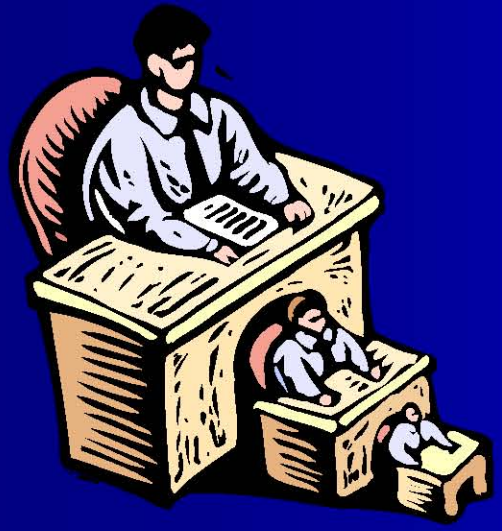


The Selection Sort

Mr. Dave Clausen

La Cañada High School

The Selection Sort Description

The Selection Sort searches (linear search) all of the elements in a list until it finds the smallest element. It "swaps" this with the first element in the list. Next it finds the smallest of the remaining elements, and "swaps" it with the second element. Repeat this process until you compare only the last two elements in the list.

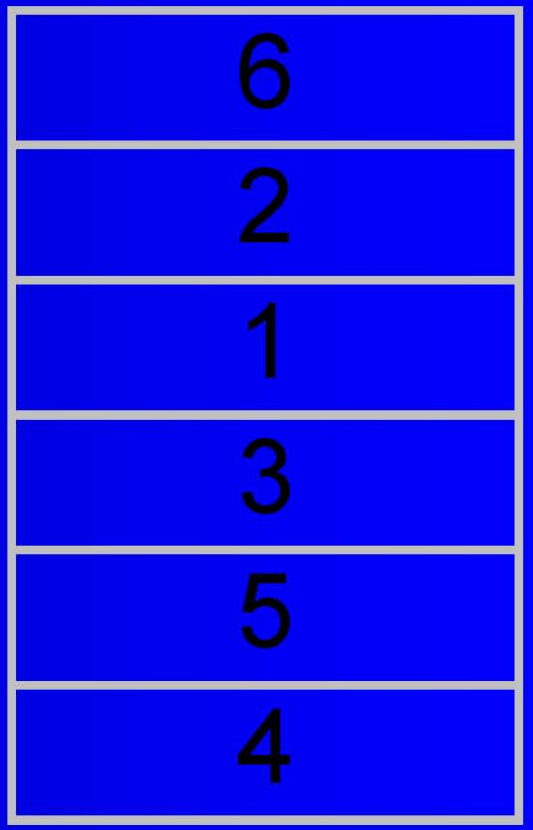
The Selection Sort Algorithm

- For each index position ***i***
  - Find the smallest data value in the array from positions ***i*** through length - 1, where length is the number of data values stored.
  - Exchange (swap) the smallest value with the value at position ***i***.

A Selection Sort Example

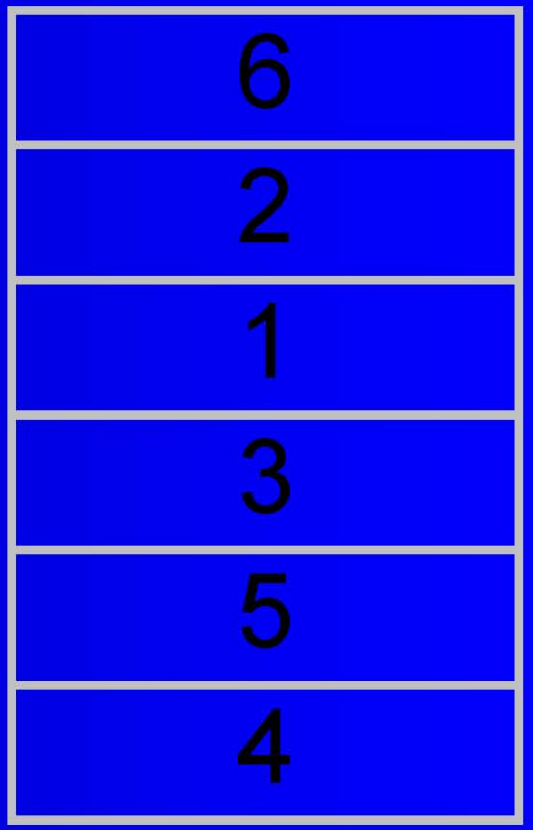
Smallest ?

We start by searching for the smallest element in the List.



