address	Variable
0x0000: a	0
0x0001: b	1
0x0002: c	2
0x0003: d[0]	0
0x0004: d[1]	0
0x0005: d[2]	0
0x0006: d[3]	0
0x0007: d[4]	0

Important Distinctions

- Arrays are not like Python `lists`!
  - Fixed length: the number of elements in an array is fixed the moment it's declared
  - Fixed type: the data type of all elements will be the SAME, and it cannot be changed after declaration
  - No inserts/prepend/append operations: access through indices ONLY

Array in memory

...	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	...
...	0x0008	0x0009	0x000A	0x000B	0x000C	0x000D	0x000E	0x000F	...
...	24	36	7	9	22	21	18	6	...

- C/C++ arrays start from index `0`: e.g. no `a[-1]`, `a[0]` is the first
- Access through index: `a[3]`
  - Look up the address of `a` (which is also `&a[0]`): `0x0008`
  - Access the address: `0x0008+3=0x000B`, retrieve value: 9

So, Why Array?

- Easier to manage that variables
    - index-based access and retrieval
    - When you have thousands of values of the same type, you might as well use array to store them
- Access speed almost as fast as accessing a regular variable

Declaration and Manipulation

Declaration of 1-dimensional Array

- Array declaration by specifying size

```
DataType arrayName[N];  // N must be const

int a[10];  // creates a[0], a[1], ..., a[9]
```

- Array declaration by initialising elements

```
int a[] = {10, 20, 30, 40};
```

- Array declaration by specifying size and initialising elements

```
int a[6] = {10, 20, 30, 40};
//create a[0]=10, a[1]=20, a[3]=30, a[4]=40, a[5], a[6]
```

Declaration of Multi-dimensional Array

- Array declaration by specifying size

```
DataType arrayName[N_1][N_2]...;  // N_i must be const

int a[10][10];  // creates a 10x10 matrix

int a[10][11][12];  // creates a 10x11x12 tensor, last element a[9]
[10][11]
```

- Array declaration by specifying size and initialising elements

```
int a[3][3] = {

    {10, 20, 30},

    {40, 50, 60},

    {70, 80, 90}

};
```

Illegal Declaration of Multi-dimensional Array

- Array declaration by initialising elements: this works

```
int a[][3] = {
    {10, 20, 30},    {40, 50, 60},    {70, 80, 90} };
```

- This **DOES NOT** work

```
int a[3][] = {
    {10, 20, 30},    {40, 50, 60},    {70, 80, 90} };
int a[][] = {
    {10, 20, 30},    {40, 50, 60},    {70, 80, 90} };
int a[] = {
    {10, 20, 30},    {40, 50, 60},    {70, 80, 90} };
```

Accessing Array Components

- General syntax:

```
arrayName[indexExp]
arrayName[indexExp1][indexExp2]...
```

- `indexExp` is called **index**, whose value is a nonnegative integer
- Index value specifies the position of the component in the array
- [] is the **array subscripting operator**
- The array index always starts at `0`

Basic Manipulations

- Initialising√
- Inputting data
- Outputting data stored in an array
- Finding the largest and/or smallest element

P2
Basic oper.

Input Data Into 1-Dimensional Array

- Consider the declaration

```
int list[100];  // array of size 100
int i;
```

- Using `for` loops to access array elements:

```
for (i = 0; i < 100; i++)    {
    // DO STUFF, process list[i]
}
```

- Example:

```
for (i = 0; i < 100; i++) {
    cin >> list[i];
}
```

Concept

Some Examples for 1-Dimensional Array

- Initialise

```
double a[100];
for (int i=0; i<100; i++) a[i] = 2.0;
```

- Print

```
for (int i=0; i<100; i++) cout << a[i] << endl;
```

- Find max

```
int maxIndex = 0;
for (int i=1; i<100; i++) {
    if (a[i] > a[maxIndex]) maxIndex = i;
}
cout << "Max is: " << a[maxIndex] << endl;
```

P2
Basic oper.

Array Index Out of Bounds

- If we have the statements:

```
double num[10];
int i;
```

- The component `num[i]` is valid if `i = 0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, or `9`
- In bounds if the `index >=0` and the `index <= ARRAY_SIZE-1`
  - Otherwise, we say the `index` is out of bounds
- In C++, there is no guard against indices that are out of bounds

Concept

Simple Algorithm:
Selection Sort

Sorting

- Given an array of numbers

```
int num[] = {12, 312, 645, 23, 74, 21};
```

- Sort in ascending order

```
{12, 21, 23, 74, 312, 645}
```

- Sort in descending order

```
{645, 312, 74, 23, 21, 12}
```

How can we do this?

