

Lecture 4 - Example Algorithm Development

Another 2 algorithms - swapping and sorting

We saw at the end of Lecture 3 how to develop an algorithm to identify the maximum value out of a vector, and the location of that value. In this lecture we will combine that idea with the ability to swap values within an array to create a simple sorting algorithm. Remember that there are already built-in Mathcad commands to sort a vector. The purpose of this exercise is to learn the basic use of control structures and algorithm development so that we can create our own more advanced programs later on.

swapping - switch the contents of two variables

Given: $x = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$ but I want $x = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ i.e. I want to swap the contents of spots 1 and 2

How can I swap the contents of x_1 with x_2 ? Will this work??

$x_1 := x_2$

$x_2 := x_1$

Why not?

$x_1 := x_2 \implies x_1 = 3$

$x_2 := x_1 \implies x_2 = 3$

What we need is some place to temporarily store one or the other

$temp := x_1$

$x_1 := x_2$

$x_2 := temp$

$temp := x_1 \implies temp = 2, x = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$

$x_1 = x_2 \implies temp = 2, x = \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix}$

$x_2 = temp \implies temp = 2, x = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$

mission accomplished.

what would **pseudo-code** look like?

- 1) identify the variables to swap
- 2) save the contents of #1 to a temporary location
- 3) replace contents of #1 with that of #2
- 4) replace the contents of #2 with that in the temporary location

From now on, this entire pseudo-code can be a one word command...

SWAP variables #1 and #2

selection sort - re-order contents of a vector from low to high, or high to low.

Combines the previous two algorithms: 1) finding a maximum 2) swapping.

we have student grades $G = \begin{bmatrix} 4.1 \\ 7.3 \\ 1.7 \\ 5.2 \\ 1.3 \end{bmatrix}$

we want to re-order from LOWEST to HIGHEST

Pseudo-code:

- 1) Find the location of the lowest value in the entire vector
- 2) Swap the contents of that spot with the #1 spot
- 3) find the location of the second lowest value in the entire vector
 - a) or...find the lowest value in the entire vector, *excluding #1*
- 4) Swap the contents of that spot with the #2 spot
- 5) Find the location of the lowest value remaining in the entire vector
 - excluding spots #1 and #2.
- 6) Swap the contents of that spot with #3
- 7) repeat steps 5) and 6), moving down one spot each time

Sorting - let's illustrate how we intend this algorithm to work ...

Given: $G = \begin{bmatrix} 4.1 \\ 7.3 \\ 1.7 \\ 5.2 \\ 1.3 \end{bmatrix}$

after iteration #1 $G = \begin{bmatrix} 1.3 \\ 7.3 \\ 1.7 \\ 5.2 \\ 4.1 \end{bmatrix}$

after iteration #2 $G = \begin{bmatrix} 1.3 \\ 1.7 \\ 7.3 \\ 5.2 \\ 4.1 \end{bmatrix}$

after iteration #3 $G = \begin{bmatrix} 1.3 \\ 1.7 \\ 4.1 \\ 5.2 \\ 7.3 \end{bmatrix}$

$$\text{after iteration \#4 } G = \begin{bmatrix} 1.3 \\ 1.7 \\ 4.1 \\ 5.2 \\ 7.3 \end{bmatrix}$$

we're done. # of iterations needed? One less than the number of values in the vector
This information will be used when we set up a loop

Making it Work

Let's build this up slowly by refining the pseudo-code and putting together one iteration at a time.

- Pseudo-code for iteration #1 only
- Fill in some Mathcad code for iteration #1
- Repeat for iteration #2
- Repeat for iteration #3
- Generalize when we **find a pattern**

iteration #1 (smallest at the top)

- Pseudo-code for iteration #1:
- Fill in some Mathcad code for iteration #1

Pseudo-code

assume smallest value in the vector is already in spot 1 (set pointer = 1)
 compare the remaining values to find minimum (set first = 2)
 when the actual smallest value is found, swap with what is in spot #1