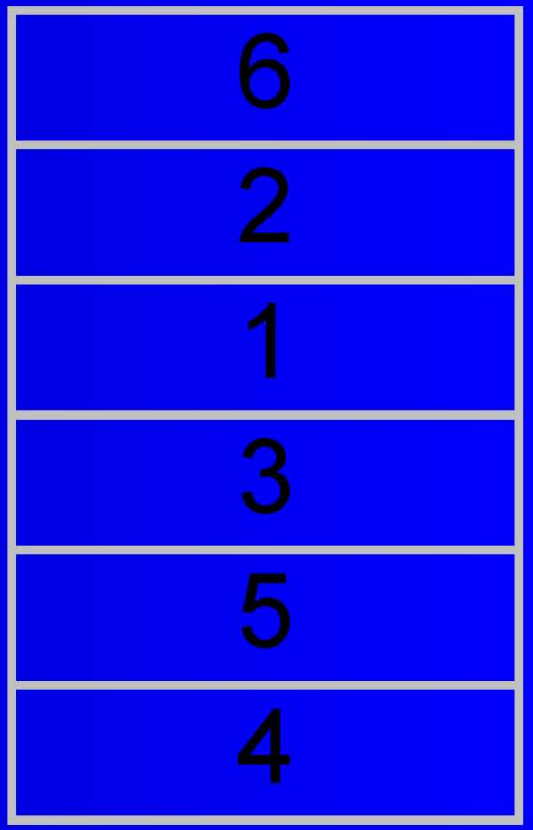
A Selection Sort Example

Smallest ?

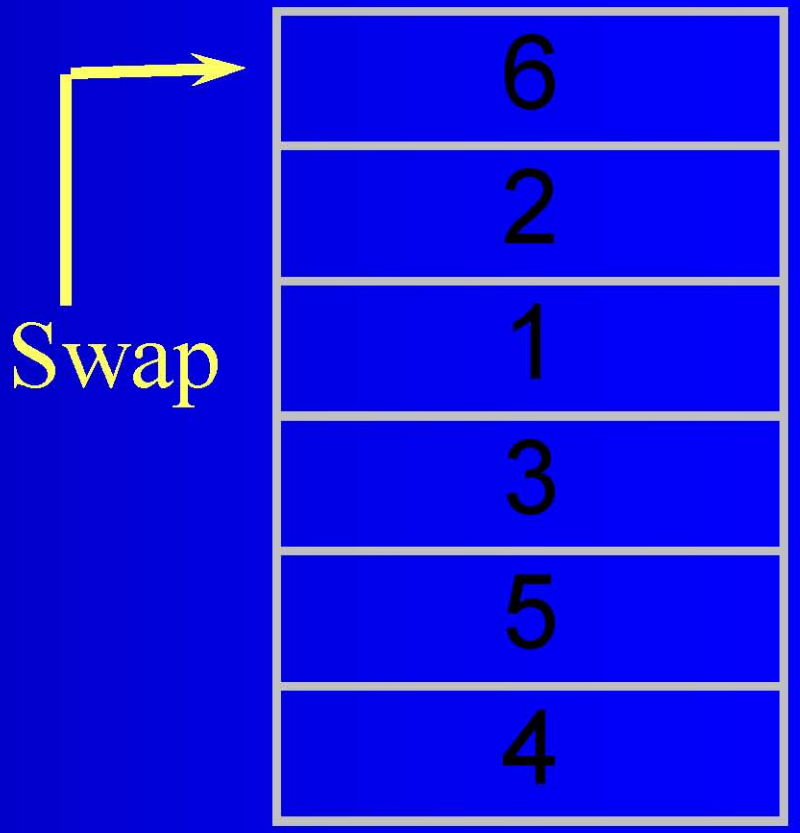


A Selection Sort Example

Smallest !