Concept

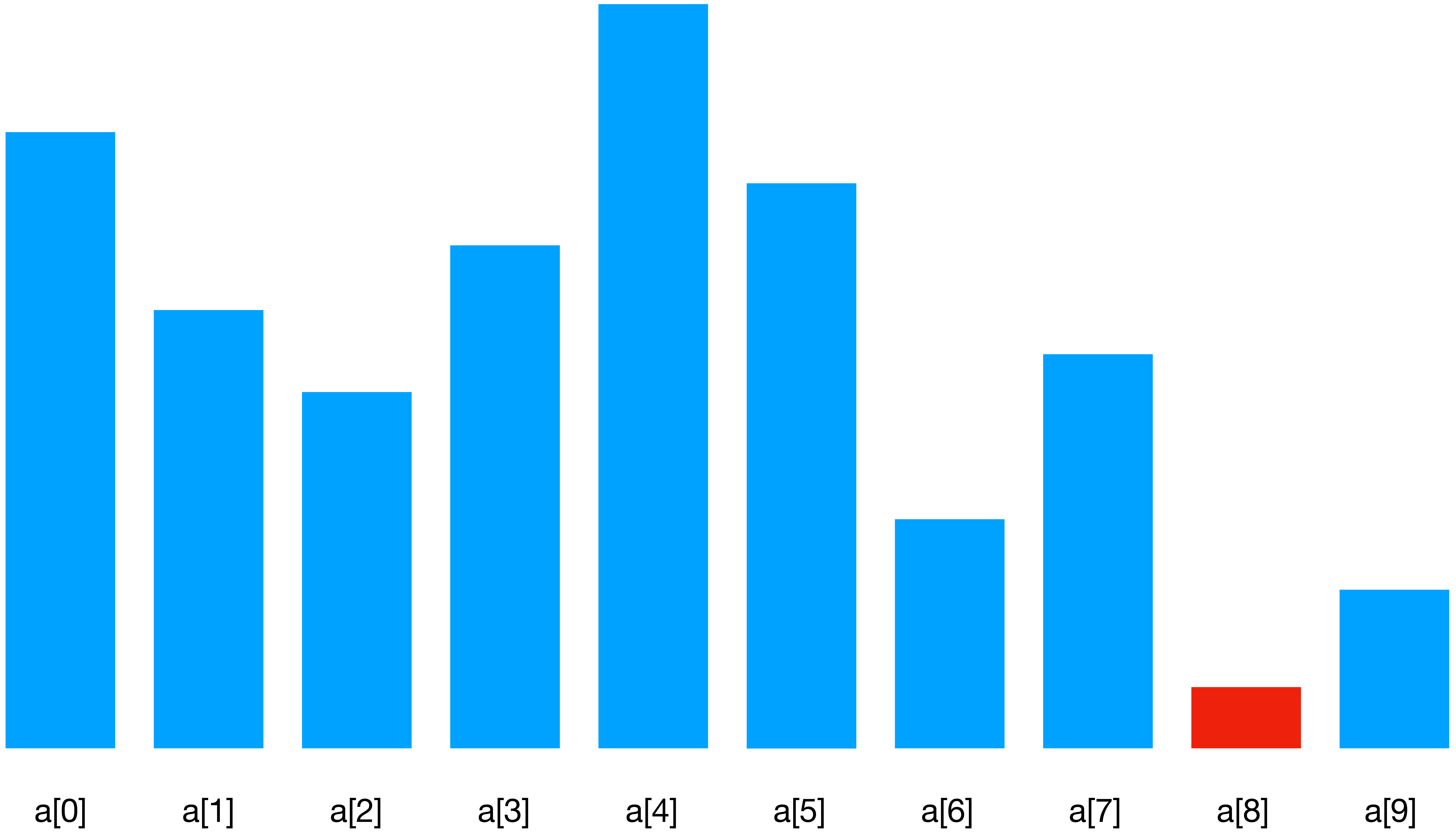
Selection Sort

- Given array `a[n]`, where `n` is an integer, let's take ascending sort

1. Find the smallest number in the array `a[0:n]` -> `a[i]`, swap it with `a[0]`
2. Continue for `a[1:n]`...

P3
Bubble Sort

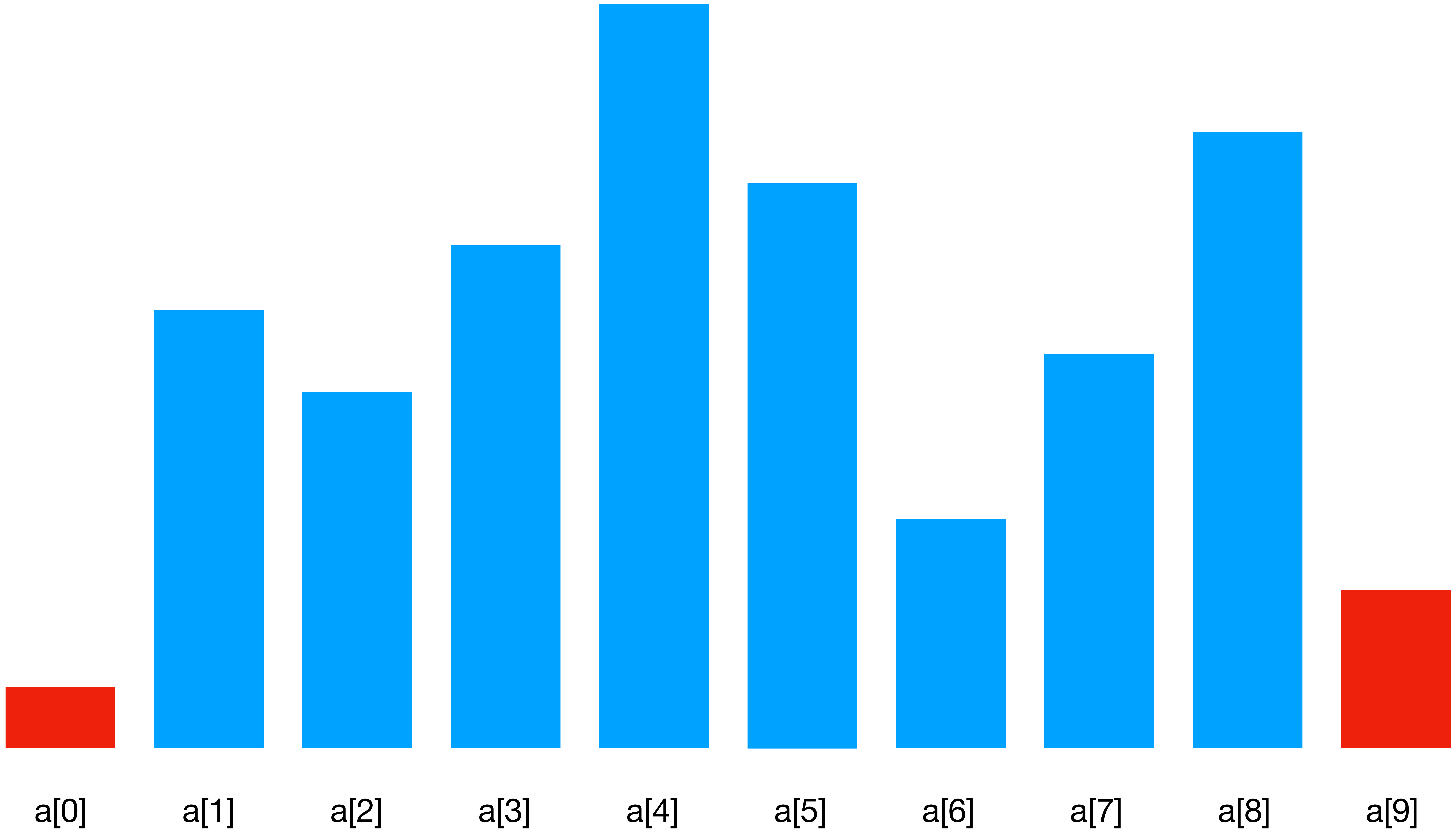
Selection Sort



Technical

P3
Bubble Sort