Mathcad code to first iteration

ORIGIN ≡ 1

$$G := \begin{pmatrix} 4.1 \\ 7.3 \\ 1.7 \\ 5.2 \\ 1.3 \end{pmatrix} \quad \text{INPUT VECTOR TO SORT}$$

```

iteration_1 (in) :=
    last ← length (in)
    out ← in
    ptr ← 1
    first ← 2
    for k ∈ first..last
        ptr ← k if outk < outptr
    temp ← out1
    out1 ← outptr
    outptr ← temp
    out
    
```

G_1 := iteration_1 (G)

$$G_1 = \begin{pmatrix} 1.3 \\ 7.3 \\ 1.7 \\ 5.2 \\ 4.1 \end{pmatrix}$$

OUTPUT OF ITERATION #1

iteration #2 (smallest remaining in spot 2)**Pseudo-code**

assume smallest is already in spot 2 (set ptr = 2)
 compare the remaining values to find minimum (set first = 3)
 when actual smallest is found, swap with #2

Mathcad code

```

iteration_2(in) :=
  last ← length(in)
  out ← in
  ptr ← 2
  first ← 3
  for k ∈ first..last
    ptr ← k if outk < outptr
  temp ← out2
  out2 ← outptr
  outptr ← temp
  out

```

$$G_2 := \text{iteration_2}(G_1)$$

$$G_2 = \begin{pmatrix} 1.3 \\ 1.7 \\ 7.3 \\ 5.2 \\ 4.1 \end{pmatrix}$$

OUTPUT OF ITERATION #2

What has changed since the first iteration? Just the assignment to 'ptr' and 'first' and the index to 'out'

iteration #3 (smallest remaining in spot 3)**Pseudo-code**

assume smallest is already in spot 3 (set ptr = 3)
 compare the remaining values to find minimum (set first = 4)
 when actual smallest is found, swap with #3

Mathcad code

```

iteration_3 (in) := | last ← length (in)
                   | out ← in
                   | ptr ← 3
                   | first ← 4
                   | for k ∈ first.. last
                   |   ptr ← k if outk < outptr
                   | temp ← out3
                   | out3 ← outptr
                   | outptr ← temp
                   | out

```

$$G_3 := \text{iteration}_3(G_2)$$

$$G_3 = \begin{pmatrix} 1.3 \\ 1.7 \\ 4.1 \\ 5.2 \\ 7.3 \end{pmatrix}$$

OUTPUT OF ITERATION #3

Time to generalize the above 3 iterations

iteration #i (smallest remaining value in spot i)

Pseudo-code

assume smallest is already in spot i (set ptr = i)

compare the remaining values to find minimum (set first = i+1)

when actual smallest is found, swap with i

```

iteration_3 (in) :=
  last ← length (in)
  out ← in
  ptr ← i
  first ← i + 1
  for k ∈ first .. last
    ptr ← k if outk < outptr
  temp ← outi
  outi ← outptr
  outptr ← temp
  out

```

What's left to do? Now we just have to put the process of performing i iterations into another loop so that iteration 1 is followed by 2, 3, etc. until the sorting is done.

'i' must be inside a loop counting from 1 to one less than the length of the vector (length(G)-1)

We will take the above algorithm to sort the ith iteration, and put it inside (nest it) of another loop which dictates which iteration we are in. The algorithm will then go through all necessary iterations in a single function call.

the selection swap routine in Mathcad

$G := \begin{pmatrix} 4.1 \\ 7.3 \\ 1.7 \\ 5.2 \\ 1.3 \end{pmatrix}$

```

sorter(in) := | last ← length(in)
               | out ← in
               | for i ∈ 1..last - 1
               |   ptr ← i
               |   first ← i + 1
               |   for k ∈ first..last
               |     ptr ← k if outk < outptr
               |   temp ← outi
               |   outi ← outptr
               |   outptr ← temp
               | out
    
```

here is the outside loop we added to keep track of whic of the 'i' iterations we are currently in

all to the right of this line is the stuff we saw on the last page.

$sorter(G) = \begin{pmatrix} 1.3 \\ 1.7 \\ 4.1 \\ 5.2 \\ 7.3 \end{pmatrix}$