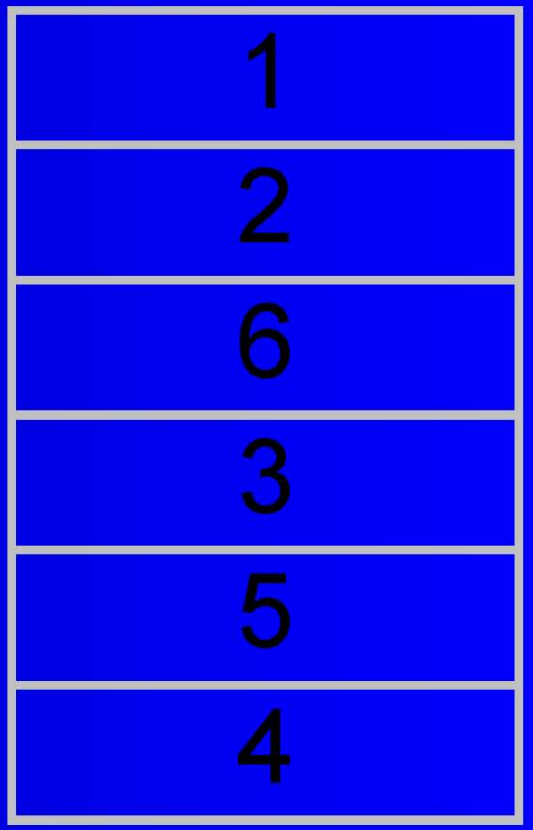


A Selection Sort Example



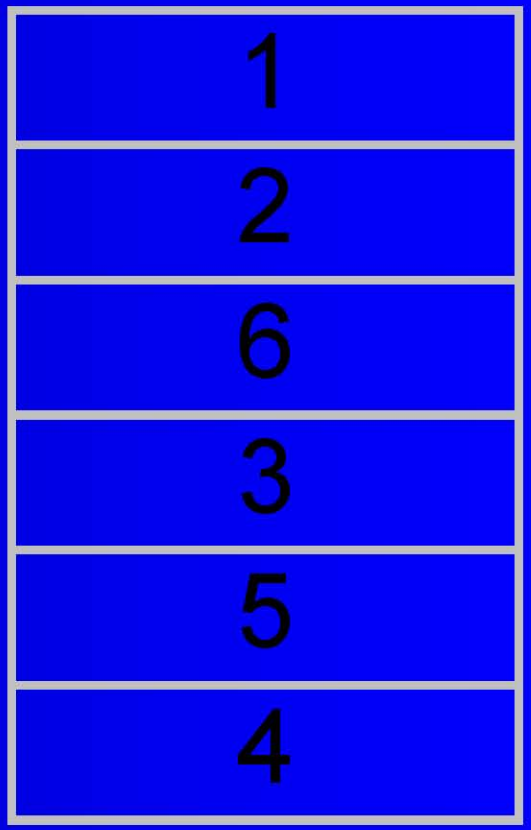
A Selection Sort Example

Swapped



A Selection Sort Example

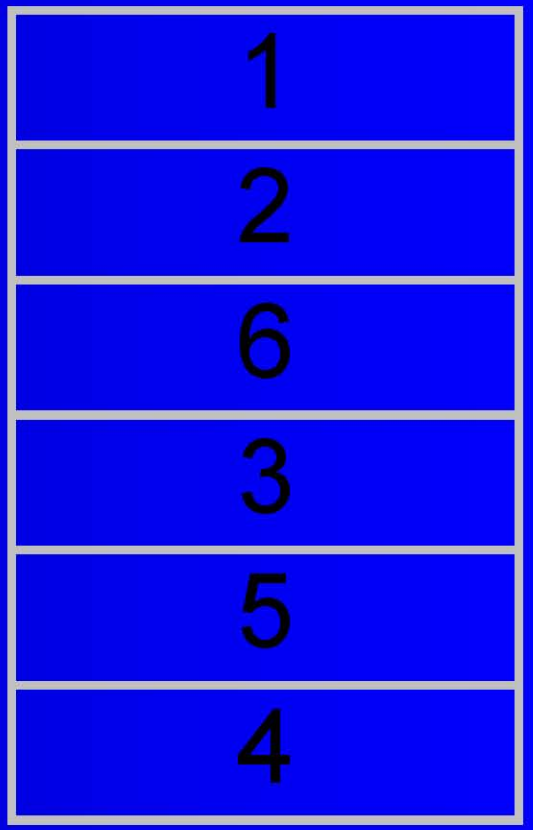
Smallest ?



After the smallest element is in the first position, we continue searching with the second element and look for the next smallest element.

A Selection Sort Example

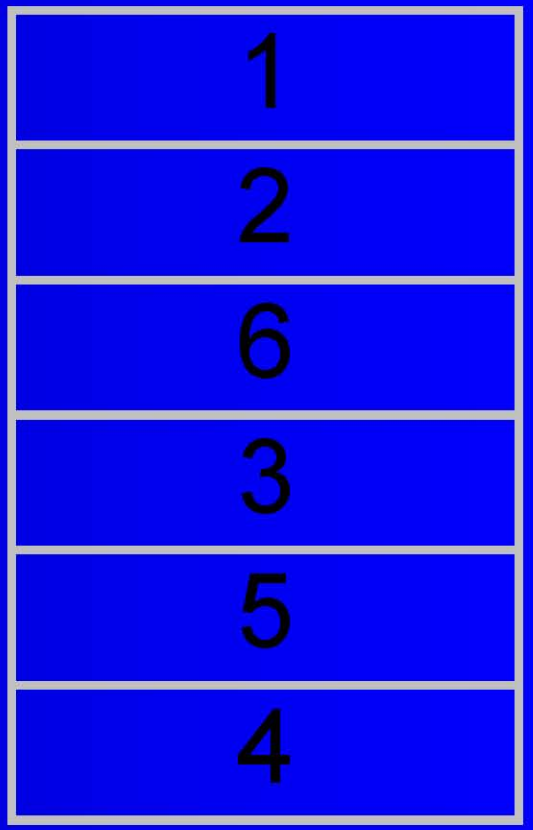
Smallest !



In this special case, the next smallest element is in the second position already. Swapping keeps it in this position.

