MATLAB: Tables of Data – Sorting data as a table – Operating on tables Extracting data from Tables – Modifying table in MATLAB

MATLAB has several built-in functions that perform set operations on vectors. These include union, intersect, unique, setdiff, and setxor. All these functions can be useful when working with data sets. Additionally, there are two is functions that work on sets, is member and is sorted.

For example, given the following vectors:

>> v1 = 2:6

v1 = 2 3 4 5 6

>> v2 = 1:2:7

v2 = 1 3 5 7

The union function returns a vector that contains all the values from the twoinput argument vectors, without repeating any.

>> union(v1,v2)

ans = 1 2 3 4 5 6 7

The intersect functioninstead returns all the values that can be found in bothof the input argument vectors.

>> intersect(v1,v2)

ans = 3 5

The setdifffunctionreceives two vectors as input arguments, and returns a vectorconsisting of all the values that are contained in the first vector argument butnot the second. Therefore, the order of the two input arguments is important.

>> setdiff(v1,v2)

ans = 2 4 6

>> setdiff(v2,v1)

ans = 1 7

The function setxor receives two vectors as input arguments, and returns a vectorconsisting of all the values from the two vectors that are not in the intersectionof these two vectors. In other words, it is the union of the two vectors obtained using setdiff earlier.

>> setxor(v1,v2)

ans = 1 2 4 6 7

>> union(setdiff(v1,v2), setdiff(v2,v1))

ans = 1 2 4 6 7

The set function unique returns all the unique values from a set argument:

>> v3 = [1:5 3:6]

v3 = 1 2 3 4 5 3 4 5 6

>> unique(v3)

ans = 1 2 3 4 5 6

Many of the set functions return vectors that can be used to index into the originalvectors as optional output arguments; for example, for the two vectors v1and v2 defined previously as:

>> v1

v1 = 2 3 4 5 6

>> v2

v2 = 1 3 5 7

The intersect function returns, in addition to the vector containing thevalues in the intersection of v1 and v2, an index vector into v1 and anindex vector into v2 such that outvec is the same as v1(index1) and alsov2(index2).

>> [outvec, index1, index2] = intersect(v1,v2)

outvec = 3 5

index1 = 2 4

index2 = 2 3

The function ismember receives two vectors as input arguments, and returnsa logical vector that is the same length as the first argument, containing 1 fortrue if the element in the first vector is also in the second, or 0 for false if not.The order of the arguments matters for this function.

>> v1

v1 = 2 3 4 5 6

>> v2

v2 = 1 3 5 7

>> ismember(v1,v2)

ans = 0 1 0 1 0

>> ismember(v2,v1)

ans = 0 1 1 0

The issorted function will return 1 for logical true if the argument is sorted inascending order (lowest to highest), or 0 for false if not.

>> v3 = [1:5 3:6]

v3 = 1 2 3 4 5 3 4 5 6

>>issorted(v3)

ans = 0

Sorting

Sort Vector in Ascending Order:Create a row vector and sort its elements in ascending order.B = sort(A) sorts the elements of A in ascending order.

A = [9 0 -7 5 3 8 -10 4 2];

B = sort(A)

B = 1×9

-10 -7 0 2 3 4 5 8 9

Sort Matrix Rows in Ascending Order: Create a matrix and sort each of its rows in ascending order.If A is a matrix, then sort(A) treats the columns of A as vectors and sorts each column.

A = [3 6 5; 7 -2 4; 1 0 -9]

A = 3×3

3 6 5

7 -2 4

1 0 -9

B = sort(A,2)

B = 3×3

3 5 6

-2 4 7

-9 0 1

Sort Matrix Columns in Descending Order: Create a matrix and sort its columns in descending order.

A = [10 -12 4 8; 6 -9 8 0; 2 3 11 -2; 1 1 9 3]

A = 4×4

10 -12 4 8

6 -9 8 0

2 3 11 -2

1 1 9 3

B = sort(A,'descend')

B = 4×4

10 3 11 8

6 1 9 3

2 -9 8 0

1 -12 4 -2

Sort String Array

Starting in R2017a, you can create string arrays using double quotes, and sort them using the sort function. Sort strings in each column of a string array according to Unicode dictionary order.

A = ["Santos","Burns"; ...

"Jones","Morita"; ...

"Petrov","Adams"];

B = sort(A)

B = 3x2 string array

"Jones" "Adams"

"Petrov" "Burns"

"Santos" "Morita"

Sort the strings in each row.

B = sort(A,2)

B = 3x2 string array

"Burns" "Santos"

"Jones" "Morita"

"Adams" "Petrov"

Sort and Index Datetime Array

Create an array of datetime values and sort them in ascending order, that is, from the earliest to the latest calendar date.

ds = {'2012-12-22';'2063-04-05';'1992-01-12'};

A = datetime(ds,'Format','yyyy-MM-dd')

A = 3x1 datetime array

2012-12-22

2063-04-05

1992-01-12

[B,I] = sort(A)

B = 3x1 datetime array

1992-01-12

2012-12-22

2063-04-05

I = 3×1

3

1

2

B lists the sorted dates and I contains the corresponding indices of A.

Access the sorted elements from the original array directly by using the index array I.

A(I)

ans = 3x1 datetime array

1992-01-12

2012-12-22

2063-04-05

Searching

Searching means looking for a value (a key) in a list orin a vector. We already have seen that MATLAB has afunction, find, that will return the indices in an arraythat meet a criterion. To examine the programming methodologies, we will inthis section examine two search algorithms:

Sequential search

Binary search

Sequential Search:A sequential search is accomplished by looping through the vector element-byelementstarting from the beginning, looking for the key. Normally the indexof the element in which the key is found is what is returned. For example, hereis a function that will search a vector for a key and return the index or the value 0 if the key is not found.

>> values = [85 70 100 95 80 91];

>> key = 95;

>> seqsearch(values, key)

ans = 4

>> seqsearch(values, 77)

ans = 0

Binary search assumes that the vector has been sorted first. The algorithm issimilar to the way it works when looking for a name in a phone directory (which issorted alphabetically). To find the value of a key, look at the element in the middle.

>> vec = randint(1,7, [1 30])

vec = 2 11 25 1 5 7 6

>> svec = sort(vec)

svec = 1 2 5 6 7 11 25

>> binsearch(svec, 4)

ans = −1

>> binsearch(svec, 25)

ans = 7

>> binsearch(svec, 5)

ans = 3