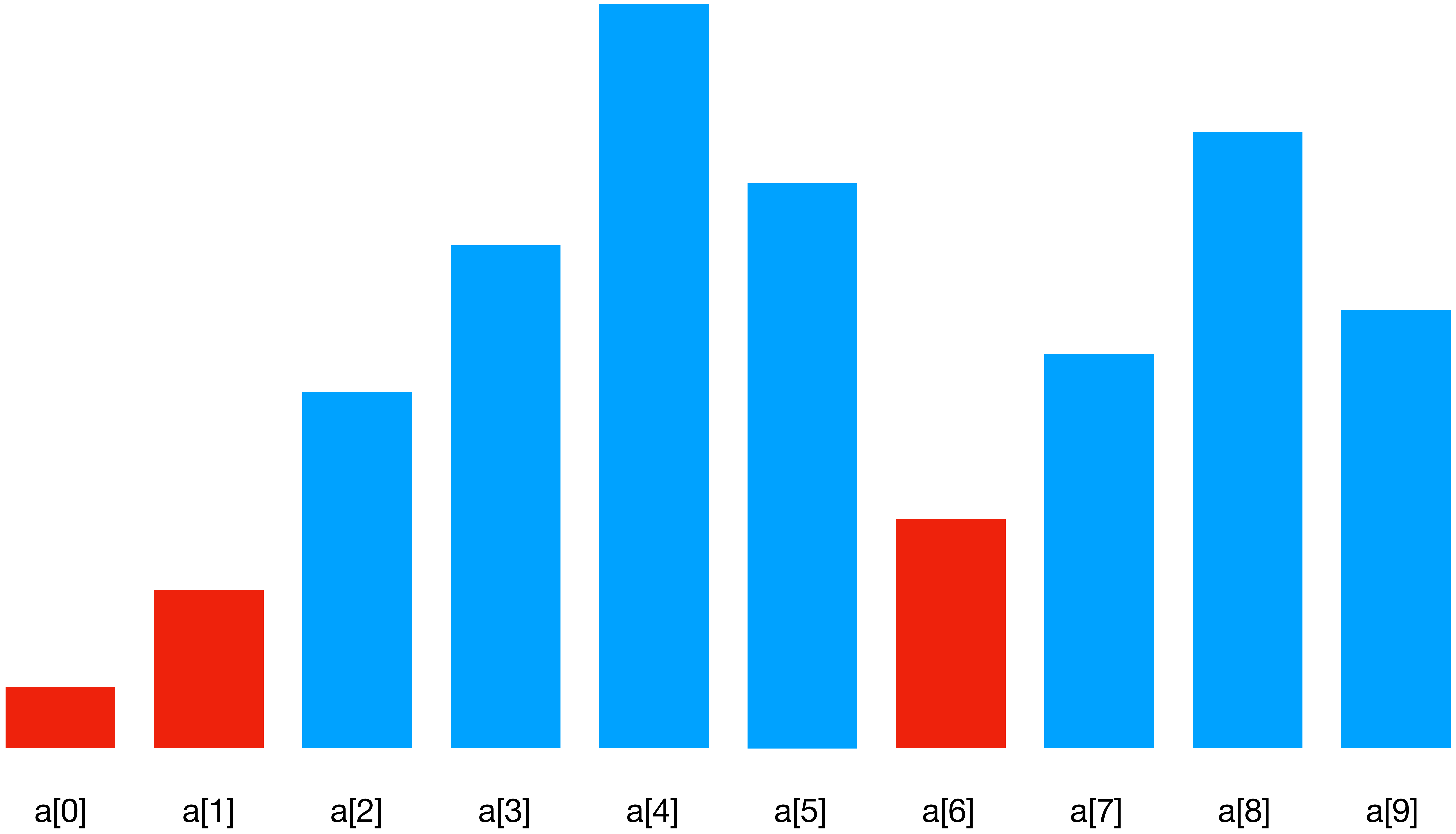
Selection Sort



Technical

P3
Bubble Sort

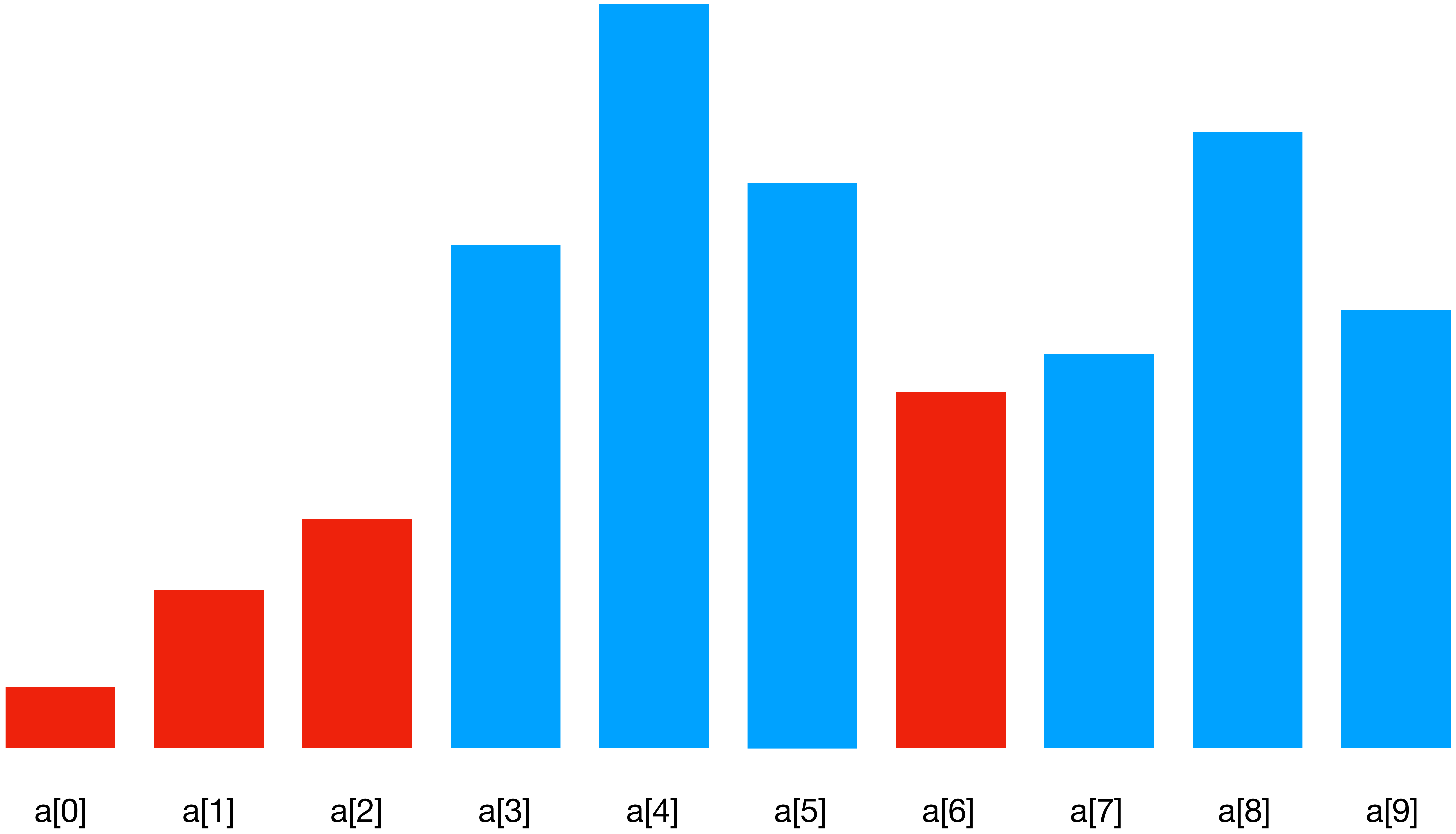
Selection Sort



Technical

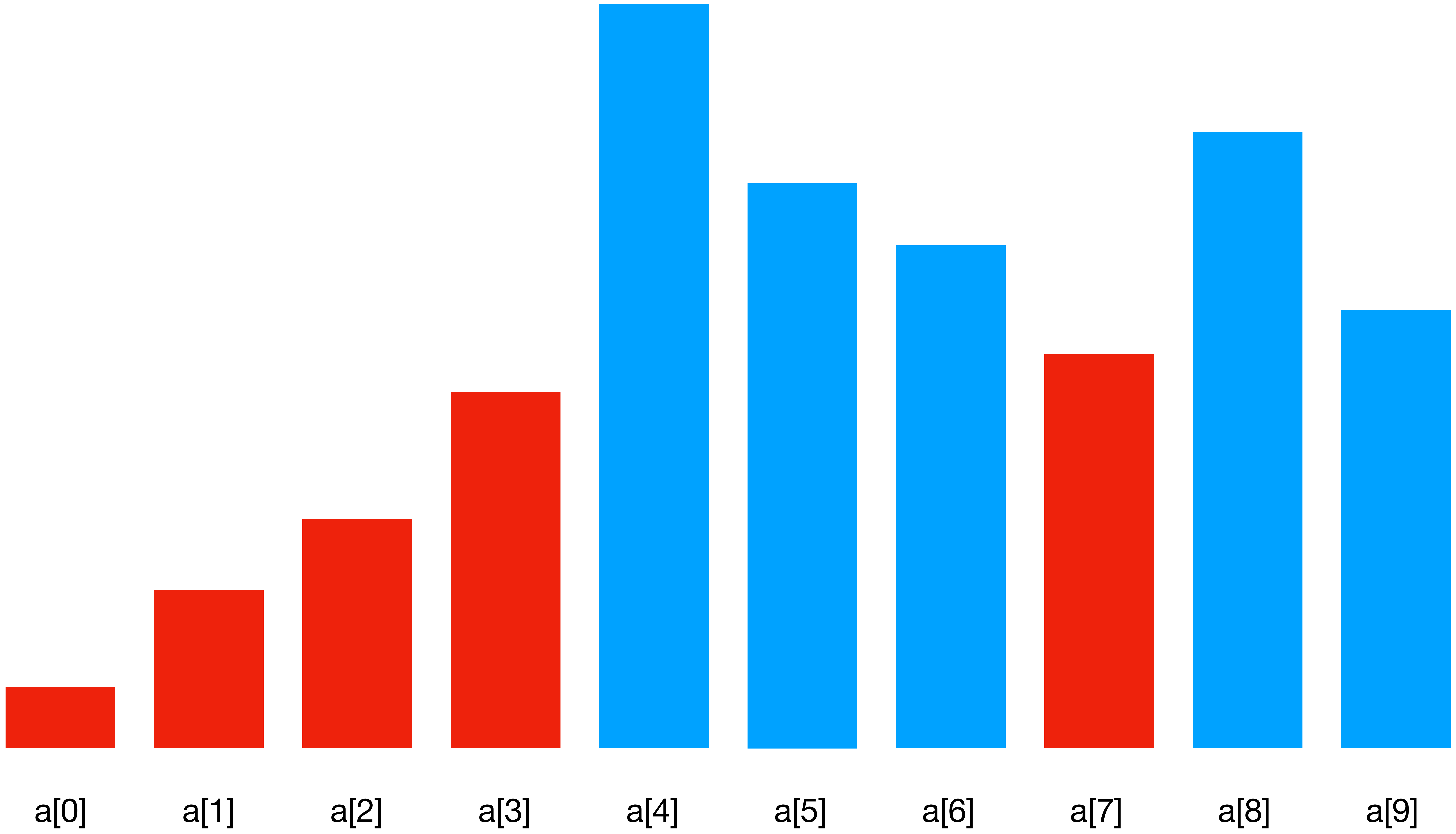