A Selection Sort Example

Swapped

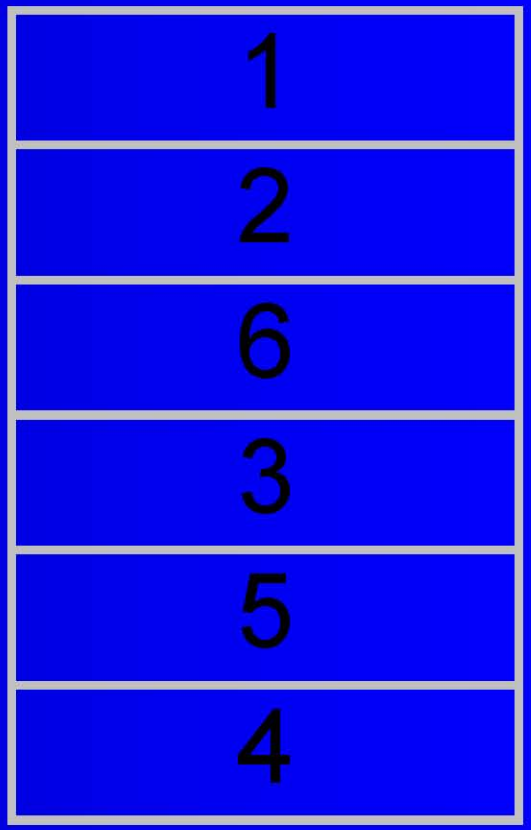


Swapping keeps it in this position.

A Selection Sort Example

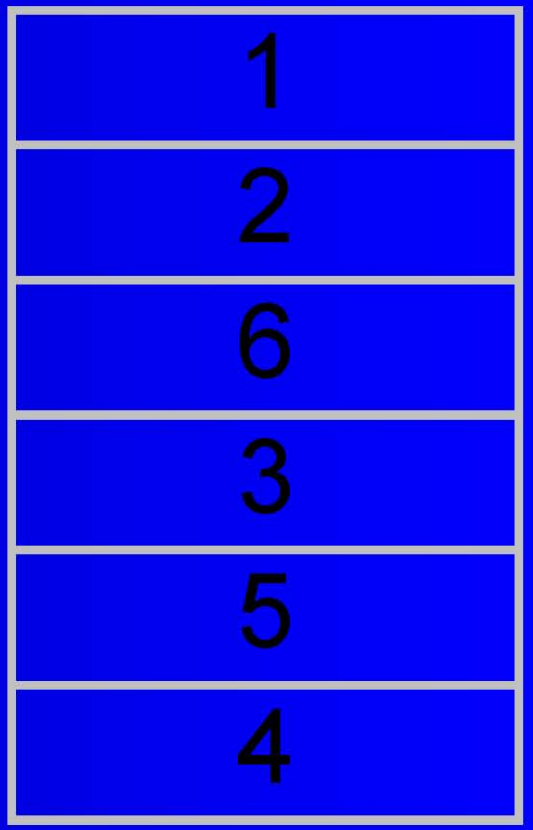
Smallest ?

After the next smallest element is in the second position, we continue searching with the third element and look for the next smallest element.

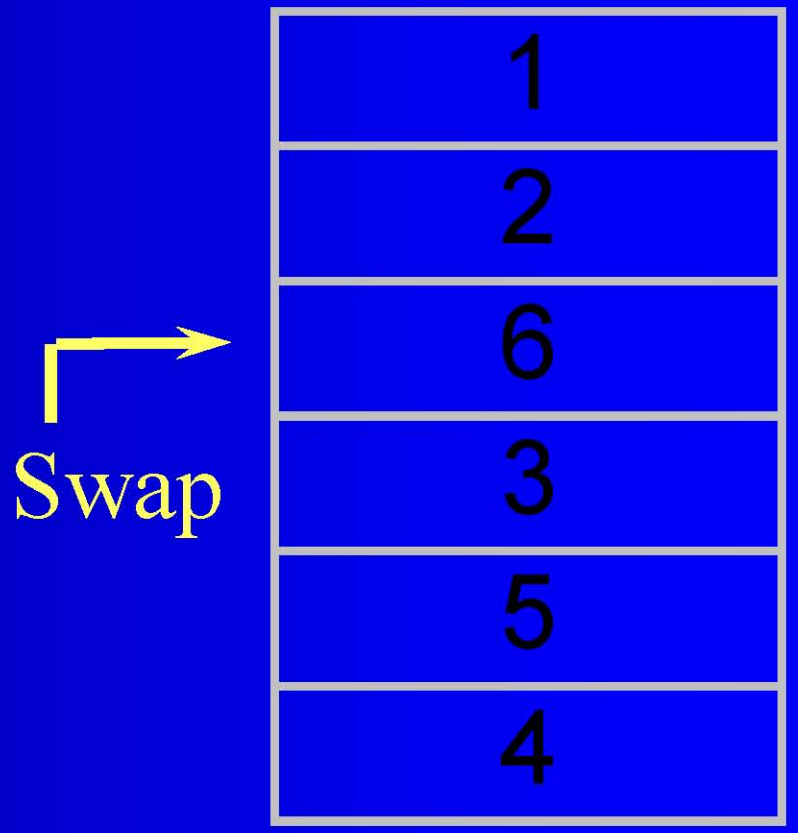


A Selection Sort Example

Smallest !

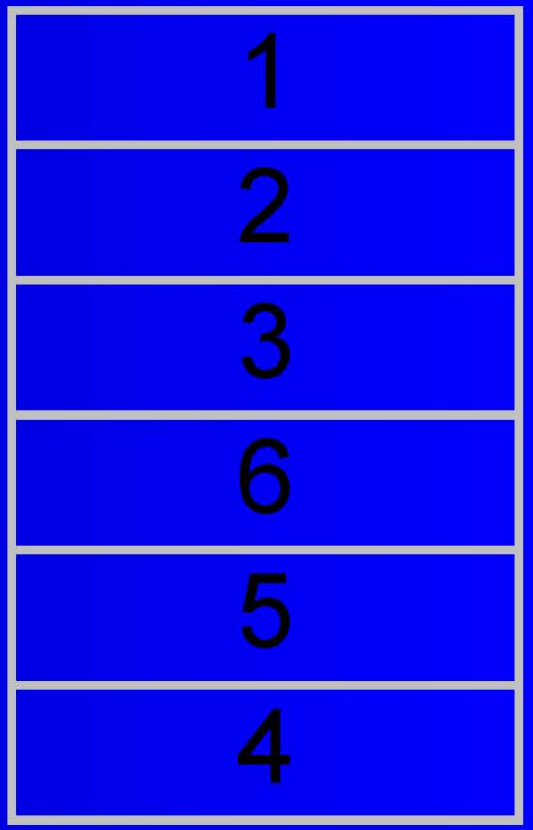


A Selection Sort Example



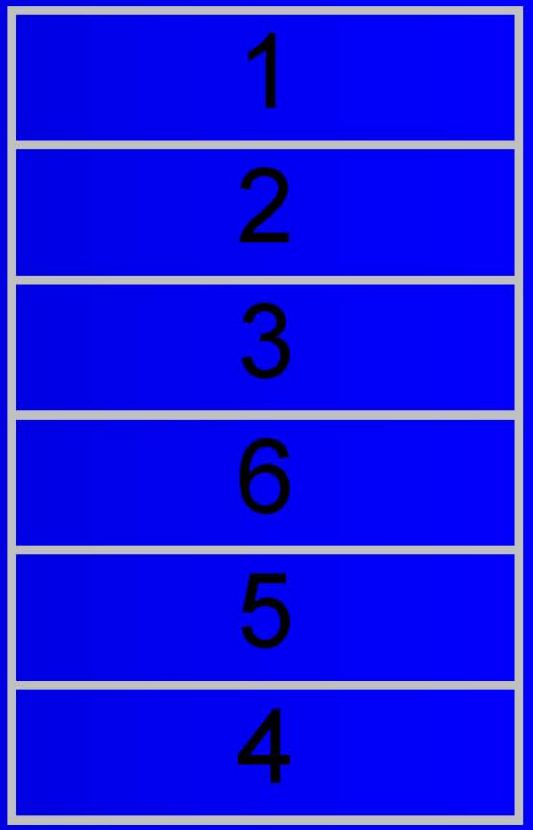
A Selection Sort Example

Swapped



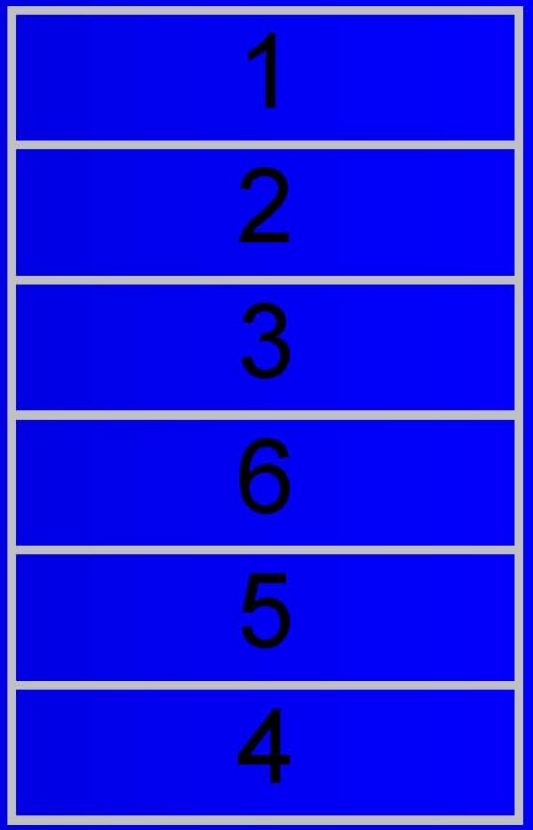
A Selection Sort Example

Smallest ?