P3
Bubble Sort

Selection Sort



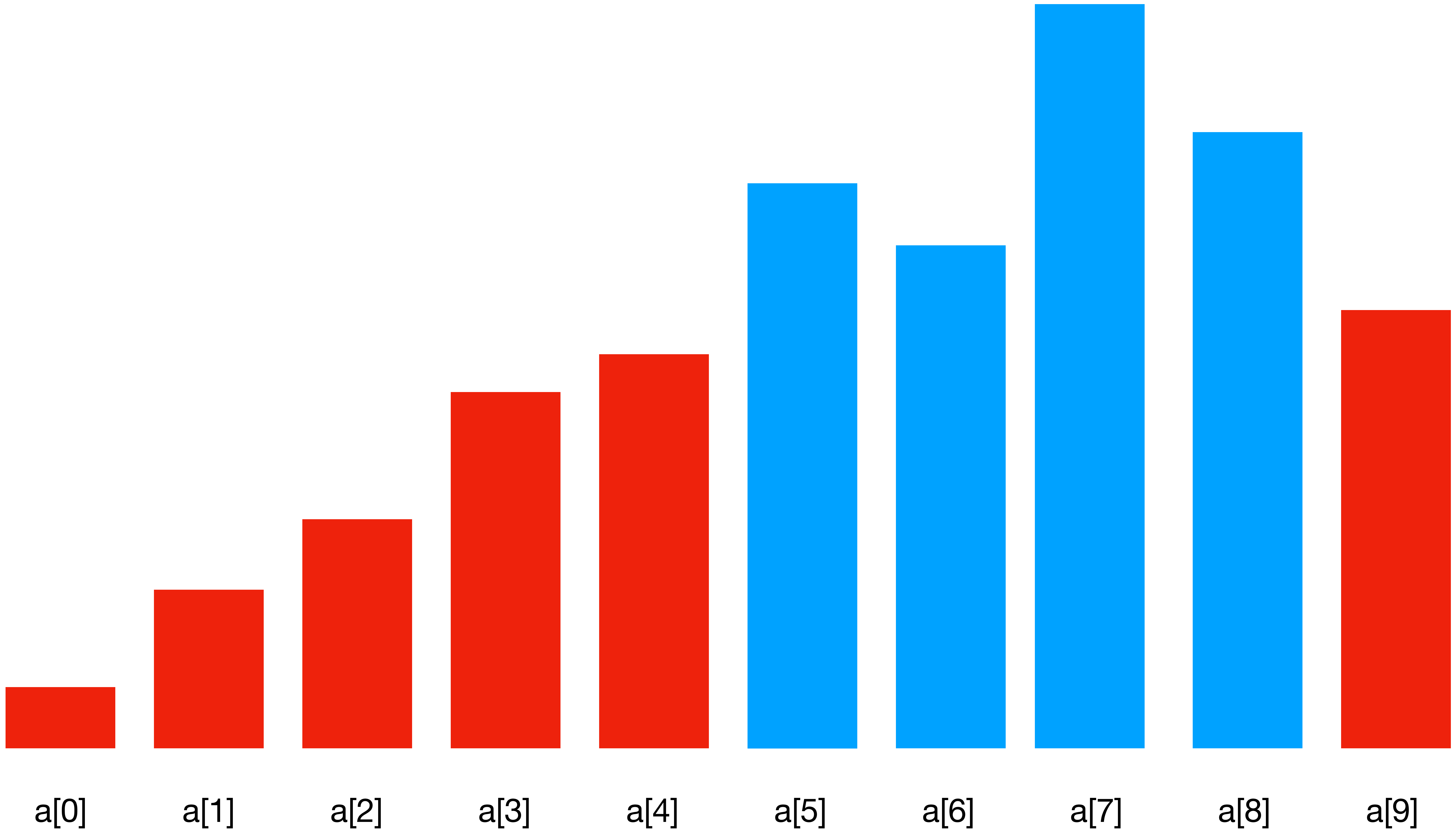
Technical

Selection Sort



P3
Bubble Sort

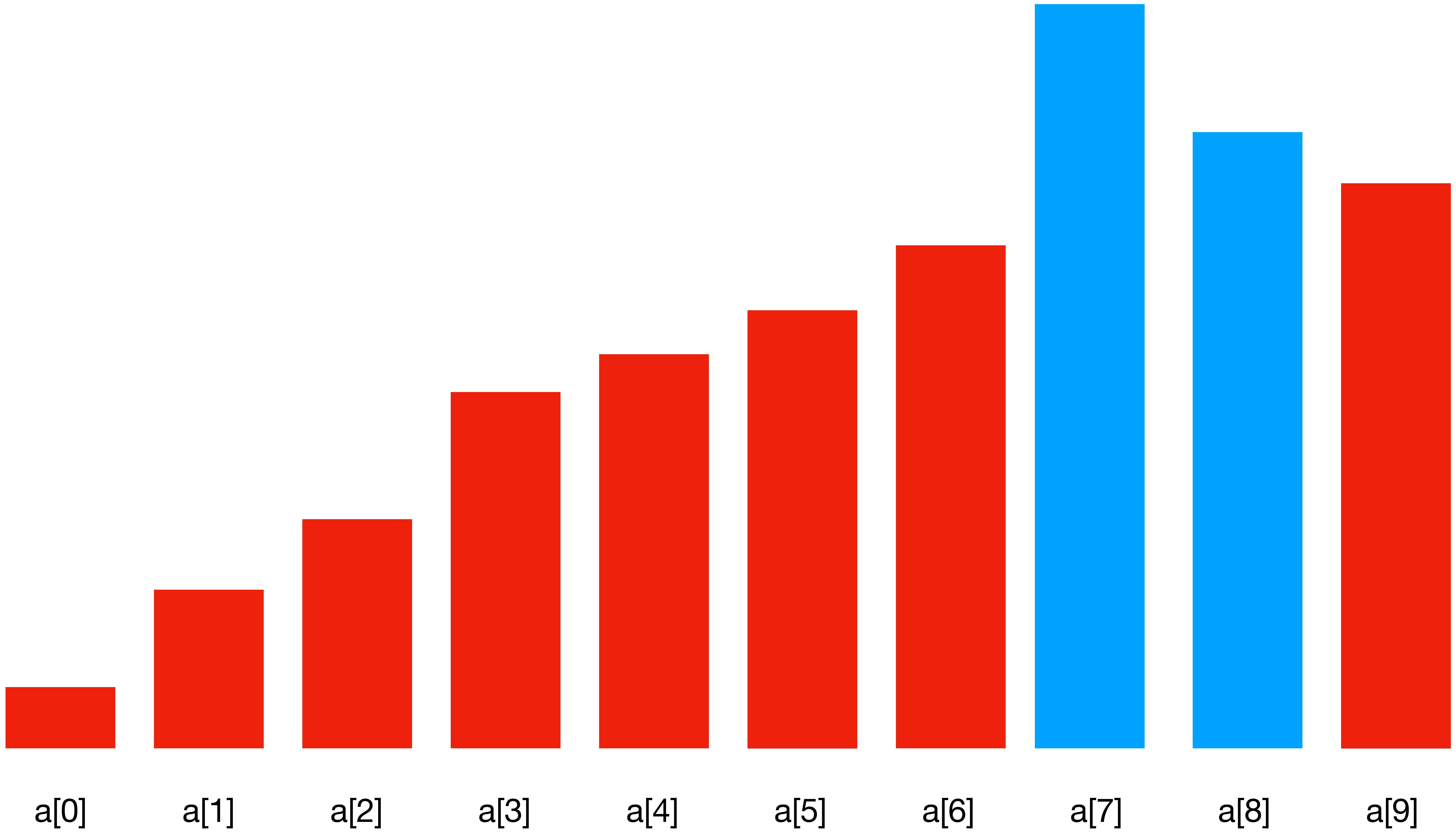
Selection Sort