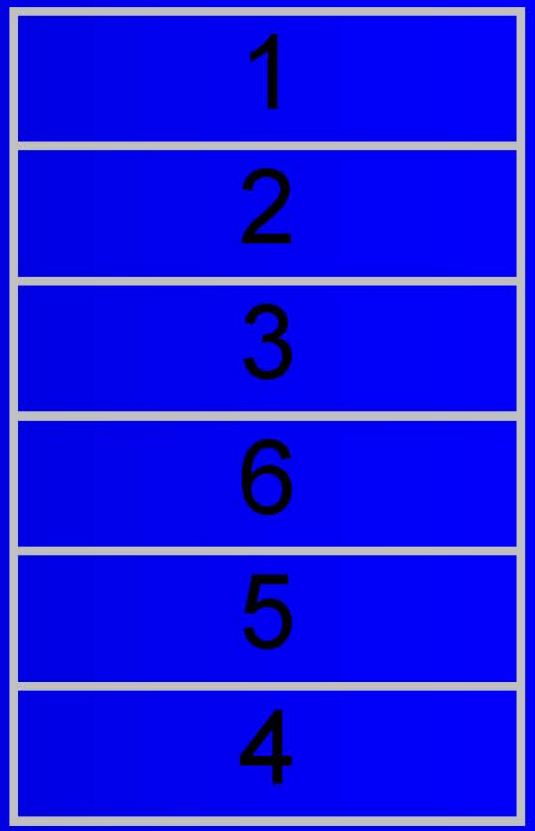


A Selection Sort Example

Smallest ?

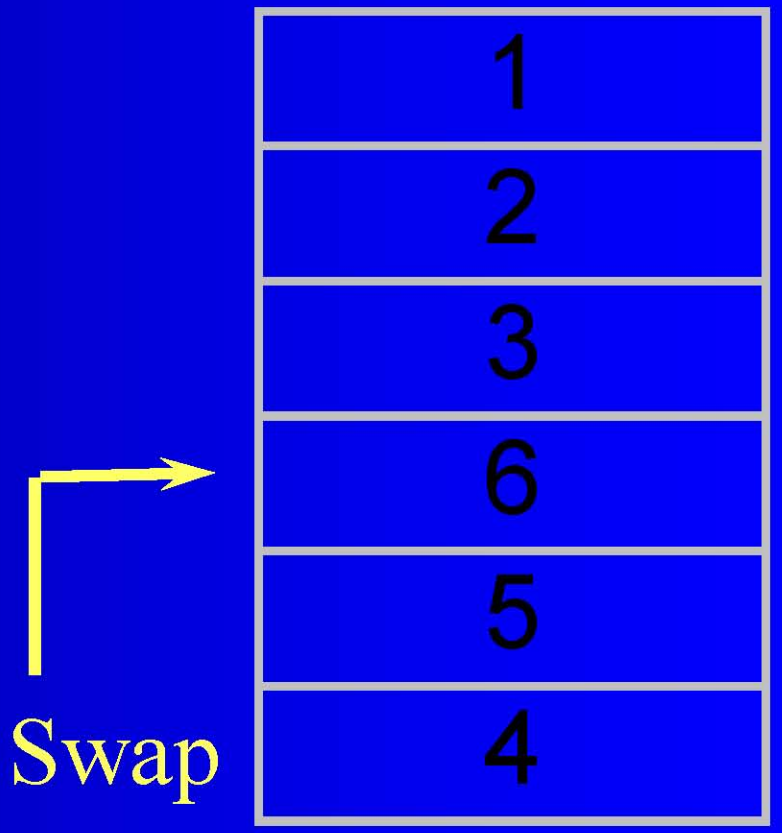


A Selection Sort Example

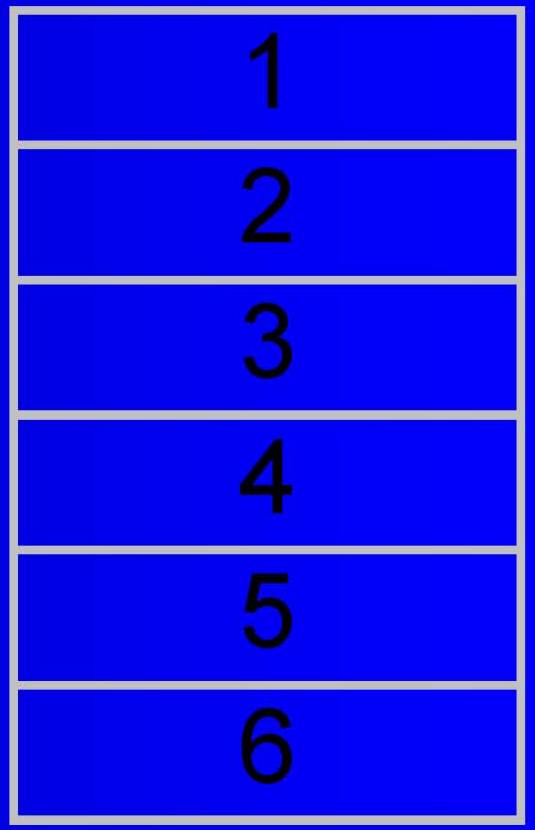


Smallest !

A Selection Sort Example



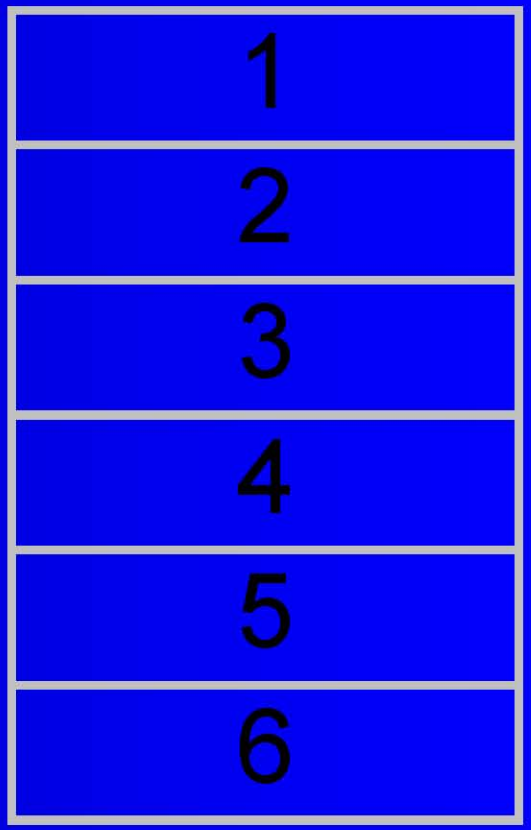
A Selection Sort Example



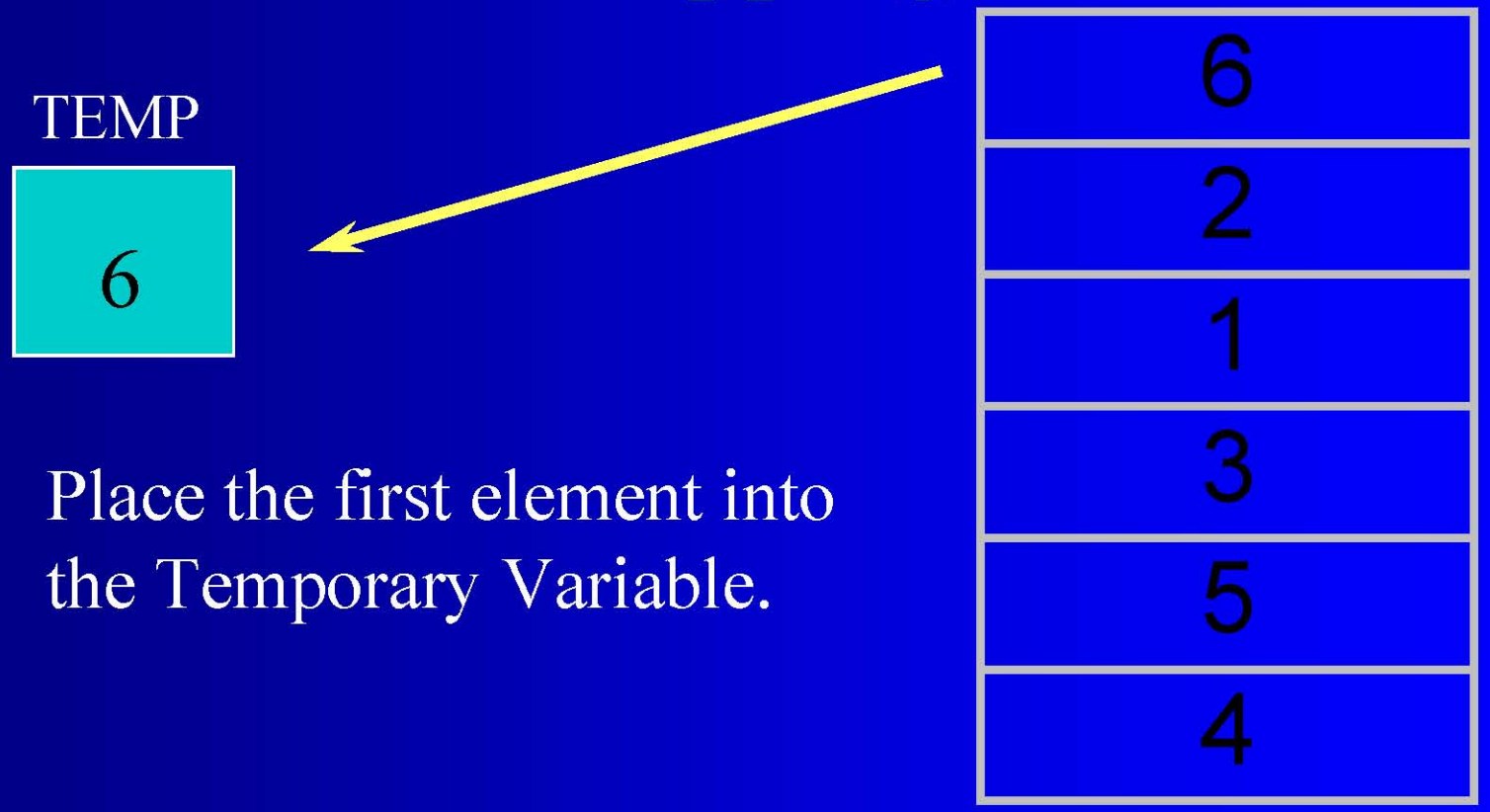
Swapped

A Selection Sort Example

The last two elements are in order, so no swap is necessary.