Technical

P3
Bubble Sort

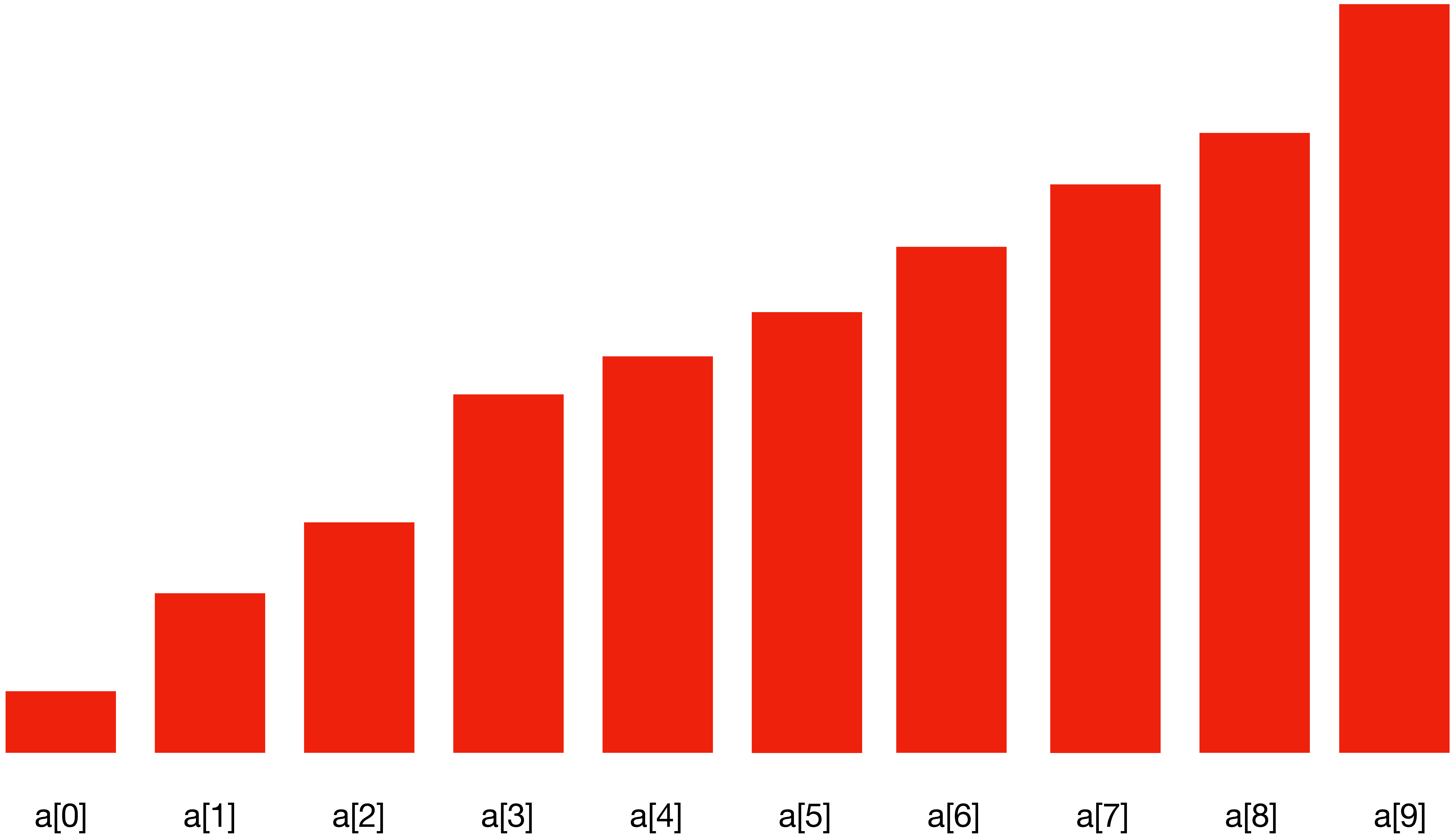
Selection Sort



Technical

P3
Bubble Sort

Selection Sort



Technical

Pseudo-Code

- Do not have strict grammar
- Simpler than programming language
- Layout a process/algorithm

Concept

Selection Sort

```
for i from 0 to n-2:
    minIndex = index of smallest number in array a[i:n];
    swap a[minIndex] and a[i];
```

- This is an example pseudocode. It is sufficient for human understanding but not detailed enough for computers since it's too simplified.

Concept