What "Swapping" Means



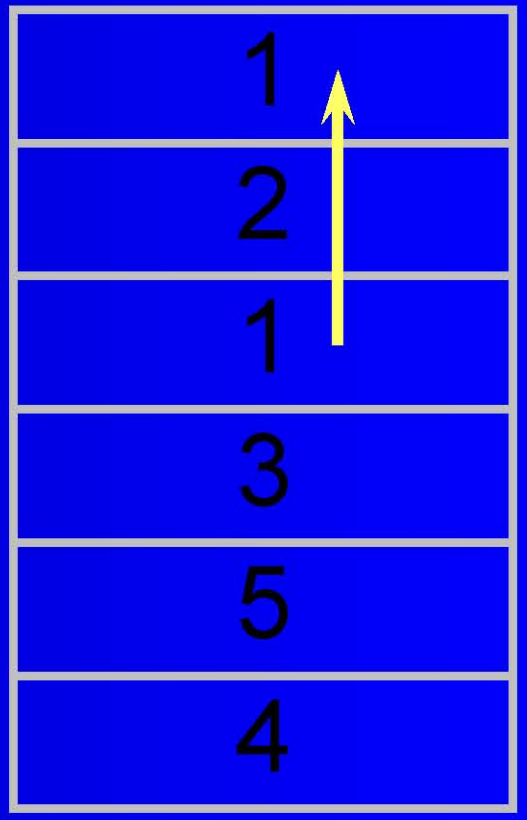
Place the first element into the Temporary Variable.

What "Swapping" Means

TEMP

6

Replace the first element with the value of the smallest element.



What "Swapping" Means

TEMP

6

1
2
6
3
5
4

Replace the third element (in this example) with the Temporary Variable.

C + + Examples of The Selection Sort

Sample C++ Program For Selection Sort:

sel_sort.cpp sel_sort.txt

Animated Examples:

Visual1.exe Visual1.cpp Visual1.txt

Visual2.exe Visual2.cpp Visual2.txt

On the Net:

http://compsci.exeter.edu/Winter99/CS320/Resources/sortDemo.html

http://www.aist.go.jp/ETL/~suzaki/AlgorithmAnimation/index.html

C + + Code For Selection Sort

```
void Selection_Sort (apvector <int> & v)
{
      int  min_index = 0;
      for (int index = 0; index < v.length( ) - 1;  ++index)
      {
          min_index = Find_Minimum (v, index);
          if (min_index ! = index)
              Swap_Data ( v [index], v [min_index])
      }
}
// Selection_Sort
```

C + + Code For Find Minimum

```
int Find_Minimum (const apvector <int> & v, int first)
{
      int  min_index = first;
      for (int index = first + 1;  index < v.length( );  ++index)
         if  ( v[index] < v[min_index])
                min_index = index;
      return min_index;
}
// Find_Minimum
```

C + + Code for Swap Procedure

```
void Swap_Data (int &number1, int &number2)
{
    int temp;
    temp = number1;
    number1 = number2;
    number2 = temp;
}
// End of Swap_Data function
```

Pascal Code for Swap Procedure

```
procedure Swap (var number1, number2: integer);
var
  temp: integer;
begin
  temp := number1;
  number1 := number2;
  number2 := temp
end;     {Swap}
```

Pascal Code For Selection Sort

```
procedure SelectionSort (var IntArray: IntNumbers);
  var    element, SmallIndex, index: integer;
begin
  for element := 1 to (MaxNum - 1) do
    begin
      SmallIndex := element;
      for index := (element + 1) to MaxNum do
        if IntArray [index] < IntArray [SmallIndex]
          then SmallIndex := index;
      Swap (IntArray [element], IntArray [SmallIndex])
    end;
end;     {SelectionSort}
```

BASIC Code For Selection Sort

```
8000 REM ####################
8010 REM Selection Sort
8020 REM ####################
8030 FOR ELEMENT = 1 TO MAXNUM - 1
8040 ::SmallIndex = element
8050 ::FOR INDEX = (element + 1)  TO MAXNUM
8060 ::::::IF N (INDEX) <  N (SmallIndex) THEN SmallIndex = INDEX
8070 ::NEXT INDEX
8080 ::TEMP = N (element)
8090 ::N (element) = N (SmallIndex)
8100 ::N (SmallIndex) = TEMP
8110 NEXT ELEMENT
8120 RETURN
```

Big - O Notation

Big - O notation is used to describe the efficiency of a search or sort. The actual time necessary to complete the sort varies according to the speed of your system. Big - O notation is an approximate mathematical formula to determine how many operations are necessary to perform the search or sort. The Big - O notation for the Selection Sort is $O(n^2)$, because it takes approximately n^2 passes to sort the elements.